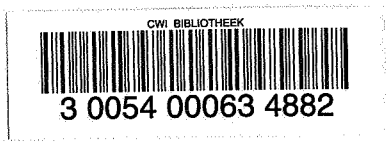
Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

T. Tomiyama, P.J.W. ten Hagen

The concept of intelligent integrated interactive CAD systems

Computer Science/Department of Interactive Systems Report CS-R8717 April



The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

The Concept of Intelligent Integrated Interactive CAD Systems

Tetsuo Tomiyama, Paul J.W. ten Hagen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In this report first we propose the concept of Intelligent Integrated Interactive CAD (*IIICAD*) systems, after having analyzed problems of present Computer Aided Design (CAD) systems. *IIICAD* is expected to be a large software system based on knowledge engineering technology. In order to develop such a complicated system we need to put emphasis on the importance of theoretical work besides implementational techniques, since these techniques cannot even solve all problems of conventional CAD systems.

1980 *Mathematics Subject Classification:* 69H12, 69H21, 69K10, 69K36, 69L60.
1982 *CR Categories:* H.2.1, I.2.1, I.3.6, J.6.

Key Words & Phrases: CAD, machine design, design theory, knowledge engineering, interactive systems, conceptual modeling.

Note: This report may be submitted for publication elsewhere.

1. Introduction

Computer Aided Design (CAD) systems are now indispensable in many industries, such as mechanical, aeronautical, electrical, automobile, architectural, and chemical industries. Especially, in mechanical engineering where geometrical information plays a crucial role, they have become not optional but necessary.

CAD in mechanical engineering is now in its third phase. Roughly speaking, the first phase began with SKETCHPAD [35] of MIT in 1963, followed by a few successful systems. The second phase was motivated by commercial (turn-key) two-dimensional drawing systems which were widely accepted by industries and, thus, successful from a point of view that they took over drawing boards. The third phase has introduced three-dimensional (wire frame, surface, or solid) modeling systems with catch-phrases like CAD/CAM (*Computer Aided Manufacturing*) integration, 3-D graphical presentation, CAD database, simulation, CAE (*Computer Aided Engineering*), etc.

Despite this popularity and commercial success, there are many deficiencies which have triggered research for a new generation CAD system. In this paper, we highlight problems and troubles with conventional systems and discuss the concepts for future systems.

Since designing activities and philosophy for CAD systems are completely dependent on the target area, those problems can be discussed only in a particular field. We take machine design as the subject area, because in many aspects a CAD system for machine design, being one of the most complicated software systems, has the strongest industrial impulses.

This paper contains three major chapters. In CHAPTER 2, we will review present CAD systems. There are already many research projects which focus on improving these. Firstly, we will briefly review the trends in significant work and make our *points of view* clear. Based on this discussion, we will formulate a new approach. This will also bring about the major differences between our study and others.

Report CS-R8717
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Next, in CHAPTER 3, we propose a new concept of CAD systems. The system is called **Intelligent Integrated Interactive CAD** (*IIICAD*; pronounced as *three-CAD*) system. Its requirements will be discussed and then its architecture will be proposed. Basic elements of *IIICAD* will be described. Finally, we will describe properties the system is expected to have, so that the reader can get an impression about the functions of the system.

In CHAPTER 4, we discuss how to develop *IIICAD*. It is anticipated that, because *IIICAD* is a large complicated software system, we need a theoretically founded- and systematic development method rather than *ad hoc* approaches. Moreover, designing is an intellectual process and knowledge engineering [1] is expected to play a crucial role in *IIICAD*. This demands theories capable of formalizing design and design knowledge. From this point of view, the relevance of some theories will be emphasized.

This report is the second of a series of CWI reports on *IIICAD*. Due to the space problem, the reader is invited to refer to the following relevant CWI reports concerning details in specifications of the *IIICAD* subsystems and theoretical achievement. At the time of writing, the following five reports are ready or being prepared. The first report discusses theory of design, although it did not clearly refer to the idea of *IIICAD* [40,44]. The third forthcoming report [47,48] will concentrate on derivation of the specifications of the kernel language of *IIICAD* called IDDL (*Integrated Data Description Language*), from theoretical aspects, while the fourth one [41,43] will discuss the specifications of SPV (*Supervisor*) in *IIICAD*. The fifth one will mainly deal with theory of knowledge [42] which will give theoretical basis to techniques used in *IIICAD*.

2. PROBLEMS OF CONVENTIONAL CAD SYSTEMS

In this chapter, first we briefly review the trends in significant work currently conducted. Secondly, our standpoint in developing *IIICAD* is firmly stated in comparison with other work. Finally, we clarify problems of conventional CAD systems, so that we can derive the specifications for future systems in the next chapter.

2.1. Trends in research

Recently the idea of intelligent CAD systems as an application of knowledge engineering is frequently discussed, because knowledge engineering seems to be a promising tool [13,15,34] for the following reasons.

(1) Knowledge engineering, by definition, aims at use of knowledge about design in a more direct way for more complicated applications.
(2) Knowledge engineering can make it possible to integrate systems at higher semantical levels. It offers methods to describe models in a flexible and powerful way.
(3) Techniques developed in artificial intelligence (e.g., various inference techniques) can improve man-machine communication.

This idea leads to concepts such as

- *intelligent CAD systems*,
- *knowledge based CAD systems*,
- *expert systems for CAD*,

etc. In our view, there seem to be three main streams in the research of this topic.

(a) The characteristics of the main streams are resulting from the recognitions, such as;

designing is a problem solving process.
designing is a decision making process.
designing is an optimization process.

Under these statements, we may implement an expert system which supports designers in the problem solving, decision making processes of designing, or optimization processes. [14,25,28].

The rest of designing processes requires intelligence and creativity, and hence will be difficult if not impossible to support. Thus, we implement an expert system which solves *overt* problems in a design process [2]. Especially in machine design, a design process can be sometimes regarded as a process to find the optimal solution under a number of constraints. From this point of view, knowledge engineering can also provide useful techniques [23].

(b) On the other hand, if we focus on various knowledge representation methods which may give a possibility to integrate descriptions, or at least, to allow flexible flow of information, we reach an idea of developing an integrated CAD framework based on knowledge engineering techniques [30, 32].

(c) Natural language processing has invented many useful theories and technologies to understand the designers' intent. They can be applied to realize flexible and intelligent user interfaces [26].

2.2. Subject of this study

In this section, we clarify our starting point and strategy for building the system as compared with current trends in research mentioned in the previous section. To do so, first we introduce some terminology.

2.2.1. Three aspects of design. There can be many different concepts and ideas implied in the term, *design*. For instance, designing a totally new product is very different from so-called parametric designing in terms of design methodology. Designing VLSI is completely different from architectural designing in terms of the design object. However, we can see characteristics common to any type of design despite all these differences; i.e., designing is a mapping process from design specifications into some concrete object [49]. This suggests that a design process utilizing CAD systems can be characterized by the following three aspects, abstracting or neglecting subtle differences in design objects and design methodology.

- **Design processes**: It is widely accepted that machine design can be categorized into the following four major subprocesses or stages (see FIGURE 1) which can be seen more or less in any other domain;
  (1) *conceptual designing* where the designer determines the basic principle and the basic mechanism of the design solution.
  (2) *basic designing* where the layout and the structure of the design solution is determined.
  (3) *detail designing* where the minute specifications of the parts are produced.
  (4) *production designing* where information needed for manufacturing is generated.
- **Design models**: During a design process, a design solution will be represented using various kinds of descriptions. A CAD system is to help designers building such descriptions. Those descriptions are called design models.
- **Design activities**: Designing contains various kinds of activities, such as *drawing sketches*, *writing technical documents*, *surveying information*, *meeting*, etc., regardless of types of design objects or design stages.

These three aspects should be considered whenever we discuss CAD systems. For example, if a system only supports a part of the design process, which is the case in the present situation, it is not very useful for designers. Thus, CAD systems must be able to support the entire design process with a unified design model in various descriptions and with powerful functions corresponding to different kinds of design activities. FIGURE 2 implies that a CAD system with different kinds of design object models, but derived from the same central model (for consistency), should be used in all the designing stages, supporting from *reasoning* to *information retrieval*. When we *criticize* the present situation of CAD systems, we consider these three aspects.

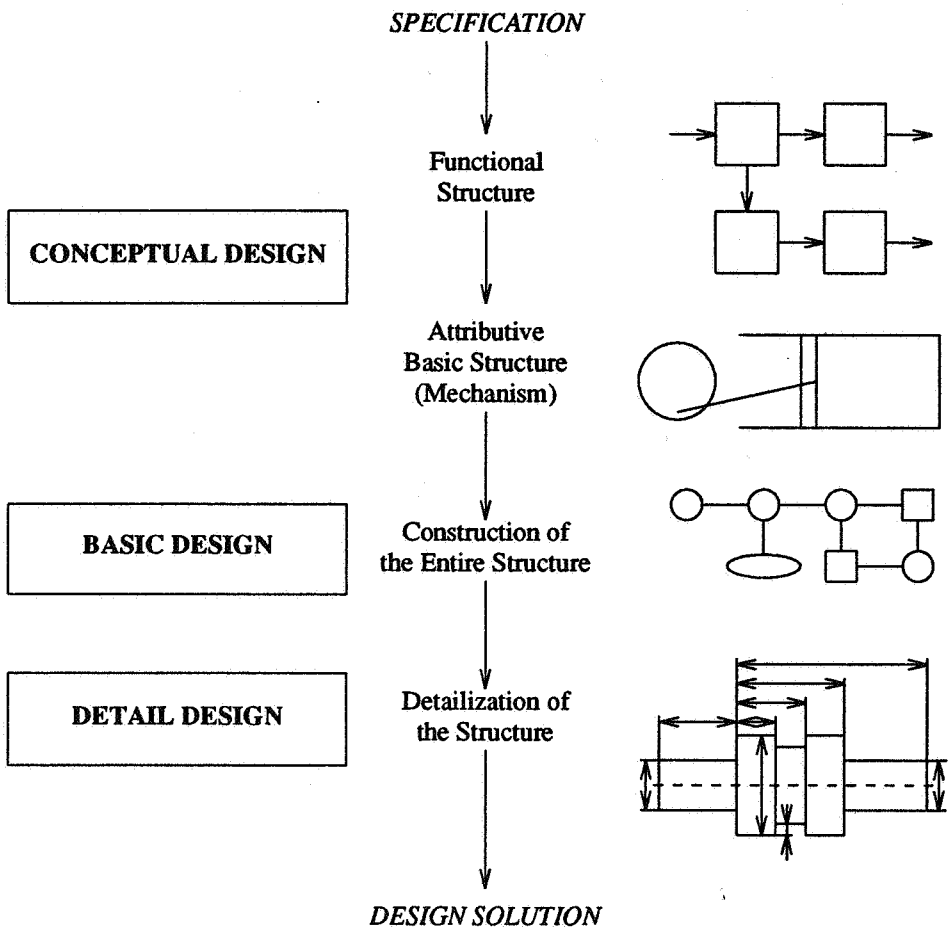


FIGURE 1. Design process

2.2.2. Semantics and syntax. Nowadays, the concepts of intelligent workstations, intelligent robots, etc., are widely discussed. When we say *intelligent*, it must at least differ from the usual *non-intelligent* system. However, in many cases, the term, intelligence, refers to distributed information processing (for example, *intelligent workstation*). Throughout this paper, we will use intelligence in a more appropriate sense. As the basis of discussion, we assume predicate logic.

- By an **intelligent system** we mean a computer system which can deal with the semantics of the application domain, because we cannot speak about *intelligence* without *semantics*.

Then, we define *syntax* and *semantics* as follows. We assume that a model of the real world domain (of the design object) can be symbolically represented using a set of symbols together with their relations. (This is an assumption on which most of our theories are based. See SECTION 4.3.1 and [44].)

- The **syntax** is mathematical relationship among the symbols in the domain.
- The **semantics** of the symbols is a function which identifies a relation between symbols as being part of that model of the real world domain.

Since we are given logics which deal with only syntactical relationships among symbols as the basis of discussion, semantics in this paper means a set of formalized functions under some sense of value. Thus, semantics is strictly related to the concepts of one who defined it and it may be abstracted or categorized from the result of having the syntax, whereas syntax simply gives a mathematical structure

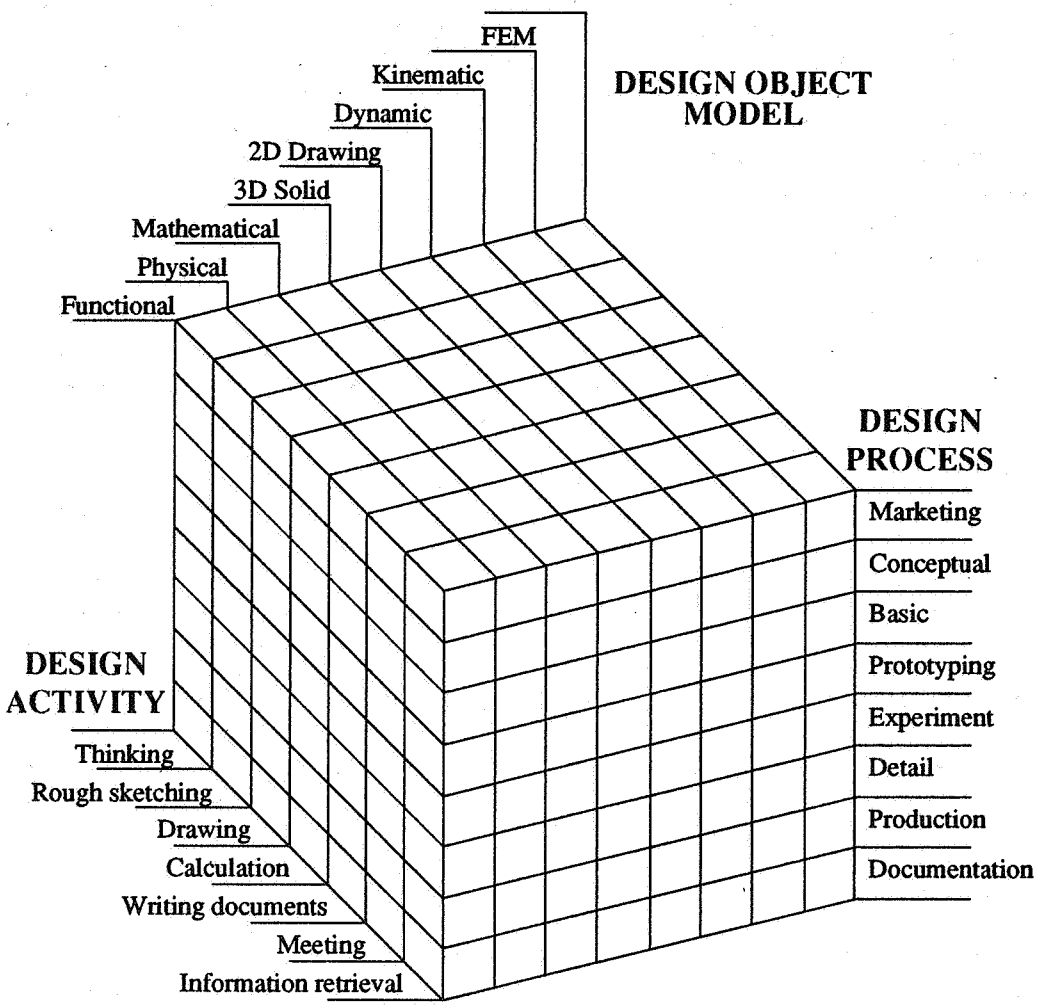


FIGURE 2. Three aspects of CAD

over the set of available symbols. By definition, semantics is a conditioned subset of syntax.

We now show an example of *semantics* and *syntax* in the context of a CAD system for mechanical engineering. Let a cylinder be known or defined precisely by mathematics. Therefore, we would know many attributes of a particular cylinder **cyl**, such as a set of equations describing its surfaces mathematically. Let **CYLINDER** be a generic name for cylinders, while **cyl** be an instance name.

Suppose *size* is giving a relation name. Then, a statement

$$size(\mathbf{cyl}) = 12.5l$$

is lexically normal and has correct syntax, whereas

$$size(\mathbf{cyl}) = -3.5l$$

has correct syntax but is judged false or unfeasible by the *semantics* of mechanical engineering;

$$size(\mathbf{CYLINDER}) \geqslant 0.$$

And, if the company produces only tiny equipment, we must judge something to be semantically wrong even with the first statement although it is not absolutely impossible. This example indicates that semantics is created from knowledge of that specific application domain.

2.2.3. Subject of this study. We have mentioned several approaches in SECTION 2.1 to build a future CAD system. In order to solve the problems of conventional CAD systems which will be pointed out in SECTION 2.3, none of them will, by itself be powerful enough. For instance, developing a sophisticated expert system may supply intelligence missing; but, this can neither solve the problem of non-integration nor of the poor man-machine interface. In fact, we need to develop a general CAD system framework providing a general solution rather than pursuing an expert system or a smart user interface management system.

As stated above we need to keep the three aspects of CAD in mind, when we develop a future system which can handle intelligence defined also in the previous discussion. This brings us to the following conclusion.

We aim at developing a general CAD system framework which should serve as a basis for integration. On top of it, we can place subsystems to provide intelligence and flexible communication. By intelligence we mean ability to operate designer's semantics as discussed previously. This aim will be justified by the discussion in the next section.

2.3. Analysis of the present situation

In order to clarify the problems of conventional CAD systems, here we review the present situation.

2.3.1. CAD supports design processes only partially without integration. TABLE 1 shows a comparison between developing software and manufacturing a machine. For software development we need an editor, for instance, and we can say an editor corresponds to a geometric modeler in machine design.

We realize that total system integration has not been achieved yet, although the need for integration was identified a long time ago and there have been substantial advances in this field [24]. The present trend in research is to integrate the system around models concerning products [22]. This implies that present CAD systems are product oriented, which further indicates that they are only good at displaying geometric information. TABLE 1 suggests that integration of CAD should mean that CAD systems are process oriented [46] (issue <1>).

If we look at the specification stage or predocumentation stage in software development in TABLE 1, we realize there is no similar tool in a CAD process. For better designing, it is also obvious that the so-called conceptual design stage should be supported by a certain CAD tool [38] which must at the same time contribute to the integration of CAD systems. Since the conceptual design stage is a highly intellectual process, this tool should be intelligent. We need an intelligent tool (issue <2>).

TABLE 1. Comparison between software development and CAD

SOFTWARE DEVELOPMENT	CAD PROCESS
Formal Specification	??
Comments, Documentation	??
Editor	Geometric Modeler
Syntax Checker	FEM
Debugger	Analysis System
Source List	*Drawing*
Compiler	NC Programming
Object Code	*NC Program*
Run-Time System	NC Machine Tool
Output Device	Robot
Output	*Product*

2.3.2. CAD supports few design activities. TABLE 2 shows how CAD systems support various design activities. Producing *final drawings* is fairly well supported by existing CAD tools. On the other hand, there is virtually no support for *sketching*. There are many systems that can accept rough sketch input such that the system recognizes inaccurate lines, circles, symbols, etc., and converts them into exact information; but there is no system which can handle inaccurate information as it is. (Some systems can handle inaccurate painting-like sketches literally as they are, though.)

Suppose we want to decide the position of a part. Since there are relationships with other parts which are to be fixed afterwards, for the moment we must leave it at specifying at best a certain area. It is difficult to express this kind of constraints in present systems (see also SECTION 2.2.9). CAD systems are not built to understand such ambiguous or vague information, hence the poor support in the conceptual design process (issue <2>).

TABLE 2. CAD support in various design activities

Activities		CAD Tools	Design Stage		
			Conceptual	Basic	Detail
Think	about Machine	?	+	++	+++
	about Function	?	+++	++	+
Meeting		?	+	+	+
Draw	Rough Sketch	?	+	--	--
	Drawing	*Drawing System*	--	+	++
Computation		*FEM*	--	++	+
Writing Text		*Word Processing*	+	+	+
Experiments		*Laboratory Automation*	++	+	--
Information Retrieval		*Database*	+	+	+

--: not necessary, +: necessary; the degree of necessity is in proportion to number of +.

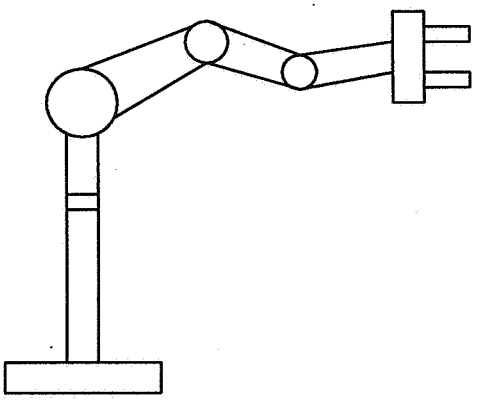
Even very advanced software systems do not always have adequate features for designing. For example, document processing systems are considered to be most advanced among current software systems. However, there is no system which allows designers to prepare complete technical documentation which must be written in *natural language* using information obtained from CAD systems and to send information described in natural language to CAD systems. This suggests that in order to support various kinds of design activities the system must have good variety of functions (issue <3>), although such functions as natural language processing are considered extremely difficult and expensive.

2.3.3. Models are not integrated. Integration of different models is significant in case of mechanical design which must deal with complicated structures. A design object must be viewed from many *points of view*. In the whole design process one object is represented in many models, such as a geometric model, a kinematic model, a dynamic model, etc., each of which is allowed to have different attributes. A typical example is shown in FIGURE 3.

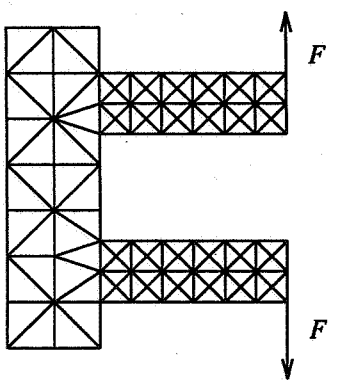
Product modelling was proposed [22] to integrate information for all CAD/CAM activities and it seems very promising. A product modeler is usually built on a geometric modeler. Technological information is, accordingly, attached to geometrical information. This approach, however, contains a strong limitation; for instance, if a piece of information is non-geometrical, it is not easy to place it in such a modeler.

Thus, integration of models should not mean the traditional approach of product modeling.

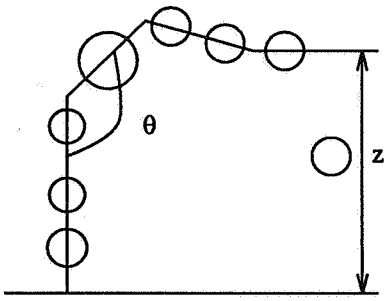
However, we have not yet obtained a unified model to guarantee the integrity and consistency of the information as an alternative of product modeling. We may call such a model *metamodel* [44] and it is a matter of great concern for researchers in this field (issue <4>).



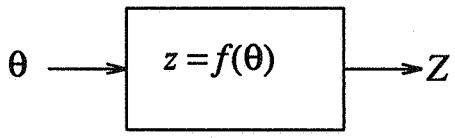
(a) 3D geometrical model



(c) FEM model



(b) Dynamic model



(d) Mathematical model

FIGURE 3. Examples of various possible models of a robot

2.3.4. Systems are not integrated. First, let us define several terms. In the *Open Systems Interconnection* (OSI) Basic Reference Model [20], there are seven layers for data exchange; i.e., physical, data link, network, transport, session, presentation, and application. We define here *physical* level to be the same as the physical layer in the OSI model. By *logical* we mean layers between data link and session in the OSI model. Additionally, in this paper, we consider *lexical*, *syntactical*, and *semantical* levels. Lexical level is equivalent to presentation level in the OSI model. On the other hand, *syntactical* and *semantical* are both included in the application layer of the OSI model. We distinguish these two, following the definitions in SECTION 2.2.2.

Local Area Network (LAN) made it easier to integrate systems at physical and logical levels, although there are standards only at very low levels. Standards such as IGES [29] and GKS [21] may contribute to the integration in distributed CAD systems based on LAN technology, since they can allow information exchange at syntactical and lower levels. However, since there is no semantical information exchange standard, higher level information exchange between different CAD systems have not yet achieved. This forces designers to use differing data schemata which cause unnecessary confusion and interface problems between different CAD systems. If a designer wants to use another subsystem which is running on even the same computer, she/he again has to generate and feed the data into the second system manually or to edit a part of the data from one system manually to make

it fit to the second system.

This problem is caused mainly by the lack of non-transferable data description methods. In other words, we need integration of *design knowledge* in both the syntactical and semantical levels. (issue <5>).

Just mechanical data exchange will cause deterioration of meaning. This is fatal for information processing, such as intelligent CAD systems, and can happen for two reasons.

The first reason is rather obvious. When you have a three-dimensional solid modeling system you cannot exchange data with other solid systems using, say, IGES, because IGES is based on two-dimensional drawings. The adequate alternative would be to build a central model which can be accessed by every CAD subsystem.

The second one is theoretical [39, 42]. Suppose we have a plane s in a three-dimensional space which can be defined by (minimum) three points, by a (normal) vector and a point, or by two vectors originating from the same point. Now, we have basically two ways of representing this fact.

(1) We consider that an entity can be represented by its properties. In this case, s will be represented by three points, p_1, p_2, and p_3, for instance. We call this representation method an *intensional* description, because an entity is represented by its intension (or properties). In a relational database system, it is expressed by a tuple

$$plane(p_1, p_2, p_3).$$

Note that s will not explicitly appear in this tuple.

(2) An entity can be explicitly represented by its relationships between it and other entities. We can use three facts; i.e., s has p_1, s has p_2, and s has p_3. We call this representation method an *extensional* description. In a relational database system, one possibility to express the case is to use three tuples $PLANE(s)$, $POINT(p)$, and $HAS(s, p)$, and the entire fact will be represented by

$$\forall s[PLANE(s)]\exists p_1, p_2, p_3[POINT(p_1), POINT(p_2), POINT(p_3),$$
$$p_1 \neq p_2, p_2 \neq p_3, p_3 \neq p_1,$$
$$HAS(s, p_1), HAS(s, p_2), HAS(s, p_3)].$$

Note that we can think of s regardless of p_1, p_2, or p_3; i.e., to define s, we do not necessarily need to know its intension.

Now, we can point out a problem. It is possible to define a plane s intensionally both by three mutually different points, p_1, p_2, and p_3, and by a normal vector v and a point p. Suppose we have a CAD system A which employs the former description, B which uses the latter description, and C which uses an extensional description method (see FIGURE 4). In system A, s will be described by

$$plane(p_1, p_2, p_3),$$

and in system B by

$$plane(p, v).$$

In system C, it will be described extensionally

$$PLANE(s), POINT(p), HAS(s, p), VECTOR(v), DEFINED\_BY(s, p_1, v),$$
$$v = (p_2 - p_1) \times (p_3 - p_1)$$

to make it possible to transfer data from system A to B via C. (It is impossible to do the other way around for the obvious reason pointed out in (1)). In this transformation we have completely lost information about p_2 and p_3. This means there will be a loss of information or at least inevitable confusion in data exchange shown in FIGURE 4, if there are two different intensional description methods and an extensional description method.

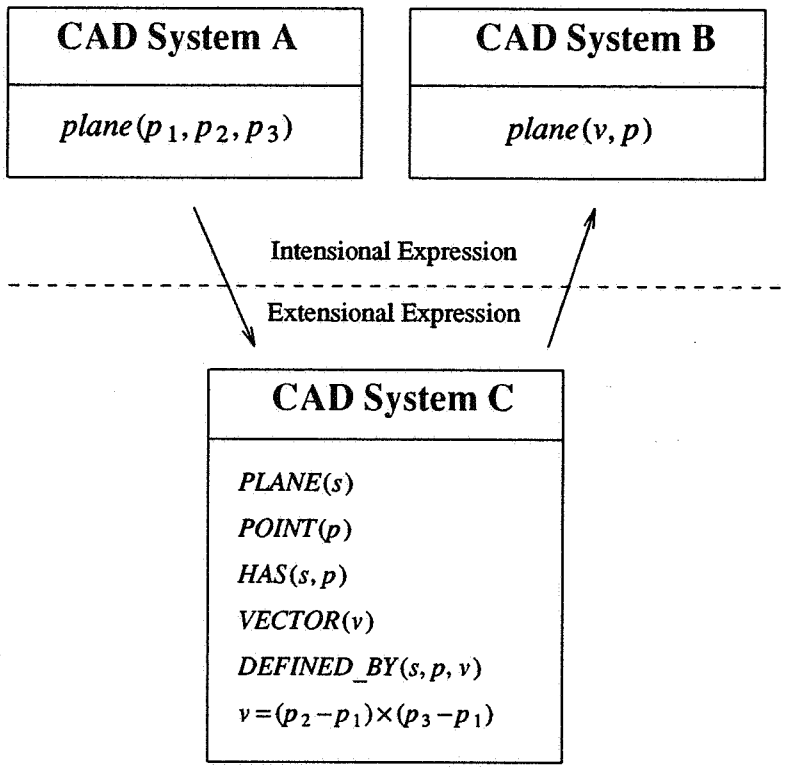


FIGURE 4. Data exchange between two CAD systems

This problem is caused by a mathematical and fundamental problem in conversion between intensional and extensional descriptions. Let us define $\downarrow x$ be an intension of x and $\uparrow x$ be its extension. Always,

$$\uparrow\downarrow x = x$$

holds, whereas

$$\downarrow\uparrow x = x$$

does not always hold. Suppose here a plane in an intensional expression using three points is x. From its extensional expression, we may create an intensional expression, using a vector and a point, which is totally different from the first one. Although these two are mathematically the same, information about points will be lost in this transformation.

2.3.5. No error check. Another big problem of conventional CAD systems is that systems do not check errors or mistakes of the designer. Sometimes, final outputs are so impressive that no one can detect errors.

A command interpreter is normally designed such that it only accepts *correct commands*. If the operator makes a syntactical mistake, the command interpreter can show the place and the reason of the error because the input does not fit to the system's vocabulary. Now the problem is that the command interpreter never suggests the correct command or the right command sequence. A few systems, like Smalltalk-80's user interface [17], can suggest possible alternatives when the user makes syntactical errors such as spelling mistakes. However, if they are semantical, there is no hope (issue <6>).

2.3.6. Data input is a problem. The amount of (geometrical) input data in CAD application is usually enormous. When an operator inputs raw data manually, errors, mistakes, and misunderstandings during man-machine communication are inevitable regardless of input devices. The ultimate solution is that the system accepts substantially reduced but comprehensive data instead of raw data. For instance, a CAD system should accept commands like

I would like to generate such and such object.
I have told you the minimum data necessary, so please do as you think fit.

In addition to this, the system should add the missing data from known information using logical reasoning based on its design knowledge. This implies that the system needs to support already the very beginning of the design process, e.g., the conceptual design stage, and it should be able to *generate* design information in cooperation with the designer (issue <7>).

2.3.7. Terminology of task domain cannot be understood. It often happens that we would like to input a command such as;

Here I want a hole in which to insert the shaft that was just created.

Saying this, we expect that the diameter of the hole is set equal to that of the shaft. Because a conventional system lacks common sense knowledge of machine design and because it cannot understand the meanings of the designer's action and intention, we are likely to be obliged to say;

Create a cylinder of diameter D.
Name it SHAFT.
Create a cylinder of diameter D.
Name it CYL.
Generate a cube with dimension, W, D, and H.
Name it PLATE.
Place PLATE at (x1, y1, z1).
Place SHAFT at (x2, y2, z2).
Place CYL at (x2, y2, z2).
Subtract CYL from PLATE.

This problem is not, however, solved by introduction of natural language techniques. What is required is to intellectualize the system so as to understand the common sense knowledge of the task domain, i.e., machine design (issue <8>).

2.3.8. Poor problem solving ability. Generally speaking, it is not quite common that a CAD system can answer questions from the designer. If the system is able to make an appropriate suggestion for designer's queries, design work will become efficient. Even simply enumerating the possibilities will be effective to avoid *sticking to one idea* and to increase the creativity of designers.

Therefore, we need to add more ability for consultation and problem solving (issue <9>).

2.3.9. Ambiguity and inconsistency are not allowed. In geometric modeling, the problem of separation of the topology and the geometry of objects has often been discussed. Most of geometric modeling systems have this separation, and quite often the dimensions can be also separated.

This issue would be generalized to the distinction between structure and value. Sometimes, we want to separate the structure of an object from the values of attributes, so that ambiguity and inconsistency are allowed and we can first decide the structure, especially, in the conceptual design stage. For instance, in rough sketch input where we do not want to specify an exact place of geometrical objects, it is much more important to recognize the existence of a point than to find its coordinates (issue <10>, see also SECTION 2.3.2).

Note that, on the contrary, quite often structural constraints influence values. Identical points are

not allowed for defining a line, for example.

2.3.10. Summary. We can categorize the aforementioned problems into three major factors as follows. From this, in the next chapter we can derive the specifications for a future CAD system. In the following list, numbers indicate the order of issues in this section.

- **Not integrated**
  - <1> CAD systems are not process oriented; CAD support in various design stages is not yet integrated.
  - <2> Early design stages are not supported; CAD support in various design stages is not yet integrated.
  - <3> Wider range of design activities should be supported; CAD support in various design activities is not yet integrated.
  - <4> Models of design objects are not integrated; model descriptions are not unified, thus not integrated.
  - <5> Systems are not integrated; this means design knowledge is not integrated and integration based on a unified model description method is necessary.
- **Not intelligent**
  - <2> Early design stages are not supported, because CAD systems are not intelligent enough to support such a stage.
  - <3> Wider range of design activities should be supported; CAD systems are not intelligent enough to provide designers with functions for such activities.
  - <6> Error detection and advisory ability are not yet sufficient.
  - <7> CAD systems should generate information in cooperation with the designer; CAD systems are not intelligent enough to do so.
  - <8> Task domain knowledge is not installed in the system.
  - <9> CAD systems have poor problem solving abilities and consultation abilities.
  - <10>Ambiguity and inconsistency in data are not allowed; CAD systems are not intelligent enough to to accept ambiguous input.
- **Poor man-machine interface**
  - <6> Error detection and advisory ability are not yet sufficient; CAD systems do not detect errors nor suggest the right way.
  - <7> CAD systems should generate information in cooperation with the designer; CAD systems are not yet able to reduce the amount of data to be put in by the designer.
  - <8> Task domain knowledge is not installed in the system; CAD systems do not understand the terminology of the designer.
  - <10>Ambiguity and inconsistency in data are not allowed; CAD systems require precise input data which may cause problems for operators.

3. Intelligent Integrated Interactive CAD

In this chapter, we propose the concept of *IIICAD* to address the problems pointed out in the previous chapter. First, we summarized requirements to *IIICAD* from discussion in Section 2.3. Then we propose general framework of *IIICAD* and its elements. Finally, how *IIICAD* should work is shown.

3.1. Requirements to future CAD systems

We have to consider a general CAD framework which includes features to solve the problems of conventional CAD systems pointed out in Section 2.3. The design requirements for such a system can be summarized as follows.

(1) The system must cover the three aspects of design; i.e, it should support designers in the *entire designing processes*, using *unified and integrated models*, with rich functions for various kinds of

design activities.

(2) The system should be **integrated**; this includes
- integration of *(sub)systems.*
- integration of *design models* based on an integrated *model description method*.
- integration of *design processes* which means computerization of even very early design stages and, as a result, integration of *design knowledge*.

(3) The system should be **intelligent**:
- The system should be able to support designers by solving design problems. (=*Intelligent problem solving*)
- The system should understand user's semantics; i.e., it should understand the task domain knowledge for more flexible man-machine communication and more powerful problem solving. This means it should be based on deeper models of machine but not so-called shallow knowledge [27]. By deeper knowledge we mean basic principle underlying the task domain. (=*Intelligent support of designers*)
- The system should understand the designer's intent even in an ambiguous context in order to provide more powerful abilities for consultation and error detection. (=*Intelligent interface*)

(4) The system should be **interactive**:
- The system should be interactive to increase the advantage of being intelligent and integrated.
- The system should directly interact on the level of meaning and intentions; not on a purely syntactical level.
- The system should provide the proper view on the design information, together with proper manipulation functions, thus exhibiting that it understands the purpose.

3.2. IIICAD architecture

3.2.1. Role of CAD in design. In order to discuss the architecture of *IIICAD*, we first clarify the role of CAD in a design process. By doing so, the *IIICAD* architecture will be justified.

An important issue even in near future is that we still use CAD (Computer *Aided* Design) rather than DA (*Design* Automation) for the following reasons.

(1) A system should never take the initiative of the design activity, because it is the designer who is responsible for the design.
(2) Assuming computers will become creative is, according to present computer technology, too optimistic.

Now, we can define one of the basic characteristics of CAD as follows. According to GENERAL DESIGN THEORY [44], we can regard a design process as a mapping from the function space onto the attribute space. This means, given specifications written in terms of functions, we find or create a new entity and we describe it in terms of attributes so that we can actually manufacture it. Therefore, we define a CAD system as a computer system which assists the designer in such a way that the designer can utilize her/his maximum ability of creating a new entity.

Consequently, a CAD system must have a **model** or descriptions about the design object which have maximum similarity to the designer's own image, and perform the best in computation and reasoning that can be expected by the present computer technology to answer questions about the model. At the same time, the system must have knowledge about design processes. This is achieved by explicit descriptions about design processes and a control mechanism to guide designers. This means a future CAD system must be controlled by an **intelligent supervisor**. As shown in FIGURE 5, these two components, i.e., the supervisor and the models, will play a crucial role in the architecture of *IIICAD*.

As far as design object models are concerned, there are two things noted.

(1) Currently, models in CAD mean exclusively geometric models or sometimes drawings.

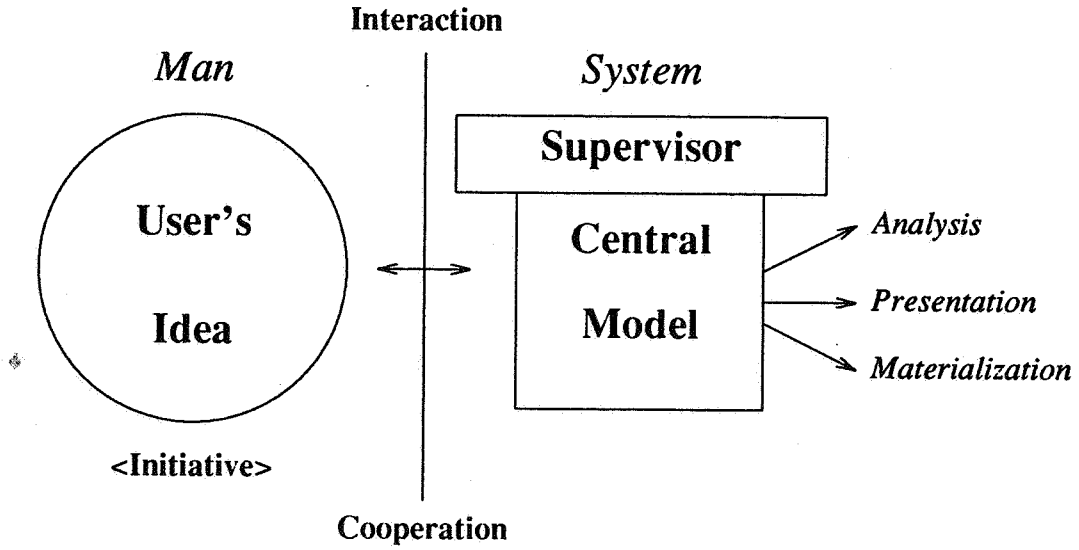


FIGURE 5. Basic CAD model

(2) Traditionally, engineers have been using drawings. However, it is obvious that drawings are different from models. Drawings are means of communication and describe a part of all the information. We need surprisingly an enormous amount of knowledge which is actually hidden to interpret drawings. Therefore, models should not mean geometric ones (and needless to say, never drawings); they should contain information telling designer's intentions, technological information, etc.

3.2.2. Basic elements of future CAD. Ideas described in FIGURE 5 can be further elaborated into the requirements following below which could be satisfied by the CAD system model shown in FIGURE 6.

(1) A mechanism for describing a central model of the design object called metamodel [44].
(2) A mechanism to derive models from the central model for the various technological evaluations.
(3) Tools to perform such evaluations.
(4) A mechanism to describe knowledge about design processes; i.e., knowledge on how to detail the model, how to derive models, how to perform evaluation, etc.
(5) A mechanism to communicate with the designer in such a way that she/he can recognize various aspects of the models.

Roughly speaking, for issues (1) and (2), we need a database and model schema (a). For (3), we need so-called design subsystems (b), such as FEM (*Finite Element analysis Method*) systems. We also need a large scale knowledge base (c) and an intelligent supervisor (d) for (4) and a good user interface system (e) for (5).

3.2.3. IIICAD architecture. The CAD system model in FIGURE 6 can be once again elaborated and extended to an architecture shown in FIGURE 7. The followings are the basic elements of *IIICAD*.

- **Supervisor** (SPV) corresponding to (d) in the previous section.
- **Integrated Data Description Schema** (IDDS) corresponding to (a) and (c) and **Integrated Data Description Language** (IDDL).
- **Intelligent User Interface** (IUI) corresponding to (e).
- **Application Program Interface** (API)
- **Applications**, such as geometric modeling systems, technological analysis systems, expert systems, etc., corresponding to (e).

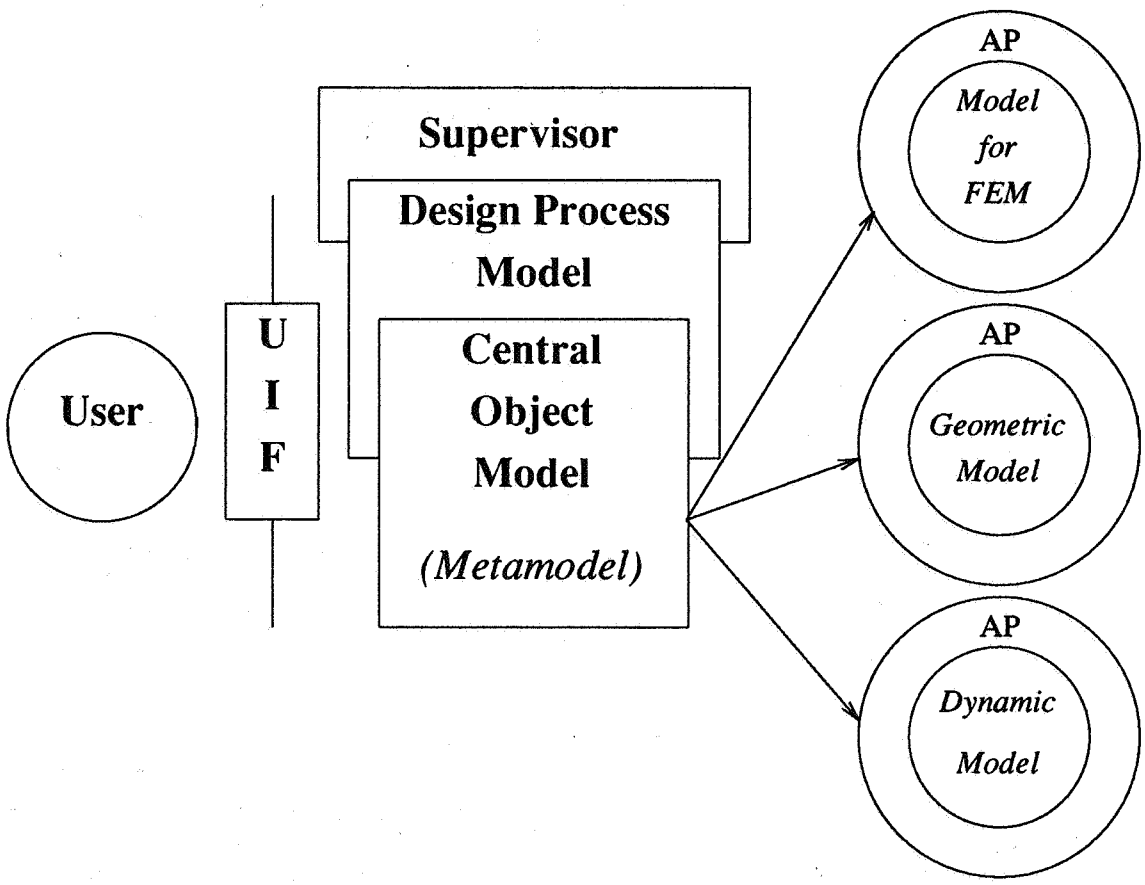


FIGURE 6. Elaborated CAD model

They will be discussed in the following sections in detail.

3.3. Elements of IIICAD

3.3.1. Supervisor. *SPV* is the kernel of the system [43]. It watches and controls all information flows in the system, such as user operations, status of the system, etc., and tries to understand the intention of the user.

The control is based on *scenarios* which describe standard designing procedures. A scenario is a set of rules which will be executed according to the situation SPV faces. When the user makes an obvious mistake, SPV should detect it by comparing the user actions with the scenario. In this way, SPV would add intelligence to the system. SPV does not have the initiative for the design process, because the final responsibility of design is left with the designer (see SECTION 3.2.).

Scenarios are, in other words, descriptions about the design knowledge. They describe knowledge about both design objects and design processes. They are written in a language called IDDL, which will be discussed later.

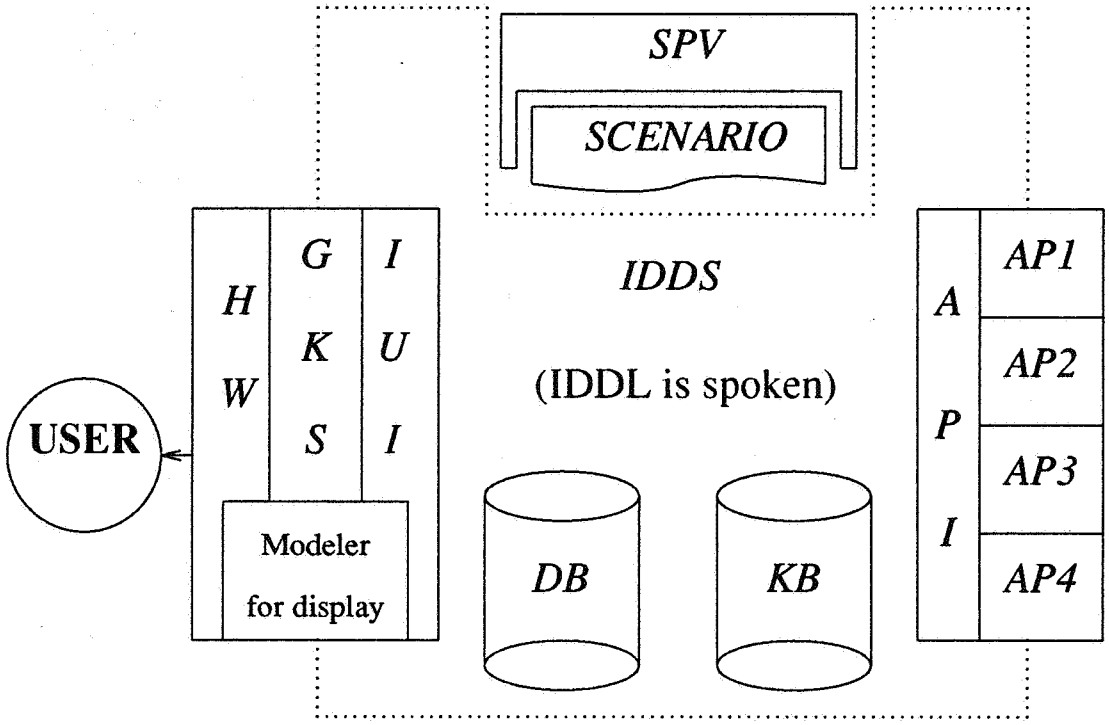


FIGURE 7. Configuration of the IIICAD system

3.3.2. Integrated data description schema. *IDDS* is a mechanism to store the information about the central model which would guarantee an integrated and unified description about the design object. At the same time, it stores knowledge about design process in a form of scenario. IDDS has a language called *IDDL* spoken by all the system elements [47, 48]

From an implementational point of view, IDDS is a gateway to *DB/KB* (*databases* and *knowledge bases*), which means the user does not have to pay attention to exactly where and how to store and retrieve information. All the information comes in and out through IDDS, and therefore, DB and KB are transparent to the user.

3.3.3. Integrated data description language. *IDDL* is a language to be spoken by all the system elements at the level of IDDS. Some important design issues of IDDL can be summarized as follows [47, 48].

(1) IDDL is based on predicate logic and uses the concepts of object to represent entities, predicate to represent relationships among entities, and function to represent attributes of entities.
(2) IDDL employs modal logic and three-valued logic.
(3) IDDL can represent system information and system control information.
(4) IDDL supports a mechanism to describe both procedural and heuristic nature of design processes.
(5) IDDL can support a multiworld mechanism for supporting nondeterministic features of design processes and for integrating models constructed from different points of view.

3.3.4. Intelligent user interface. The interface between *IIICAD* and the user has relatively complicated structure in order to realize flexible and intelligent interaction. There must be several interfacing systems reflecting the necessity to access information from very low level input/output to very high abstracted level.

The highest level interface system is called *IUI* which is driven also by scenarios written in IDDL and which accepts messages from SPV or other subsystems and sends them to the lower level interface systems, such as the DIALOGUE CELL system [5, 37, 45] and GKS, to control the physical device. It accepts a user's input from the DIALOGUE CELL system and translates semantical information of the input to descriptions in IDDL which in turn will be sent to SPV.

As described in SECTION 2.2.2, *IIICAD* has a distinction between syntax and semantics. Therefore, syntactical errors of the user can be processed by IUI, whereas semantical errors can be processed by SPV.

3.3.5. Application interface. API translates the central model descriptions about the design object stored in IDDS into an individual model used by an application program. It also passes data requests from an application program to SPV. This means API should have;

(1) descriptions about applications, such as their functions, what kind of model description is required, what data is returned from applications, etc.
(2) descriptions about the translation from the central model to the model used in an application program.
(3) descriptions about data requests from applications; i.e., how to request SPV to give a set of data to an application program.

Therefore, API is more or less similar to a run-time system monitor. It should also have scenarios written in IDDL to perform its tasks.

3.3.6. Applications systems. IIICAD should be able to have, at least, the following design subsystems in terms of mechanical engineering applications.

(1) Conceptual design system; it might be a knowledge based system to handle vague information in this stage [38].
(2) Consultation systems or problem solving systems for engineering applications [34].
(3) Basic/Detail design system; this can be also a knowledge based system [36].
(4) Geometric modeling system and its applications.
(5) Engineering analysis systems, such as FEM, etc.; they can be specific to the application.

Application systems are accessed only by API. SPV inquires API about a fact in forms of a clause of predicate logic. API translates this inquiry into an application program execution and returns the results again in the form of a clause. Therefore, API together with application systems behaves as an assertion mechanism for predicate logic [43]. It returns not only YES or NO to an inquiry but also a set of objects which instantiates the clause as in Prolog [11].

Among the subsystems mentioned above, the geometric modeling system will play a crucial role in mechanical engineering applications. It will be used in the heart of *IIICAD* to provide most important information. However, at the same time, this will raise a problem. Since usually a geometric modeling system is primarily used to produce pictures on the display, it may have a direct contact with IUI neither via SPV nor via IDDS.

This should be avoided, because the geometric modeling system must be used only for handling geometrical information. If a display of the design object is directly produced by the geometric modeling system, information displayed there only reflects geometrical information and what we want to display might be different. Its function to produce displays should be used separately. This means we need more than two geometric modeling systems in the *IIICAD* architecture, i.e., one for handling geometrical information and the other for display, although they can be the same system.

3.3.7. Syntax and semantics in IIICAD. As we have already pointed out in SECTION 2.2.2, it is indispensable for *IIICAD* to manipulate semantics of the target domain (e.g., mechanical engineering) to behave intelligent. SPV, knowing the intent of the user (designer) and standard scripts of design as scenarios, detects mistakes, gives suggestions and alternative solutions, etc. IUI, on the other hand, gets a request from SPV, knowing only,

- *what to get from the user*

and

- *what to give to the user,*

and it controls the dialogue using low level input/output systems. In this context, a *User's Conceptual Model* (UCM) [31] is badly needed for SPV.

Now, it is clear that SPV should manipulate the semantics, while IUI manipulates only the syntax (and lexical information). IUI should transform very low level information into one described in IDDL. Whether the information is semantically all right or not will be judged by SPV. This means the following.

(1) There must be information about how to translate low level input/output information into IDDL.
(2) SPV should know that only *semantically* valid information can be used. This requests semantical information validation whenever new information is reported to SPV (or entered to IDDS).

Thus, when IUI is given a request for an input of the designer from SPV, it controls the I/O system (graphics system), prompts, and accepts a user's response. The request can include, for instance, the descriptions about the type, expected range of the value, standard prompt and messages, and (local) echos of the input. The syntax itself is written in scenarios for IDDS.

There is a dictionary which describes conceptual correspondences or correlations of the terms, feasible values, sets of equations, and so on, in IDDS. These descriptions are the semantics and we can call it common sense of a specific domain. (This requests us to investigate such topics as natural language processing, language understanding, etc., by the way.) The semantics may vary from domain to domain and thus, inevitably, SPV will be implemented as a knowledge based system with common sense reasoning [18]. For instance, SPV is expected to have knowledge about basic characteristics of solid, when the scenario deals with metal.

IUI, on the other hand, understands much broader but shallow knowledge. It may know something about cylinders, rectangulars, etc., but not about shafts in mechanical design. This means, when SPV asks IUI to get a user input, SPV tells only syntactical information. Otherwise, we cannot implement IUI for general purpose.

SPV controls the split between syntactical transactions entirely taken care of by IUI and semantical transactions which involve often components such as IDDS or SPV itself. The split level may change dynamically.

3.4. IIICAD system behavior

3.4.1. IIICAD system cycle. The *IIICAD* system has a basic cycle as follows.

(1) Every system component, such as IUI, API, etc., reports its own status or requests to SPV. Typically, this report can be a user's request from IUI or a data request from an application system through API.
(2) SPV asks DB/KB whether there is a proper scenario for the situation created by status reports from the subsystems. This can be done also by an explicit user command.
(3) SPV executes the selected scenario. Execution is done in the following cycle.
    (3-1) Recognition of the situation, i.e., selection of the most matching rule. Again, typically, a

report can be a user's request from IUI or a data request from an application system through API.

(3-2) Execution of the selected rule. Execution of a rule results either in assertion of facts or in passing requests from one subsystem to the most proper one. The response from the inquired system will be returned to the system from which the request originated. If it was satisfied by the response, it reports satisfaction to SPV and SPV will proceed to the next step, i.e., go to (3-1). If not, SPV must execute exception handling rules which might or might not be described in the scenario.

(3-3) If no rule, stop with this scenario.

(4) If there is no more rules which can be applied or SPV encounters an explicit *end* command, it returns the control to the higher scenario and go to step (1).

In order to implement this cycle, one has to use meta-level knowledge, i.e., knowledge concerning how to use which scenario. In each scenario, there must be a description about when this particular scenario should be active.

3.4.2. Expected usage of IIICAD. The *IIICAD* cycle described in the previous section will make the following usage of the system possible.

(1) The designer would type in or select from a menu (or by any possible interaction method).

Design winch

(2) When IUI gets this command, it creates a request to SPV telling that the designer wants to design a winch.

(3) Receiving this request, SPV asks DB/KB to give it a scenario for designing a winch. If not found, something should happen to tell the designer that the system knows nothing about a winch. The most reasonable thing SPV could do is to ask the designer to select an example which is similar to a winch from the system library and to specify the difference between it and a winch.

(4) SPV begins the designing process of a winch based on the selected scenario;

What is the name of the winch? **WINCH**
How much load do you expect for WINCH? **100**
Do you mean 100kgf? **YES**
.....

(5) The dialogue described here took place in the following way. First, SPV needed to know the name of the winch and it sent a message to IUI to ask the designer. IUI, without knowing what in fact a winch semantically is, asked the question. Because what IUI needed was a name from the user for a winch and because there was knowledge written in the scenario for IUI that a name is given by a character string, IUI invoked a graphics system to put a prompt and to fetch a designer's input probably from the keyboard. IUI got an input from the user and sent it directly to SPV.

(6) If IUI is requested by SPV to get an integer value between 0 and 8, for example, and if the designer types a letter key ("s", for instance), this is her/his obvious error, a mistake, or a misunderstanding. In this case, an error recovery at some level is done by IUI, for example, by showing possible input to her/him. However, it must be also reported to SPV telling that she/he made an error and this is the fifth of this type and the twenty-eighth in total. This information should be used firstly by IUI to control the dialogue dynamically and secondly by SPV to judge whether the designer really understands the design process of a winch. For instance, if the designer gave an unexpected answer to a question asking the usage of a winch, it should be interpreted either as a misunderstanding or as trying to design some new which the system does not know, but not as an error because she/he is a beginner. This is an example of

SPV's understanding the intent of the designer.

(7) During the design process, the designer always takes the initiative. Thus, basically she/he must always enter commands to indicate what to do next. If SPV finds a difference between the scenario and the designer's action, i.e., if SPV cannot find a rule matching to the input, it would be;
   (a) an error of the designer,
   (b) the designer does something new, or
   (c) the scenario is wrong.

   In any case, SPV must warn the designer. If the designer admitted an error or if he asked a question, SPV must;
   (A) give help,
   (B) give suggestions, and
   (C) ask the designer to specify more precisely his intent.

3.5. Summary

Here we check how *IIICAD* can provide abilities to solve the problems of conventional CAD systems pointed out in SECTION 2.3.

(1) **Intelligent**: SPV and IUI will add intelligence to the system in the aspect of interaction. SPV will understand the designer's intent by selecting the most appropriate rule to the situation. Problem solving ability and consultation facilities will be given by knowledge based systems built on top of IDDS which feeds unified knowledge about the task domain to all those systems. A system for conceptual design, for instance, will be included in those knowledge based systems.

(2) **Integrated**: All the (sub)systems will be integrated on top of IDDS through API. Models (of design objects) will be uniformly described by IDDL, which helps integration of models. This is expected to result in integration of knowledge.

(3) **Interactive**: An intelligent interactive user interface will be realized by SPV and IUI. It should be possible to detect errors, make suggestions, handle incomplete input data, etc.

IIICAD is only a framework to realize such functions. This means we need to develop intelligent, integrated, and interactive domain specific design subsystems, and we need to implement task domain knowledge, i.e., design knowledge for mechanical engineering, in them. Note that *IIICAD* only makes it easy and implementing *IIICAD* itself cannot bring us to the goal.

4. METHODS OF DEVELOPING IIICAD

How to develop *IIICAD* is discussed in this chapter. We think a theoretical basis should be firmly established, because problems we found out in CHAPTER 2 are theoretical rather than implementational. We will, therefore, clarify theoretical problems in developing *IIICAD* and then mention implementational problems.

4.1. Three players in the development of IIICAD

Before we discuss the development of *IIICAD*, we make our position clear. As stated in CHAPTER 3, *IIICAD* is a general CAD framework, although design subsystems should be provided to demonstrate the feasibility. This means we will concentrate on system development and then this general *IIICAD* framework will be passed on to a system designer of the task domain who will implement domain dependent subsystems and knowledge for those systems. After this, the system can be used by end users who are actually designers in the task domain.

4.2. Theory and technology

As the history of science shows, it is obvious that *ad hoc* approaches can only serve very limited problem areas but not problems in general. So far, the importance of studying theories has been less emphasized than that of implementation technology. We do not think that all the problems of conventional CAD systems can be solved only by implementational techniques.

Because more or less designing is an intellectual process of humans which may require a deep investigation, we must aim at building general and robust theories besides implementation technology; i.e., we need

- **theory of CAD**

and

- **implementation technology.**

Theory of CAD will be discussed in detail in the next section. As implementation technology, following items seem relevant.

- **Knowledge engineering**: Since *IIICAD* is supposed to rely heavily on semantical information processing, knowledge engineering is expected to be the key technology.
- **Computer graphics**: In an application like mechanical CAD, geometrical information is indispensable and computer graphics (among other things, geometric modeling techniques) will be playing a crucial role.
- **Geometric modeling**
- **User interface management**: For an interactive system like *IIICAD*, so-called user interface management is important.
- **Database technology**: Together with knowledge engineering techniques, advanced database techniques are necessary to manage huge amount of design knowledge.
- **Software engineering**: *IIICAD* will be a big software system. In order to develop efficiently such a big system, we need sophisticated techniques from software engineering, such as rapid prototyping, requirement engineering, etc. Furthermore, since the maintenance of knowledge bases is becoming a big problem [4], this issue is very important also in *IIICAD*. The methodology for incremental and orthogonal development of knowledge bases is crucial [47].

These items are equally important; but, studies on knowledge engineering can be most emphasized among other things. (For more general discussions, see [7].)

4.3. Theory of CAD

As discussed in SECTION 2.2.1, there are three aspects of design, i.e., processes, models, and activities. This implies we need theories corresponding to them. A **theory of CAD** must indispensably be constructed from the following three elements.

(1) A theory which describes the design process and design activity itself: **Theory of design.**
(2) A theory which tells how to describe design objects (or models): **Theory of design objects.** In our case, this can be **theory of machines.**
(3) Moreover, we need an abstract or meta level theory about how to describe our knowledge about design: **Theory of knowledge.** While theory of design and theory of design objects concentrate on design at a concrete level, theory of knowledge deals with more abstract discussions.

Note that we must be able to build CAD systems for VLSI design, for example, by replacing theory of machines with theory of VLSI. This implies both theory of design and theory of knowledge must be general and independent from the design task domain.

4.3.1. Theory of design. Theory of design is necessary;

(1) to clarify what design is,
(2) to formalize design processes,
(3) to formalize design knowledge.

We often realize that, if we do not know what design is, we are obliged to use *ad hoc* approaches to design a CAD system which might be powerful in a particular field but not in general. As pointed out in CHAPTER 2 and 3, *IIICAD* should support designers in wider range of design processes in an integrated way. This forces us to describe a design process, preferably, in a mathematical way, but we also realize we have no such method.

At the same time, when we try to apply techniques invented in knowledge engineering so far to CAD systems, we do not have a guide to formalize design knowledge so that those techniques can be used. This results in hopeless trial-and-error research to find out the most suitable knowledge representation technique. Therefore, the formalization of design processes and design knowledge (particularly, by mathematical means, such as logic) is extremely crucial.

We have developed GENERAL DESIGN THEORY [44] and its most recent result is the mathematical formulation of the design process and the justification of knowledge representation techniques in a certain situation. In GENERAL DESIGN THEORY, a design process is regarded as a mapping from the function space onto the attribute space both of which are defined on an entity set. From this formalization based on axiomatic set theory, we can mathematically derive interesting theorems which can well explain a design process as follows.

(1) A design process is an evolution process about a metamodel. A metamodel is a set of attributive descriptions of an entity. Evolution means that during designing those attributive descriptions will be added and finally the metamodel will converge to the design solution (FIGURE 8).
(2) Although it is widely accepted that an attribute always has a value especially in knowledge engineering (e.g., MYCIN [33], and OPS5 [8]), it can be proven that attributes do not have always values by GENERAL DESIGN THEORY. This implies that the concept of an entity having an attribute should be distinguished from the concept of an attribute having a value. In terms of geometric modeling, this can be translated into the need to separate geometry and topology (see SECTION 2.3.9). This further suggests there is a possibility to represent concepts such as entities, properties, and relationships between entities in a new knowledge representation scheme (see SECTION 3.3.3 and 4.3.3).
(3) There is a possibility of representing the design process in a general and mathematical way, provided a mathematical representation method for a design process is found as suggested in (1). Unfortunately, this is not yet achieved [47]. However, this may result in innovative techniques, such as a new modeling technology beyond (and including) geometric modeling, representation of designing procedures in a mathematical way, etc. These two mean we may have an integration method for various kinds of CAD systems at a very high semantical level.

4.3.2. Theory of design objects. Secondly, we need a theory of design objects, in our case, theory of machines;

- because we need deeper knowledge of the task domain.
- because we do not know how to describe basic concepts, such as machine, function, attribute, etc.

Consider developing an expert system which can perform a conceptual design of machines. The system must understand how basic functions are realized, e.g., by what kind of mechanism. This means the system requires deeper knowledge about mechanisms constructed through very fundamental understanding of physical phenomena. In other words, semantics of machine design should be explicitly represented.

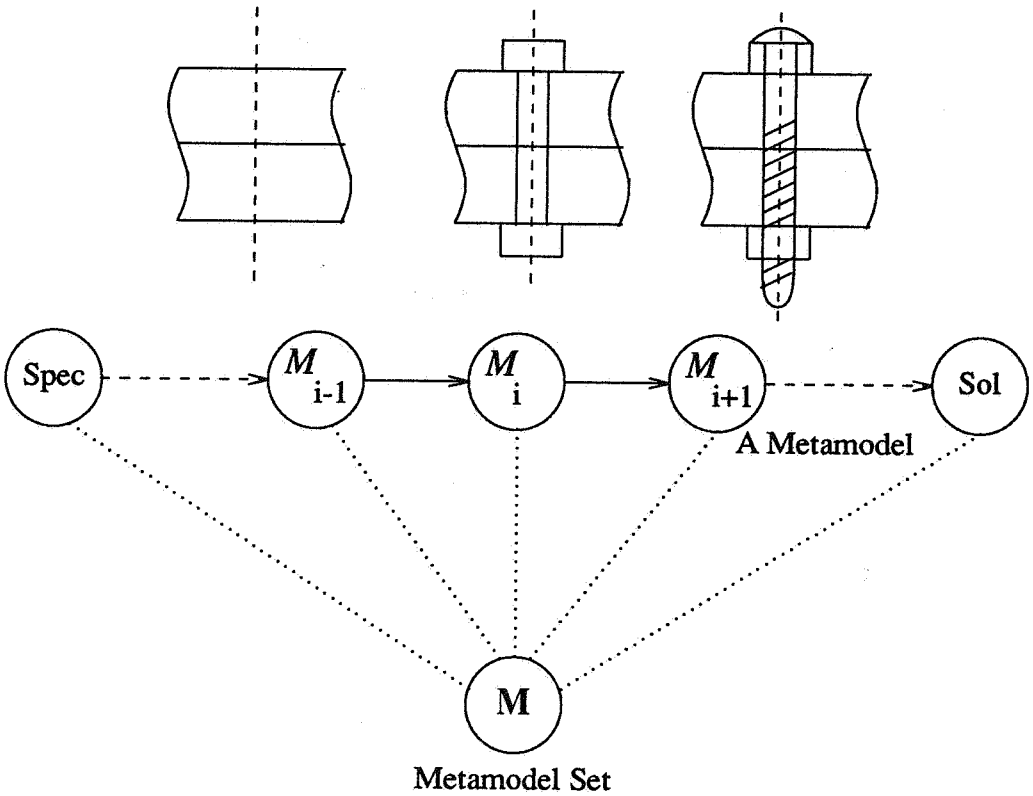


FIGURE 8. Evolution model of a design process

At the same time, we may realize that we simply do not know how to describe basic concepts, such as machine, function, attribute, etc. To solve this problem, we need to establish a theory of machine which can consist of;

- underlying principles such as qualitative physics [3],
- theory of mechanical systems as well as so-called theory of methodological design (e.g., [19]).

The task domain knowledge, i.e., knowledge about machines and machine design can be summarized by the latter. There are many proposals resulting from generalization of design experience. Further work is necessary to computerize those theories.

4.3.3. Theory of knowledge. Because *IIICAD* must be implemented as a tool for an intellectual process, we need theories that are more directly related to the human thinking process. Perhaps, we need epistemological theories about our perception from philosophy, cognitive science, psychology, etc., which may contribute to formalize our knowledge about design. Traditionally in the database technology, this aspect has been called *conceptual modeling* [6].

FIGURE 9 shows how a database or knowledge base system is constructed. This is a well known diagram both in knowledge engineering and in database technology. When we want to encode a piece of knowledge in a computer, we need to follow this diagram.

(1) First, we must build the most appropriate representation schema to the application. In terms of database technology, this process is called conceptual modeling. This means we outline the target knowledge using available templates, such as Entity-Relationship model [10], relational datamodel [12], etc. In case of knowledge engineering, we may have a choice from production rule paradigm, frame paradigm, predicate logic representation paradigm, semantic network

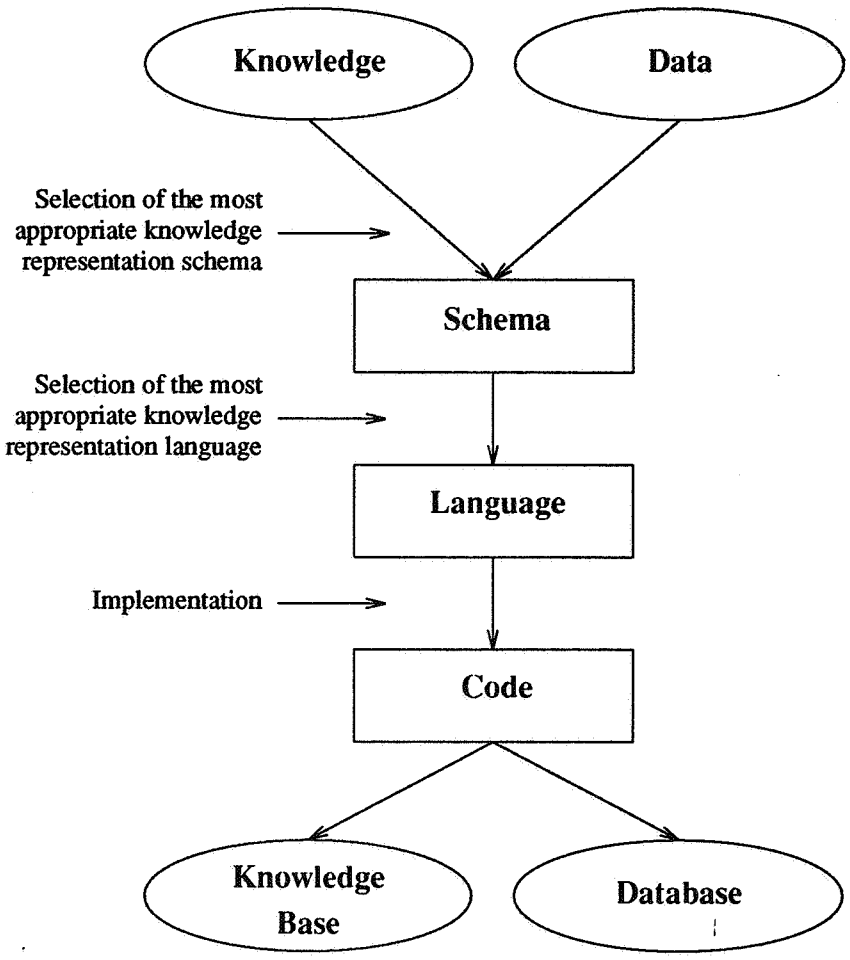


FIGURE 9. Implementation of data/knowledge

paradigm, etc., and then we must reorganize the target knowledge so that we can pick up the most suitable representation language in the next step.

(2) Second, we must select a representation language. If you have selected a production rule paradigm for example, you have to select the most appropriate language from a number of available languages or shells. In case of database systems, we may have relational database language, CODASYL type database language [9], etc.

(3) Finally, we start coding of knowledge. The actual coding takes place in three phases again. Firstly, we organize the knowledge based on the selected representation schema. Secondly, this templated knowledge will be translated into the selected language. Thirdly, this will be further translated into actual codes so that computer systems can understand it.

To decide which modeling method is the best in the step (1) and which language is the most suitable in the step (2), we need to know the nature of data and knowledge in CAD, how they are applied, etc. This discussion will be supported by theory of knowledge as a guiding principle;

(1) to generate the representation schema,
(2) to organize the knowledge,
(3) to select the representation language,
(4) to utilize the knowledge in a certain language.

In this context, theory of knowledge particularly resides on a more abstract and metaphysical level than conceptual modeling. It contributes to designing a CAD knowledge representation framework together with theory of design and theory of machines which tell about matters specific to machine design.

In the history of artificial intelligence, there were many remarkable and interesting debates on fundamental issues [1], such as *procedural vs. declarative*, *(first order) logic vs. special knowledge representation techniques* (frame, semantic network, etc.), and so on. The second debate is still going on, and there is no final conclusion. For instance, advocates for logic claim that anything can be represented by (formal) logic, whereas defenders for a particular knowledge representation technique insist that there exist some concepts or *meaning* which cannot be captured by logic. This is a problem in fact, because we cannot be sure if our knowledge representation method is suitable for CAD applications.

In our opinion, however, there are two issues which should not be dropped in this series of debates. Firstly, those who say logic cannot capture *meaning* have forgotten that logic is an entirely syntactical system. The meaning of what logic denotes should be understood in the context of a model world. Secondly, those who attack poor performance of one system have forgotten its advantages which can be obtained only by sacrificing performance. At the same time, although in principle logic can denote what other systems can represent, it never means that comprehensiveness or easiness to implement of the system (such as readability) is equivalent to that of special representation techniques.

These issues will be included also in theory of knowledge. An example of theory of knowledge was described in SECTION 2.3.4, where we introduced extensional and intensional description methods. Further study suggests that [42], for instance, for CAD applications extensional descriptions are more proper because of its dynamic changeability, whereas intensional descriptions can provide simpler (and faster) retrieving methods. This results in an idea of combining both extensional and intensional descriptions, on which IDDL is based, by introducing objects, predicates, and functions (see SECTION 3.3.3). This idea fits well to the object oriented programming paradigm (as in SMALLTALK-80 [16]) and can be implemented in such a language.

4.4. About knowledge engineering

As an implementation technique, knowledge engineering is supposed to play a crucial role in developing *IIICAD*, primarily because all the elements are supposed to behave very differently from a so-called algorithmic and deterministic way. However, knowledge engineering is not a magic box. It should be understood in the context of *formalized deductive symbolic manipulation*.

Let us define three concepts.

- A problem is *described* when there is its description in any language.
- A problem is *formalized* when there is its description in any formal language, such as mathematics, computer languages, etc.
- A problem is virtually *solved* when there is a method (or algorithm) to solve it.

Note that if a problem is formalized, it is also described. If a problem is solved, it is also formalized and, thus, described.

Since knowledge engineering is (still) based on computers which allow formalized deductive symbolic manipulation, there is a certain limit to what it can handle. Normally, techniques of knowledge engineering involve at most;

- pattern matching.
- heuristics by non-deterministic methods.
- deductive algorithm for non-deductive reasoning, e.g., simulation of induction or abduction.

Thus, in our view:

> *Knowledge engineering is an implementation technique for a class of problems which can be described and formalized but cannot be solved in an algorithmic way, or for those which cannot be*

formalized well in conventional programming languages.

For instance, if the problem can be both described, formalized, and solved, writing programs in FORTRAN, C, or Pascal is the best. If the problem is not described, naturally it is impossible to solve it (even with knowledge engineering).

Therefore, in developing *IIICAD*, we need to guarantee that

(1) all the problems are at least formalized and can be solved in some way (not necessarily algorithmically).
(2) IDDL or any other subsystems provide with such abilities that are demanded to solve those problems (not necessarily algorithmically).

5. Conclusion

In this paper the concept of *IIICAD* was proposed. The main issues can be summarized as follows.

(1) Problems of conventional CAD systems were analyzed originating from various, not only technical but also theoretical, treatments. It is concluded that conventional systems are not intelligent, they are not integrated, and they have poor man-machine communication.
(2) We build a general framework of a future CAD system. Comparing with other work, we put emphasis on fundamental and theoretical work besides implementational issues such as developing a design expert system.
(3) The concept of *IIICAD* was proposed as a general framework for a future system to solve those problems; i.e., future CAD systems must be intelligent, integrated, and interactive.
(4) The architecture of *IIICAD* was defined and its subsystems were clarified. The framework of *IIICAD* will be constructed having supervisor, integrated data description schema which has a language called integrated data description language, intelligent user interface, and application program interface. However, to achieve intelligence required to *IIICAD*, we need to implement also intelligent design subsystems on top of the framework.
(5) In order to develop *IIICAD*, we need both theory of CAD and implementation techniques. Theory of CAD may consist of three theories, i.e., theory of design, theory of design object, and theory of knowledge. Some results of those theories were mentioned.

Reference

[1] A. Barr and E. A. Feigenbaum (eds.), *The Handbook of Artificial Intelligence I, II, III*, William-Kaufmann, Los Altos, California, (1981).

[2] Bijl, A., "An Approach to Design Theory," in *Design Theory for CAD, Proceedings of the IFIP W.G. 5.2 Working Conference 1985 (Tokyo)*, H. Yoshikawa and E. A. Warman (eds.), North-Holland, Amsterdam, (1987), pp. 3.

[3] D. G. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, North-Holland, Amsterdam, (1984).

[4] Bobrow, D. G., S. Mittal and M. J. Stefik, "Expert Systems: Perils and Promise," *Communications of ACM*, **29**(9), September 1986, pp. 880.

[5] Borufka, H. G., H. W. Kuhlmann and P. J. W. ten Hagen, "Dialogue Cells: A Method for Defining Interactions," *IEEE Computer Graphics and Applications*, **2**(7), 1982, pp. 25-33.

[6] M. L. Brodie, J. Mylopoulos and J. W. Schmidt (eds.), *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer, New York, Berlin, Heidelberg, Tokyo, (1984).

[7] M. L. Brodie and J. Mylopoulos (eds.), *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Springer, New York, Berlin, Heidelberg, Tokyo, (1986).

[8] Brownston, L., R. Farrell, E. Kant and N. Martin, *Programming Expert Systems in OPS5*, Addison-Wesley, (1985).

[9] CODASYL DATA DESCRIPTION LANGUAGE COMMITTEE, Journal of Development, (January 1978).

[10] CHEN, P. P., "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transactions on Database Systems*, 1(1), March 1976, pp. 9-36.

[11] CLOCKSIN, W. F. and C. S. MELLISH, *Programming in Prolog*, Springer, Berlin, Heidelberg, New York, (1981).

[12] CODD, E. F., "A Relational Model of Data for Large Shared Data Banks," *Communications of ACM*, 13(6), 1970, pp. 377.

[13] J. S. GERO (ed.), *Knowledge Engineering in Computer-Aided Design, Proceedings of IFIP WG5.2 Working Conference in 1984 (Budapest)*, North-Holland, Amsterdam, (1985).

[14] J. S. GERO (ed.), *Optimization in Computer-Aided Design, Proceedings of IFIP WG5.2 Working Conference in 1983*, North-Holland, Amsterdam, (1985).

[15] J. S. GERO (ed.), *Expert Systems in Computer-Aided Design, Proceedings of IFIP WG5.2 Working Conference in 1987 (Sydney)*, to be published from North-Holland, Amsterdam, (1987).

[16] GOLDBERG, A. and D. ROBSON, *Smalltalk-80: The Language and its Implementation*, Addison Wesley, New York, (1983).

[17] GOLDBERG, A., *Smalltalk-80: The Interactive Programming Environment*, Addison Wesley, New York, (1984).

[18] HAYES, P. J., "The Second Naive Physics Manifesto," in *Formal Theories of the Common Sense World*, J. R. HOBBS and R. C. MOORE (eds.), Ablex Publishing, Co., Norwood, NJ, USA, (1985).

[19] HUBKA, V., *Theorie der Konstruktionsprozesse*, Springer, Berlin, Heidelberg, New York, (1977).

[20] ISO, "Open Systems Interconnection, Basic Reference Model," ISO International Standard 7498, (May 1983).

[21] ISO, "Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description," ISO International Standard 7942, (August 1985).

[22] KIMURA, F., T. SATA and M. HOSAKA, "Integration of Design and Manufacturing Activities Based on Object Modelling," in *Advances in CAD/CAM*, T. M. R. ELLIS and O. I. SEMENKOV (eds.), North-Holland, Amsterdam, (1983), pp. 375.

[23] KIMURA, F., H. SUZUKI and L. WINGARD, "A Uniform Approach to Dimensioning and Tolerancing in Product Modelling," in *Proceedings of CAPE '86, Second International Conference on Computer Applications in Production and Engineering*, K. BO, L. ESTENSEN and E. A. WARMAN (eds.), Copenhagen, (1986), pp. 165.

[24] D. KOCHAN (ed.), *Integration of CAD/CAM: Proceedings of IFIP WG5.3 Working Conference in 1983*, North-Holland, Amsterdam, (1984).

[25] LATOMBE, J., "Artificial Intelligence in Computer Aided Design: The TROPIC System," in *CAD Systems: Proceedings of IFIP Working Conference in 1976 (Austin, Texas, USA)*, J. J. ALLAN, III (ed.), North-Holland, Amsterdam, (1977), pp. 61.

[26] S. G. LEAHEY (ed.), *Proceedings of COMPINT 85 (Computer Aided Technologies) (Montreal)*, IEEE Computer Society Press, Montreal, Quebec, Canada, (September 1985).

[27] MACCALLUM, K. J., "Knowledge-Based Systems for CAD," *Proceedings of CAPE '86, Second International Conference on Computer Applications in Production and Engineering*, Copenhagen, (1986), pp. 903.

[28] MISTREE, F., H. KARANDIKAR and S. KAMAL, "Knowledge-Based Mathematical Programming: A Hybrid Approach for Decision Making in Design," in *Design Theory for CAD, Proceedings of the IFIP W.G. 5.2 Working Conference 1985 (Tokyo)*, H. YOSHIKAWA and E. A. WARMAN (eds.), North-Holland, Amsterdam, (1987), pp. 201.

[29] NATIONAL BUREAU OF STANDARD, "Initial Graphics Exchange Specifications (IGES), Version 2.0.," NBSIR-82-2631, (1983).

[30] OHSUGA, S., "Conceptual Design of CAD Systems Involving Knowledge Bases," in *Knowledge Engineering in Computer-Aided Design, Proceedings of IFIP WG5.2 Working Conference in 1984 (Budapest)*, J. S. GERO (ed.), North-Holland, Amsterdam, (1985), pp. 29.

[31] G. E. PFAFF (ed.), *User Interface Management Systems*, Springer, New York, Berlin, Heidelberg, Tokyo, (1985).

[32] REHAK, D. R., H. C. HOWARD and D. SRIRAM, "Architecture of an Integrated Knowledge Based Environment for Structural Engineering Applications," in *Knowledge Engineering in Computer-Aided Design, Proceedings of IFIP WG5.2 Working Conference in 1984 (Budapest)*, J. S. GERO (ed.), North-Holland, Amsterdam, (1985), pp. 89.

[33] SHORTLIFFE, E. H., *Computer-based Medical Consultations; MYCIN*, American Elsevier, New York, (1976).

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

[35] SUTHERLAND, I. E., "SKETCHPAD - A Man-Machine Graphical Communication System," *Proceedings of Spring Joint Computer Conference*, (1963), pp. 329.

[36] TAKESHIGE, A., T. TOMIYAMA and H. YOSHIKAWA, "An Application of Frame System to CAD," in *WDK 12, Proceedings of ICED 85 (Hamburg) - Theory and Practice of Engineering Design in International Comparison*, V. HUBKA and PROGRAMME COMMITTEE (eds.), Heurista, Zurich, (1985), pp. 763-770.

[37] TEN HAGEN, P. J. W. and J. DERKSEN, "Parallel input and feedback in dialogue cells," CWI Report No. CS-R8413, Centre for Mathematics and Computer Science, Amsterdam, (July 1984).

[38] TOMIYAMA, T. and H. YOSHIKAWA, "Requirements and Principles for Intelligent CAD Systems," in *Knowledge Engineering in Computer-Aided Design, Proceedings of IFIP W.G. 5.2 Working Conference 1984 (Budapest)*, J. S. GERO (ed.), North-Holland, Amsterdam, (1985), pp. 1-23.

[39] TOMIYAMA, T. and H. YOSHIKAWA, "Knowledge Engineering and CAD," *FGCS*, 1(4), June 1985, pp. 237-243, North-Holland.

[40] TOMIYAMA, T. and H. YOSHIKAWA, "Extended General Design Theory," CWI Report No. CS-R8604, Centrum voor Wiskunde en Informatica, Amsterdam, (January 1986).

[41] TOMIYAMA, T. and P. J. W. TEN HAGEN, "Organization of Design Knowledge in an Intelligent CAD Environment," CWI Report, Centre for Mathematics and Computer Science, Amsterdam, (in preparation, 1987).

[42] TOMIYAMA, T. and P. J. W. TEN HAGEN, "Representing Knowledge in Two Distinct Descriptions: Extensional vs. Intensional," CWI Report, Centre for Mathematics and Computer Science, Amsterdam, (in preperation, 1987).

[43] TOMIYAMA, T. and P. J. W. TEN HAGEN, "Organization of Design Knowledge in an Intelligent CAD Environment," in *Expert Systems for Computer-Aided Design Proceedings of IFIP W.G. 5.2 Working Conference 1987 (Sydney)*, J. S. GERO (ed.), to be published from North-Holland, Amsterdam, (1987).

[44] TOMIYAMA, T. and H. YOSHIKAWA, "Extended General Design Theory," in *Design Theory for CAD, Proceedings of the IFIP W.G. 5.2 Working Conference 1985 (Tokyo)*, H. YOSHIKAWA and E. A. WARMAN (eds.), North-Holland, Amsterdam, (1987), pp. 95.

[45] VAN LIERE, R. and P. J. W. TEN HAGEN, "Introduction to Dialogue Cells," CWI Report No. CS-R8703, Centre for Mathematics and Computer Science, Amsterdam, (January 1987).

[46] VAN DEN KROONENBERG, H. H., "CAD Applications in the Creative Phases of the Methodological Design Process," in *Proceedings of CAPE '86, Second International Conference on Computer Applications in Production and Engineering*, K. BO, L. ESTENSEN and E. A. WARMAN (eds.), Copenhagen, (1986), pp. 339.

[47] VETH, B., "An Integrated Data Description Language for Coding Design Knowledge," in *Intelligent CAD Systems 1: Theoretical and Methodological Aspects*, P. J. W. TEN HAGEN and T. TOMIYAMA (eds.), Springer, (in preparation, 1987).

[48] VETH, B., "The Specifications for Integrated Data Description Language," CWI Report, Centre for Mathematics and Computer Science, Amsterdam, (in preparation, 1987).

[49] YOSHIKAWA, H., "General Design Theory and a CAD System," in *Man-Machine Communication in CAD/CAM: Proceedings of IFIP WG5.2/5.3 Working Conference in 1980 (Tokyo)*, T. SATA and E. WARMAN (eds.), North-Holland, Amsterdam, (1981), pp. 35.