MIND: A Design Machine. Conceptual Framework

Abstract: We define ICAD (Intelligent Computer-Aided Design) as the instantiation of the interactive relation between designer and a variable, category-based, extendible machine, based on a parallel configuration of Intelligent Processors. The paper provides an outline of a computational design theory, followed by a set of observations and recommendations as specifications for an ICAD system. Issues of methodology and configuration conclude the conceptual analysis. In place of 'expert systems for design,' implemented as partial solutions to the design problem within limited domains, we propose 'expert systems of design, meta-expert systems containing knowledge of the design process. Expert systems of design can be built using current technology, and can be used to organize and guide the development of conventional, domain-specific, expert systems modules. The design team is the metaphor around which we developed the conceptual model of a multifunctional intelligent design (MIND) machine.

Keywords: concurrency, intelligent processor, artificial intelligent processor, natural intelligent processor, visual processing.

1. INTRODUCTION

In contemporary society design plays an increasingly important role. The visual component has proven to be not just a way of embodying results, but primarily the means for a new way of thinking. In spite of the very promising capabilities of computers for supporting the visual, design has benefited little from the advent of computers. The computer industry recognized early on the design potential of this technology but never took the time to address the specific needs of the very complex human activity that design is. General purpose machines were declared design workstations once software packages with graphic capabilities were made available.

CAD systems continue to be developed under the "general computing" paradigm. They offer an ever greater list of capabilities, but continue to be problematic because:

1. While design specifications are complex, people in the computer industry felt entitled to act on behalf of designers and made their specifications a standard for the trade.
2. Hardware has rarely been conceived in view of the specific purpose for which a particular computer is built.
3. The variety of design activities makes the determination of a common denominator quite difficult.
4. In view of the short-term success strategy that the computer industry pursued, long-term development (such as the type needed for developing a design station) was and still is being avoided.

Computer-aided design systems currently available on the market are moving towards increased integration of functions operating on a single database, or on closely related databases. The GDS integrated system offered by McDonnell-Douglas, for example, combines modules for site-planning (as understood by the civil engineer), heating, ventilating and cooling (HVAC), heat load calculations, structural analysis, space planning, facility management - all built around a central architectural modeler. Information can flow back and forth between the modules. The modules contain a considerable portion of the knowledge that an architect's consultants bring to a large-scale project or that the architect has available "in-house" for small to medium scope projects.

This raises the following questions:

1. What is it that the designer actually does? While it is clear what each of the engineering disciplines does, it is much less clear what the designer's 'territorial' task actually is. Insofar as design is not simply the sum of the parts, the increased availability of 'expert modules' makes pressing the question of emphasis and difference. Each specialized module that can be added to an integrated system clarifies what design is by precisely pointing out what design is not. Aesthetic and semantic issues become increasingly important as more and more of the technical and functional aspects of design approach resolution and automation. Accordingly, we should be able to evaluate these qualities and support them in a future design machine.

2. To a certain extent, the designer is an organizer of all the different disciplines involved in the realization

of a project. Accordingly, two questions must be analyzed:
a) Is this part of the ‘necessary and sufficient’ contribution of design?
b) Should a design machine provide means that facilitate this aspect of design?

In this paper we will first describe the requirements for a computational theory of design and the characteristics of design that such a theory must address. We will then present a model for intelligent CAD systems, followed by a methodology for extracting information about design intelligence. Finally, we will discuss the configuration of Intelligent CAD systems using present and future technologies.

1.1. Premises

A basic characteristic of the conceptual model we propose is that design is a major activity of the mind. Intelligence, which we define as generalized pattern recognition and the associated interpretation and transformation of patterns as forms of shared experience, participates in design in various stages: identification of needs, formulation of goals, critical evaluation of previous models and self-evaluation, as well as the ability to orchestrate various components participating in the design process. The interdisciplinary nature of design is made possible by intelligence insofar as it allows for elements often quite difficult to relate to each other (such as aesthetic and economic considerations, technological requirements and sociological prescriptions, high performance and ease of use, to name only a few) to belong to and to actually constitute a design identifiable through its general qualities and through its originality. During our research, we established several premises:

a) Design is a dominantly visual activity, involving not only intelligence in general, but also a specific form of intelligence related to the visual.
b) Design is an open-ended activity.
c) Although some design aspects can be represented in forms of linear relations, design by its nature is nonlinear.
d) Design is redesigning, i.e., it never starts from zero.

These premises established, we notice that quite often issues of design are treated in computer-aided design by accepting restrictions that are not design-specific but computationally motivated. We make an epistemological decision which might affect actual implementation but which in the long run assures that we deal with design and not with some operations which are accidentally performed by designers.

This decision can be expressed as follows: Instead of dealing with a subset of design, assumed closed, complete, consistent, we deal with design as open-ended, as process, and in which inconsistencies are part of the final product. The kind of design we decided to cover, i.e., the problems of the mind it involves, imposes the need to conceive of a hybrid system. We have in mind an evolving structure, continuously reconfigured in the process of interaction between the machine intelligence as provided in programs and the natural intelligence of the designer. We call this a “tunable” design machine with an important autopoietic component.

2. TOWARD A DEFINITION AND THEORY OF DESIGN

First, let us examine a structural theory of design and then extend the requirements implicit in the theory into the area of design computing. Design covers various fields of activity, such as architecture (landscape, interior, urban, monumental), visual communication, engineering, and industrial design. It is one of the most pervasive components of human activity. The following simplified representation of almost any kind of design evidences the relation between design, designer, and beneficiary (Fig. 1).

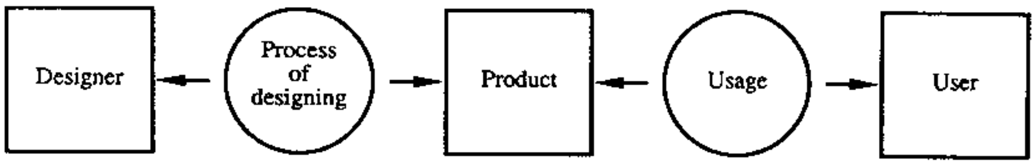

Fig. 1. Designer and user

The diagram can be slightly improved if instead of defining the object of a designer's work as the product we deal with a higher level concept: the problem. In this case, design is identified as problem-solving, one of today's dominant paradigms. Problem-solving, however, is not sufficient to cover the full extent of design.

The process of designing is quite difficult to describe due to the interdisciplinary nature of design. This interdisciplinarity can be represented, although the representation we have chosen (viz. Venn diagrams in set theory) is not necessarily exhaustive (Fig. 2).

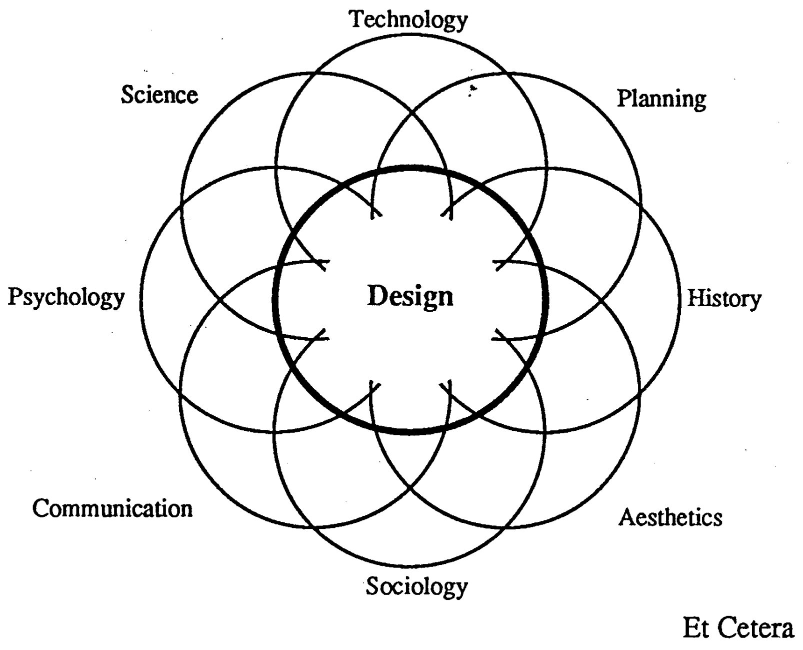

Fig. 2. Interdisciplinarity of design

The "specialized" components (such as the planning component, aesthetic quality, the social and psychological aspects of design and the product designed, communication, science, technology, etc.) require an integrated procedure (how to bring them together while maintaining the integrity of design or as a means to achieve it) as well as a self-critical moment. The components mentioned have elements in common. On a higher-level representation dealing with these common elements, we should be able to identify structural equivalences that are quite critical in the design of the intelligent component of the system that will participate in the integration. The self-critical moment of design is represented by the fact that designers as well as users of design compare new designs to previous work and situate design in the broader context of culture and civilization. It is useful at this juncture to point, through an additional diagram to the environment (in a broad sense, i.e., social, natural, cultural) in which and in relation to which designs are generated, i.e., to the pragmatic framework (Fig. 3).

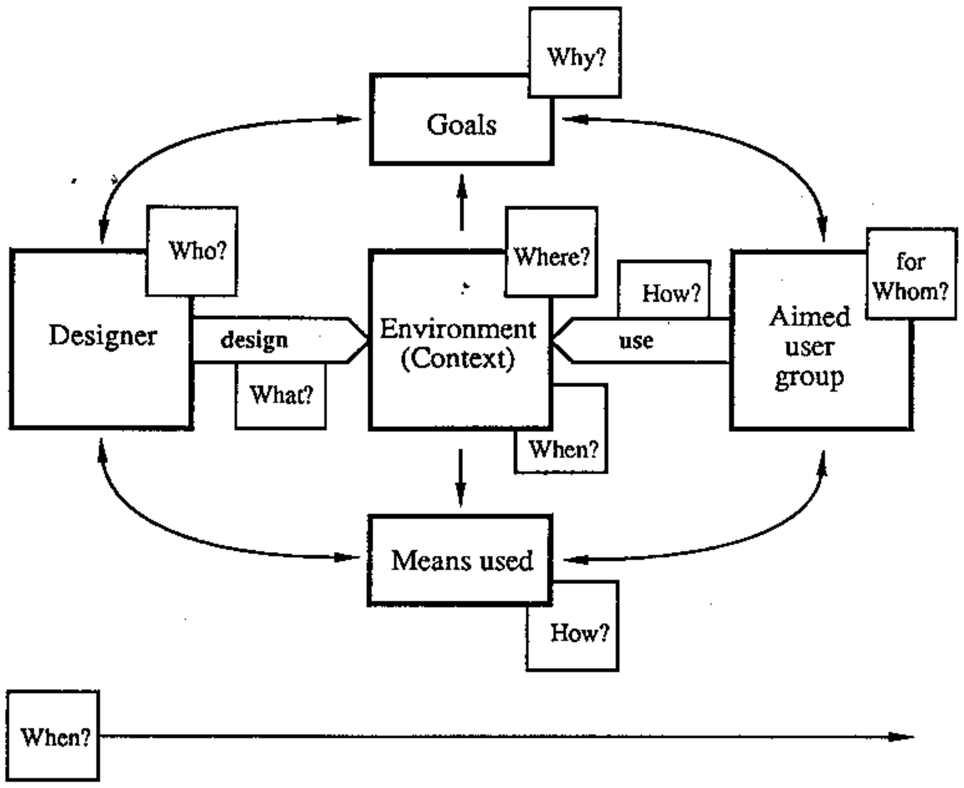

Fig. 3. Context and signification of design

In their work, designers use:
a) specialized knowledge acquired through design education,
b) general knowledge (belonging to culture),
c) tools (simple to complex, such as pencils, rulers, production tools, and more recently, computers).

They also apply what is called “aesthetic sensitivity,” i.e., knowledge concerning form and its expressive power according to ideals historically acknowledged. In the interdisciplinary design work, a hierarchical structure, within which aesthetics occupies a high position, is evident. By no accident, almost all design theories refer to this relation between the aesthetic component and the scientific and industrial aspects of design. Historically, design *can* be seen as the interaction of the aesthetic component, which encompasses art, craft, and the components of science and industry.

The design process, in its close relation to design products and their use, implies *design intelligence, cultural sensitivity,* and a *critical attitude.*

The designer works towards a goal (product) to be achieved with the help of the representation of this goal (Fig. 4). Representation can be in words, written statements, drawings, models, diagrams, etc. The dominant form of representation is visual. Some authors go so far as to speak of *visual thinking,* a metaphor that describes the participation of visual representations in thought processes.

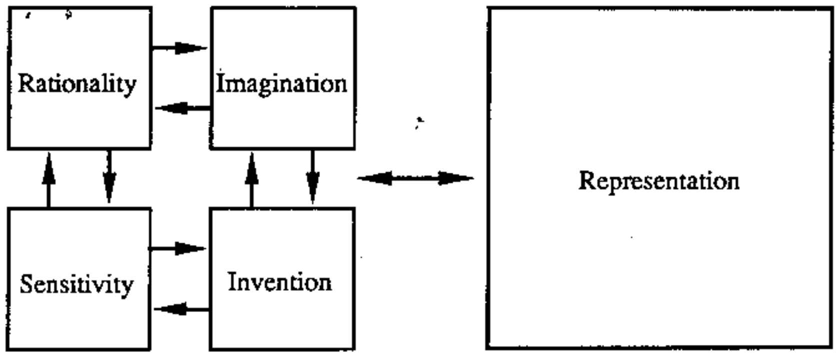

Fig. 4. Aspects of representation

3. DESIGN AND COMPUTING

To formulate a computational theory of design is not the goal in itself, but the means necessary for accomplishing the global aim of providing a foundation for computer-aided design. A computational theory of design must address the components of design in their most general expression and state the relationships between them in a way similar to the way communication theory deals with communication, for example. As with any theory, once one grasps the general elements and the fixed and variable relations of those elements, one should be able to' enter values specific to the system so as to accommodate, explain, or predict the behavior of particular design efforts. A computational theory of design must be a theory of design foremost; it must be computational in that its expression allows the computer to "understand" design and to "behave" accordingly.

On its own, such a theory will be a contribution to all design fields. Most designers have noticed the need for a design theory as a prerequisite for design education, design criticism, and design evaluation. In the process of building such a theory, the contribution from other fields (communication, aesthetics, cognitive science, sociology, and others) is important. The theory should not be hardware-dependent, but should suggest ways of improving understanding of hardware requirements specific to different applications. Computational design, together with other computational disciplines, has become necessary precisely due to the challenges to design in today's society.

4. CATEGORIES FOR A COMPUTATIONAL DESIGN THEORY

Technology advances toward faster computers, increased memory, parallel processing, expert systems containing knowledge within specific problem domains, artificial intelligence techniques employing heuristics for problem-solving, natural language recognition and computer-generated speech, pattern recognition, vision systems, and others. It also makes available increasingly powerful graphics processors with 2D and 3D and even holographic capabilities and real-time responses. As a result we see the possibility for an alternative approach to the study of the use of computers in design along the line of the premises introduced at the beginning. Since the goal of these approaches is clearly to produce "intelligent processors," *IPs,* it is possible to approach this problem top down rather than bottom up. Instead of waiting for all the techniques implied in current research to be resolved, we can assume that they have been resolved (to any desired degree of resolution, as we shall see) and proceed to find how they should be employed. This is possible because, if the goal is to produce very powerful "intelligent processors" with the capabilities mentioned above, it is possible to simulate those artificial "intelligent processors," *AIPs,* with actual designers using tools (for efficient work), natural intelligent processors, *NIPs.*

The problem can be simplified to one familiar with a design studio: How does the designer in charge

communicate with the design team
1) assuming that we already have a group of "intelligent processors" (simulated by a group of designers);
2) assuming that we have identified a (tentative, hypothetical) common language which can be arbitrarily complex but not infinite.
What kinds of communication are necessary for these processors to design an object? What is the role of the components (speech, written language, drawings, models, animation) in defining a design language? Obviously we have in mind behavioral aspects characteristic of the design teamwork, and we try to find ways to capture these in forms that would make intelligent computation possible.

We do not expect to identify the "absolute" primitives, contexts, codes, interactions, etc. Rather, we intend to arrive at an understanding of the conceptual categories, the basic elements pertinent to the pragmatics of design. The reason for this can best be understood through analogy. If we try to design a chair, we can either try to design one for the "average" (though non-existent) person and produce a chair with a tolerable degree of discomfort for most people; or we can design an adjustable chair that allows any individual to adjust the chair to one's own dimensions. In the first case, we attempt to design for the "ideal" shape, i.e. within the logical model of categories. In the second case, we attempt to identify the nature, locus, and magnitude of adjustments that will accommodate the full variability of human dimensions, the experiential component to which Lakoff (1987) makes reference in his recent work.

The second approach is in keeping with the goal of creating a "design machine." This machine should allow the designer to adjust it according to that designer's approach to design, which hopefully will not remain constant but will change as the designer "matures." The design machine must be useful to designers with different viewpoints and must allow, even encourage, the formation of new theories and approaches to design (some of which *can* be extended to several forms of design).

The creation of a hardware/software configuration for use in any task involves the embodiment of a theory, even if the configuration contains its own "inference engine.'' Thus any system relies on our ability to state explicitly what it is that we do when we do activity X. That is, the creation of any system relies on an act of communication, which we evaluate through the behavior it leads to. In the case of design, this implies that the explanation and interpretation of the behavior as studied by previous protocol analysis approaches is irrelevant. What is important is our ability to communicate things characteristic of our interaction with computers: how do we program the computer for a design task or how do we use the program? Or, in a more general way, what would an intelligent program have to accomplish in order to fulfill a design task? Research in this direction must try to uncover the *categories* of *communication,* the *categories* of *contexts* within which *communication* is meaningful, the *categories* of *conceptual primitives* that form the tokens of exchange in this communication, and so on. Ultimately, the creation of a "design machine" requires that we establish codes of communication between the designer and the computer, which means establishing the pragmatic framework of design.

Traditional protocol analysis applied to design has relied on the possibility of deriving an *explanation* of design processes from the *interpretation* of the observed behavior of the designer as well as from *concurrent explanations* given by the designer during design and from subsequent *recollections* of the thinking that prompted certain design decisions. These protocol analyses rely on the explanation or interpretation of the design behavior relevant to the problems of using computers in design. If we extrapolate from current research directions, it is possible to conduct research that does not rely on the explanation or interpretation of design behavior, *but that studies the actual relevant behavior itself,* as long as it can be studied at all. Furthermore, we conclude, this behavior is the only one that can be studied without studying the actual architecture of the human brain.

5. OBSERVATIONS AND RECOMMENDATIONS

We have defined the general character of design, and outlined the requirements of a theory of design that can be used to guide the development of intelligent CAD systems. We also suggested an approach for gathering information related to the way designers communicate in the process of designing. It is now necessary to look at some specific and critical characteristics of design as they result from the premises established at the outset of this paper. Based on the assumptions made we present observations and practical recommendations for an actual machine.

5.1. Concurrency: The Primacy of the Visual

OBSERVATION: The predominance of visual representations in design is well established, and the effectiveness of visual interfaces, even in systems that are not directly related to design, is clear. The primacy of the visual representation in design applications stems from two elements fundamental to design: *concurrency of input and output* and *concern with form.* Visual representations, unlike verbal and other representations, are primarily concurrent rather than sequential. Beyond simply allowing the designer to see what something will "look like," this allows the state of the whole design, or at least several parts of it to be inspected at once, and progress and problems to be monitored continuously. Furthermore, the visual representation allows designers to concentrate on the form of the design, where form is used not to describe the surface characteristics of the evolving design but the abstract structure (as in the form of a novel, the form of an argument, etc.). In this way the visual can be seen as a multi channel parallel means for input and output that has the capacity to handle both concrete and abstract information.

RECOMMENDATION: If the visual is understood in this way, then the primary interface between designer and machine will be visual, and specifications of high-speed interactive graphics are natural, as are requests for full color for the rapid presentation and manipulation of complex information. The capacity to support multiple visualization techniques through fast mapping is critical to any ICAD system.

5.2. Analysis vs. Synthesis

OBSERVATION: While analysis is important at several points in the process of designing, the act of designing is first and foremost an act of synthesis. As the word *synthesis* implies, it is an act of bringing together, comparing, combining, fitting, assembling a multiplicity of parts into a whole. Characteristically, the parts are not *homogeneous.*

RECOMMENDATION: Not only must a system allow the designer to manipulate many different elements, but it must also put few restrictions on what those elements may be and how they are to be manipulated. Special tools must be provided for the retrieval, superimposition, combination, and transformation of different elements.

5.3. Normative and Procedural Aspects of Design

OBSERVATION: Procedural theories of design fall short of explaining design to any depth. Normative considerations ("ought to be" theories) guide the design of architecture, products, communications, and other fields where a semantic dimension is important. Even in engineering design, as in science **(Kuhn** 1962) there are dominant paradigms which guide what should be done. There is evidence that such normative concerns offer clear benefits by providing the designer with a conceptual platform from which to launch the first attack against the problem (Rowe 1987).

RECOMMENDATION: Although normative concerns are *de facto* parts of much of the generation and evaluation of design and form a great part of the context within which designs are understood, they have been ignored almost completely by computer-aided design systems developers. An ICAD system must allow the designer to formulate normative, as well as procedural, theories and must provide tools for their application to the emerging design. Such tools must allow the designer to construct design schemata easily, check for consistency of application of design principles, allow the designer to make reference to previous solutions and ideas, etc.

5.4. Parallel and Episodic Design Process

OBSERVATION: As soon as they are given a problem, before collecting information, analyzing it, and arriving at a reformulation of the problem, designers begin to generate and evaluate solutions. Later, even as the project nears completion they speak of finally understanding what the “real problem” within the problem was. The design process that designers follow is usually presented as a rigid-state flowchart with arrows that denote backtracking, and a sequence of phases such as: problem statement, programming, data collection, data analysis, problem restatement, alternative generation, alternative evaluation, design development, evaluation, and communication. This conception of design, even with backtracking, is essentially sequential and assumes rigid boundaries between stages.

RECOMMENDATION: We propose an alternative view: a parallel design process with soft boundaries between tasks and traversed in an unpredictable, episodic manner. The designer keeps all “stage files” open all the time, simply shifting attention from one to the other, or assigning information to one stage or another. The design is not complete until all the stages have run their course. This view of the design process accounts for the “aha!” phenomenon of sudden realization, as a piece of information that is literally out of phase falls into place in another stage in the process.

5.5. Personal, Evolving Approaches

OBSERVATION: Not only is the design process not sequential, not only does it not have hard boundaries, but it is not even constant. The process a designer uses “matures” with the designer, just as the designer’s normative and procedural theories of design evolve. The process may vary both over the short range and over the long range. In addition, the process varies from designer to designer, and even from mood to mood.

RECOMMENDATION: An ICAD machine needs to have a variable architecture that recognizes and even encourages the evolving approach. This is particularly important in view of the various levels of the visual at which designers work.

5.6. Streams of Association: The Flow State

OBSERVATION: The episodic traversal of the parallel process stages and the variability of the design process itself follow the designer’s “train of thought,” that is, the stream of associations made during the effort to understand and solve the problem. Psychologists describe a state where the mind is free to move fluidly from concept to concept, and during which a person’s sense of the passage of time is diminished, as a “flow state.” Given flexible tools, such as tracing paper and pencil, a designer can enter this state with ease in the process of concentrating on the problem. Present CAD systems scarcely recognize (let alone respond to) this issue.

RECOMMENDATION: An intelligent CAD system must provide means for encouraging the designer to enter this state quickly and easily, by providing means of interaction that are sufficiently fast so that they can follow the stream of associations the designer is working on. Moreover, the ICAD system must be sensitive to the designer’s pace, actively provoking the designer by inquiries, suggestions, recommendations, in addition to passively responding to the designer’s requests.

5.7. Problem Reformulation and Alternative Generation: Multiple Representations

OBSERVATION: Designers often attempt to solve problems by shifting their viewpoint with respect to the original design specification and by playing games of hypothesis of the form “what-if-y-were-z?” When generating alternative solutions they often apply the “what-if?” form to certain parts of the design but not to others, or not to the same degree setting up, in effect, informal generative systems. Finally, if the situation in which they must design is in itself problematic--for example, if there is not enough time or there are not enough funds to complete a project--they apply the “what-if?” form to the design process itself, thus moving to a meta-design level. We see that in the reformulation of the problem, the generation of alternatives, and the design of an approach to design, multiple representations are maintained and mappings are constantly being revised and updated.

RECOMMENDATION: An ICAD system must allow multiple representations of problems, solutions and processes, and provide a panoply of tools for moving from one to the other.

5.8. Management of Partial or Previously Rejected Solutions

OBSERVATION: Designers frequently return to solutions that were sketchy, partial, or rejected. Traditional media leave a physical record that allows for this kind of "hill-climbing." This can occur either diachronically or synchronically. The following is a good example of diachronic use of partial solutions: Alvar Aalto, a leading figure in modem architecture, is said to have designed on a continuous roll of paper which was set up so that, using cranks, the history of a design's development could literally roll before his eyes. The synchronic use of partial design solutions is evident in the multiple "tracings" designers make, where selective copying allows information from previous attempts to be recovered easily. Both approaches have implications for ICAD that are relatively easy to implement.

RECOMMENDATION: The diachronic record of partial solutions is simply a history of commands; the synchronic requires provisions for recognizing and extracting partial solutions of elements that, as we mentioned previously, may not be homogeneous. Further intelligence could be added if the system could analyze partial solutions and bring them up as particular conditions were met.

5.9. Problem/ Solution Relationship

OBSERVATION: Many design problems involve the adaptation of previous solutions to similar problems. It is often possible, therefore, to gather previous solutions and classify them into different types, creating a typology of solutions. Solving a problem involves searching for an appropriate type of solution and modifying it to suit the requirements of the problem at hand.

Many design problems do not have direct precedents, and one cannot simply modify a pre-existing type. However, since the solution to a problem stands to the problems it addresses as figure stands to ground, it can be seen that a parallel, invisible knowledge of problems is implicit in our knowledge of the world. When the designer is faced with new problems it is this knowledge that provides access to the solution space. This knowledge is usually implicit and unarticulated, yet it is indexed by partial choice strategies. These strategies mediate between problems and solutions.

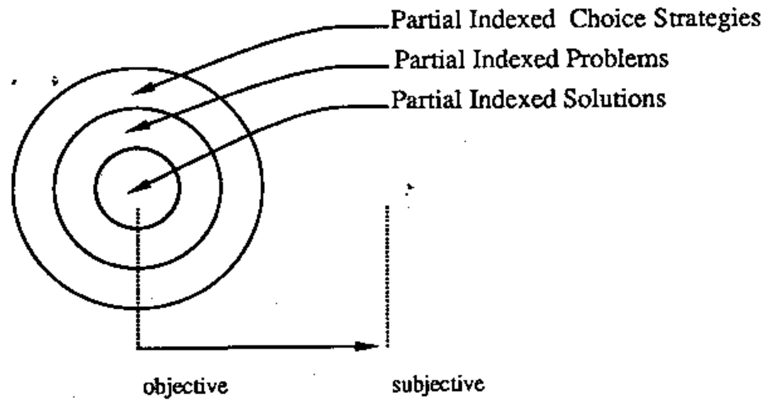

Fig. 5. Hierarchical relation of solutions, problems, and choice strategies

RECOMMENDATION: A typology of problems can be constructed that allows the designer to manipulate the problem itself, as well as to have access to “indexed partial solutions” (Brown and Chandrasekaran, 1985), indexed partial problems, and indexed choice strategies (Fig. 5). Indexed solutions represent objective answers, while choice strategies embody “intelligence” and support subjective components.

5.10. Roles and Performances

OBSERVATION: Designers very often “cast” either the developing design or themselves in “roles,” relying on accumulated experiences of typical situations to provide backgrounds for visualization or simulation of situations, and as means for achieving empathy with the users of designs. Designs are often judged by their “performance.”

RECOMMENDATION: The frame and script representation of knowledge familiar to artificial intelligence researchers should be extended to allow the representation of knowledge of “roles” and “performances”; a key operation using this representation must be “casting” and “performing.” Unlike contexts, which passively illuminate meaning, *frame/script/role/performance* representations can allow the designer to place objects in situations dynamically.

5.11. Classification

OBSERVATION: The abilities to classify, and act upon, selected features of iconic and symbolic aspects of design problems are characteristic of design intelligence. Suppression and enhancement of features, feature extraction, and recognition of objects, attributes, relations and actions are classification acts central to design cognition.

RECOMMENDATION: It is necessary to provide efficient visual classification tools via semantic networks. At a more general level, classification within problems has to be supported through frames, scripts and “roles” and “performances,” as described above. Action upon features of the iconic and symbolic realm has to be supported by adequate computer architecture. In addition to vertical, hierarchical classification through inheritance or other means, which *can* be restricting, a flexible, horizontal, scheme of classification through reuse of attributes across object descriptions is desirable. Objects must allow for an open-ended, extensional (Veth, 1987) description. New ’attribute set’ objects should be created automatically by the system (Fig. 6).

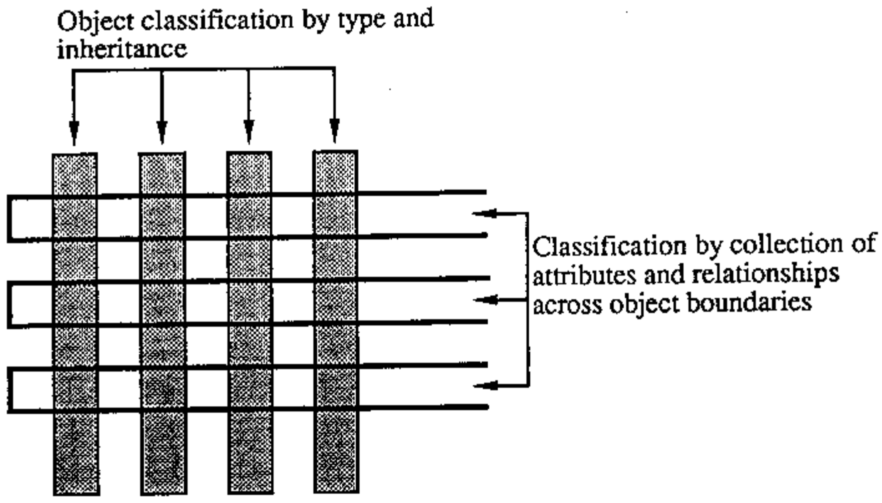
Object classification by type and inheritance
Classification by collection of attributes and relationships across object boundaries

Fig. 6. Classification by attributes and relationships

6. A MODEL FOR ICAD

We can now suggest a model for intelligent CAD systems based on a structure uniting IPs, categories of communication, and the concept of a variable and extendible machine (Fig. 7). More precisely, the model proposed consists of several structural components: flow information metaphor, context interpreter, process and problem management, variable interface, and intelligent processors.

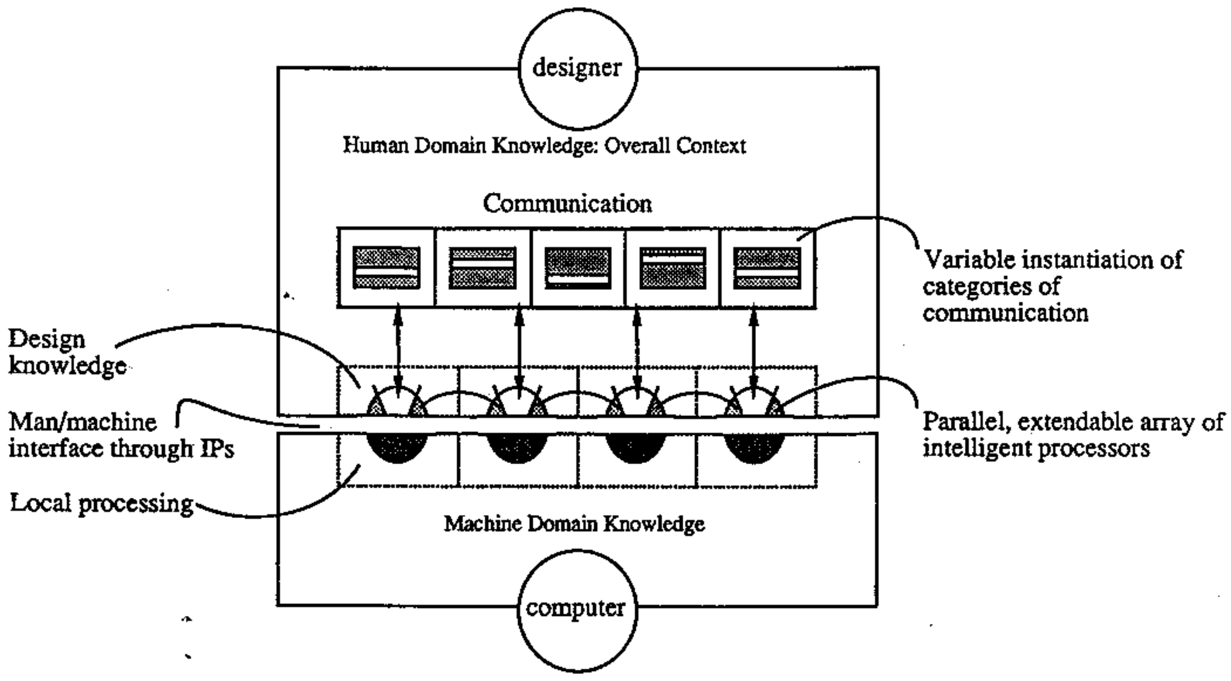
designer
Human Domain Knowledge: Overall Context
Communication
Variable instantiation of categories of communication
Design knowledge
Man/machine interface through IPs
Local processing
Parallel, extendable array of intelligent processors
Machine Domain Knowledge
computer

Fig. 7. System overview

6.1. Information Model: The Flow State

The primary goal an ICAD system must strive to achieve is the maintenance of a state in which concentrated mental activity occurs, the sense of the passage of time is diminished and uninterrupted, “fluid” thought is possible. To achieve this goal, maximum transparency of interface and representation is required and real time response with high-resolution graphics is necessary (Fig. 8). Furthermore, the system must be active in making observations, suggestions, provocations, associations, and otherwise providing stimuli that keep the designer’s mind engaged with the problem.

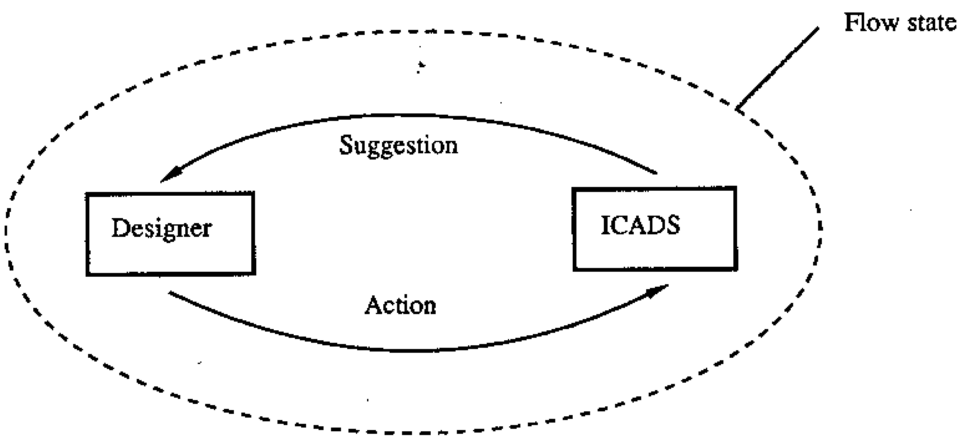

Fig. 8. Flow state interaction

6.2. Context: Interdisciplinarity - Synchronic/Diachronic

Design can only be understood in an interdisciplinary context which must be synchronic as well as diachronic, procedural as well as normative, domain-specific as well as general. It must contain information from many disciplines and must be easily expandable, both through local editing of its knowledge base and through the addition of modules representing other disciplines. Some of the modules need only be virtual modules--the actual information is provided by electronic network information or other databases. These can be visualized as consisting of a modular but interrelated knowledge base and a context interpreter. The context interpreter's task is to attempt to find as many matches as possible between the user/system interaction and the context knowledge modules and to use this information to monitor and control the system's variable configurations (Fig. 9).

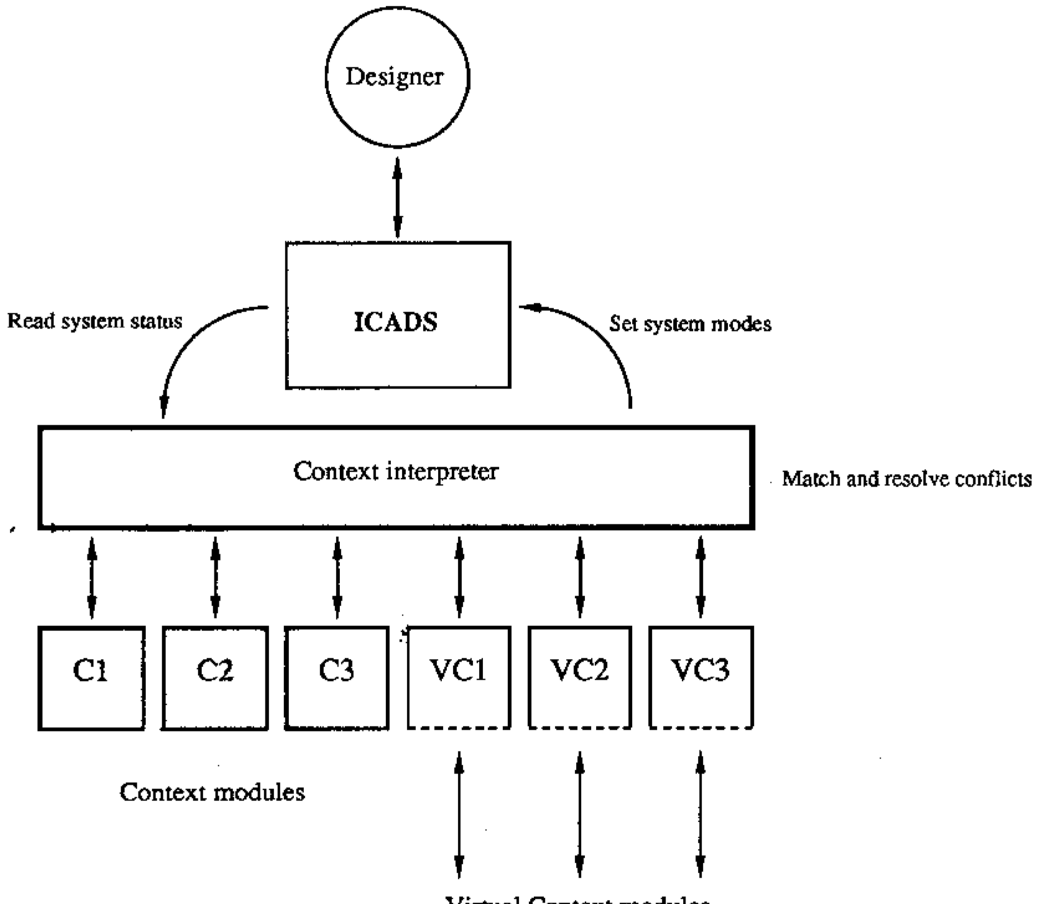

Fig. 9.9. Context interpreter

Fig. 9. Context interpreter

6.3. Process: Process Management - Parallel, Episodic, Individual, Evolving Process

The design process in this model is a parallel process. All stages of the process commence at the onset of the design effort and remain active throughout the duration of the project. Even though the designer may focus attention on one aspect of the design at a time, observation indicates that, at any point, the designer may divert attention to another part of the process, apparently following a stream of associations. In any case it seems clear that when one works on one task one does not necessarily stop thinking of another task. Moving from task to task therefore needs to happen as quickly as possible and with the least restriction in terms of timing, task completion, and so on.

As shown in Fig. 10 a & b, the parallel process involves overlapping, concurrent steps. Each step may be thought of as a stack that needs to be filled. The idea of *progress stacks* is used to indicate that the design is not completed until all stacks are full, though each stack may have items removed or inserted at any point along the design period. Thus, for example, the designer may make some progress in problem definition, think of some alternative solutions, consider communication options--adding tokens to the progress stacks--then realize that the problem definition is inadequate--removing tokens from that stack. He could return to ideas about communication and perhaps not complete the problem definition stack until the design is almost complete. Visual feedback could allow the designer to monitor the overall progress of the project.

Unlike previous models, this model assumes that the stages of the design process are not rigid, but that they have soft boundaries. In addition, the design process is seen as being episodic (Rowe, 1987), individual, and evolving. An ICAD system cannot impose a fixed process on the user, but must instead provide tools for inspecting, evaluating and modifying it as needed. In that sense, a meta-process level is implied which will allow the designer to re-design the design process.

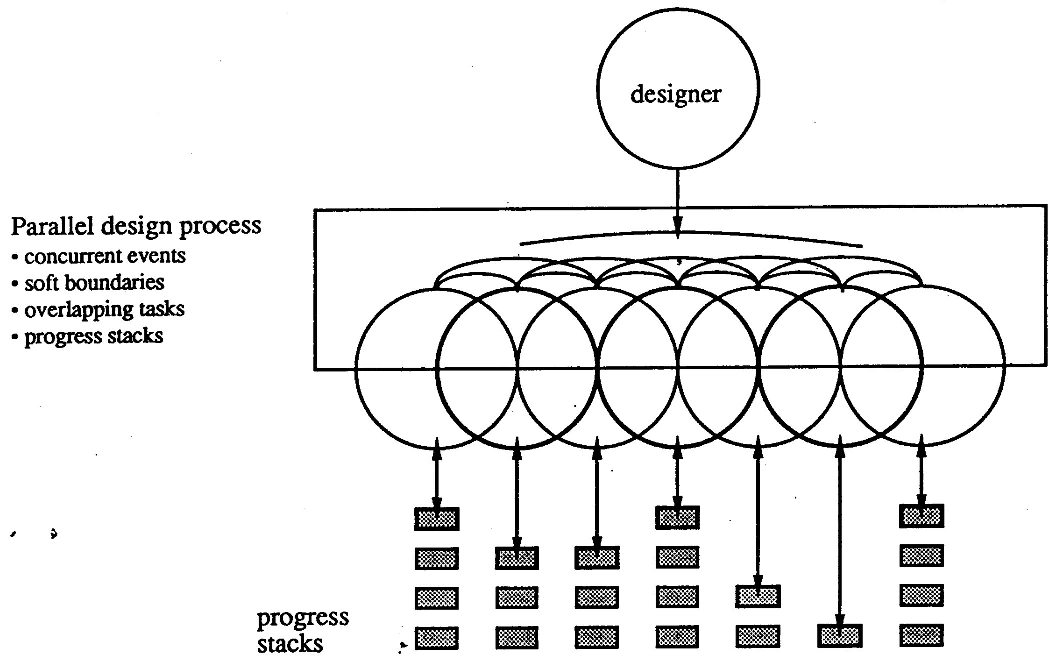

Fig. 10(a). Parallel design process

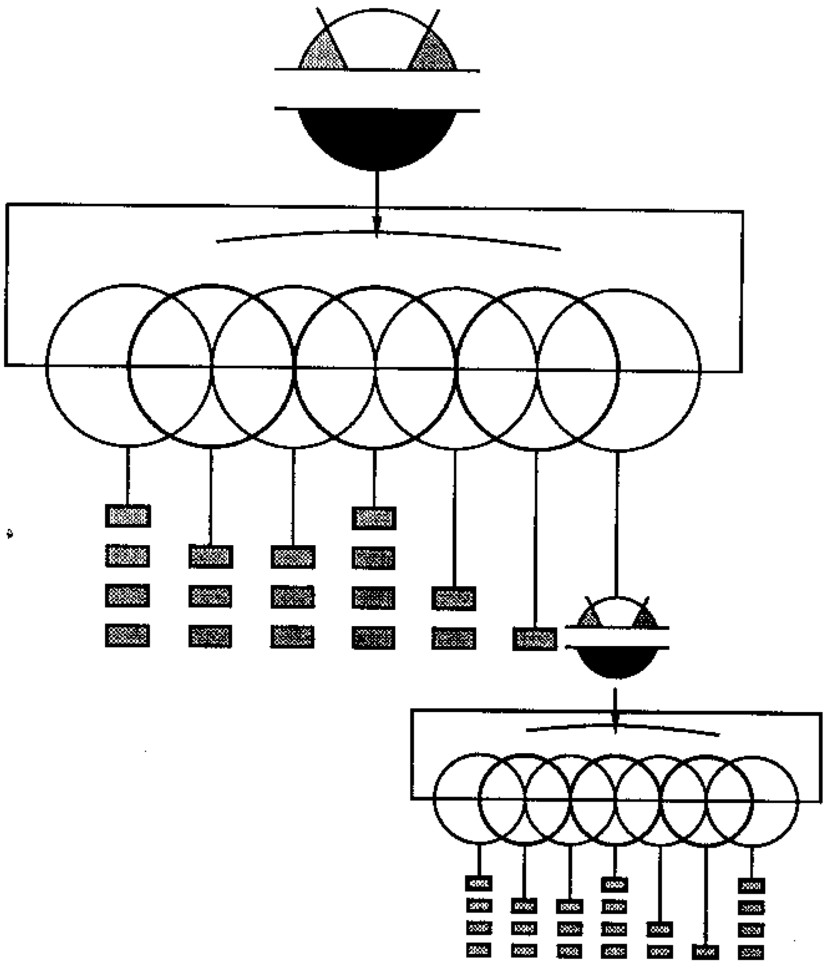

Fig. 1O(b). IPs in parallel design process

6.4. Interface: Interface Management - Variable Machine

Based on what is already known about designers and how they work, it is expected that the Design Machine will be a variable machine. As such, it must not only be adjustable by the designer, but must also be able to "recognize" the designer's "state of mind," the place of the designer's actions in the design process, the context appropriate to understanding these actions. It must "adjust" its interface as necessary to support the designer, different designers, at different times, facing different design problems, and similar changes in the context of design. We shall especially consider the role of information related to the transitions between states (of thinking, of the design process, of scope, scale, category, association, etc.).

A human intelligent "processor" learning design (in college, in a studio, etc.) can follow a train of thought suggested by the experienced designer (procedural learning) and often will attempt to anticipate the next step. The knowledge that allows this to happen will also be researched, formalized, and incorporated in the Design Machine. This knowledge is expected to be relative, and to vary from designer to designer, from task to task, from time to time, from domain to domain, and from context to context.

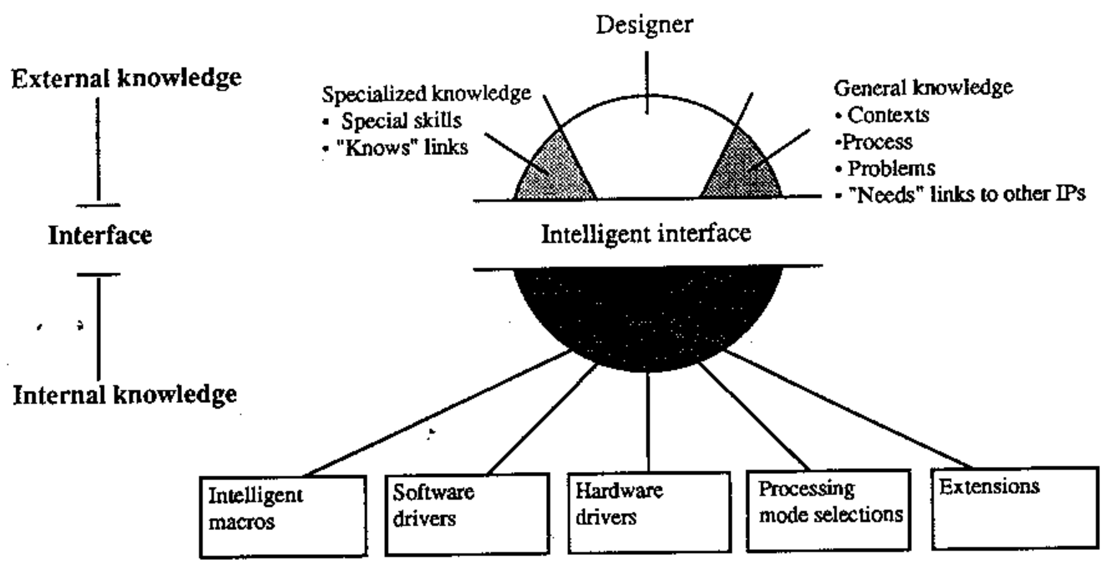

Fig. 11. Intelligent processors

6.5. Intelligent Processors: General/Specific IP

The model (Fig. 11) illustrates the architecture of the intelligent processors themselves: each processor has external, internal, and interface knowledge. The external knowledge is divided into general, specialized, and designer characteristic knowledge, while the internal knowledge contains information about the computer, peripherals, and software. Knowledge connecting the external knowledge and the internal knowledge is contained in the interface section of the IPs and provides an intelligent connection between the designer and the lower level software and hardware. The general knowledge of the IPs is made up of information allowing access to the context knowledge, access to other IPs, and knowledge of the design process and its progress stacks. The specialized knowledge component allows each IP to act as a specialist when needed, while still acting as a normal member of the IP design team under most circumstances. Knowledge of the designer allows it to recognize idiosyncrasies, anticipate questions, and adjust to different users. Knowledge of the computer, peripherals, and software constitute the other major portion of an IP. Between the external and the internal knowledge is a section containing intelligent interface knowledge, knowledge which provides the designer with a mapping of actions which is not one-to-one, but one-to-many, so that one action by the designer may elicit many actions by the system.

The relationship between the IPs is reconfigurable, so that parallel, sequential, and hierarchical connections can be established as needed (Fig. 12, a, b, c). A stacking mechanism allows recursion to occur by keeping track of partial results, so that ring and loop connections can also be established. Such connections are evident in design activity, especially during problem definition, ideation, and alternative generation, when IPs interact rapidly with the designer and with each other.

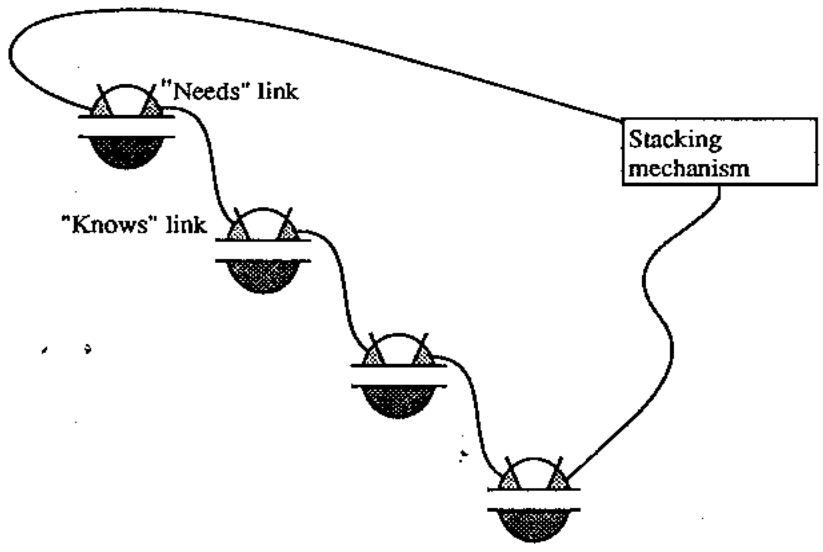

Fig. 12(a). IP link/loop configuration

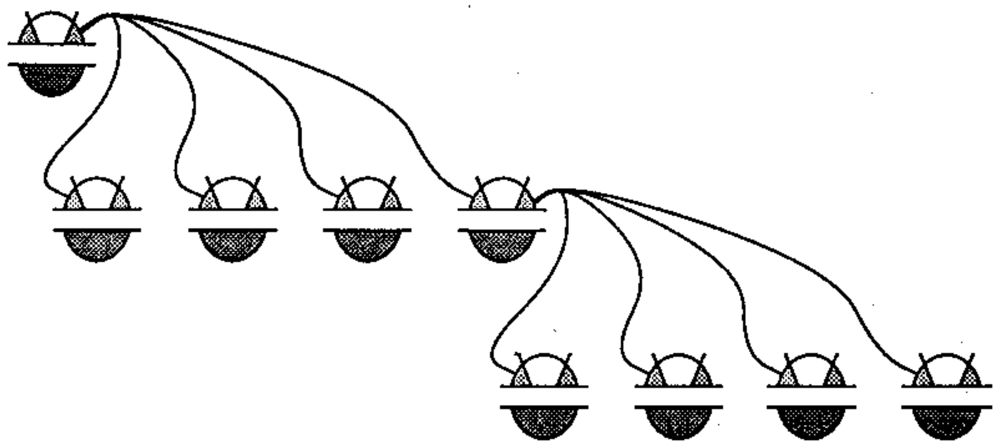

Fig. 12(b). IP hierarchical/networked configuration

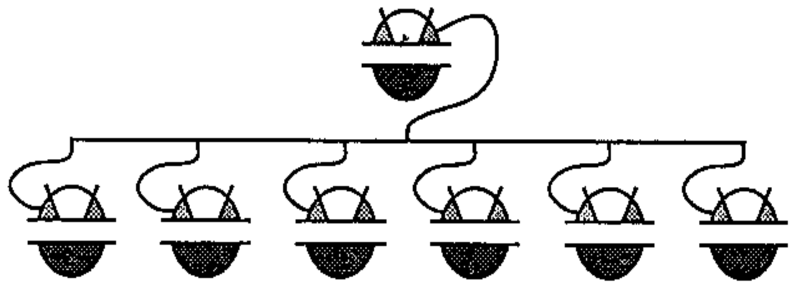

Fig. 12(c). Concurrent IP configuration

6.6. Problem: Problem Management

A problem management utility is provided. This utility contains knowledge about problem-solving approaches, i.e., top-down, middle-out, bottom-up, as well as heuristic tools, rules of thumb, rules for divergence and convergence. It also contains other problem solving strategies particular to the designer and tools for choice of strategies. It is used to monitor the designer's interaction and recognize the approach taken. At times it can interrupt the designer and propose another strategy, or suggest another viewpoint. It is responsible for maintaining the "flow state" mentioned earlier.

6.7. Representation: Representation Management

The representation management utility works in parallel with the problem management utility (Fig. 13). It provides the designer and the IPs with a broad selection of representations that allow the designer to look at the information in a variety of different ways. In order to maintain the flow state, and based on the progress being made and the designer's place in the design process, the representation management utility can suggest alternative representations to the designer. An integral part of the representation management subsystem is to maintain partial solutions for review and reuse, and to provide tools for their manipulation and combination.

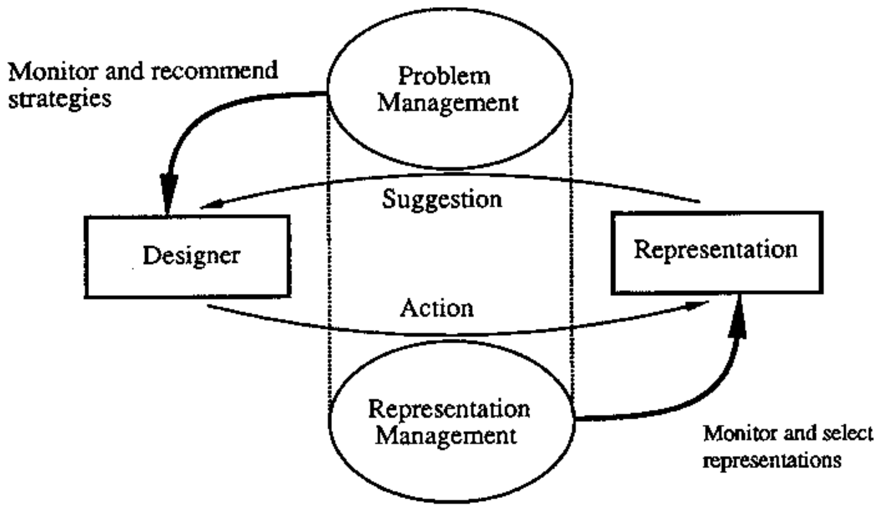
Monitor and recommend strategies
Problem Management
Suggestion
Designer
Representation
Action
Representation Management
Monitor and select representations

Fig. **13.** Problem and representation management

7. METHODOLOGICAL APPROACH

By focusing on NIPs, it is possible to collect the information needed for the implementation of the AIPs and to arrive at the necessary formalization for the creation of a system. The following methodology forms the basis for research aimed at identifying the categories of communication used in design.
.

7.1. Analysis

The results of the analytical stage may be of only limited usefulness as synthesis is likely to be very different from analysis. Nevertheless, the following approach can be suggested:
a) Identify possible communication categories by involving designers in explaining why designs acknowledged as "good" and "valid" are "good" and "valid"; identify conceptual primitives in each category.
b) Identify contexts relevant to design within which categories and primitives are meaningful.
c) Identify prior knowledge implicit in contexts.
d) Identify classes or representations used--iconic, symbolic, indexical--and their relation to semiotic categories of interpretation.
e) Identify how all these are communicated, the degree of precision (i.e., vagueness), the influence of precision, etc.

7.2. Synthesis

a) Instruct the designer to "lead" a team of "intelligent processors" in the design of simple objects; observe aspects of communication, especially related to how designers "externalize."
b) Identify representations, their relation to prior knowledge, conceptual categories, primitives, and contexts used.
c) Identify means of communication: verbal, mimetic, visual.
d) Identify ways of thinking employed: association, analogy, metaphor, simile, synecdoche, etc.
e) Identify modes of communication: affirmative, negative, declarative, imperative, interrogative.
f) Identify approaches to problem solving: top-down, bottom-up, breadth first, depth first, special heuristics.
g) Identify manner of problem reformulation.
h) Identify aesthetic considerations and their formulation in the design process
i) Identify choice strategies for knowledge acquisition and distribution.

j) Combine results with those of analytical phase.

7.3. Hypothesis and Testing

a) Create a hypothetical "language" model and use it for specifying design programs.
b) Instruct a group of designers and the design leader on this model; clarify which elements of the hypothetical language can be used; observe discrepancies, assumptions, shared knowledge, etc.; test using programs.
c) Improve language model.
d) Repeat as necessary.

7.4. Extract Patterns: Operational Knowledge

In order to facilitate implementation on computers, we need to operationalize the knowledge acquired. The steps to follow are:
a) Identify categories of conceptual adjustments.
b) Formalize these categories in the hypothetical design language.
c) Test and improve the language using intelligent processors and prototypical software/ hardware configurations.
d) Search for patterns of frequency, distribution, interconnection, proportion, and especially transition in the ranges of variables identified; search also for constants, minima and maxima, averages, and other such quantities that may be related to quantifiable measures of performance.

In general, analyze the information gathered from the use of prototypical systems in order to find what settings are important for different phases of a design task, for different tasks, and for different task domains.

A rapid hypothesis/test cycle should be set up to allow fragments of hypothetical languages to be rapidly formalized and tested.

1) The designer communicates with IPs using an initial hypothetical language derived from the analysis of existing designs, other interviews with designers, and conventional design concepts and terminology. NIPs carry out commands manually. Hypothetical language is augmented by communication directed towards synthesis.
2) The designer communicates with IPs. IPs act as front ends for fast interactive workstations.
3) Designer/IP communication is formalized into a sketch "language" on artificial intelligence/expert system workstations.
4) The designer uses sketch language to design. Discrepancies and inadequacies are noted. Hypothetical language is updated. Process is repeated with new hypothetical language.

8. CONFIGURATION

In addition to the requirements discussed in the recommendations mentioned above, we have to add requirements pertinent to the visual processing specific to design. There are two levels at which visual processing takes place: the iconic level (realistic images), treated mainly in bitmapped two- and three-dimensional representations, and a symbolic level of abstracted images. The two levels require different kinds of processing in order to support optimum interaction. One should be highly vectorizable, supported by single instruction multiple data (SIMD) architectures, dealing with image processing. The other is supported by multiple instruction multiple data (MIMD) architectures, in the symbolic domain. Since the designer goes from one level to the other frequently, the IPs should support the change in mode. We suggest here a configuration appropriate for this purpose (Fig. 14).

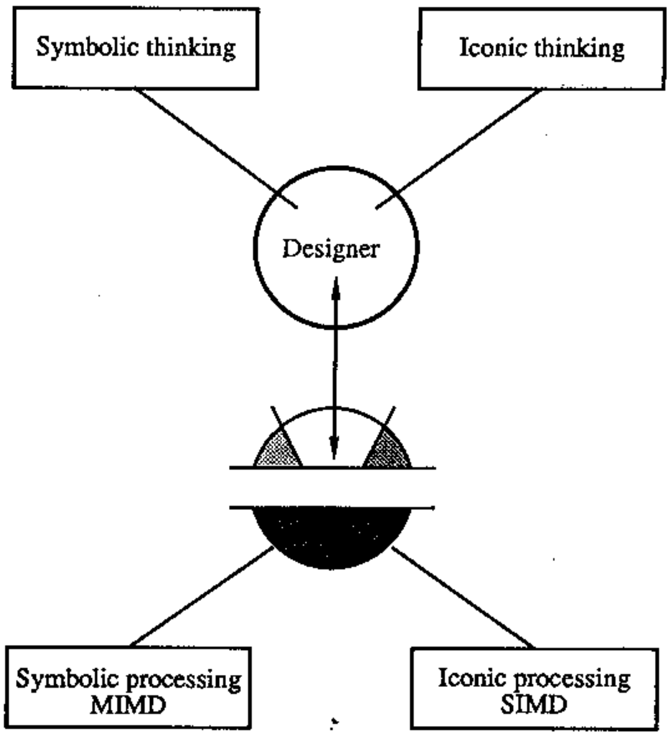

Fig. 14. Processing configuration

The iconic domain of our real-time design machine requires intense computation. Conventional computer architectures, based on the von Neumann machine, are limited in their speed. However, they are flexible. SIMD uses direct memory access and thus avoids time-consuming MOVE instructions.

The two-dimensional arrays used in image processing are well suited for the vector processing of SIMD. The only condition is to keep the pipeline operation uninterrupted. The MIMD architecture is a multiprocessor system useful where the SIMD architecture with pipeline processors or systolic arrays is not recommended because of the predominantly scalar nature of the problem. When we need multiple small window processing, as in design applications of a symbolic nature, the MIMD architecture is better suited (Giloi, 1986).

Having this architecture in mind, we have to define different levels of software, starting with the operating system and all its utilities, drivers, communications software, file management, etc. Since it is supposed to be a real-time machine, we have to provide real-time capabilities.

The applications software level is designed having in mind programming languages in which design applications are programmed. Together with appropriate languages, we have to provide a library of generic and specific routines for image analysis and image synthesis. In view of the machine's variability ("tunable"), the user interface will be built up by the user. It seems quite probable that the programming style will be the result of using languages with new data types characteristic of iconic image processing and symbolic image processing pertinent to the designer's activity. These types are provided in addition to conventional data types such as integers, strings, and arrays.

9. CONCLUSION

The complexity of design makes the attempt to build ICAD machines unlikely to result from a piecemeal approach. We discovered that what is necessary is an encompassing, but not enclosing, concept. This is represented by the suggested variable, category-based, extendible machine based on IPs, for which this paper proposed a model. We do not try to emulate ways in which designers solve partial problems but rather suggest an environment in which design intelligence is supported in the interaction between designers and the design machine.

REFERENCES

Brown, D. C. and Chandrasekaran, B. (1985). Expert systems for a class of mechanical design activity, in *Knowledge Engineering in Computer-Aided Design,* Proceedings of the IFIP WG5.2 Working Conference 1984 (Budapest), Gero, J. S. (Ed.). Amsterdam: North-Holland, pp. 259-290.

Giloi, W. K. (1986). *Functionality and Architecture of Machine Vision Systems,* Lecture Course (notes for the USLA computer vision course).

Kuhn, T. (1962). *The Structure of Scientific Revolutions.* Chicago: Chicago University Press.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* Chicago: Chicago University Press.

Rowe, P. (1987). *Design Thinking.* Cambridge MA: MIT Press.

Veth, B. (1987): An integrated data description language for coding design knowledge, *Intelligent CAD Systems I.* Theoretical and Methodological Aspects. Berlin/New York: Springer Verlag.