Yehuda E. KALAY
Anton C. HARFMANN
Lucien M. SWERDLOFF
Computer-Aided Design /
Graphics Laboratory
School of Architecture
and Environmental Design
State University of New York
at Buffalo

Knowledge-Based Computer-Aided Desgin: The Computer as Design Partner

SOMMAIRE

Il est possible de definir le design comme un processus a la recherche d'un projet physique ou organisationnel qui, une fois réalisé, accomplira certains objectifs et se conformera à certaines contraintes. Ce processus, qui d'habitude s'applique à des situations non triviales, est caractérisé par l'exercise d'une pensée créatrice et d'une faculté de jugement. Des fonctions de représentation et d'analyse facilitent ces deux fonctions à haut niveau grâce à une mise à jour constante du projet émergeant, et an fournissant des mesures spécifiques, à la fois quantitatives et qualitatives, des performances qui en sont attendues.

Dans cette communication, il est suggéré que les ordinateurs pourraient assister les fonctions de pensée créatrice et de jugement en design, s'ils se fondaient sur des types de connaissance et d'expérience similaires à ceux sur lesquels se fondent les designers. Ces connaissances peuvent être représentées et fichées sous forme de critères de performance, objectifs, et représentations graphiques. Il n'est pas nécessaire pour ce genre d'assistance que toutes les étapes du processus de design soient assistées par ordinateur. Certaines opérations devraient être exécutées par le designer, tandisque d'autres sont exécutées au mieux par l'ordinateur. Une allocation des tâches dynamique entre le designer et l'ordinateur permettra une attitude plus flexible devant le problème de l'automatisation du design, pouvant en particulier s'adapter à des besoins changeants, des problèmes imprévus et des possibilités nouvelles au fur et a mesure qu'ils se manifestent au cours du processus de design.

Une méthodologie est présentée - ainsi que sa traduction en PROLOG - pour développer des assistants de design automatisés avec connaissances. Telle que présentée, cette méthodologie se différencie d'autres facons d'aborder l'utilisation des ordinateurs dans les fonctions de pensée créatrice et de jugement en design de par son étendue et sa flexibilité. Elle comprend toutes les étapes du processus de design, et incorpore des méchanismes d'acquisition des connaissances qui permettent au designer d'accroitre et de modifier la représentation des connaissances du système de façon dynamique.

Il est suggéré que des systêmes de conception assistée par ordinateur modélisés suivant cette méthodologie représentent une adaptation meilleure des ordinateurs afin d'assister le design d'objets physiques, ceci conduisant ainsi à une réalisation meilleure des promesses de la CAO pour améliorer la productivité des designers ainsi que la qualité du produit final.

ABSTRACT

Design can be defined as a process of searching for a physical or organizational schema which, when realized, will achieve certain goals and abide by certain constraints. This process, which is usually applied in complex situations, is characterized by creative thinking and judgment. These two high-level functions are facilitated by representational and analytical functions, which keep track of the emerging schema and provide specific quantitative and qualitative measures of its expected performance.

In this paper it is suggested that computers could assist in the creative and judgmental functions of design if they had access to knowledge and experience similar to that which designers rely upon. This knowledge can be represented and stored in the form of performance criteria, goals, and design plans. Such assistance does not necessitate that all steps in the design process be computer-aided. Some design operations should continue to be performed by the designer, while others are best performed by the computer. The dynamic allocation of tasks between the designer and the computer will enable a more flexible approach to design computability, particularly in responding to changing requirements, unforeseen problems, and emerging opportunities, as they arise during the design process.

A methodology, and its PROLOG implementation, for developing knowledge-based computerized design assistants is presented. This proposed methodology differs from other approaches to the employment of computers in the creative and judgmental functions of design in its scope and flexibility: it spans all phases of the design process, and incorporates knowledge-acquisition facilities which enable the system's knowledge-base to be dynamically expanded and modified.

We believe that computer-aided design systems modeled after this methodology represent an improved adaptation of computers for assisting in the design of physical artifacts; therefore, they may lead to a better realization of the promise CAD holds for improving designers' productivity and product quality.

1. INTRODUCTION

The use of computers in the design professions has attained a fair amount of success in the two major areas of their employment: the representation and the analysis of design solutions. In general, however, CAD has failed to meet its proponents' high expectations for increasing productivity and product quality. Arguments for assigning to computers the role of design partners, rather than the role of design tools, have been set forth in the past few years to address this situation (1,2). This major shift requires both an acceptance by designers of the possibilities of "intelligent" design machines, and the development of fundamental paradigms of design as a computable process. While the former can only be attained through time and favorable results, the latter is the focus of this paper.

At the core of any such paradigm is a basic understanding of the process of design. Although many models have been developed, one which seems workable for our purposes defines design as a process of search for a physical or organizational schema which, when realized, will achieve certain goals and abide by certain constraints (3). Since this process is usually applied in complex situations, design is characterized by creative thinking and judgment - two of the most distinguishing functions of the human mind (4). These high-level functions are facilitated by representational and analytical operators, which keep track of the emerging schema and provide specific quantitative and qualitative measures of its expected performance.

The complexity of design and its attenuating uncertain qualitative results gave rise to the eighteenth century legacy that design is a process driven by divine inspiration and intuition, whose practitioners attain the status of artists (5). Even though this legacy discouraged the development of a design theory whose basic principles could be systematically taught, many attempts have been made to rationalize the design process, in the hope of reducing its unpredictability. These attempts have resulted in the formulation of several well-established models of design (6,7,8,9).

The realization that computers have the ability to assist in intellectual processes previously considered the exclusive domain of humans (10,11), prompted researchers to investigate the possibilities of design as a formal computational process (4,12,13,14,15,16). By defining a plausible "computable" design paradigm, it is believed that the role of computers can be expanded from their established ability to represent and evaluate design solutions, such that they will be able to assist in the creative and judgmental functions of the design process as well (1).

The paradigms thus developed have demonstrated that certain aspects of the process of design could indeed be computed, or at least computer-assisted, provided that design is considered from particular points of view. They have also reaffirmed the complexity of design, whether computer-aided or not, and the value of finding methods by which designers could be relieved from some of the responsibilities for making, tracking, and verifying the myriad decisions that comprise the creation of even trivial artifacts.

In this paper we present a paradigm of design as a computable process, in which computers assist, rather than merely facilitate design, by performing some of the creative and judgmental functions of the design process. This paradigm can be achieved by providing computers with knowledge and experience

similar to that of designers, stored in the form of performance criteria, goals, and design plans. It is not necessary for all steps to be computer-assisted in order to significantly improve the utility of computers in the design process. Some steps should continue to be performed by the designer, while others may be delegated to the computer. We believe that the dynamic allocation of tasks between designer and machine, in which the strengths of each partner are fully utilized, will result in a more flexible approach to design computability. Morever, it will allow the designer and the system to respond to changing requirements, unforeseen problems and emerging opportunities as they arise during the design process.

The proposed paradigm differs from other approaches to the employment of computers in the creative and judgmental functions of the design process in its scope and flexibility: it spans all phases of the design process, and incorporates methods for knowledge acquisition. It does not prescribe a specific design process, nor limit in any way the development of design solutions.

2. METHODOLOGY

The methodology which we propose for developing knowledge-based computer-aided design systems has its basis in Simon's state/transition model of design as a problem-solving search process (16). In his model states represent design solutions, and transitions represent the processes that produce new candidates for consideration as potential solutions to the design problem. The model proposed here differs in that states represent design goals instead of design solutions. Goals are comprised of context-dependent sets of constraints, which define the conditions that candidate solutions must meet. The major advantage of this approach is that a goal can be achieved by a variety of design solutions, whose particular composition is not explicitly prescribed. Transitions are defined in terms of processes which determine search strategies among alternative goals (Figure 1).

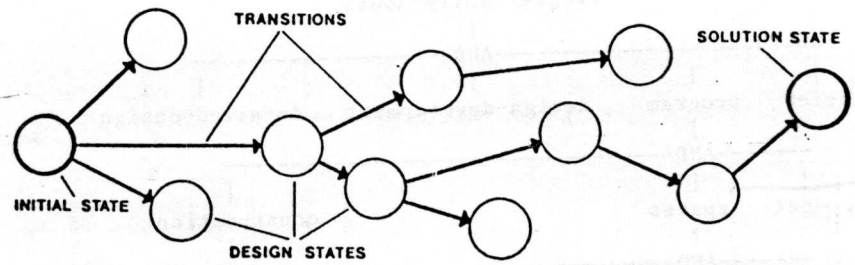

Figure 1: Alternative design plans

The design process as a whole is represented by the sequence formed by transitions through a set of goals. A design planning strategy can be mapped out using both pre-defined goal sequences, when structured methods exist, or dynamic goal sequences which will support creativity and unexpected situations.

2.1 Goals

Each goal represents a particular design objective at some level of abstraction. This objective is stated in terms of constraints that a candidate design solution must satisfy in order to achieve the goal. The goals represent specific design knowledge, which candidate design solutions are evaluated against to establish the accomplishment of the design objectives.

Since the constraints in each goal represent actual design knowledge, they are likely to be inexact and internally conflicting. Therefore, a candidate design solution which satisfies some constraints, may also violate others. To solve this problem, as well as to enhance the flexibility of the system, the goals include a set of rules that represent knowledge about the constraints themselves. These rules are in the form of context-dependent weight factors. They establish the relative importance of satisfying various sets of constraints to the achievement of the goal as a whole, thereby introducing the dynamics of particular circumstances, prior actions and previously acquired information into the design process. The achievement of each goal may thus be accomplished by several different design solutions, whose quality depends on particular design conditions.

There are two types of constraints, "hard" and "soft," which differ in the conditions for their satisfaction (17). A hard constraint is either satisfied or violated; a soft constraint is expresses as a numerical value which qualifies the expected performance of the candidate design solution in that category. Evaluation of constraints is performed by an open-ended set of "experts" which constitute part of the system's knowledge-base. Each one of these "experts" consists of a task-specific algorithm or rule-based production system, depending on the nature of the task it performs.

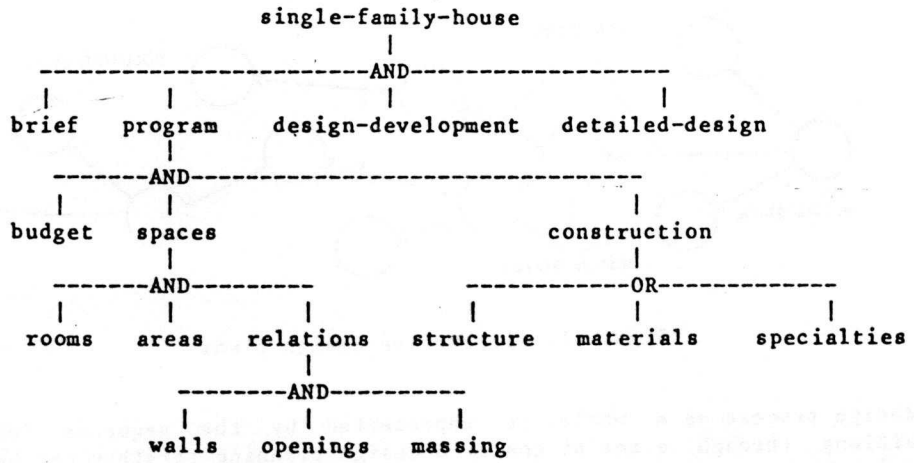

Figure 2: Goal hierarchy for the design of single family houses

Since goals are often defined at a high level of abstraction, it may not always be possible to directly generate a solution. In these cases, the goal can be decomposed into a set of more specific sub-goals, whose combined achievement will constitute achievement of the higher level goal. Sub-goals thus represent a decomposition hierarchy of design objectives, which form an AND/OR tree structure (such as the example depicted in Figure 2).

2.2 Transitions

The goals and sub-goals in this schema represent a model of the design process in terms of "milestones" that must be reached in order to bring it to a successful conclusion. According to this model, the method of transition between goals is analogous to the process of developing or following an overall design plan: a search strategy for traversing the goal network. The particular strategy which is used depends on the designer's judgment and preference, the system's knowledge base and the opportunities and weaknesses of the evolving solution.

Determining the method of generating a candidate solution for a given goal, or whether goal decomposition should be pursued to simplify the solution generation process, is achieved by the following algorithm (which is depicted schematically in Figure 3):

1. If the designer chooses to produce a solution, the system will present him with facts and data (e.g. plan, elevation, or contextual information) that are relevant to generating a candidate solution which achieves the current goal. The actions taken by the designer will be evaluated in order to determine their impact on other parts of the design solution, and to determine if the solution satisfies the constraints of the goal.

2. If the designer chooses to delegate to the system the production of a solution which achieves the current goal, an appropriate solution may be generated if sufficient information exists in the system's database and if the means for generating the solution exist in the system's knowledge-base. The method used by the system to generate a solution may be algorithmic or heuristic.

3. If the designer has delegated the production of the solution to the system, but the system is unable to comply due to lack of information or lack of knowledge, a sub-goal must be chosen for achievement, using the goal decomposition hierarchy as a "road map" for determining the next step in the search strategy.

Once a goal in the sequence has been achieved, the design process proceeds to achieve the next goal which will bring it closer to completion. The decision as to which goal to pursue next is also subject to task allocation between the designer and the system. The designer may choose any goal based on his experience and training, or on arising design opportunities. Alternatively, the system may be assigned the task of choosing the next goal. In this case,

it may rely on either pre-defined or dynamic plans. Pre-defined goal sequences, representing existing alternative design strategies, can range from extremely specific to very general plans. Plans for given problems can be selected and then "filled-in" dynamically, dependent on the particular context (18).

A dynamic search strategy is pursued by following a means-end analysis to choose the next goal in the network. By this process a goal is chosen according to the current status of the design solution and the available actions that will advance it closer to the conclusion of the design process. The system compares the current design solution to the pre-conditions for all solution generators that are applicable at the current phase of the design process, and determines their expected post-conditions. The system can then decide which actions are most likely to advance the process towards its conclusion, and hence which goal to pursue next.

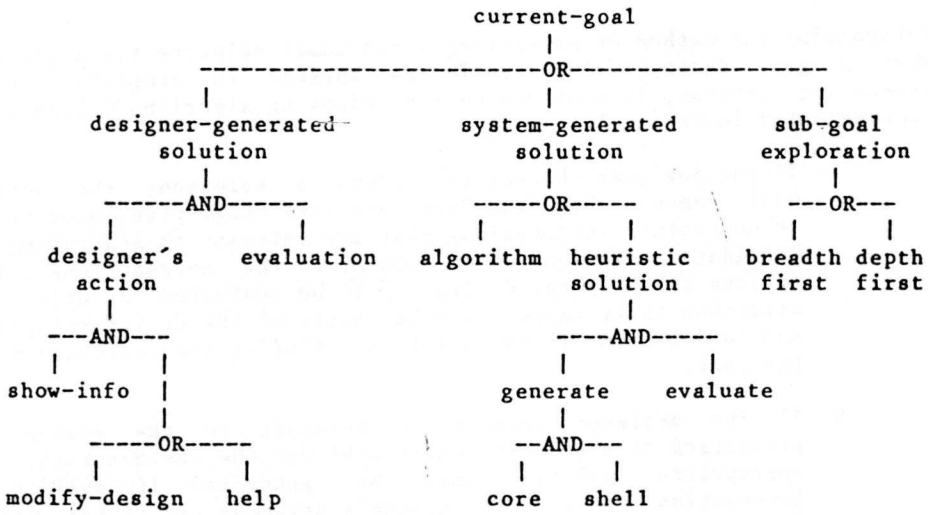

Figure 3: The schema of the design process control algorithm

Evaluation of design solutions that are generated as the process unfolds, and exploration of the impact each action has on previously generated solutions, renders the entire search process iterative, but not repetitive. Each action adds some information to the evolving design database, which may or may not cause goals that were achieved earlier to become dis-achieved, thereby necessitating their re-achievement (or their modification). When these goals are reconsidered, the circumstances for their achievement have changed due to addition of information and constraints to the design database by intermediate design actions. The designer may choose to reconsider these goals immediately, or to temporarily ignore them with the expectation that some further design developments will re-achieve them. Alternatively, the designer may modify or relax the constraints of certain goals, thereby further enhancing the flexibility of the process. Such modifications may

reflect insight into formerly unrecognized opportunities and pitfalls, gained as the design process evolves. Such opportunistic planning ensures creativity by allowing the process to be instigated by changes which are noticed as the solution progresses (19).

3. IMPLEMENTATION

The implementation of the goal hierarchy and the design process control mechanism utilize PROLOG's strong commitment to trees as its basic data structure, and its ability to reason about unknown objects, provided they satisfy certain equalities and inequalities (20).

Each goal is implemented by a frame structure, as depicted in Figure 4. This form of knowledge representation accounts for both the descriptive knowledge which is associated with each goal (constraints), and its procedural knowledge (relative weighting functions that facilitate the process of preferring some performances of a candidate design solution over others). This representation can accommodate modification and addition of constraints and weighting functions as the system's knowledge-base expands.

```
goal ( GOAL-ID,                    constraint ( CRITERIA-ID,
       CRITERIA-LIST,                           PRE-CONDITIONS,
       WEIGHTING-FUNCTIONS ).                   WEIGHTING-FUNCTIONS,
                                                EVALUATION-PROCEDURE,
                                                CRITERIA-STATUS ).
```

Figure 4: Goal and constraint frames

The constraints are represented in logical groups outside the specific goals that use them. Several goals may use the same constraints, whose satisfaction need not be re-established for each goal separately. The respective importance of satisfying each constraint is determined by the goals, through their procedural knowledge, which provides the particular context for constraint evaluation. The constraints thus represent the expected performances of the design solution at all times, much like vital statistics (which are used by multiple specialists) represent the status of a patient in the hospital.

Evaluators use the database query facilities to extract the information they need to perform their function, triggered through PROLOG's inference mechanism and by the constraints of the goal which the solution attempts to achieve. The information that is available in the database is usually augmented with rules-of-thumb and default values, which enable the evaluators to function in many levels of design development, and reduce the number of evaluators needed for establishing the satisfaction of individual constraints.

The system's solution generators operate in a manner similar to the evaluators, but they modify the database in addition to querying it. Examples of such system generators include space allocation, area distribution and layout of mechanical subsystems.

The design process as a whole is monitored and regulated by a system module called the Design Process Controller (DPC), which assumes control whenever the designer explicitly forfeits the option to generate a candidate solution or to choose the next goal. A PROLOG pseudo-code implementation of the DPC mechanism is shown in Figure 5.

4. APPLICATION

To illustrate the operation of the DPC and the process as a whole, the search for a particular solution that achieves the PROGRAM goal for a single family house as depicted in Figure 2 will be described. The following examples are taken from a prototype system which has been implemented to verify the methodology (21).

In most cases, the PROGRAM goal is chosen after the BRIEF goal (which establishes a base of client data) has already been achieved. For single family houses, the solution to the BRIEF goal is often as simple as a statement by the client such as "I want a 4 bedroom house for $125,000."

This brief constitutes a basis for defining the parametric constraints of the PROGRAM goal. The complexity of this goal leads to consideration of its sub-goals, which include BUDGET, SPACES, and CONSTRUCTION. Since the budget set by the client while achieving the BRIEF goal is not contradicted by other information in the database, the BUDGET goal is already achieved. If the designer chooses the SPACES goal to be achieved next, it is expanded into the sub-goals of ROOMS, AREAS, and RELATIONS. The ordering of sub-goal sequences may or may not be fixed, depending on their pre-conditions. For example, ROOMS must be achieved before AREAS or RELATIONS can be assigned, but these two can be encountered in any order.

Precedent suggests that a 4-bedroom, $125,000 house in the northeastern USA will have an area of approximately 2000 square feet, and include 2 bathrooms, a kitchen, a living room, a dining room, a basement, a garage, and perhaps a family room. The ability to make such inferences resides in the system's knowledge-base, and constitutes part of its "design experience." Using this knowledge to achieve the ROOMS goal, the DPC then presents the user with the AREAS goal. As the user inputs areas for each room the system will evaluate them against minimum functional requirements. If the available area, based on budget and cost per square-foot, is depleted, the system will respond with several options including reducing certain areas, increasing the budget, or reducing construction costs. These interactions suggest the relations and trade-offs which must be made between goals.

Having achieved the AREAS goal, the RELATIONS goal is selected next. Its satisfaction can be achieved by a space allocation algorithm (based on conventional adjacency and privacy constraints between rooms), or by user interaction. Both methods will result in the generation of a schematic layout of walls and openings, and may introduce multi-level massing to optimize the grouping or the separation of rooms.

```
/*  ************************  */
/*  design process controller  */
/*  ************************  */

goal(G) :- user-generated-solution(G).
goal(G) :- system-generated-solution(G).
goal(G) :- sub-goal-expansion(G).

user-generated-solution(G) :-
        show-current(G),
        modify-solution(G),
        evaluate-user(G).

/*  ************************  */
/*  goal specific procedures  */
/*                            */
/*    show-current(G)         */
/*    modify-solution(G)      */
/*    help(G)                 */
/*  ************************  */

evaluate-user(G) :- evaluator(G).
evaluate-user(G) :- user-advice(G), !, fail.

system-generated-solution(G) :- algorithmic(G).
system-generated-solution(G) :- heuristic(G).

algorithmic(G) :-  algorithmic-generator(G), show-current(G).

heuristic(G) :-
        heuristic-generator(G),
        show-current(G),
        evaluate-system(G).

/*  ************************  */
/*  goal specific procedures  */
/*                            */
/*    algorithmic-generator(G) */
/*    heuristic-generator(G)  */
/*  ************************  */

evaluate-system(G) :- evaluator(G).
evaluate-system(G) :- system-advice(G), !, fail.

/*  ************************  */
/*  goal specific procedures  */
/*                            */
/*    evaluator(G)            */
/*    system-advice(G)        */
/*    user-advice(G)          */
/*  ************************  */
```

Figure 5: A PROLOG pseudo-code implementation of the DPC

With the achievement of the RELATIONS goal the process proceeds to achieve the CONSTRUCTION goal, using the information generated from the SPACES goal and conventions that are stored in the system's knowledge-base. If the conventions are accepted by the designer - the PROGRAM goal is achieved. If, on the other hand, the designer intervenes by specifying materials and specialties (e.g. a fireplace) that exceed the default square-foot cost, constraints that were satisfied earlier may no longer be satisfied. For example, the increased cost will dis-achieve the formerly achieved BUDGET goal, which will have to be modified (by increasing the total budget) or else will trigger changes in the SPACES and/or CONSTRUCTION goals, such as elimination of certain rooms (eg. the family room), reduction in size of other rooms, changing the number of floors, or specifying less expensive materials.

5. CONCLUSION

In this paper we have presented a knowledge-based paradigm for design which is intended to promote computers to the role of partners in the design process. A design process involving the search for candidate design solutions has been modeled as a problem-solving process composed of states and transitions. States represent design objectives (goals) in various levels of abstraction and transitions represent the means for choosing and achieving goals. Dynamic allocation of design tasks between the designer and the system is proposed as a means to computationally model the design process, while providing for flexible responses to unforeseen situations arising as the search for a design solution evolves.

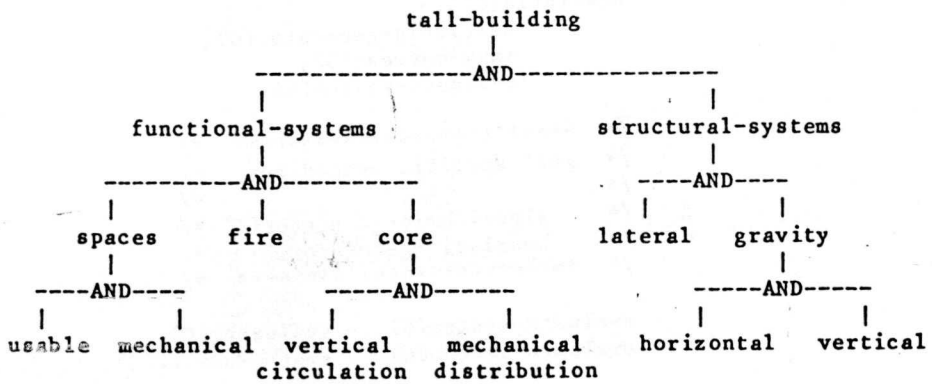

Figure 6: A goal hierarchy for the design of tall buildings

A PROLOG implementation of the methodology has been used to illustrate the theory, which was exemplified through the achievement of a specific goal sequence in the design of single family houses. The application of the methodology to other design tasks is a matter of substituting the appropriate knowledge-base, a task which is akin to filling a "shell" of a generalized

expert system with domain specific knowledge (22). For example, substituting the goal hierarchy for the design of a single family house with the goal hierarchy (and associated knowledge) depicted in Figure 6 will lead to a system for the design of tall buildings.

Although the application that was discussed (and our particular area of interest) concerns architectural design, we believe that the design paradigm described in this paper is applicable to decision-making processes of many kinds, and may prove to be an alternative computational model of design. Furthermore, we believe that the potential of CAD systems based on this methodology for assisting in the design of physical artifacts and environments may lead to an improvement in designers' productivity and product quality.

6. REFERENCES

(1) Y. Kalay, "Redifining the role of computers in architecture: from drafting/modeling to knowledge-based assistants", Computer-Aided Design, Vol. 17, No. 7, September 1985, pp 319-328.

(2) J. Orr. "The merits of design automation", Computer Graphics World, January 1985 pp 83-84.

(3) A. Newell and H. Simon, Human Problem Solving, Prentice-Hall, Englewood Cliffs, NJ, 1972.

(4) C. Eastman, "Recent developments in representation in the science of design", Research Report No 83, Institute of Physical Planning, Carnegie-Mellon University, Pittsburgh PA, April 1981.

(5) W. Mitchell, "The logic of architecture", Draft, Carnegie-Mellon University, Pittsburgh PA, 1980.

(6) O. Akin, "How do architects design?", Artificial Intelligence and Pattern Recognition in Computer-Aided Design, Latombe, ed., IFIP, North Holland Publishing Co., 1978.

(7) D. Koberg, and J. Bagnall, The Universal Traveler, William Kaufmann, Inc., Los Altos, California, 1974.

(8) T. Maver, "Simulation and solution teams in archtectural design", Design Participation, Nigel Cross, ed., Academy Editions, London, 1972, pp. 79-83.

(9) D. Schon, The Refective Practitioner - how professionals think in action, Basic Books Inc., New York, 1983.

(10) E. Feigenbaum and J. Feldman, eds. Computer and Thought, McGraw Hill, Inc., New York., 1963.

(11) J. Haugeland, ed. Mind Design: philosophy, psychology, artificial intelligence, Bradford Books, Montgomery, Vermont, 1981.

(12) N. Cross, The Automated Architect, Pion Limited, London, 1977.

(13) J. Gero and R. Coyne, "Knowledge-based planning as a design paradigm", Working Paper, Computer Applications Research Unit, University of Sidney, NSW, Australia, 1985.

(14) W. Mitchell, "The theoretical foundations of computer aided architectural design", Environment and Planning B, Vol. 2, 1975, pp. 127-150.

(15) N. Negroponte, The Architecture Machine, MIT Press, Cambridge, Mass., 1970.

(16) H. Simon, The Sciences of the Artificial, MIT Press, Cambridge, Mass., 1969.

(17) M. Maher, "HI-RISE: a knowledge-based expert system for the preliminary structural design of high rise buildings", Ph.D. Dissertation, Carnegie-Mellon University, Pittsburgh PA, 1984.

(18) P. Friedland, "Knowledge-based experiment design in molcular genetics", Ph.D. Dissertation, Report No 79-771, Computer Science Department, Stanford University, 1979.

(19) B. Hayes-Roth and F. Hayes-Roth, "Cognitive processes in planning", Report No R-2366-ONR, Rand Corp., Santa Monica, California, 1978.

(20) A. Colmerauer, "Prolog in 10 figures", Communications of the ACM, Vol. 28, No, 12, December 1985.

(21) Y. Kalay, A. Harfmann and L. Swerdloff, "ALEX: a knowledge- based architectural design system", ACADIA Workshop '85 Proceedings, P. McIntosh, ed., 1985.

(22) B. Buchanan and E. Shortliffe, Rule-Based Expert Systems, Addison-Wesley Publishing Co., Reading Mass., 1984.

Yehuda E. Kalay
Anton C. Harfmann
Lucien M. Swerdloff

Computer-Aided Design / Graphics Laboratory
School of Architecture and Environmental Design
State University of New York at Buffalo
Buffalo, New York, 14214 (USA)

Tel. (716) 831-2291