INTERNATIONAL JOURNAL OF ELECTRONICS; MECHANICAL and MECHATRONICS ENGINEERING Vol.2 Num.4 pp.(335-343)

COMPUTABILITY OF DESIGN BY MEANS OF ITS COGNITIVE CONTENT

Derya Gulec OZER[1]

[1]Istanbul Aydin University, Engineering- Architecture Faculty
Architecture Department

E-mail: deryagulec@aydin.edu.tr

Abstract-*Design activities based on intuition and induction cannot, presently, be substituted by computational means. Computers can only assist designers by providing examples, precedents, case studies, prototypes and their derivatives adapted specifically to the context of the problem. However choosing and adapting the appropriate solution remains a human prerogative. The failure to automate design as a whole is due, in a large measure, to the difficulty of finding computational means that can support learning, creativity, and judgment, which comprise much of the cognitive aspects of design. Computer support for innovative design must overcome the problem that designers necessarily make extensive use of situated* ***tacit*** *understanding while computers can only store and display explicit representations of information. This paper aims to understand tacit to explicit knowledge transformations in computed aided architectural design process.*

Keywords: *Tacit / explicit knowledge, design knowledge, design process, computability of design, knowledge based computer aided design*

1. INTRODUCTION

Architectural Design is much more integrated with computer, than it was twenty years ago. Computation directs the whole part of the process. But this method brings us many problems to deal with. One of the most important issue is how to tackle with the tacit / explicit knowledge transformations. There has been many researchers pointing out this problem. Gerry remarked that computer support for innovative design must overcome the problem that designers necessarily make extensive use of situated tacit understanding while computers can only store and display explicit representations of information. The automation techniques used for routine design are not applicable: techniques are needed to support creative, tacit human understanding with explicit computer representations. [Stahl, p.3] Likewise, in their book "Knowledge-Based Computer-Aided Architectural Design" editors Carrara and Kalay pointed out that design activities based on intuition and induction cannot, presently, be substituted by computational means. Computers can only assist designers by providing examples, precedents, case studies, prototypes and their derivatives adapted specifically to the context of the problem. However choosing and adapting the appropriate solution remains a human prerogative [Carrara, Kalay, p.151]. They adress the difficulty of finding computational means that can support *learning*, *creativity*, *and judgment*, which comprise much of the cognitive aspects of design.

Cognition in architectural design process is the central issue for computation, pointing out in this paper. Challenge for understanding the many facets of design has been the target in attempting to computationally define design processes and knowledge. [Kalay, Swerdlof, p. 47]

In means of computation, the most used tools are CAD programs in architecture. From education to professional life their extended use shapes the comtemporary architecture. But while these programs exhibit remarkable levels of compactness, efficiency, sophistication, and power unthinkable only a few years ago, they fail to support the cognitive aspects of design. In fact, the use of current CAD tools forces architects to use a more precise and systematic mode of design, which follows the logic and methodology offered by available software, rather than a model more suitable for the non-deductive,

often irrational and not easily computable architectural design process [Carrara, Kalay, p.393].
Nigel Cross in his book "Designerly Ways of Knowing" also points out the importance of cognition in architectural design. He mentions where the goal is to develop interactive systems that support designers, then knowledge of the human designer's cognitive behaviour obviously is of fundamental importance, because the users of the interactive systems (that is designers) must be able to use them in ways that are cognitively comfortable. So the systems must be designed on the basis of models of the cognitive behaviour of the system uses. [Cross, p.40]

2. KNOWLEDGE FOR ARCHITECTURE

In order to deal with knowledge representation in computation, first it would be useful to understand the process of architectural design. The process of architectural design aims to define a physical form that will achieve certain functional and behavioral objectives in a particular context. It comprises three distinct, but highly interrelated, operations:
Definition of the desired objectives
Production of alternative design solutions
Evaluation of the expected performances of the solutions and comparing them to the predefined objectives. [Carrara, Kalay, p. 147]
Likewise Cross defines 5 aspects of designerly way of thinking as follows:
Designers tackle ill defined problems.
Their mode of problem solving is solution focused.
Their mode of thinking is constructive.
They use codes that translate abstract requirements into concrete objects.
They use these codes to both read and write in object languages. [Cross, p. 12]
According to Mitchell, It is useful to regard architectural design as a special kind of problem-solving process, and to discuss design within the framework of a general theory of problem-solving. The view of problem-solving that will be introduced is one which has gained wide currency in recent years. It assumes that we can construct some kind of a representation of a system that interests us, and that problem solving can be characterized as a process of searching through alternative states of the representation in order to discover a state that meets certain specified criteria. This view is not without its limitation, but it provides an appropriate starting point for discussion. [Mitchell, p.27]

2.1. Explicit and Tacit Knowledge In Design

Knowledge can be classified broadly as either explicit or tacit. The dictionary (Merriam-Webster, 1991) provides the following definitions:
Tacit knowledge; expressed or carried on without words or speech; implied or indicated but not actually expressed.
Explicit knowledge; fully revealed or expressed without vagueness, implication, or ambiguity; leaving no question as to meaning or intent; verbal plainness and distinctness such that there is no need for inference and no room for difficulty in understanding.
Explicit knowledge consists of facts, rules, relationships and policies that can be faithfully codified in paper or electronic form and shared without need for discussion. By contrast, tacit knowledge (or intuition) defies recording. This kind of knowledge underlies personal skill, and its transfer requires face-to-face contact or even apprenticeship. [Wyatt, p.6]

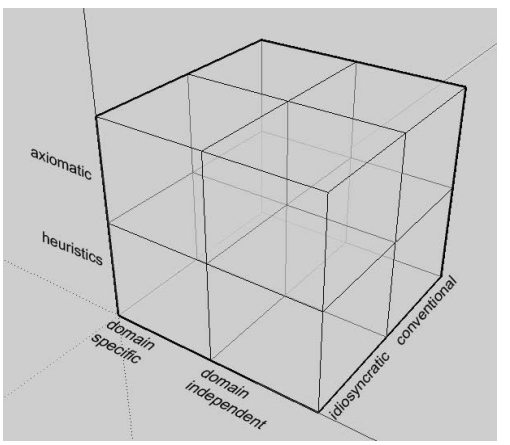

Figure 1. Three of six categories of knowledge as a three-dimensional space [Coyne, p.141]

Table I. Comparison of strategies to manage explicit and tacit knowledge

Comparison of strategies to manage explicit and tacit knowledge [Hansen]		
	Codification for explicit knowledge (people to documents)	**Personalization for tacit knowledge (people to people)**
Intended result for organization	Uniform, high quality solutions tto most problems; contain current risks and costs	Unique, appropriate, creative solutions to strategic problems; exploit opportunities and contain future costs and risks
Type of problem targeted and solution preferred	Routine, short-term, low-risk problem for which a good enough solution is available but is not usually applied	One-off, medium to long-term, high risk, strategic problem with no precedent needing a novel, customized solution
Knowledge management goal	Re-use of explicit knowledge by capturing, codifying, classifying and making available knowledge to support routine problem solving	Sharing of tacit knowledge by helping staff to identify relevant experts and enhance conversations to create novel solution

3. UNDERSTANDING INTERPRETATION: THREE METHODOLOGIES IN DESIGN

The process by which designers transform their **tacit** preunderstanding into **explicit** knowledge is termed "*interpretation*" [Stahl, p.3]. To say that interpretation is central to innovative design is to stress that in order to design the designer must to some degree understand and be able to articulate the significance of the artifact being designed. All of this takes place primarily in *tacit* ways. However, one's tacit understanding of something can be partially articulated or expressed *explicitly* in spoken, written, or graphical language—either to deepen one's own understanding or to communicate with others. Two aspects of the process of interpretation can be distinguished:
There is a tacit *preunderstanding* based on previous background knowledge; items from this preunderstanding can be articulated explicitly.
There is the possibility of revising that preunderstanding based on *discoveries* that are opened up by it. [Stahl, p.10]
The process of understanding in design has the following three features:
Understandings of a design arise from interactions with the *situation* of the task in the world;
The designer's unique interpretive *perspectives* grow out of traditions which pass on viewpoints for relating to the world, skills for behaving in the world and languages for talking about the world; and
Explicit articulations of interpretations in *language* emerge from situated, tacit understanding and then re-submerge.

Table II. The structure of human interpretation [Stahl, p.12]

	(a) situated	(b) perspectival	(c)linguistic
(1)preunderstanding	expectations	focus	conceptualization
(2)discovery	surprises	deliberations	refinements

Table III. Computer-based mechanisms to support interpretation in design [Stahl, p.14]

	(a) situated	(b) perspectival	(c)linguistic
(1)reuse	hypermedia network	perspectives mechanism	end-user language
(2)plasticity	revising representations	merging multiple perspectives	defining new expressions

Although computers cannot understand things the way people do, they can serve as a computational medium to support people's interpretive processes. The computer support mechanisms listed in Table 2-3 can augment

cooperative design in a number of ways, including:
a-1 As a long-term memory or repository for information that was created in past designing and is now available to be shared by designers using the repository.
a-2 As an external memory for representing and revising designs to see how alternative variations appear.
b-1 As a retrieval mechanism for organizing and managing design knowledge and filtering through just what is relevant.
b-2 As a display mechanism to define new personal and shared views of designs.
c-1 As a linguistic medium for expressing knowledge in a canonical form that can be used for computations by the software.
c-2 As a communication medium to generate new knowledge to be shared with others.
A comparison of Table 2-1 and Table 2-2 shows that *the mechanisms of computer support are based on the structure of unaided human interpretation*. The computer support is intended to extend the power of designers to operate under conditions of "information overload," in which it is becoming increasingly difficult to work effectively without the use of computers. [Stahl, p. 15]
Stahl points out in his dissertation the insights of three people who have provided insightful and influential interpretations of the design process: Christopher Alexander, Horst Rittel, and Donald Schön. Significantly, each has been concerned at some point with the issue of providing computer support for design. Also, they emphasize the themes of this paper: Alexander focuses particularly on the problem of representation; Rittel emphasizes the consequences of people's differing perspectives; and Schön is concerned above all with how explicit reflection arises from tacit understanding. *Alexander* recognizes the need to combine mathematical methods and analysis of patterns with intuitive sense grounded in architectural practice [Alexander, 1964]. In pushing the paradigm of objective analysis as far as he can, he is nevertheless frank about the limits of empirical research and the importance of prioritizing human needs that are less susceptible to empirical evaluation. Finally, the pattern language he proposes is meant as a basis for every culture and every person to build their own unique and appropriate representations of design situations. *Rittel*'s analysis of the "wicked" problems of design does not suggest the elimination of method in favor of arbitrary personal whim. Rather, it stresses the complexity of continually framing the problem and solving it in parallel. One's interpretation of the problem must not only be based in the specifics of the situation, but must also grow out of the exploration of potential solutions. The argumentative process of design is not simply one in which everyone is entitled to their own opinion. Rather, it is a process in which initial prejudices are supposed to be subjected to critique from other viewpoints so that they will be refined. At the same time, Rittel recognizes that people have differing perspectives for various legitimate reasons, and that agreement will not always be possible even with the best processes of deliberation [Rittel & Webber, 1973].
Schön can be seen as a resolution of the objective and subjective approaches, for he stresses the interplay or dialogue between the designer (who brings tacit skills and personal perspectives) and the materials of a design situation (which provides surprises for the moves of the designer that could not have been anticipated but that constrain the design) [Schön, p. 52]. Schön's theories about the roles of tacit knowing and explicit reflecting, drawing upon important philosophical sources, flesh out both Alexander's notion of intuition and Rittel's sense of how judgments can be deliberated. Schön's theory of design focuses on the movement between the designer's skillful preunderstanding ("knowing-in-action") and explicit articulation ("reflection-inaction"). This is precisely the movement that is called interpretation.

3.1. Alexander: The Structure of Design Situation

Deliberation on the question of whether and how computers should be used to support the work of designers has raged for several decades. In the beginning of the 1960's Alexander (1964) pioneered exploration of this possibility by running a series of computer programs for the hierarchical decomposition of systems into subsystems, diagrams, or patterns. This kind of decomposition was central to the methods he proposed for design, and it seemed logical and necessary to use computationally powerful

equipment to implement such analysis. However, within several years, Alexander was discouraged about the use of computers to support design. He complained that, "the people who are messing around with computers have obviously become interested in some kind of a toy. They have definitely lost the motivation for making better buildings" (Alexander, 1971, p.309). In his 1971 Preface to the paperback edition of his original work, he characterized the problem with attempts at computer support in terms of a broader problem of separating the study of design methodology from the practice of designing (Alexander, 1964).

The issues surrounding the appropriate use of computers go to the heart of what design is and should be. In his now classic *Notes on the Synthesis of Form*— which presents his dissertation work incorporating the early computer programs— Alexander reviews the history and even the prehistory of design in order to argue that the field reached a second watershed in the mid-twentieth century. The profession of design had originally emerged when society started to produce new needs and innovative perspectives too rapidly to allow forms to be developed through "unselfconscious" activities of slowly evolving traditions. Now, the momentum of change has reached a second qualitatively new stage: Today more and more design problems are reaching insoluble levels of complexity. (Alexander, 1964, p.3)

Alexander's patterns provide the representational or computational basis today for computerization. In an obvious sense, computers are a natural tool for storing large amounts of information. But at a deeper level, computer languages and applications are designed to manage complexity. It is no coincidence that the movement toward structured programming was contemporaneous with Alexander's emphasis on functional decomposition.

Alexander saw a major advantage of the systematic use of structures or patterns in what he referred to as a "loss of innocence". Recognizing the power of both formal representations and non-formalizable tacit knowledge, he did not propose that design methods substitute for the practice of design or for the designer's practical intuitions. Rather, he recognized that intuition and rationalism were equally necessary, and argued for a proper balance: "Enormous resistance to the idea of systematic processes of design is coming from people who recognize correctly the importance of intuition, but then make a fetish of it which excludes the possibility of asking reasonable questions". Alexander felt that the fetishism of intuition as some kind of inalienable artistic freedom of the designer functioned as a flimsy screen to hide the individual designer's incapacity to deal with the complexity of contemporary design problems. While computers may be necessary to manage this complexity, the tacit knowledge of human designers must also be brought to bear with their intuitions.

3.2. Rittel: Deliberating From Perspectives

When Rittel declared in his *Dilemmas in a General Theory of Planning* that "planning problems are inherently wicked" (Rittel & Webber, 1973, p.160), he thereby spelled out that characteristic of planning and design tasks that has subsequently become the central source of perplexity in trying to imagine a computer system that can effectively support the challenging aspects of design. Computer programs have traditionally been devised in accordance with the classical example of *tame* science and engineering problems—precisely the paradigm that Rittel argued is not applicable to the problems of open societal systems with which planners and designers are generally concerned. This inadequate approach assumes that a problem can first of all be formulated as an exhaustive set of specifications. Then, based on such a problem statement, possible solutions can be evaluated to see which are optimal solutions to the problem. Computer programs based on this paradigm must represent in advance the space of problems and solutions for a well-defined type of design problem in an explicit, comprehensive, and non-controversial (objective) manner. However, as Rittel points out, in order to program such a computer system, you would have to anticipate all potential deontic judgments ahead of time before the machine could run. But if you did that you wouldn't need the computer because you would have had to have thought up all the solutions ahead of time. Therefore it is almost ridiculous to claim that there will be a

designing machine if design is thought of in this sense. (Rittel, 1972, p.323)

Rittel claimed that the wicked problems of planning could not begin to be understood in the first place until one had already started to explore directions for solutions. He described what Heidegger calls the *hermeneutic circle* of understanding when he argued, "that you cannot understand the problem without having a concept of the solution in mind; and that you cannot gather information meaningfully unless you have understood the problem, but that you cannot understand the problem without information about it" (Rittel, 1972, p.321). Suppose, for instance, that you are asked to plan a mission to the moon for four astronauts for a period of 45 days. According to NASA, the purpose has been specified as: to explore long-term stays for crews of international backgrounds and mixed gender and to conduct some scientific research and some site work to prepare for future moon bases. In thinking about the design of the lunar habitat for this mission, you might begin to discuss the importance of privacy issues with other people on your design team. You might feel that not only was some physical privacy needed for cultural reasons, but psychologically there would be a need to structure a careful mix of public and private spaces and opportunities. These privacy issues might become paramount to your design even though they had not been included in the original problem statement. In this way, the set of issues to be investigated and concerns to be balanced would emerge and evolve as the planning process took place. Your ability to interpret the problem as one of privacy would have been based on your tacit preunderstanding of privacy as part of human life.

Computer systems may be useful for storing, organizing, and communicating complex networks of argumentation—as long as they do not stifle innovation by imposing fixed representations of the ideas they capture or limiting diversity of interpretive viewpoints. Computer support for planning and design processes as Rittel conceived of them must allow team members to articulate their individual views and judgments, to communicate these to each other, and to forge shared perspectives. It must support deliberation or argumentation. Rittel concluded that the proper role for computers and information systems generally is that of an enhancer of natural (human) intelligence, not an artificial substitute for it. In *Designing Crutches for Communication* (Kunz & Rittel, 1984), he uses the image of prosthetic devices like crutches or eye glasses: "The glasses do not see instead of you, or on your behalf. Neither does the automobile relieve you from traveling. They are prosthetic devises which support, reinforce, enhance some capacity or activity". Because the role of information science is not to automate problem-solving but to augment human problem-solving, it must be based on an analysis of how people use information and solve their problems: "Here lies the central task of information science: to develop methods for exploring its users' knowledge and their modes of reasoning, i.e., the systems analysis of problem solving and information". Given Rittel's view of design as argumentation from perspectives, this means computers should support people's perspectival interpretation processes.

3.3. Schön: Tacit Knowing and Explicit Language

Schön argues in his seminal work, *The Reflective Practitioner* (1983), that much design knowledge is tacit, rather than being rule-based. He views the design process as a dialogue-like interaction between the designer and the design situation, in which the designer makes moves and then perceives the consequences of these design decisions in the design situation (e.g., in a sketch). The designer manages the complexity that would be overwhelming if all the constraints and possibilities were formulated as explicit symbolic rules by using professionally-trained skills of visual perception, graphical sketching, and vicarious simulation. Note that these skills bypass the process of analyzing everything into primitive elements and laying it out in words and propositions.

Schön recently addressed the question of computer support for design in an article descriptively entitled *Designing as Reflective Conversation with the Materials of a Design Situation* (Schön, 1992). He argued for a necessarily limited role for computers in design because one of the most important things that designers do is to construct the design situation itself. Not only is this something that computers cannot do by themselves, but it also precludes programmers

of computer systems from predefining a generic design situation for the computer, prior to the involvement of the designer with the task.

4. COMPUTER SUPPORT FOR ARCHITECTURAL DESIGN

4.1. Supporting Situated, Perspectival, Linguistic Interpretation

The analysis of interpretation mentioned before suggests that computer support for design should:

- Capture computer representations of tacit situated understanding at the points when it becomes articulated as explicit interpretations.
- Provide multiple perspectives for analyzing and understanding designs.
- Allow users to evolve and refine interpretive expressions in language without starting from scratch or accepting predefined frameworks.

Accordingly, three hermeneutic principles will be adopted in trying to develop computer-based environments to support the work of designers:

- Provide facilities so designers can create representations of the design *situation* during the process of solving the task.
- Provide facilities so designers can define multiple interpretive *perspectives* on design problems.
- Provide facilities so designers can articulate explicit conceptualizations in *language* expressions for their work and submerge this new knowledge into tacit forms of knowledge for future use.

Therefore, a computational method in design must feature;

- An extensible *computational medium for representing and evolving* artifact constructions, design rationale, computational critics, and other forms of design knowledge.
- A mechanism for sharing *group and personal interpretive perspectives* to support collaboration and deliberation.
- A *language for explicitly defining computations* and for hiding information that can then function in a tacit way.

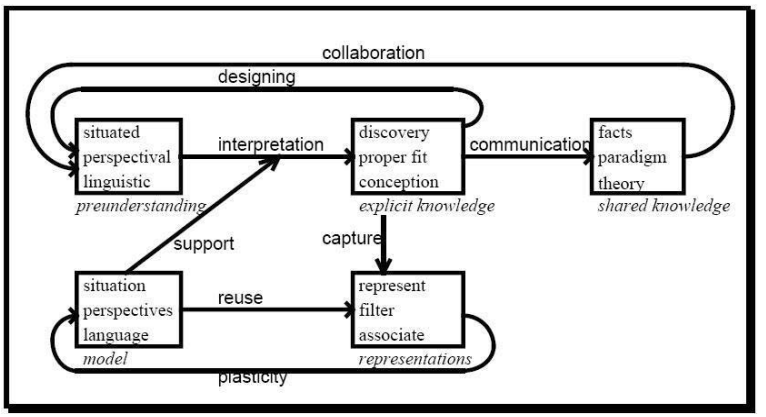

Figure 2. Computer support for interpretation in design. [Stahl, p.220]

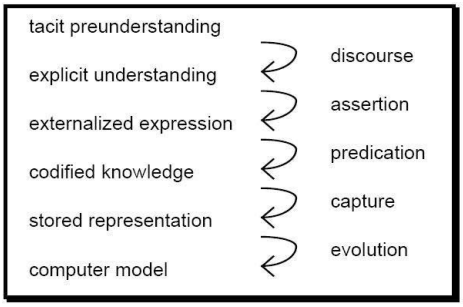

Figure 3. Successive transformations of knowledge.
The left-hand column lists consecutive forms of information. The right-hand column indicates the transformation processes from one form to another.[Stahl,p.200]

5. CONCLUSION

The key concept for a theory of computer support is interpretation. Support for interpretation is the ingredient missing from most traditional AI programs.
Knowledge-based system design inevitably raises the question of the nature of knowledge. First, the varieties of knowledge or information have been categorized in terms of their origins in various phases of the process of interpretation. This includes not only tacit and explicit understanding, but also shared understanding and captured computer representations. Second, the idea of domain knowledge has been critiqued. Not only does knowledge in a design domain change as the related technologies and styles change and as the expertise of the field matures and grows, but every designer and every design team has their own domain knowledge. It is not simply that they each have different pieces of an underlying knowledge. Rather, to know is to know from a perspective, so *there is no objective body of domain knowledge independent of what people know in their own ways*, within their many perspectives. Third, the role of language in expressing knowledge has been emphasized. [Stahl, p.386] *The emergence of interpersonal or operationalized knowledge from tacit experience takes place through discourse and assertion* within situated interpretation. Correspondingly, an end-user language has an important role to play in computer support.

ACKNOWLEDGEMENT

This paper is developed within ARCH 626-Issues in Architectural Research course in METU. I would like to thank course instructor Prof. Dr. Zeynep Mennan for her contributions.

6. REFERENCES

[1]. Alexander C , *Notes on the Synthesis of Form*. Cambridge: Harvard University Press,1964.

[2]. Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl-King I, Angel S , *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press,1977

[3]. Rittel H, Webber M, "Dilemmas in a General Theory of Planning", *Policy Science*. 4, 155-169.,1973

[4]. Schön D , *The Reflective Practitioner*. New York: Basic Books,1983
Schön D, The Design Studio. London: RIBA Publications, 1985.

[5]. Bronowski, J., *The origins of knowledge and imagination*, Yale University Press, New Haven, Connecticut,1978.

[6]. Eastman, C.M., McCracken, W.M.(eds.), *Design Knowing and Learning: Cognition in Design Education*, W.C. Newstetter, Elsevier, 2001.

[7]. Carrara,G., Kalay,Y.E. (eds.), *Knowledge Based Computer-Aided Architectural Design*, Elsevier, Netherlands, 1994.

[8]. Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., Gero, J.S., *Knowledge Based Design Systems*, Addison Wesley, USA, 1989.

[9]. Cross,N., "Designerly Ways of Knowing: Design Discipline versus Design Science", *Design Issues*, 17(3)49-55, 2001.

[10]. Cross,N., *Designerly Ways of Knowing*, Springer, Germany, 2006.

[11]. Cross,N., Christiaans, H., Dorst, K.,(eds.) *Analysing Design Activity*, Wiley, Great Britain, 1996.

[12]. Hansen M.T., Nohris N., Tierney T. "What is your strategy for managing knowledge?" *Harvard Bus Rev* 1999;77:106-16, 187

[13]. Kalay,Y.(ed.), *Computability of Design*, Wiley Science, New York, 1987.

[14]. Kalay,Y.,Swerdloff,L.,Majkowski,B., "Process and Knowledge in Design Computation", *Journal of Architectural Education*, 43(2)47-53,1990.

[15]. Kunz W, Rittel H , How to Know *What is Known: Designing Crutches for Communication. In Dietschmann HJ (Ed) (1984) Representation and Exchange of Knowledge as a Basis of Information Processes*. North-Holland: Elsevier. 51-60, 1984

[16]. Mitchell,W.J., *Computer Aided Architectural Design*, Van Nostrand Reinhold, New York, 1977.

[17]. Stahl,G., "Interpretation in Design: The Problem of Tacit and Explicit Understanding in Computer Support of Cooperative Design", *Phd Thesis, Faculty of Graduate School of the University of Colorado*, 1993

[18]. Wyatt, C. J., "Management of explicit and tacit knowledge", *Journal of the Royal Society of Medicine*, 94: January 2001.