PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 30/93

A Formal Approach to Design Based on Cognition

Bruno Feijó
Carlos J. P. Lucena
João Bento

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 30/93
Editor: Carlos J. P. Lucena December, 1993

A Formal Approach to Design based on Cognition*

Bruno Feijó
Carlos J. P. Lucena
João Bento**

* This work has been sponsored by the Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

** CMEST, Instituto Superior Técnico, Lisboa, Portugal

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio — Departamento de Informática
Rua Marquês de São Vicente, 225 — Gávea
22453-900 — Rio de Janeiro, RJ
Brasil
Tel. +55-21-529 9386 Telex +55-21-31048 Fax +55-21-511 5645
E-mail: rosane@inf.puc-rio.br
techrep@inf.puc-rio.br (for publications only)

A FORMAL APPROACH TO DESIGN BASED ON COGNITION

B.Feijó
Intelligent CAD Laboratory - ICAD
Pontifícia Universidade Católica do
Rio de Janeiro, SCT/PR
Fax: +21 511 5645
Email: bruno@inf.puc-rio.br

C.J.P.Lucena
Computer Science Department
Pontificia Universidade Catolica do
Rio de Janeiro, SCT/PR
Fax: +21 511 5645
Email: lucena@inf.puc-rio.br

J.Bento
CMEST
Instituto Superior Tecnico, Lisboa
Fax: +1 899242
Email: joao@civil2.ist.utl.pt

Abstract - Most of the current general design methodologies do not capture the cognitive nature of design. The present paper proposes an approach based on an evolving theory of the design process inspired on a cognitive interpretation of design. This approach provides a pragmatic guideline for developing and evaluating design methodologies used in connection with CAD and CASE systems. Furthermore, it seems that several current design models are interpretations of the proposed theory.

Keywords: design, CAD, CASE

Resumo - Atualmente a maior parte das metodologias gerais de *design* não captura a natureza cognitiva de *design*. Este artigo propõe um tratamento baseado em uma teoria evolutiva do processo de *design* inspirado em uma interpretação cognitiva de *design*. Este tratamento provê uma orientação pragmática para desenvolvimento e avaliação de metodologias de *design* usadas em sistemas de CAD e CASE. Além do mais, é sugerido que vários modelos atuais de *design* são interpretações da teoria proposta no presente trabalho.

Palavras-chave: design, CAD, CASE

1. INTRODUCTION

Many design models and design methodologies have been proposed during the last thirty years of work in design research. Cross (1984) contains one of the most complete collections of papers in design methodology. Jones' work (Jones 1963, 1970, 1977, 1981) is a remarkable example of a steady flow of contributions to that area. Some of the literature attempted to provide a cognitive interpretation of design. The work by Jeffries et alii (1981) and Akin (1979) exemplify this type of approach.

Despite all the above mentioned efforts progress in design automation has been kept behind expectations. This fact manifests itself in the proliferation of proposals which either do not address the complete process of design (they focus only sub-parts of the process such as preliminary design) or face design as a strict problem solving activity (e.g.: design as planning and design as constraint satisfaction).

The present paper proposes the basic ideas for a theory of the design process based on a cognitive interpretation of design. In fact, what is proposed here is a meta-theory, since it can be instantiated to become a theory of a particular design process (and associated methodology) when particular methods for synthesis, analysis and evaluation are provided.

The figure below (Figure 1) illustrates the phases involved in moving from a characterization of the nature of design to a Particular methodology and its associated CAD/CASE working system.

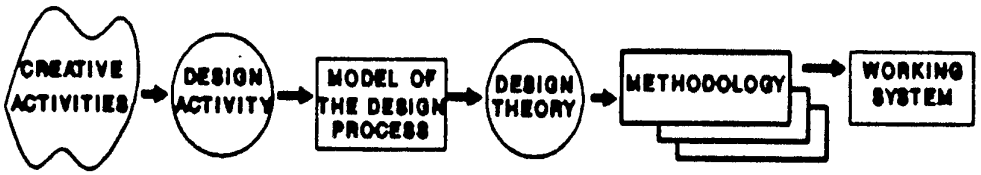

Fig.1 Design from abstractions to working systems

2. DESIGN ACTIVITY

In a sense, our work is a followup on the work by Goel and Pirolli (1989) who have proposed a characterization of the design problem space based on invariants of the design task environment. Firstly their characterization of design points out that design is not a ubiquitous

activity (e.g. playing chess is not a design activity). Secondly they characterize design activity as what Lakoff (1987) calls a radial category, that is a category in which a "central, ideal, or prototypical case exists as well as some unpredictable but motivated variations". Design activity as a radial category means that there is no chap boundary between what is and is not a design activity. thirdly they analyse the task environment of a few prototypical design professions to be able to describe the design task environment on the basis of the following set of invariants (here stated in a simplified way): *inv1*= Many degrees of freedom; *inv2* = Limited or delayed feedback from the world; *inv3*= Input consisting substantially of goals and intentions and output is a specification of an artifact; *inv4*= The artifact must function independentJy of the designer; *inv5* = The specification and dell very of the artifact are temporally separated; *inv6* = Costs are associated with every action; *inv7* = Answers are neither right nor wrong only better or worse; *inv 8*= Problems tend to be large and complex.

We adopt the above definition *of* design activity and consider it the class of problems to be modelled, i.e.

> *Def. A Design Activity is* a radial category of problems whose task environment is characterized by the set of invariants inv_i

3. A MODEL OF THE DESIGN PROCESS

Goel and Pirolli (1989) examined a number of designers at work to reconstruct their problem space and make an "explanatory connection" between the features evident in their problem spaces and two other invariants: the invariants of the design task environment *(inv.)* and the invariants of the information processing system (Pi). The invariants of the inflammation processing system are: *pl* = Limitations on the expressive power of the language of thought; p2= Sequential processing; p3= STM (Short Term Memory) capacity. They came up with a set of eight features that are "entailed" or "enabled" by the invariants envoy and Pi. Goel and Pirolli (1989) claim that these features are invariants in the problem spaces of design situations and collectively constitute a DPS (Design Problem Space). The notion of design by Goel and Pirolli (1989) is motivated within Newell and Simon's (1972) information-processing theory. The DPS invariants proposed by Goel and Pirolli (1989) are clearly inspired by the paradigm of problem solving.

We find that Goel and Pirolli's (1989) DPS invariants are a major empirical finding since it adds a new cognitive perspective to the current studies of the design process. However, we have a different view of them. i.e.: firstly we call them *characteristics of the Design Problem .Space* secondJy we simplify them (as presented in Table 1) and finally we see them as an informal model of the design process. Strictly speaking, we take them to be an informal interpret anon of a design theory (a *language* and a *deductive apparatus)* associated to it. All the axioms of this design theory

interpret to true in this interpretation (i.e. an interpretation C is a model for a theory if each axiom Ax_i of the theory is true under C (Manna & Waldinger, 1985)).

---

c1 = Extensive Problem Structuring;
That is: there is an extensive process of finding missing information. The following steps proposed by Goel and Pirolli (1989) are used by designers: studying the design brief; soliciting information and clarification; applying legislative constraints (e.g. design codes); applying technical constraints; applying pragmatic constraints (e.g. money, resources at hand); applying personal knowledge constraints; negotiating constraints (to fit personal ideas).

c2= Extensive Performance Modelling,
Mainly: idealized cognitive models (e.g. the concepts one has about what a draughts man is), scenario immersion, pictorial models, mock-ups, mathematical and numerical models and computer simulations.

c3= Personalized Stopping Rules; i.e.: rules which are used to terminate a design process or sub process;

c4= Evaluation in Three Contexts;
A generated or focused component may be evaluated in three contexts: local context; current context (of the current stage of the design); future context (i.e. projecting the complete artifact in its final stage).

c5= Making and Propagating Decisions (or Commitments);
Decisions must be made, recorded and propagate in order to produce an artifact description.

c6= Solution Decomposition;
As pointed out by Goel and Pirolli (1989), this process should consider the interconnections between entities at a functional and/or physical level.

c7= Abstraction Hierarchies;
That is: working with entities - functional and physical attributes - at various levels of detail. Descending too soon or not descending at all this hierarchy is a common mistake by novice designers.

c8= Use of Symbol Systems;
For instance: bubble diagrams, rough sketches and natural language.

Table 1 Characteristics of the Design Problem Space

We propose the following definitions:

Def. A Model of the Design Process is a set of characteristics C supported by a *Design Cognitive structure.*

Def. A Design Cognitive Structure for a set of characteristics C of the design process is determined by a *design paradigm* and the following *entailment relations:* $K = INV \times C$ and $I = P \times C$, where *INV* is a set of invariants of the task environment and P is a set of invariants of the information processing system.

We adopt the *paradigm of problem solving* and the following relations *K* and I which are determined by the "explanatory connections" suggested by Goel and Pirolli (1989):

K= {<inv1.c1>,<inv,c2>,<inv3,c7>,<inv3,c5>,<inv4,c2>,
<inv5,c4>,<inv5,c2>,<inv6,c2>,<inv7,c3>,
< inv8,c6>,<inv8,c7> , < inv8,c8>};

I= {<p1.c8>,<p2,c8>,<p2,c4>,<p3,c8>,<p3,c6>,<p3,c7>,<p3,c4>}.

4. THE PARADIGM OF PROBLEM SOLVING

Design was first identified with problem solving in Simon (1969). According to his approach, a state space represents all possible states of the problem (i.e. all possible problem descriptions) that need to be considered when a solution is attempted. Besides, he claims that it is practically computable to cover all the space.

Traditional problem solving are search processes within a state space. In this context, design knowledge is to be expressed in terms of goals and operators. Nevertheless, as Maher (1990) points out, "search does not directly address some of the intricacies and idiosyncrasies of the design problem". The difficulties are related to the variation of goals during the problem solving process and to the problem of predetermining the relevant operators. In conclusion, the notion of design as problem solving needs to be discussed in a broader setting, that is. one that sees it as more than a traditional search process.

In the present work we adopt the view of problem solving proposed by Minsky (1988). According to his view, machines can be made to solve any problem whose solution we are able to identify (Minsky's Puzzle Principle). The basic ways to improve upon blind trial-and-error search are: the progress principle (i.e. the ability to detect when progress is being made), goals and subgoals (i.e. the decomposition procedure that reduces the problem space) and the use of knowledge (i.e. "The most efficient way to solve a problem is to already know how to solve it. Then one can avoid search entirely").

Also the notion of goal needs more elaboration. The difference engine scheme proposed by Minsky (1988) contains a description of a desired state and a set of subsystems (agents) that are aroused by various differences between the desired state and the actual situation. Each subsystem must act in a way that tends to decrease the difference that arouse it. The idea of a difference-engine embodies the main ingredients of goal: (1) to have some image or description of a wanted or desired state; and (2) persistence (in pursuing the desired state). The well-known GPS (general Problem Solver) by Ernst & Newell (1969) and Newell & Simon (1972) is the classical Implementation of a difference-engine. Minsky (1988) claims that the difference-engine scheme remains the most useful conception of goal, pileups, or intention yet discovered.

Our idea here is to focus on the set of subsystems of a difference engine given the impossibility of having, in a design environment, a set of well defined operators in the traditional GPS sense. However, in the present paper we abstract the way in which difference between goals and actual situations are ultimately implemented. This issue belongs to the methods of synthesis, analysis and evaluation and does not need to be specified by the proposed design theory.

5. ELEMENTS OF DESIGN CONSIDERED IN THE THEORY

In the present section we are going to discuss three important aspects that we will try to capture in the semantics of the design language to be proposed as part of the forthcoming design theory: a representation of the states of the design space; the synthesis/analysis/evaluation process of design (the SAE process); and the role of short term and long term memories (STM/LTM).

5.1 The Representation of the States

The concept of goal discussed in the previous section is closely related to the question of decomposition (goals into sub-goals - i.e. problem into sub-problems). From the design process point of view, goals can be decomposed in terms of *structures* (physical or conceptual ones - which are always represented by a set of attributes) or *functional specifications* (i.e. description of the functions to be performed **by the structures).** The question of decomposition based on the structure (form) versus functional decomposition may pose a dilemma in terms of design knowledge representation: the knowledge base would need a different organization for each case of goal decomposition.

As discussed by Maher (1990), in a problem solving approach to design this situation does not occur because representations of goals may capture both the notions of fail ill and function. We propose here to represent goals by means of design entities. The properties of those entities are described both in terms of functional specifications and attributes. i.e.

> *Def.* We say that a *design entity is* a·pair $E = (F,A)$ where E stands for an entity, F for a set of *function specifications* and A for a set of *attributes.*

The notion of entity as introduced above represents goals, sub-goals and partial solutions. A partial solution is a resolved goal. i.e.: a solution to a sub-problem or a sub-problem which requires no further decomposition or transformation. Consequently, a state in the problem space of a design problem may be defined as the union of a number of such goals. sub-goals and/or partial solutions. Figure 2 presents the evolution of states schematically.

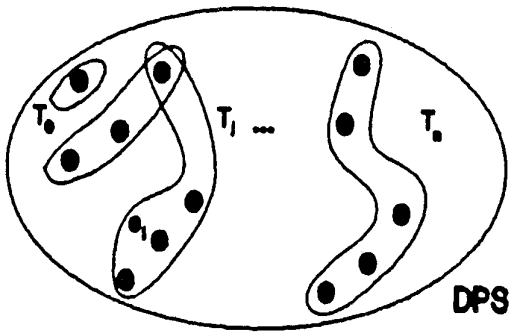

Fig.2 Evolution of states within a Design Problem Space

In the context of the present work design is an evolutionary process that starts with a set of input specifications, T_O, generates a kernel idea in the early stages of the process and refines it towards the artifact description T_n, i.e.: T_O -> T_1, -> ... -> T_n. Relationships between entities as well as constraints within or across the entities may be defined both in terms of F and A.

The sequence of states $<T_O, T_1, ...>$ represents one of the many trajectories through the design problem space. This sequence may enter the design problem space, get lost and never arrive to an artifact specification. In this regard, constraints are needed to navigate in the problem space. We shall use the DPS characteristics to explain movements in the problem space and support the evolution of the states T_i.

5.2 The SAE Process of Design

The evolution of the states T_i is carried out by a recursive process of three sub processes *Synthesis-Analysis-Evaluation*, which are themselves recursive processes (Figure 3). These sub processes represent a decomposition of the design problem and each of them has a particular *method* of reasoning. Furthermore, they are organized in levels of abstraction. The completion of a *S-A-E* design cycle at any level of recursion and at any level of abstraction corresponds to a change of state Tn -> T_{n+1}, by asserting or refuting any particular belief (for example, the acceptance of a value for a physical attribute of a given entity).

Examples of processes at different levels of abstraction are the generation of a kernel idea, which is a higher level process, and detailing which corresponds to a lower level process. In the

case of structural engineering design. the following processes can be identified: conceptual design, preliminary design, structural analysis and detailing.

The process SAE should also contemplate the decomposition of entitles in terms of functional specifications and Physical attributes Moreover. the process SAE should be able to consider the three contexts of' design, i.e. local context, curent context (of the current stage of the design) and future context (the complete artifact in its final stage).

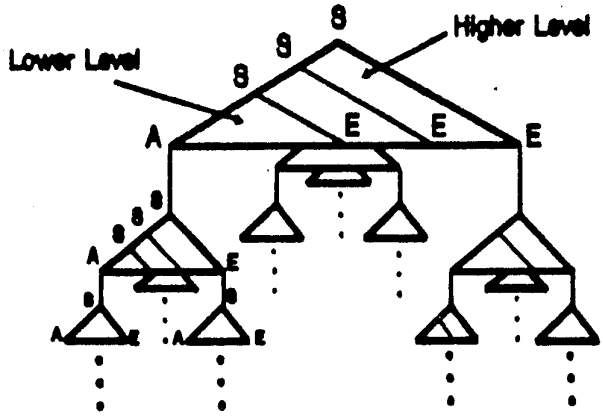

Fig.3 The Recursive Process SAE

5.3 The Role of STM/LTM

The role of *short term* and *long term memory* in the process of design is inherent to the information processing system which underlies the cognitive interpretation of any activity of mind. A theory of the design process should consider interactions with those two types of memory.

Very few models are general, in the sense that they cover both types of interactions. Schneiderman's (1980) Syntactic/Semantic model for software development is a remarkable example of a general model. A design system based on Anderson's (1989) spreading activation memory in ACT* has been reported by Steir (1990).

6. TOWARDS A FORMAL THEORY OF DESIGN

We consider the set of characteristics *C* above mentioned as an interpretation (i.e. a model) of a design theory *dt:*

$$dt = (dl, dp)$$

where *dl* is a *design language* and *dp is* an associated *deductive apparatus*. The design language is given by:

$$dl = (alphabet, syntax, semantics)$$

6.1 The Design Language

The *alphabet* of the design language *dl is* the following set of symbols:

$$alphabet = \{\ \#, [,], (,), nil, T_{x,l}^{y}, T_{x,2}^{y}, \ldots\}$$

where $x = S, A, E$ and $y = 0, 1, 2, 3, \ldots, m$.

The *syntax* of *dl* can be expressed through the following simple syntactic metalanguage:

wff= design_abstractions, #;

design_abstractions= [process_SAE | process_SAE, design_abstractions];

process_SAE= $T_{S,i}^{level}$(process_SAE), $T_{A,i}^{level}$(process_SAE), $T_{E,i}^{level}$(process_SAE) | nil |
design_abstractions;

i= integer;

level= integer,

integer= 0 | 1 | 2| ...;

This syntax is illustrated by the following example:

wff2= $[T_{S,0}^{0}()T_{A,0}^{0}()T_{E,0}^{0}()$
$T_{S,1}^{1}(T_{S,2}^{1}()T_{A,2}^{1}()T_{E,2}^{1}())T_{A,1}^{1}()T_{E,1}^{1}()$
$T_{S,3}^{2}()T_{A,3}^{2}()T_{E,3}^{2}()]$ # ;

where () denotes *nil* (i.e. an empty list).

The *semantics* of the design language is given by the following assertions:

i 1. The well-formed formulas (wff) represent the Design Problem Space (DPS);

i2. $T_{x,l}^{j}$ is a state in DPS and is represented by a set of entities $e_l^j(F,A)$;

i3. $T_{x,l}^{j}$ is a state derived by the application of a *method* of x (S for synthesis, A for analysis and E for evaluation) on the previous state (at least one new entity is created).

i4. $T_{x,l}^{j}$ is a state within the j-th level of abstraction and the i-th cycle of Synthesis-Analysis-Evaluation;

i5. $T_{S,0}^{0}$, also called T_0, represents the INPUT SPECIFICATION;

i6. "(...)" denotes levels of recursion in the process SAE and "[...]" denotes a set of abstraction levels;

i7. "()" denotes an empty sub process (a recursion stop);

i8. # is a stopping criterion for the design process;

i9. Entity decomposition can only be performed by a synthesis process S;

i10. $T_{E,i}^{j}(...)T_{S,k}^{l}$ is a change of abstraction level (j to l) determined by an evaluation process;

i11. An abstraction level is not supposed to be changed by either a synthesis process or an analysis process;

i12. Every state is supposed to be consistent with the previous one.

This semantics captures the nature of the design process.

6.2 The Deductive Apparatus

The two components of a deductive apparatus are the *Axioms* and the *Inference Rules*. In what follows we give the flavour of our theory by presenting two axioms, one inference rule and a theorem. The theory is currently evolving and we shall provide more details about it in a forthcoming paper. We assume the reader will be able to assess its power by figuring out, as an exercise. a few candidate axioms.

Axl. $...[...SA_{ij}...T_{E,k}^{j}()SA_{ij}T_{E,k}^{j}()...]...;$
within the same set of levels of abstractions and where $SA_{ij} = T_{S,i}^{j}(X)T_{A,i}^{j}(Y)$ and X, Y and Z stand for arbitrary SAE processes;

Ax2. $...[...T_{S,i}^{j} ... T_{S,k}^{j} ...]...; k > i$
within the same set of levels of abstractions.

R1. If $X_1T_{E,i}^{j}(Y_1)Z_1$ and $X_2T_{E,k}^{l}(Y_2)Z_2$ are wffs and $T_{E,i}^{j} = T_{E,k}^{l}$
then $X_1T_{E,i}^{l}(Y_2)Z_2$ and $X_2T_{E,i}^{j}(Y_1)Z_1$ are wffs.

According to *Ax1* any evaluation state can be next to higher levels of specification. This is, in essence, an evaluation in different contexts (characteristic *c5* of the meta-model). A_{x2} allows the return to previous levels of abstraction. In essence, this is problem structuring (the characteristic *c1* of the meta-model).

The following theorem allows a formal interpretation of artifact specification:

Teo. If $[T_0,...T_{E,n}^{m}()]\#;$ is a wff then $[HT_a]\#;$ is a wff
where $H= T_0,...T_{E,n}^{m}()$ is called DESIGN HISTORY and
$T_a= T_0T_{A,0}^{0}(X)T_{E,n}^{m}()$ is called ARTIFACT SPECIFICATION.

7. THE SAE METHODS

The methods of synthesis (S), analysis (A) and evaluation (E) to be used by a specific methodology should conform to the proposed design theory.

Table 2 is a general template to guide the implementation of a particular methodology. This table presents the main tasks and operations associated with SAE methods. However, this is not a strict classification, since the recursive nature of the design process makes the characteristics of each method to overlap (e.g. strictly speaking, what is the analysis process of an evaluation process ?).

SYNTHESIS	
Tasks:	Creation
	Exploration
	Induction
	Decomposition
	Case-Based Reasoning (mainly analogical reasoning)
Operations:	Generation of Kernel Ideas
	Generation/decomposition of Functional Specifications
	Generationdecomposition of Physical Attributes
	Functional Specifications ⇒ Partial Solutions
	Functional Specifications ⇒ Performance Specifications
	Performance Specificahons ⇒ Partial Solutions
	Combination of Partial Solutions
ANALYSIS	
Tasks:	Simulation
	Modelling (generating behaviours)
	Case-Based Reasoning (behaviour of retrieved solutions)
Operations:	Physical Attributes ⇒ Actual Behaviour
	Entities ⇒ Actual Behaviour
EVALUATION	
Tasks:	Judgement
	Case-Based Reasoning (test of retrieved solutions)
	Diagnostic
	Modelling (comparing behaviours)
	Deduction
	Test of Kernel Ideas
	stopping rules
Operations:	Actual Behaviour <> Performance Specifications
	Altenative Solutions <> Performance Specifications
	Artifact Specification <> Performance Specifications

Note:"A ⇒ B" denotes "B is obtained from A", "<>" denotes "compared with"; A Performance Specification is a value (or a range of values) which provides a measure of the behaviour that an artifact is expect to produce in terms of a particular functional specification; Actual Behaviour is the actual measure of the behaviour of an artifact in terms of attributes;

Table 2 General Tasks and Operations Associated with SAE Methods

Synthesis is the crucial question in design. The degree of artificial intelligence of a CAD/CASE system is proportional to the degree of independence that each process of synthesis has from human intervention.

Interactive computer graphics and intelligent user interface techniques can be understood as the link between the incomplete artificial synthesis (null in conventional CAD/CASE systems) and the external human synthesis (Figure 4). Human synthesis is considered one of the methods to be used in the implementation of any methodology based on the present theory.

In the context of the present work, automation means the substantial replacement of human action by machine action without removing the human interference in the key stages of the process. This concept discards any attempt of pursuing total automation, which is, as a principle, undesirable, technically (and perhaps theoretically) unattainable and, above all, renders the concept of professional liability difficult to apply in law (British Computer Society, 1985). From a cognitive point of view, total automation is unattainable because of the following facts: (1) knowledge that is not open to introspection has been reported by several researchers and, as far as design is concerned, Goel and Pirolli (1989) report this sort of knowledge in both general and domain specific levels; (2) there is evidence that specialists do not need to have formalized representations in order to act (Winograd and Flores, 1986, p.99); (3) Dreyfus and Dreyfus (1985) argue that expertise cannot be captured in any collection of formal rules.

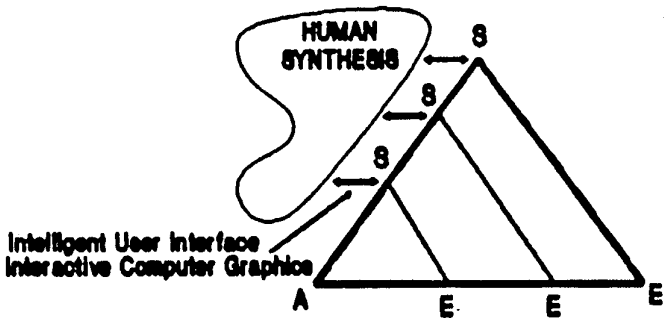

Fig.4 Human and Artificial Synthesis

8. EVALUATION OF SOME DESIGN METHODOLOGIES

It is not the intention in this short evaluation of current design methodologies to prove that the proposed design theory is a totally new approach to design. The idea here is to show that there is a common root for some of the well-known design methodologies and, moreover, that it can be formalized.

The following design methods were evaluated by the authors in the light of the framework proposed in the present paper: Jone's (1963) Analysis-Synthesis-Evaluation Model; Formulation-Synthesis-Evaluation Model, Decomposition Model, Case-Based Reasoning Model and Transformation-by-Grammar Model of Design Synthesis presented by Maher (1990); Propose-Critique-Modify family of methods proposed by Chandrasekaran (1990); IIICAD methodology built around the General Design Theory (B.Veth, 1987 and Veerkamp et alii, 1989); Gero's Design Prototypes Model (1987, 1990); Coyne's (1988) Logic Model. A common fact amongst these models is that they lack a cognition basis. However, they can be recast as instances of the meta-theory proposed here of which they constitute particular methodologies. The following remarks are worth being mentioned: Jones' proposal misses the recursive nature of design and Coyne's formalism for a design language associated with logic is based on particular aspects of first-order logic and shape grammars.

9. CONCLUSIONS

In a sense, the present work is a follow-up on the work by Goel and Pirolli (1989) who suggest that their analysis should be pushed further toward a process model of design. We adopt the paradigm of design as problem solving and propose a model that supports a theory (a meta-theory) of the design process. The formal system is far from completion but it proved itself to be feasible and useful. The proposed theory captures the cognitive foundations and the recursive nature of design. Furthermore, the theory seems to represent a common root for some of the well-known models of design. Also the theory represents a pragmatic guideline for the investigation of ways in which design methodologies can be built and CAD/CASE systems enhanced.

The set of characteristics *C* represent cognitive needs of the designer that have to be satisfied by every CAD/CASE system. The ideal system should satisfy all those characteristics. However, current CAD/CASE systems only satisfy one or two of those cognitive needs.

Specific SAE methods yield to specific technologies. In this regard, we believe that hypertext tools might support problem structuring *(c1)*, object-oriented languages might help the implementation of *c4*, *c5*, *c6* and *c7*, and intelligent user interface techniques might help in the use of symbol systems (*c8*).

BIBLIOGRAPHY

Anderson,J.R.,1989. A theory of the origins of human knowledge, Artificial Intelligence,40(13),pp.313-351.

Akin,0.,1979. An exploration of the design process,Design *Methods and Theories,13.*

British Computer Society, 1985. Seminar on social implications of artificial intelligence and expert systems, Specialist Group on Expert Systems,Oxford.UK.

Chandrasekaran,B.,1990. Design problem solving: a task analysis,AI *Magazine,winter* 90,pp.59-71.

Coyne,R.,1988. Logic Models of Design, Pitman Publishing London,UK.

Cross,N.,1984. Developments in Design Methodologies,Cross,N.(ed.)John Wiley and Sons.

Dreyfus,H.L. and Dreyfus,S.E.,1985. Mind over Machine, Macmillan~The Free Press,New,USA.

Ernst,G.W. and Newell,A.,1969. GPS: a Case Study in Generality and Problem Solving,

Academic Press,New York,USA.

Gero,J.S..1990. Design prototypes: a knowledge representation scheme for design,*AI Magazine,winter* 90,pp.26-36.

Goel,V. and Pirolli,P.,1989. Motivating the Notion of Generic Design within InformationProcessing Theory: the Design Problem Space, *AI Magazine*,**Winter 90**,pp.19-36

Jones,J.C.,1963. A Method for Systematic Design,in Jones,J.C. and Thornley,D. (eds), Conference on *Design Methods,Pergamon.*

Jones,J.C.,1970. Design Methods, Wiley Interscience,London,UK.

Jones,J.C.,1977. How my Thoughts about Design Methods have Changed During the *Years,Design Methods and Theories,11(1).*

Jones,J.C.,1981. Design Methods: Studies of Human Future,J.Wiley.

Jeffries,R.;Turner,A.A.;Polson,P.G. and Atwood,M.E.,1981. The Processes Involved in Designing Software,in Anderson,J.R.(ed.),Cognitive *Skills and Their* Acquisition,Lawrence Erlbaum,New Jersey.USA.

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

Maher,M.L.,1990. Process models of design synthesis, Al Magazine, **winter 90**,pp.49-58.

Manna,Z. and Waldinger,R.,1985. The Logical Bases for Computer Programming: Deductive Reasoning (volume 1),Addison-Wesley.

Minsky,M.L.,1988. The Society of Mind, Pan Books,London,UK.

Newell, A. and Simon,H.A.,1972. Human Problem Solving, Prentice-Hall,USA.

Simon,H.A.,1969. The Sciences of the Artificial, MIT Press, Massachusets,USA.

Steir,D.,1990. Creating a scientific community at the interface between design and AI,AI *Magazine,winter* 90,pp.18-22.

Schneiderman,B .,1980. Software Psychology, Winthrop, Cambridge, Massachusets, USA .

Veerkamp,P.;Akman,V.,Bernus,P. and ten Hagen,P.J.W.,1989. IDDL: a language for intelligent interactive integrated CAD systems, in Akman.V. et alii (eds.),*Intelligent* CAD *Systems II Implementational* Issues, Springer-Verlag, Berlin,pp.58-74.

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

Winograd,T. and Flores,F.,1986. Understanding Computers and Cognition: a New Foundation for Design, Ablex Publ.,Norwood,New Jersey,USA.