MOTIVATING THE NOTION OF GENERIC DESIGN WITHIN INFORMATION PROCESSING THEORY:

THE DESIGN PROBLEM SPACE[‡]

By

Vinod Goel
Department of Architecture

and

Peter Pirolli
School of Education
University of California at Berkeley

Cognitive Science Program

Institute of Cognitive Studies

University of California at Berkeley

February 1989

[‡]Article to appear in an upcoming issue of *AI Magazine*.

The production of the Berkeley Cognitive Science Report Series is supported by a grant in cognitive science to the University of California at Berkeley from the Alfred P. Sloan Foundation.

MOTIVATING THE NOTION OF GENERIC DESIGN WITHIN INFORMATION PROCESSING THEORY: THE DESIGN PROBLEM SPACE†

Vinod Goel
Department of Architecture
&
Peter Pirolli
School of Education
University of California at Berkeley

ABSTRACT

The notion of generic design, while it has been around for 25 years is not often articulated, especially within Newell and Simon's (1972) Information Processing Theory framework. Design is merely lumped in with other forms of problem solving activity. Intuitively one feels that there should be a level of description of the phenomenon which refines this broad classification by further distinguishing between design and non-design problem solving. However, Information Processing Theory does not facilitate such problem classification. This paper makes a preliminary attempt to differentiate design problem solving from non-design problem solving by identifying major invariants in the *design problem space*.

There are four major steps in the strategy: (1) characterize design as a radial category and flesh out the task environment of the central or prototypical cases; (2) take the design task environment seriously; (3) explicate the impact of this task environment on the design problem space, and (4) argue that, given the structure of the information processing system as a constant, the features noted in the problem spaces of design tasks will not all occur in problem spaces where the task environment is vastly different. This analysis leads to the claim that these features are invariants in the problem spaces of design situations, and collectively constitute a design problem space.

Descriptive protocol studies are used to explore the problem spaces of three prototypical design tasks from the discipline of architecture, mechanical engineering, and instructional design. The following eight significant invariants are identified: (A) extensive problem structuring, (B) extensive performance modeling, (C) personalized/institutionalized evaluation functions and stopping rules, (D) a limited commitment mode control strategy with nested evaluation cycles, (E) making and propagating commitments, (F) solution decomposition into leaky modules, (G) role of abstractions in the transformation of goals to artifact specifications, and (H) use of artificial symbol systems. The paper concludes by drawing some morals for the development of computer-aided design (CAD) systems, noting

† The authors are indebted to Dan Berger and Susan Newman for much useful discussion and argumentation on the contents of this paper. Vinod Goel gratefully acknowledges the support of the following sources of funding during the course of this work: an Andrew Mellon Fellowship, the Myrtle L. Judkins Memorial Scholarship, a Canada Mortgage and Housing Corporation Scholarship, and a summer internship at the Institute of Research on Learning at Xerox Parc. Portions of this research were also funded to Peter Pirolli by the Office of Naval Research under contract N0014-88-0233.

Correspondence should be directed to Vinod Goel at the Institute of Cognitive Studies (Bldg T-4), University of California at Berkeley, Berkeley, CA 94720.

the limitations in the work, and indicating directions for further research.

1. INTRODUCTION

The term *generic design* denotes two related ideas. It suggests that design as an activity has a distinct conceptual and cognitive realization from non-design activities, and that it can be abstracted away from the particulars of the knowledge base of a specific task or discipline and studied in its own right. It has its origins in the design methodology research of the 1960s (Cross, 1986). At this time the observation was made that the various design methods, while they differed in particulars, shared a common pool of assumptions which conceived the design process as moving through the following sequence of steps: (i) an exploration and decomposition of the problem (i.e., analysis); (ii) an identification of the interconnections between the components; (iii) the solution of the subproblems in isolation; and finally, (iv) the combination (taking into account the interconnections) of the partial solutions into the problem solution (i.e., synthesis). On the basis of this observation many researchers concluded that "the logical nature of the act of designing is largely independent of the character of the thing designed" (Archer, 1969, p.76). However, they did not go on to develop the concept to any significant extent.

Subsequently these assumptions were questioned by other researchers (Akin, 1979; Lawson, 1979) working in the different framework of Newell and Simon's (1972) Information Processing Theory (IPT).[1] While the concern of the earlier researchers was with the development of "systematic design methods" to help designers (often working in teams) to deal with the increasing amount and complexity of project information (Cross, 1986), the concern of the latter is with explicating the internal structures and procedures individual cognitive systems use during design activity, with what Eastman (1969) called *intuitive design*.

The study of intuitive design, within an IPT framework, has become a dominant mode of research into design activity.[2] But this research is moving in two directions which are rather dissatisfying from the perspective of developing a cognitive theory of design. First, the research tends to be discipline-specific and even task-specific (e.g., Kant and Newell, 1984; Kant 1985; Steier and Kant, 1985; Jeffries et al. 1981; Ullman et al., 1986; Akin, 1979, 1986). Second, there is a proliferation of disciplines and activities being labeled as "design." Thus for example, Perkins (1986) labels the process of knowledge acquisition "design." Thomas (1978) analyses communication as a design process. Thomas and Carroll (1979) assume that letter writing, naming, and scheduling are all design activities. The first of these trends flies in the face of the intuition lying behind the notion of *generic* design.

[1] This discussion assumes considerable familiarity with IPT as presented by Newell and Simon (1972). The uninitiated reader is well referred to this original work.

[2] There are two major reasons for this. First there is a realization by industry that the development of effective CAD tools requires a model of the user's (designer's) cognitive processes (Ballay et al. 1984). Second there is the hope that the study of human designers will lead to insights into the automation of the design process (Kant and Newell, 1984).

The second threatens to drain the word *design* of all meaning.

One reason for these trends is the nature of IPT itself. Within IPT design is problem-solving activity. But problem solving encompasses a wide range of cognitive activity; indeed, according to some theoreticians, all of symbolic cognitive activity (e.g., Newell, 1980). Intuitively one feels that there must be a description of design problem-solving activity which both captures the similarities in the problem-solving process across the various design disciplines and also recognizes the differences between design and non-design problem solving. This is the level which the term *generic design* informally tries to characterize. In the vocabulary of IPT, there must exist a *design problem space*—a problem space with major invariant characteristics across all design situations. However, as has been observed by a number of researchers (e.g., Greeno, 1978), the theory does not easily facilitate such classification. We see three interrelated reasons for this "shortcoming."

1) In some ways the vocabulary provided by IPT seems to be missing a layer. At the top level of the theory one can talk about Information Processing Systems, Task Environments, and Problem Spaces. But the next level down takes one directly to the implementation details of specific programs where one must talk about states and transformations at the level of the elementary information processes. Differentiation of problem types is readily possible only at this lower level. There is a gap in the middle where one intuitively feels there should be several intermediate levels of psychologically interesting concepts—such as generic design.

2) The structure of the information processing system is underdeveloped. Except for the size of short-term memory (STM) and read/write times, it does not impose many significant constraints on the problem space. Thus the problem space tends to be substantially task determined.

3) But the notion of task environment has not been fully explored and exploited within the theory. While the theory does say that the task environment consists of (i) the goal or desire to solve the problem, (ii) the problem statement, and (iii) *any other relevant external factors*, the fact remains that historically, the goal or motivation of the problem solver has simply been assumed, and the "other relevant external factors" have been effectively ignored.[3] The emphasis has been on how the problem statement gets mapped onto the problem space.

Within IPT there are two possible sources of invariants on the design problem space. One is the *structure of the task environment*, the other is the *structure of the information processing system* (IPS). One way of motivating a DPS is to identify task environments and information processing structures that are particular to design situations. This is precisely the strategy that will be pursued here. However, we will have little new to say about the

[3] This is undoubtedly due to the influence of game-playing problems on which the theory grew up.

structure of the IPS. Most of the paper is concerned with explicating the structure of the design task environment and specifying its impact on the DPS.

Organization and Overview of Paper: In this paper we would like to play out the intuition which says that the design problem space is an interesting and "natural" categorization of problem spaces. Our strategy will be to (i) characterize design as a radial category and flesh out the task environment of the central or prototypical cases; (ii) take the design task environment seriously; (iii) explicate the impact of the structure of the task environment and the structure of the IPS on the problem space of subjects from three different design disciplines; (iv) suggest that the features noted in these problem spaces will not all occur in a problem space where the task environment is vastly different; (v) claim that these features are invariants in the problem space of design situations and collectively constitute a design problem space. Two aspects of our strategy differentiate this work from much of the current research in design: (1) we take the structure of the design task environment very seriously, and (2) we examine data from three different design disciplines. The paper concludes by drawing some lessons for the development of CAD systems, noting some methodological limitations, and suggesting avenues for further research. We begin by characterizing design and the design task environment.

2. CHARACTERIZING DESIGN & THE DESIGN TASK ENVIRONMENT

In this section we would like to claim that design is *not* a ubiquitous activity. We no more design all the time than we read all the time, play chess all the time or engage in scientific research all the time. But the characterizations of design in the cognitive science literature would have us believe that most of us do engage in design activity most of the time. We briefly review some of this literature and conclude by offering our own, rather different, analysis.

Perhaps the most encompassing characterization of design is due to Simon (1981, p.130):

> Everyone designs who devises courses of action aimed at changing existing situations into preferred ones.... The intellectual activity that produces material artifacts is no different fundamentally from the one that prescribes remedies for a sick patient or the one that devises a new sales plan for a company or a social welfare policy for a state.

On this account, anyone dissatisfied with existing states of affairs and attempting to transform them into "preferred ones" is engaged in design activity. The domain of design would seem to be coextensive with the domain of problem solving.[4]

An early attempt at circumscription is due to Reitman (1964). In a paper on ill-defined problems, Reitman (1964) suggested a categorization of problems into six types based upon

[4] Which in turn is coextensive with thinking in IPT.

the distribution of information within a *problem vector*. A problem vector is a tuple of the form [A, B, =>], where components **A** and **B** represent the start and terminal states respectively, and the component => denotes some transformation function. Reitman's Type2 problems correspond to our intuitive notion of design. Typical Type2 problem statements are:

compose a fugue
design a vehicle that flies
write a short story
design a building
make a paper airplane

While these statements encompass widely varying activities, Reitman observed that they constitute formally similar problems by virtue of the amount and distribution of information among the three components of the problem vector. In the case of the Type2 or design problems, the invariant characteristic is the lack of information. For instance:

(i) The start state **A** is unspecified (e.g., design a vehicle...with what? putty? cardboard? prefabricated parts from GM?).

(ii) The goal state **B** is incompletely specified (e.g., how long should a story be? what should the plot be? how should it end?).

(iii) The transformation function => is unspecified (e.g., how should the airplane be made? by folding the paper? by cutting and pasting?).

After this seminal paper design problems became identified with ill-defined problems.

Continuing the investigation of ill-defined problems, Simon (1973) argued that problems in the world do not come prelabeled as "well-defined" or "ill-defined." Furthermore, according to Simon, "well-defined" and "ill-defined" are not mutually exclusive categories. They constitute a continuum. Where a given problem falls on this continuum is a function of the *stance* the problem solver takes to the problem. That is, the problem solver may ignore existing information or supply missing information from long-term memory or external aids. The conclusion that follows from Simon's discussion is that what constitutes a design problem is determined by the intentions and attitudes of the problem solver. This is an interesting position that has found some acceptance in the literature (e.g., Thomas and Carroll, 1979). It does however, have the effect of once again opening up the flood gates as to what constitutes design activity.

Each of these attempts at delimiting or characterizing design is due to cognitive science researchers. Designers typically offer very different definitions. A rather well-accepted one among designers is due to Eastman (1981, p.13): "Design is the specification of an artifact that both achieves desired performances and is realizable with high degrees of confidence." This statement emphasizes that the product of design is an artifact specification, and that considerations of performance and realizability are integral to the process.

While each of these definitions is interesting in its own right and has a role to play in our understanding of design, none of them is sufficient for our purposes here. Design is too complex an activity to be captured in a one-line definition; particularly a one-liner that purports to specify necessary and sufficient conditions. As such, our characterization of design starts with the observation that design as a category exhibits what Rosch (in Lakoff, 1987) calls "prototype effects." Furthermore, it is what Lakoff (1987) calls a radial category—a category in which there is a central, ideal or prototypical case and then some unpredictable but motivated variations. On this assumption, if one shows people a list of professions—e.g., medicine, legal work, architecture, teaching, engineering, and research—and asks them which are the *best examples* of design professions, they will all invariably and consistently pick out the same few cases. In this list we believe the "best examples" would be architecture and engineering. We propose to call these "good" or "central" or "prototypical" examples of design professions.

Having made this observation, we propose to take a serious look at the task environment of these prototypical design professions. In so doing we will be using the term "task environment" much more broadly than it is generally construed in IPT. We want to use it to encompass much of what is relevant and external to the problem space and the information processing system. The danger with this move is that either it results in a theoretically uninteresting term—because in some sense everything is relevant—or one is obliged to say what matters and what doesn't. We go the latter route and attempt to specify some of the more important aspects of the design task environment.

Fig. 1 approx here

The structure of the design task environment as we construe it is depicted in Fig. 1. As a first approximation one can note the following overt features:[5]

1) There are many degrees of freedom in the problem statement. (This is just a positive reformulation of Reitman's (1964) earlier point about a lack of information in design problem statements.)

2) There is delayed/limited feedback (on the order of many hours to many months) from the world during problem solving.

[5] No attempt is being made to be exhaustive.

3) The input to the design process substantially (though not completely) consists of goals and intentions. The output is a specification of an artifact.

4) The artifact must function independently of the designer.

5) There is a temporal separation between the specification and delivery of the artifact (with specification preceding delivery).

6) There are costs associated with each and every action in the world. (i.e., There are penalties for being wrong.)

7) There are no right or wrong answers, only better and worse ones.

8) The problems tend to be large and complex.

We claim that these are significant invariants in the task environments of prototypical design situations, and we can use them as a template to identify other cases of design. To the extent that the task environment of a given problem situation meets or conforms to this template, that problem situation is a good or prototypical example of a design situation. To the extent that a task environment varies from this template—by omission of one or more of the requirements—to that extent it is a less central case of design activity.

Some problem-solving situations which fit well into the schema are instructional design, interior design, "text book cases" of software design, and music composition. Some tasks that deviate slightly are writing and painting. Here there is usually no separation between design and delivery. The problem solver actually constructs the artifact rather than specifying it. Some activities that deviate more radically are classroom teaching, spontaneous conversation, and game playing.

Note that we are not stipulating what is and is not a design activity. To do this we would have to insist that the eight task environment characteristics, or some subset of them, constitute necessary and sufficient conditions for design activity. We make no such claim. Rather, all we are suggesting is that we have here a template of some salient characteristics common to the task environment of problem situations that are consistently recognized by people as good examples of design activity. Problem situations in which the task environment fails to conform to this template on one or more accounts are deviations from the central case. In this paper we are only interested in central cases and thus have no interest in saying how far one can deviate from the prototype and still be "really" designing. Thus, we will use the label *design* to refer to situations that closely conform to the prototypical or central cases.

There are two reasons why the above might be a reasonable characterization of design for our purposes. First, it is descriptive. We look at the task environment of *some* designers and try to take it seriously. The task environment of an activity is usually overtly visible with minimal theoretical commitments (though it does require some immersion in the activity and the ability to specify the more relevant factors). Second, in IPT the structure of the information processing system is relatively underdeveloped, leaving the task environment as the major tool/resource for structuring the problem space. Furthermore, the theory asserts that people "are severely stimulus-bound" (Hayes and Simon, 1974, p.197) with respect to representation and construct a naive/transparent model of the problem based upon "the surface features of the external environment..." (Newell, 1980, p.714). Thus given the accessibility and the importance of the task environment to IPT, it seems like a good basis for classification. In the next section we examine each of the invariant features of the design task environment and hypothesize about their impact on the DPS.

3. A CASE FOR GENERIC DESIGN: THE DESIGN PROBLEM SPACE

In the previous section we identified eight interesting invariants in the structure of the design task environment. These invariants are external features of design activity that have been noted by various researchers at various times and places in the design methodology literature. But we are unaware of any studies in the IPT literature in which these factors are taken seriously and their cognitive implications sketched out. We undertake this task in this section.

Our strategy is to examine a number of designers at work and (i) reconstruct their problem space; (ii) make an "explanatory connection" between the features evident in their problem spaces and the above noted invariants of the design task environment (DTE); and (iii) make the standard argument that the problem space is as it is because of the structure of the DTE and the structure of the IPS. This last point implies that, taking the structure of the IPS as a constant, the features noted in the problem spaces of these tasks will not all occur in a problem space where the task environment is vastly different. This leads to the claim that these features are invariants in the problem spaces of design situations, and collectively constitute a Design Problem Space. We are actually able to identify eight interesting invariants in the problem spaces of three different design disciplines. To anticipate and overview, we will claim the following:

A) The many degrees of freedom in design problem statements entail extensive problem structuring.[6] (section 3.1)

B) The delayed/limited feedback from the environment, coupled with the cost of action, and the independent functioning requirement on the artifact entails extensive performance modeling of the artifact in the problem space. This modeling is made possible by the fact that there is a temporal separation of specification and delivery phases. (section 3.2)

---

[6] By the use of the terms *entail* and *necessitate*, we are throughout making contingent causal claims, not logical necessary claims.

C) The fact that there are no right or wrong answers to design problems entails the use of personalized evaluation functions and stopping rules. (section 3.3)

D) The requirements of extensive performance modeling, along with the constraints of sequential processing and short-term memory (STM) capacity entail a limited commitment mode control strategy with nested evaluation loops. This strategy is enabled by the temporal separation of specification and delivery. (section 3.4)

E) The necessity of having to specify an artifact means that designers must make and propagate commitments. There is a tension between the limited commitment mode control strategy and the need to make commitments. (section 3.5)

F) The size and complexity of design problems combined with the limited capacity of STM require solution decomposition. However, the decomposition is not complete. The modules are "leaky." (section 3.6)

G) A phenomenon closely related to solution decomposition is the mediation of goal and artifact by abstraction hierarchies. It is entailed by the complexity of the problem, STM capacity, and the fact that the input to the design process substantially consists of goal statements while the output is an artifact specification. It is also related to the phenomenon of personalized/institutionalized stopping rules and the making and propagating of commitments. (section 3.7)

H) The last problem space invariant we note and discuss is the use of artificial symbol systems. It is entailed by the limitations on the expressive power of the "language of thought," STM capacity, sequential processing, and problem complexity. It is related to and has consequences for the phenomenon of solution decomposition, abstraction hierarchies, the making and propagating of commitments, and performance modeling. (section 3.8)

All these invariants, their interconnections, and their connections to the invariants of the DTE and the information processing system are explicated in the diagram in Fig. 2. While no claim of completeness is made for this list, it is our contention that collectively these invariants differentiate design problem spaces from non-design problem spaces. But before actually presenting and discussing each one, a word about methodology is in order.

Fig. 2 approx here

Method: The method of investigation adapted here is that of protocol analysis (Ericson and Simon, 1984). The data base consists of 12 protocols from 3 different design disciplines—architecture, mechanical engineering and instructional design. To illustrate and substantiate our claims for the purpose of this paper we will draw upon one protocol from each of the three disciplines. The decision as to which three of the protocols to use was made as follows: In the case of mechanical engineering, there was only one protocol. There were multiple protocols for instructional design and architecture. The decision among them was made on the basis of the completeness of the artifact specification and the fluency of the verbalization.

Task Descriptions: The architecture task involved the design of an automated post office (where postal tellers are replaced by automated postal teller machines) for a site on the UC-Berkeley campus. The mechanical engineering task was to design the automated postal teller machines (APTM) for the post office. The instructional design task was unrelated. It called for the design of some stand-alone text based instruction to prepare the secretaries of a medium-sized company for a transition from typewriters to the Viewpoint[7] computer environment. In each case, the subjects were given a design brief which stated the client's requirements and encouraged to probe the experimenter for further information and clarification. They were asked to "talk aloud" as they proceeded with the task. The sessions were taped on a video recorder.

Each of the tasks are complex, real world problems requiring on the order of weeks to months for a complete specification of the artifacts. We asked the architecture and mechanical engineering subjects to restrict their sessions to approximately 2 hours and gave the instructional designers approximately 3 hours. As a result, we received solutions specified to an incomplete level of detail.

Subjects: Each of the 3 subjects volunteered to participate in the study. The architect (Subject-A) is a Ph.D. student in the Department of Architecture at UC-Berkeley. He has had six years of professional experience. The mechanical engineer (Subject-M) is a Ph.D. student in the Department of Mechanical Engineering at Stanford University. His professional experience is more limited, but it has included the design of automated bank teller machines. The instructional designer (Subject-I) is a professional with over 10 years experience in designing industrial training material.

Coding Procedure: The analysis of the protocols to date has been qualitative and descriptive. We are still in the process of identifying the major components of the design problem space and arranging them in an explanatory fashion so as to build a model of the design process. We are not at a stage where we can engage in any quantitative or predictive analysis. But on the other hand, we are not limited to noting and relating everything we see. We have a rather explicit and constrained agenda: We want to know how the identified aspects of the DTE impact the DPS.

---

[7] Viewpoint is an icon-based computer environment for Xerox Stars. It supports such functions as electronic mail, filing, word processing, and graphics.

3.1. Extensive Problem Structuring

As noted earlier, many degrees of freedom exist in a design problem statement (or to put it in Reitman's terms, there is a lack of information). This lack of information impedes the creation of a problem space. Problem structuring is the process of finding the missing information and using it to construct the problem space (Simon, 1973a). It is the first step in any design activity. Large projects may require an alternation between problem-structuring and problem-solving phases. While some structuring is required in all problem situations, one of the hallmarks of design problems is that they require extensive structuring. The extent to which problem structuring is necessary and successful determines the nature and extent of the problem solving that will occur.

Each of the subjects in our experiment began by articulating and fleshing out their respective problem statements. This process proceeded through the follwoing steps: (1) gathering information from the design brief; (2) soliciting information and clarification from the experimenter through questions; (3) applying knowledge of legislative constraints (e.g., building codes, in-house company standards); (4) applying knowledge of "technical" constraints (e.g., "laws" of structural soundness, "laws" of learning); (5) attending to pragmatic constraints (e.g., time, money, resources at hand); (6) bringing to bear self-imposed constraints or personal knowledge; and (7) negotiating constraints. While each of these can bear considerable discussion, only the latter two will be addressed here. With respect to (6) two questions are raised: (i) what is the form and structure of this personal knowledge, and (ii) how and when is it brought to bear on the construction of the problem space? While we have no definitive answers to these questions, we do offer some preliminary observations. In the case of (7) we illustrate the process of negotiation and comment on when and why it might occur.

3.1.1. Form and Organization of Personal Knowledge

The personal knowledge our subjects used to construct their problem spaces was organized in rich, intricate chunks or schemas. Two types were discernible: *general schemas* and *domain specific schemas*. Generally, neither surface explicitly in protocols[8] but both are easily inferred from the situation-specific statements that the subjects make.

General schemas contain knowledge about the way(s) the world is. They are acquired over the course of a lifetime and are our primary means of dealing with the world. They consist of at least *procedural knowledge*, *abstract conceptual knowledge*, and knowledge of

[8] Unless the subject stops to explain or rationalize, as one of our architects frequently did. Here is a typical excerpt from him: "Now, every building fitting into a site should be harmonious with that site. Nobody argues with that. The next thing, and compatible with the other buildings. Ah. We are going from a very antisocial period, where buildings were very antisocial and withdrawn, and aggressive, and impolite, such as the one we are standing in, to, ah, buildings which are pleasant, outgoing, gentle, ah, sophisticated and cultured...."

thousands of *patterns* (pictorial, linguistic, musical, etc.). Procedural knowledge is not open to introspection (Anderson, 1982) and thus does not surface in the protocols. However, both the abstract conceptual knowledge and some of the patterns are visible.

Abstract conceptual knowledge is the generalized knowledge—principles, laws, heuristics—which we extract and carry away from the totality of our worldly experience. While there is much structure and coherency in the organization of this knowledge, it does not necessarily constitute a theory. It is perhaps better characterized as knowledge fragments or "knowledge in pieces" (diSessa, 1985). It is instantiated and discernible in the problem space as *situation-specific* conceptual knowledge. For example, here is an excerpt from Subject-A's protocol:

(PF1) S-A: You, after all, you probably have your parcel or your precious letter and you want to get it out, stamp it, or ah, have a dialogue with a machine and see what, how much you have to pay. Your probably have to take it out from your bag, or whatever. So you do need a sort of protection.... I don't want them to get wet....

Underlying this verbalization are two knowledge fragments at the abstract, conceptual level—beliefs about the use of post offices and beliefs about when and where people do and do not like to get wet.

Knowledge of patterns is knowledge that is stored in such a direct way so that much of the original pattern or form is preserved (i.e., there is little generalization or abstraction). This may be voluntary, such as when students of poetry memorize lines of text or when architecture students draw and commit to memory the forms of specific buildings, or it may be involuntary, as in the case of a stimuli which the cognitive system is unable to fully comprehend and generalize. Instances of specific patterns are visible in all the protocols. Subject-A for instance, in attempting to reason about an automated postal interface, immediately retrieved and repeatedly used the "image" of an automated bank teller machine. But it was not some general conception of an automated bank teller but the specific Bank of America Versateller on Telegraph Avenue which he regularly uses.

(PF2) S-A: I don't want to have one booth after the other and having the lines, ah, like it were a Versateller, ah, kind of a service. Bank of America has that kind of approach, here on Telegraph. You have two, two Versatellers and usually have this long lines on the, ah, walk path. And who ever, ah, leaves first in one of the two, ah, then. So you have one single line for two machines. I am trying to avoid that....

Domain-specific schemas are built on top of the general schemas. They constitute the knowledge acquired during the years of professional training. They also consist of procedures, abstract conceptual knowledge, and patterns. Again, the procedures are not visible in the protocol. The abstract conceptual knowledge here seems to be less fragmentary and more "theoretical"[9] than in the case of the general schemas. (This is not surprising

[9] By 'theoretical' is meant only that it is more elaborate, complete, consistent, and organized.

considering that it was acquired as an organized, systematic body of knowledge.) For example, Subject-A has an elaborate mini-theory about the use and organization of space between buildings. His first sentences on viewing the site are:

(PF3) S-A: Well, what comes to my mind immediately, as I told you before when I was waiting [for] you, I was looking at, how this is set by pathways, this, this open space in between the sports court yard and these three buildings. And in thinking about the missed opportunity that people had here, of having a sort of more relaxed plaza, instead of being just a cross between these two directions. Which makes it very efficient, ah, but for sure it didn't, ah, give any contribution to the urban open space....

Similarly, Subject-I has a mini-theory about motivating, teaching, and imparting knowledge.

(PF4) S-I: The first thing we want to do with these people is try and sell them on a system. Any time you change somebody from an old system to a new system, or from what they are doing to what they're going to be doing, or what you're expecting them to be doing, you've got to give them a good positive reason. Why do I really? What's in it for me, you know.... This is positive reinforcement....

Several of these protocol fragments (PF1, PF2, PF3) are also examples of what we call *scenario immersion*. Scenarios are frequently occurring episodes in which designers recall and immerse themselves in rich intricate images from their past experience. The experience in question could have been acquired directly or vicariously through some symbolic medium (e.g., reading, watching TV). These episodes seem to play an absolutely crucial role in the process of generation and evaluation. For instance, the scenario in PF1 is used to generate the functional requirement "protection from rain." In PF2, the scenario is used to evaluate a proposed spatial configuration of APTMs. We say more about scenario immersion in section 3.2.

3.1.2. Application of Personal Knowledge

Personal knowledge structures and procedures are stored in long-term memory (LTM). Their indexing and retrieval is not well understood. Problem structuring is the process of finding and retrieving "relevant" schemta and instantiating them into the problem space. By *instantiation* is meant nothing more than the process by which a proposition of the content "All public buildings are required to have ramp access for the handicapped" is transformed into the proposition that "This building requires a ramp access." As it is construed here, problem structuring is not itself a problem-solving activity. But the extent to which it is successful does determine the amount of problem solving that will need to occur.

Subject-I was able to find, retrieve and instantiate a single powerful schema for designing training programs. The template came with slots marked for lessons, sections, subsections, etc. He merely had to fill in the blanks with the content of the particular course. He generates the required content by (i) asking the client (experimenter) for a list of tasks the secretary would be required to perform; (ii) drawing upon his own personal knowledge of Viewpoint; and (iii) consulting the Viewpoint manuals.[10] Finally, the selection of content is

[10] Some of the instructional design subjects actually used the Viewpoint manual to structure the task.

guided by an idealized cognitive model (ICM) (Lakoff, 1987) of what a secretary is; for example:

(PF5) S-I: All this isn’t going to stay in this create and edit documents [lesson]. This is just looking at what’s available, and what we are going to have to do. Because within this table of contents we’ve got related information—hardware requirements and so forth that has nothing to do with the secretaries, and foundation and environment. Secretaries couldn’t care less.... And the logoff sheet properties, I, I wouldn’t even teach the secretaries. That’s none of their business. They have no need for that information. That I would teach your systems administrator....

Subject-A on the other hand seemed to find his design problem more of a challenge and exhibited somewhat different behavior. His initial structuring process took twenty minutes and resembled a "brainstorming" session. If the protocol for this phase is recorded as a directed graph, with the nodes forming individual "ideas" as they are uttered in temporal sequence, and the arcs connecting related nodes, then the result is a lattice structure. The density and distribution of the links suggest that there are really four smaller structures. First there are some site related constraints:

(PF6) S-A: You plotted those trees and that would really be a sin to touch them, I think. At least, the evergreens.... As far as seating space goes, the one just below the evergreens, I wouldn’t touch all that corner....

Second there is a kernel idea:[11]

(PF7) S-A: And what I thought is I shouldn’t necessarily think of an enclosed building. Cause, I am in the middle of an open space. It would be a contradiction to place a formal building there.

Third there are some ideas about the integration of site and structure:

(PF8) S-A: since this is the view towards the sports field, things happen over there after 5:00 p.m. I have seen people playing softball and ah, frisbee, and a lot of spectacular kind of activities. And I might take the opportunity of using this. So that people can be out there looking at the field. The sunset is going to be, ah, watched. Ah, my guess is that it would be a good opportunity to use it. And then now that I think of it, I am saying, well, I could even, ah, sort of think of something, some structure that might use the roof of my post office to be on a sort of more privileged position towards the field....

Lastly, there are some functional ideas about the flow of mail.

(PF9) S-A: I have to be concerned about the pick up service.... Ah, I need to be able to service the machine from behind and to have enough space to do so....

Thus, he was unable to retrieve a single unified plan or schema to guide his subsequent problem solving. He had to start his problem solving with at least four schemas and

[11] The early generation and faithful development of a kernel idea is an intriguing phenomenon. It has been reported by several researchers, including Kant and Newell (1984) and Ullman et al. (1986). We do not have the space to pursue it here.

integrate them as he proceeded. This is a much more challenging situation than the one encountered by Subject-I.

Sometimes the domain-specific knowledge of the designer is insufficient to structure the problem. In such a case he first tries to use his general world knowledge; if this fails the problem may be avoided, abandoned or not even recognized. For example, the architect (Subject-A) had no experience in designing user-transaction interfaces, but he was explicitly requested to do so in the design brief. He chose to assume a "Versateller type interface." When pressed by the experimenter to provide further details, he gave the following "explanation" for avoidance:

(PF10) S-A: the philosophy of it is that I hate an interface which is not human.... Let's leave it open. It might be through a keyboard, through a menu where you have a multiple selection and you have a ah, sort of Versateller mode to answer....

3.1.3. Negotiation of Problem Space Boundaries

Constraints as they occur are not always desirable. Negotiation of problem space boundaries is an interesting resultant phenomenon exhibited by most of our subjects. It is an attempt to shift problem space boundaries. Often it is done to minimize search effort by transforming the problem to fit an existing plan or template. This seems to be the motivation behind Subject-I's attempt. Subject-I, based on past experience, believes that training programs need some minimal instruction interaction. The instruction he was requested to design on this occasion was to be completely self-contained (i.e., no instructor interaction). He attempted to make the current task conform to his normal mode of operation:

(PF11) S-I: Ok. We can't negotiate you, ah, considering bringing these people in, ah, in possibly two groups of five, after hours, paid overtime or something, or is this already....

Sometimes negotiation is also used to enlarge and complicate the problem. Subject-A attempts to do this. On viewing the small triangular site he has been given for the proposed post office, he is not content to just build a post office but wants to redesign the whole area.[12]

(PF12)

S-A: So, given the fact we have that triangle [i.e., the site for the post office] over there as a limit. And I cannot exceed that I suppose?

E: Right, that, that....

S-A: I have to take that for granted?

---

[12] The subject is standing on a 9th floor balcony and has a bird's-eye view of the site.

E: I, I would think so.

S-A: That's the boundary of. You do not allow me to, to exceed in, in my area of intervention?

E: No, I think you should restrict it to that.

S-A: So, I am constrained to it and there is no way I can take a more radical attitude. Say, well, look, you are giving me this, but I actually, I, I'd come back to the client and say well look, I really think that you should restructure actually the whole space, in between the building. I'd definitely do that, if that was the case. You come to me as a client, and come to me with a triangle alone, I will give you an answer back proposing the whole space. Because, I, I think the whole space should be constructed. So, that there is an opportunity to finally to plan and that space through those, ah, this building, open up Anthropology and, and plan the three buildings together. So, as to really make ah, this ah, a more communal facility....

The motive here is more difficult to speculate about. It could be a belief that this will result in a more effective artifact; a desire for a larger fee; exuberance and enthusiasm for rebuilding the world in one's own image; etc.

3.2. Extensive Performance Modeling

Four important aspects of the DTE converge to necessitate extensive performance modeling of the artifact (in its intended environment) in the design problem space.

1) Penalty for being wrong: It is a fact about the world that every action occurs in real time, consumes real resources, and has real consequences. In other words, it is impossible to set the world back as it was before the action. At best one can only take additional action (at additional cost) to remedy the situation, but traces of the original action will invariably remain. This is as true of bending one's little finger, uttering a sentence, walking to the grocery store, building a house or a freeway, or putting a man on the moon. The difference in each of these cases is in the cost and residue—the penalty for error. As the penalty for error increases, we respond by thinking through and anticipating as many consequences of an action as possible—*before acting*.

2) Autonomy of artifact: The artifact has an independent existence from the designer and must "make it on its own." The designer cannot be there to explain its significance or perform its function. For example, in the case of the stand-alone instruction, the instructional designer will not be in the classroom to respond to difficulties and questions of comprehension. He must anticipate the necessary interaction and respond to it in the structure of the artifact. Such anticipation/prediction requires extensive models of the artifact interacting in its intended environment.

3) Delayed/Limited feedback from world: Feedback from the environment is a major mechanism used by adaptive systems to enhance goal achievement in the

face of variable environmental factors. One of the most dramatic consequences of the structure of the DTE is that the feedback loop is delayed. The design is being developed between time *t* and *t+1* (see Fig. 1) but it does not interact with the world until time *t+3*. But this for all practical purposes is a point of no return. Resources have been expended and the damage has been done. The feedback from this point can not guide the designer in the current project, but only the next "similar" project. To guide the current problem solving the designer must simulate or generate his own feedback between times *t* and *t+1*.

4) Temporal separation of specification and delivery: There is a linear, temporal separation between artifact specification and delivery. In Fig. 1 the specification is complete at time *t+1* and the artifact constructed in the world at time *t+3*. Ideally the artifact is completely specified before construction begins.[13] This temporal separation enables the designer to model artifact performance—in the problem space or some external medium—to minimize damage and the expenditure of more substantive resources.

Performance modeling is necessitated by the first three aspects and enabled by the fourth.

Modeling is both internal and external to the problem space. Some of the possibilities, and the sequence in which they are used, are as follows: (i) entailments of designer's ICMs, (ii) scenario immersion, (iii) pictorial models, (iv) mathematical models, (v) mock ups, (vi) surveys, (vii) computer simulations, etc. Our subjects did not have the time or resources to make use of all these modeling devices—though they all pointed out when they would normally use them. They were basically restricted to their problem space and paper and pencil. This meant that they could take advantage of only the first four types of models. We will restrict our discussion to a few comments about the first two.

The designer's ICM of the world allows for quick and automatic inferences. We have already encountered an example in PF5 where Subject-I uses his "secretary ICM" to quickly evaluate whether to include certain material in the lessons. Such inferences do not seem to require any effort. They fall out automatically from the designer's idealized cognitive model of the world.

Scenario immersion is a more elaborate process whereby the designer pulls out a relatively concrete scenario from his past experience and immerses himself in it. Knowing how the scenario actually transpired, he draws upon similarities between the scenario and the current situation to calculate the entailments of the current situation. It is a strategy of both the first and the last resort. For example, we saw in PF2 how Subject-A evaluated a one possible spatial configuration of APTM machines by doing a mapping between it and a

[13] This is of course not always the case. Fast-tracking is a case of substantial parallel processing. But, even here, there are significant self-contained modules—and errors are expensive.

previously encountered similar situation, the consequences of which he has had first-hand experience. Subject-M, in determining the size and height of APTM machines wanted to do a formal study to see how people would use the machine. However, (perhaps knowing a formal study is not possible in the circumstances), he immediately and without prompting indulges in scenario immersion.

(PF13) S-M: Ok. I think [we need...] user group studies about how.... they would do the transaction. I think there is something about how...they're going to use it. Maybe, most student maybe riding bikes sometimes. Or most people, we expect them to walk, walk in. But sometimes maybe students [are] kind of lazy, or maybe they ride their bike or moped....

While their external models varied according to task demands and their pre-existing notational systems, the scenario immersion strategy was common across all subjects.[14]

3.3. Personalized/Institutionalized Evaluation Functions and Stopping Rules

It has been noted by many people (e.g., Rittel and Webber, 1974) that there are no right or wrong answers in design situations, but only better and worse ones. This has two interesting consequences at the level of the problem space. First, it means that evaluation functions are often personalized or at least institutionalized.[15] This is quite apparent in the above uses of ICMs and scenario immersion. Second, the point at which a design is complete is a function of cognitive and personal resources. Subject-I asked to stop because he was tired. Subject-M reported he could not proceed any further without doing a mock-up of the APTM. As we did not have the resources there for him to do so, he used this as a reason to terminate the session.

3.4. Limited Commitment Mode Control Strategy with Nested Evaluation Cycles

In section 3.2 we discussed the importance of performance modeling. Ultimately the purpose and value of this is to enable the designer to anticipate the performance of the artifact and the consequences of releasing it in the world. Since what matters is the performance of the final, complete artifact (at time $t+3$), one possible strategy is to delay evaluation until the specification is complete (at the end of time $t+1$). Evaluation at this point would certainly yield as good a value as possible, short of direct feedback at time $t+3$. But given the time, cost, and complexity involved in the design phase itself, it is neither optimal or feasible. That is, quite apart from the time and costs involved in generating a complete design and then having to scrap it and start all over again, it is a fact about adaptive systems

[14] Not only does the scenario immersion phenomenon play a crucial role in performance modeling, it also seems to be instrumental in generation. However, we do not discuss this aspect of it here.

[15] By 'institutionalized' is meant accepted by a group or organization with which the designer associates himself. For example, in the case of Subject-I, this means in-house company standards and practices. In the case of the architect, it might be some "movement" such as Bauhaus, Postmodernism, etc.

that they require continual feedback when engaged in any goal seeking endevour. It is simply not possible for people to work for months on end without having any indication as to the value and status of the work with respect to the goal. So not surprisingly, we found that our subjects did not wait until the artifact was completely specified to evaluate its performance.

Since the design unfolds in a quasi-linear sequence, generally starting with a kernel idea that is transformed and augmented until the final form emerges, another possible strategy is to evaluate components of the artifact as they are being generated. This would result is a linear sequence of short generate-evaluate cycles. While this is cognitively a very tractable strategy it can arrest design development by requiring strict adherence to earlier decisions. That is, a decision made at one point, while attractive in that local context, may be inappropriate in a later, more complete context. With this strategy one would be stuck with the earlier decision. Our subjects did not use this control strategy either.

Instead, all our subjects used a *limited commitment mode control strategy* (LCMCS) which incorporates the best of both worlds. It is cognitively tractable, enhances design development, and gives good evaluation results. It is necessitated by the essentially sequential nature of symbolic processing and made possible by the fact that the design phase is separate from, and prior to, the delivery phase.

If one looks at the design process at any given time, one finds that there are at least three contexts that the designer needs to attend to: (i) the component of the artifact currently being generated or focused on; (ii) the complete artifact in its current state (i.e., the design so far); and (iii) the projection of the artifact in its complete state (i.e., the final design). The LCMCS allows the designer to take each of these contexts into consideration.

As a first option, the designer can evaluate a generated or focused component on its own and make a decision to accept or reject it. For example, the instructional designer thought of including the component "start with basics and finish with more complex" in a subsection entitled "What will be Trained." He rejects it even before verbalizing it. (It surfaces only when the experimenter intervenes with his question.)

(PF14)

S-I: Ok, we've overviewed the course now just as far as the selling features. Now we're going to do a little bit of overview of what to expect. [writing: "What Will be Trained"] Ah, now what we will train. Ok, and we put that over.... [writing: "Six 1-hour Sessions"] We're going to, oh hell, that's bullshit.

E: What was bullshit?

S-I: Start with basics and finish with more complex. Well of course. What in the hell else would you be doing? I am not going to step you right off the end of the Titanic and ask you to swim....

What matters for present purposes is that the evaluation of the component was not done in the context of the design but strictly locally, on its own terms.

Second, the designer can evaluate a generated or focused component in the current context (i.e., the context of the design so far). This practice results in a better evaluation function and an increase in the number of options. He can choose to reject or accept the current component or he can choose to reject or modify some previous decision to make the current one acceptable. For example, at one point, Subject-I makes a decision to the effect that secretaries don't need to know about "waste baskets" (an icon used to delete computer files). A little further down he decides that they should know how to recover deleted icons. Then he realizes that the only way they can do this is if they know how to use "waste baskets." At this point he can simply reject the later decision of teaching the secretaries about recovering deleted icons, but instead he decides to undo the previous decision and include a section on "waste baskets." This now makes it possible to stick to the second decision of teaching about the recovery of deleted icons.

Finally, the designer can evaluate the generated or focused component in a later more complete context (at a later time), further increasing accuracy and options. In this situation, he can accept or reject the current component, as in the first case; modify some previous decision to make the current one acceptable, as in the second case; but in addition has the option to modify some future decision to make the current one acceptable. For example, Subject-A during his initial structuring phase had an idea for using the roof structure of the post office as a seating platform for viewing the sports field:

(PF15) S-A: I could even, ah, sort of think of something, some structure that might use the roof of my post office to be on a sort of on more privileged position towards the field....

But later when he calculated the size of the structure and realized how small it would be (i.e., reevaluated it in the current, more complete context), he abandoned the earlier idea:

(PF16) S-A: The thought that I had before, that I might use, the envelope itself, the form, the roof, ah, the walls, to, to implement some sort of, ah, landscape element, so as to have a major view towards the sports field. That I am denying now.... I really am coming back to this and seeing that, after all, I won't have huge lines. After all I just have 3 booths and a roof. That's what I really have here. So, I'm sort of seeing the extent, ah, to which this problem will be heading to.

3.5. Making and Propagating Commitments

A design task is not complete until the artifact is completely specified. A specification is a complete, procedural, and declarative description, which when executed by an external agent results in the construction of the artifact. It is not sufficient to wave one's hands and talk about the artifact in some general terms. One must actually make, record, and propagate decisions as one proceeds, otherwise one will have nothing to show at the end of the session. Each of our subjects did explicitly record and propagate their decisions.

An interesting tension exists between the LCMCS and the need to make commitments—between not acting and acting rashly; between being Hamlet and being Laertes. Designers are adept at negotiating this tension between keeping options open for as long as possible and making commitments.

3.6. Solution Decomposition into Leaky Modules

A major cognitive strategy for dealing with large complex problems is through decomposition. Decomposition was a major step in the normative models of the design methodology movement (e.g., Alexander, 1964). It has since been questioned and discredited as overly simplistic and even harmful to the design process. As Alexander (1965) subsequently noted, "A city is not a tree; it is a semi-lattice." Or in Simon's (1962, 1973b, 1977) vocabulary, the world is only *nearly* decomposable. But what is to be made of the *nearly*? Some interpret it to mean that one can not talk about solution decomposition in any significant sense. Others assume it can be ignored and continue to do clean, tree-like decompositions (e.g., Brown and Chandrasekaran, 1985).

Our data shows extensive decomposition. Each of our subjects quickly and automatically decomposed their problem and developed their solution in a dozen or so modules. Subject-M's modules were things like key pad, screen, stamp dispensary, parcel depository, and weighing mechanism. The decompositions were discipline specific. They were not invented anew for the problem but seemed to be part of the designers' training and practices. However, equally important, the subjects did not treat the modules as strictly encapsulated but rather as *leaky modules*. A decision made in one module could have consequences in several others. The subjects seemed to have some sort of ongoing monitoring process that looked for interconnections across modules.

The subjects dealt with the problem of "leaks" in one of two ways. One method was to plug the leaks by making functional level assumptions about the interconnecting modules (see section 3.7). This method enabled them to bring closure or encapsulation to a module and make it cognitively tractable. For instance, in designing the first lesson, Subject-I did not have to attend to the details of the third lesson. It was sufficient to make some high-level functional assumptions about it. Similarly, in considering the height and angle of the APTM key pad, Subject-M did not attend to the details of the stamp dispensary. A second method of dealing with "leaks" was to engage in opportunistic behavior—to actually put the current module on "hold" and to attend to some of the interconnecting modules right there and then.

3.7. Abstraction Hierarchies Mediate Transformation of Goals to Artifact

The input to the design process is generally a set of goals or intentions. The output of the process is generally a specification of an artifact. The goals come substantially from the client (though are elaborated in discussion with the designer) and are a statement of the behavior he wants the artifact to support. The artifact specifications are substantially generated by the designer (though the client's brief may provide some guidelines at the level of the artifact) and specify those aspects of the artifact that he considers to be causally relevant in the given circumstances. Conceptually or logically, it is tempting to say that the transformation from goals to artifact specifications is mediated by functional specifications (see Fig. 3). On this account one gets a story whereby the intentions are carried out *by means of* the functioning of the artifact, and the function is carried out *by means of* the causal structure of the artifact. Both function and causal structure have to fit the intentions, but they are only constrained, not determined, by them. In fact the intentions constrain (underdetermine) function, and function constrains (underdetermines) causal structure (see Fig. 3).

Fig. 3 approx here

Such explicit mediation is sometimes visible in our data. For example, Subject-M, when determining the components and configuration of the APTM, began with a very functional vocabulary:

(PF17) S-M: I think that functioning-wise we have some criteria. Ah, it's supposed to fulfill the requirement of user to purchase the stamps, mail the letters, and weigh parcels and mail it. Certainly there also will be register, should be something that can do the function for registering letters. And ah, certainly we expect it to be user-friendly and without requiring any training, and transparent to user....

At this point there is no indication of how these functions will be realized. A few minutes later they are mapped onto device components on a one-to-one basis:

(PF18) S-M: So I would assume there is input and output devices...and we got to also have depository...for letters and parcels, and something for...delivering device for stamps.... And we also need some device to weight....

But generally, the story that emerges from the data is not quite so clean and is closely connected to the near-decomposability phenomenon of the previous section. The functional specifications and the causal structure specifications are not two distinct ontological categories but the same category under different descriptions. Functional specifications treat the artifact, or some component of it, as a black box and attend only to the input and output. They basically answer the question "what function will this artifact, or this part of it, accomplish?" Artifact specifications detail the causally efficacious structure of the artifact. They answer the question "how is the function to be accomplished?" For example, during the course of designing the first lesson in the training package, Subject-I worked with several different modules, interconnected in various ways. Some of these modules were: lessons, sections, subsections, paragraphs, sentences, and the choice and arrangement of lexical and grammatical elements. This corresponds to what we called a solution decomposition in the previous sub-section. In addressing each of these modules the designer may choose to do it at various levels of abstraction or detail. The functional-causal structure distinction is just a special case of this abstraction process.

The status of any module vis-à-vis the functional-causal structure distinction depends on whether a *what* or *how* question is asked of the module. For example:

- What is the function of this lesson?
- How is it going to achieve this function?
(By means of these sections.)
- What is the function of these sections?
- How are they going to achieve their function?
(By means of these subsections.)
- What is the function of these subsections?
- How are they going to achieve their function?
(By means of these paragraphs.)
- etc.

In asking the different questions, the designer is choosing to attend to different levels of detail. Ultimately, this regress must bottom out at a level where the artifact is completely specified. There are some interesting observations to be made as to where it bottoms out and the number of levels a designer explicitly considers.

Our data indicates that the number of levels explicitly attended to by a designer is a function of his experience and familiarity with the task, availability of relevant knowledge, and personal preferences. The more routine a task is, the more quickly and directly the designer can get to the low-level details, if he so chooses. He knows by experience what type of artifact supports what type of goals and does not have to reason through it via "first principles." Of our three subjects, Subject-I found the task quite routine and traversed the abstraction hierarchy quite quickly. Subject-M, as noted earlier (PF17 & PF18), did cascade down several levels of function-artifact specifications. Subject-A, when confronted with designing the automated mail-handling system for the post office, dealt with it in strictly functional terms. He simply did not have the knowledge to specify lower-level details.

However, Subject-A consciously did something that was rather interesting. In determining the configuration and location of the post office building, he purposefully stayed at a highly abstract level for an extended period of time so as not to crystalize or commit himself too soon to low-level details:

(PF19) S-A: I am constantly referring to that sketch by the way. As you can see it's ah, although it's the lousiest of them all, it still, still something that I, I, I, and I am not willing to do any other sketch at the moment. Because I, I am really, trying to figure it out and I am doing it at an abstract level. So, that...the flow is not affected by the crystalization of an idea....

Thus training, personal preferences, style, and a number of pragmatic factors can affect the number of abstraction levels that are considered and how quickly one descends the hierarchy. This point is tied to the personalized evaluation function and stopping rules observation discussed in section 3.3. Descending too soon or not descending[16] at all is a common mistake of novice designers. This relates to the earlier point about the tension between the LCMCS and the making of commitments.

The level of detail at which the designer chooses to bottom out depends on professional conventions and standards, personal preferences, style, and a host of pragmatic factors. Subject-I, for example, did not stop at the specification of the actual words and sentences but went on to also specify page layout and typeface. But he did not have to stop there either. He could also have specified the chemical composition of the ink or the tensile strength of the paper. He chose not to. He left it as someone else's responsibility. He simply assumed that they would function in the "normal way"—that the ink would not dissolve and the paper would not fall apart—and did not feel the need to provide any specifications for them. Every design profession has some conventions in this respect, and there is always some freedom either way that the designer may exercise at his discretion.

3.8. Use of Artificial Symbol Systems

Designers often use artificial symbol systems to filter and focus information and augment memory and processing. These systems are so crucial for the problem-solving process that if they do not pre-exist they have to be invented before the design can proceed.[17] Their use and importance can be seen most dramatically in the case of architecture. It is possible to recognize at least seven different symbol systems (six of them artificial) in the architect's repertoire (see Fig. 4). Roughly they are (i) natural language, (ii) topology ("bubble diagrams"), (iii) similarity geometry (rough sketches), (iv) Euclidean geometry (plans, elevations, sections), (v) affine geometry (isometrics), (vi) projective geometry (perspectives), and (vii) models or mockups. (Admittedly, the correspondence between the formal geometries and the architect's various drawings is only approximate, but it does serve to highlight the richness and variety of artificial symbol systems that are actually used.)

Fig. 4 approx here

The symbol systems from topology to Euclidean geometry form a sort of a hierarchy. In fact, they map onto and support the abstraction hierarchy discussed in the previous

[16] One of our instructional designer subjects stayed at a very high abstract level and refused to come down. The result was that he had no artifact specifications to show at the end of the period.

[17] One of our subjects realized that he did not have an appropriate symbol system for the development and specification of the artifact and tried to develop one as he went along. The development of symbol systems can be seen on an institutional scale in the case of the emerging scripting and mazing systems for interactive videodisc.

section. It is possible to make and represent distinctions at the lower levels which the higher levels do not support. Similarly, it is possible to make and represent distinctions at the higher, more abstract levels, which can only be made at the lower levels in a hidden or obscure fashion. For example, metric distinctions are preserved in Euclidean geometry but not in topology, and while every proposition of topology is trivially true in Euclidean geometry, topology does not come into its own until one abstracts away from metric and other details.

Subject-A in his two-hour session used the symbol systems of natural language and similarity geometry. There are two interesting things to note in his use of these systems. (1) Moving between the systems *automatically* commits him to a level of detail by selectively highlighting and hiding information. (2) Within a single symbol system he constructs multiple representations of the artifact. In both cases we want to note that these external representations are not for communicating something after the fact. They serve an indispensable role in the generation, evaluation, and decision-making process. Once decisions are made, symbol systems serve to record and perpetuate them.

As an illustration of the first point, consider the following sequence of protocol fragments and the accompanying diagrams in which Subject-A determines the form and configuration of the post office building:

(PF20) S-A: But I could eventually have one single space, where all the, ah, mail is, is delivered. Which eventually would open up in a single way and have the booths orbiting around it. So that a given line might occur here, another one here, and another one there.... Now what I see is a more enclosed to itself structure. By that I want to say is that there is an inner core and then this roof extending around it....

Along with this verbalization was the concurrent realization of the geometric form in Fig. 5.

Fig. 5 approx here

The relationship between the verbalization and the diagram is a one-to-many mapping. The diagram contains several elements which the verbalization does not. It contains and makes very explicit information on the rough size (relative to users) and shape of each unit, the configuration of the units, and how the designer envisions the lines forming. This is not an accident. It is simply not possible to draw the artifact in similarity or Euclidean geometry without making commitments on these issues, whether you are ready to or not.[18] In fact, a few minutes later, while examining Fig. 5, Subject-A expresses surprise when he realizes the full extent of his commitment and commences to modify it.

[18] In actual similarity geometry, size, of course, is not preserved.

(PF21) S-A: I don't want to, to affect the type of line that might happen. Why did I draw this, ah, like something that sticks out? Ah, no. I actually want to minimize even more. So, the way I see it now is I'll have to, ah, the booths [are], conceived, probably in such a way that, the element itself is, is really minimized as, as, ah, formal or volumetrical type of ah, intervention. We have a main structure and 1, 2, 3 interfaces, and the main axis.... This seems to work well....

Accompanying this verbalization is the diagram in Fig. 6. While substantially different from the diagram in Fig. 5, it is consistent with the original verbalization in PF20.

Fig. 6 approx here

Each sketch highlights information not explicit in the verbal description. As the information is explicated, it can be attended to in subsequent generate-evaluate cycles. Much of Subject-A's problem solving involves traversing abstraction levels via the corresponding symbol systems. Learning to traverse this hierarchy has some serious consequences for design development and crystalization. One must know when to use which system so as not to commit oneself too soon and thereby prematurely arrest design development. At the other extreme one must learn not to stay at the higher abstract levels for an overly extended period of time and thereby produce nothing. This observation is of course related to the earlier mentioned tension between the LCMCS and the necessity to make commitments.

To illustrate the second point, we note that Subject-A constructed four distinct representations of the artifact within the system of similarity geometry: site plans, floor plans, elevations, and sections. Furthermore, he attended to the various aspects of the building as they were being drawn; for example, he calculated the vertical dimensions of the structure when drawing the elevation (see Fig. 7), not when working on the plan:

(PF22) S-A: So, maybe, ah, I should go on to a section now and see how this is ah, happening, with more precise measures [meaning roof overhang and the glare on the monitors].... Ah, 6 feet I envisioned this to be very low anyway.... Probably 2.4 meters, 2.2 meters even.... So I'd say that 8 feet will be the maximum height.... Ah, probably we need about 2 or 3 feet to have all the equipment.... And the lower part of the display monitor and, and keyboard will be perhaps 3 feet, 3.5 feet perhaps from the ground level.

Fig. 7 approx here

----

4. CONCLUSION

This study has identified eight significant invariants in the design task environment and characterized their impact of the design problem space. Fig. 2 serves as a succinct summary of both our strategy and findings. To repeat, our major empirical findings are the following characteristics of the design problem space: (A) extensive problem structuring, (B) extensive performance modeling, (C) personalized/institutionalized evaluation functions and stopping rules, (D) a limited commitment mode control strategy, (E) the making and propagating of commitments, (F) solution decomposition into "leaky modules," (G) the role of abstractions in the transformation of goals to artifact specifications, and (H) the use of artificial symbol systems. But in addition to noting these features, we also made "explanatory connections" between them and the invariant features of the design task environment. We make no claim for completeness and fully expect our characterization to grow and evolve as we examine more of our data. But we do expect our strategy of viewing design as a radial category, taking the design task environment seriously, and examining data from several design disciplines to be of continuing value in the future. At this stage, we cautiously suggest that while singularly these features may be found in non-design problem spaces, collectively they are the invariant hallmarks of the design problem space. We now conclude the paper by indicating some implications for CAD systems, noting some methodological shortcomings and suggesting directions for future research.

4.1. Implications for Computer-Aided Design Systems

Typically, CAD systems provide designers with a variety of tools for modeling the anticipated performance of an artifact during the design process. Our characterization of generic design and our empirical observations suggest that there are several ways in which such systems could be enhanced.

We noted that design characteristically involves problems with many degrees of freedom, requiring substantial collection of information, problem structuring, and negotiation. Much of this information comes from external sources or the prior experience of the designer. At first blush, hypertext tools would seem to be appropriate for such activities. However, as noted by Halasz (1988), making hypertext systems that permit cheap input and restructuring is still a major research issue.

Design inherently involves the use of design abstractions, nested generate and evaluate cycles, and a limited-commitment mode control strategy. This suggests that designers should be able to inexpensively specify design abstractions and evaluate designs at any level of abstraction. The CLU language for software design is an attempt along these lines (Liskov and Guttag, 1986). It is essentially a variant of an object-oriented programming language that allows software designers to develop procedural and data abstractions and specify the preconditions and entailments of these abstractions without immediate concern for their implementation. The fact that designers appear to mix formalisms in their representations of artifacts suggests that we have substantial work to do in this area.

Representation is an important issue in itself. First generation CAD systems viewed the designer's notes and drawings only as communicative devices. Our studies confirm the findings of Ballay et al. (1984) and Ullman et al. (1986) that this is simply not the case. The designer's notes and drawings play a crucial role in design development by selectively focusing and filtering information and augmenting memory and processing. This speaks for the need to develop computational environments which can support a wide range of symbol systems.

Finally, we should remark on the potential role of AI in CAD systems. AI is especially appropriate for propagating the entailments of closed-world models, as is typically done in theorem-proving programs or problem solving programs that deal with well-structured problems. It does not fare as well in tasks with changing world models; ones that are continually influenced by knowledge brought in from the external world or from past experience. This would seem to imply that we should not expect AI to provide highly automated design systems for anything but the most routine and well-structured problems that arise during design. However, research on hierarchical planning could provide tools for representing and evaluating abstract design plans. Research in knowledge acquisition tools could influence the development of CAD systems that acquire new design abstractions and evaluations. Research in case-based retrieval and reasoning could provide tools to augment designers' use of prior knowledge in design. Intelligent advice or help systems that use knowledge of particular design tasks, and on-line "pattern books" might be particularly useful aids for novice designers or as warehouses for the design knowledge in particular disciplines or institutions.

4.2. Principle Shortcomings and Limitations

As the work currently stands, there are three principle shortcomings. The first is that the whole analysis is based substantially on three protocols, one each from three of the many design disciplines. In the short term we justify our experiment design by noting that the methodology used is qualitative rather than quantitative. It does not require large numbers of subjects. As has been argued by Anzai and Simon (1979) there is much to be gained by the detailed analysis of a single protocol. Over the long term, we recognize the shortcoming and are continuing to analyze additional protocols.

The second shortcoming is that we have not used a formal procedure for coding the protocols. Neither has there has been any independent coding of the protocols. Again, over the long term, we recognize this shortcoming as serious. In the short term, we note that the categories and conclusions were arrived at through much argumentation and compromise with colleagues with first-hand knowledge of the data.

The third shortcoming is that only design problem protocols have been examined. This only allow us to make the weak claim that we have identified certain invariants in the design problem space. It does not permit the additional claim that these invariants are not (collectively) found in nondesign problem spaces. This latter claim is desirable for the motivation of generic design as a useful theoretical construct. But it requires the examination of nondesign protocols. The comparison of nondesign problem spaces with design problem spaces is a matter of ongoing concern.

4.3. Future Work

This investigation has been a first-pass, breadth-first look at design problem solving. We have tried to lay out the major pieces of the design problem space and explain or justify them by an appeal to the design task environment and the structure of the IPS. A logical extension of this work would be to push the analysis further and to derive a process model of design from it.

In concentrating on the big picture, we have had to resist the temptation to delve deeply into any single feature of the problem space. Of particular interest to us are the phenomena of scenario immersion, leaky modules, and the use of artificial symbol systems. Each of these promises to be a rich and intricate field of study.

Also, we have not said anything about the differences in the problem spaces of our three subjects. We have noticed some interesting differences in their knowledge bases, the external symbol systems they use, and their cultural and professional values and practices. However, any conclusions in this regard must wait until we gather and analyize additional data.

References

Akin, Omer, ‘‘An Exploration of the Design Process,’’ *Design Methods and Theories*, vol. 13, no. 3/4, pp. 115-119, 1979.

Akin, Omer, *Psychology of Architectural Design*, Pion Limited, London, 1986.

Alexander, Christopher, *Notes on the Synthesis of Form,* Harvard University Press, Cambridge, MA, 1964.

Alexander, Christopher, ‘‘A City is not a Tree,’’ *Architectural Forum*, vol. 122, pp. 58-62, April-May, 1965.

Anderson, John R., ‘‘Acquisition of Cognitive Skill,’’ *Psychological Review*, vol. 89, no. 4, pp. 369-406, 1982.

Anzai, Yuichiro and Simon, Herbert A., ‘‘The Theory of Learning by Doing,’’ *Psychological Review*, vol. 86, no. 2, pp. 124-140, 1979.

Archer, L. Bruce, ‘‘The Structure of the Design Process,’’ in *Design Methods in Architecture*, ed. G. Broadbent & A. Ward, George Wittenborn Inc., N.Y., 1969.

Ballay, Joseph M., Graham, Karen, Hayes, John R., and Fallside, David, ‘‘CMU/IBM Usability Study: Final Report,’’ Communications Design Center Technical Report No.11, Carnegie-Mellon University, 1984.

Brown, David C. and Chandrasekaran, B., ‘‘Knowledge and Control for Design Problem Solving,’’ Tech. Report July 1985, Laboratory for Artificial Intelligence Research, Dept. of Computer and Information Science, The Ohio State University, 1985.

Cross, Nigel, ‘‘Understanding Design: The Lessons of Design Methodology,’’ *Design Methods and Theories*, vol. 20, no. 2, pp. 409-438, 1986.

Eastman, Charles M., ‘‘On the Analysis of Intuitive Design Processes,’’ in *Emerging Techniques in Environmental Design and Planning*, ed. G. Moore, MIT Press, Cambridge, MA, 1969.

Eastman, Charles M., ‘‘Recent Developments in Representation in the Science of Design,’’ *ACM & IEEE 18th Design Automation Conference*, pp. 13-21, 1981.

Ericsson, K. Anders and Simon, Herbert A., *Protocol Analysis: Verbal Reports as Data*, The MIT Press, Cambridge, Massachusetts, 1984.

Greeno, James G., ‘‘Natures of Problem-Solving Abilities,’’ in *Handbook of Learning and Cognitive Processes, Volume 5: Human Information Processing*, ed. W. K. Estes, Lawrence Erlbaum Associates, Hillsdale, N.J., 1978.

Halasz, F. G., ‘‘Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems,’’ *Communications of the ACM*, vol. 31, pp. 836-852, 1988.

Hayes, John R. and Simon, Herbert A., ‘‘Understanding Written Problem Instructions,’’ in *Knowledge and Cognition*, ed. L. W. Gregg, Lawrence Erlbaum Associates, Potomac, Maryland, 1974.

Jeffries, Robin, Turner, Althea A., Polson, Peter G., and Atwood, Michael E., ‘‘The Processes Involved in Designing Software,’’ in *Cognitive Skills and their Acquisition*, ed. J. R. Anderson, Lawrence Erlbaum, Hillsdale, N.J., 1981.

Kant, Elaine and Newell, Allen, ‘‘Problem Solving Techniques for the Design of Algorithms,’’ *Information Processing and Management*, vol. 20, no. 1-2, pp. 97-118, 1984.

Kant, Elaine, ‘‘Understanding and Automating Algorithm Design,’’ *IEEE Transactions on Software Engineering*, vol. SE-11, no. 11, pp. 1361-1374, 1985.

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

Lawson, Bryan R., ‘‘Cognitive Strategies in Architectural Design,’’ *Ergonomics*, vol. 22, no. 1, pp. 59-68, 1979.

Liskov, B. and Guttag, J., *Abstraction and Specification in Program Development*, MIT Press, Cambridge, MA, 1986.

Newell, Allen and Simon, Herbert A., *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, N.J., 1972.

Newell, Allen, ‘‘Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category,’’ in *Attention and Performance VIII*, ed. R. S. Nickerson, Lawrence Erlbaum, Hillsdale, N.J., 1980.

Perkins, D. N., *Knowledge as Design*, Lawrence Erlbaum Associates, Publishers, Hillsdale, N.J., 1986.

Reitman, Walter R., ‘‘Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems,’’ in *Human Judgements and Optimality*, ed. M. W. Shelly II & G. L. Bryan, John Wiley and Sons, N.Y., 1964.

Rittel, Horst W. J. and Webber, Melvin M., ‘‘Dilemmas in a General Theory of Planning,’’ *DMG-DRS Journal*, vol. 8, no. 1, pp. 31-39, 1974.

diSessa, Andrea A., ‘‘Knowledge in Pieces,’’ *Constructivism in the Computer Age*, pp. 1-24, 1985. (Fifteenth Annual Symposium of the Jean Piaget Society, Philadelphia, June 1985.)

Simon, Herbert A., ‘‘The Architecture of Complexity,’’ *Proceedings of the American Philosophical Society*, vol. 106, pp. 467-482, 1962.

Simon, Herbert A., ‘‘The Structure of Ill-Structured Problems,’’ *Artificial Intelligence*, vol. 4, pp. 181-201, 1973a.

Simon, Herbert A., ‘‘The Organization of Complex Systems,’’ in *Hierarchy Theory*, ed. H. H. Pattee, G. Brazileer, N.Y., 1973b.

Simon, Herbert A., ‘‘How Complex are Complex Systems?,’’ *Proceedings of the 1976 Biennial Meeting of the Philosophy of Science Association*, vol. 2, pp. 507-522, 1977.

Simon, Herbert A., *The Sciences of the Artificial (Second Edition)*, MIT Press, Cambridge, MA, 1981.

Steier, David M. and Kant, Elaine, ‘‘The Role of Execution and Analysis in Algorithm Design,’’ *IEEE Transactions on Software Engineering*, vol. SE-11, no. 11, pp. 1375-1386, 1985.

Thomas, John C. Jr., ‘‘A Design-Interpretation Analysis of Natural English with Applications to Man-Computer Interaction,’’ *International Journal of Man-Machine Studies*, vol. 10, pp. 651-668, 1978.

Thomas, John C. and Carroll, John M., ‘‘The Psychological Study of Design,’’ *Design Studies*, vol. 1, no. 1, pp. 5-11, 1979.

Ullman, David G., Stauffer, Larry A., and Dietterich, Thomas G., ‘‘Preliminary Results of an Experimental Study of the Mechanical Design Process,’’ Tech Report 86-30-9, Dept. of Computer Science, Oregon State University, 1986.