Cognitive processes in architectural design problem solving

Chiu-Shui Chan

Department of Architecture, Carnegie Mellon University, Pittsburgh, PA 15213, USA

The purpose of this research is to explore the cognitive mechanisms involved in architectural design problem solving. Special attention is given to how design constraints operate throughout the whole process. A proposed cognitive model describes the general cognitive procedure. Schema theory applied from cognitive psychology simulates the representation of domain-specific knowledge. The application of production system detects the control strategy, and a laboratory experiment of a residential design is conducted for protocol analysis. Evidence collected from the analysis yields two results. First, that design solutions are generated by means of activating the design constraints and associated rules in memory. Second, that the design ability is determined by the ability to select rules in schema as well as the ability to develop new schema to test newly generated design units.

Keywords: semantic, schema, production system, protocol analysis

Architectural design is one kind of problem solving which primarily involves a series of actions that must be performed in order to solve a design problem. Typically, a designer does not know in advance what the goal state is or whether a candidate solution is indeed a solution. The design problem is therefore categorized as an ill-defined problem[1–3]. Current research in design as an ill-defined problem has studied problem structuring[4], solution structure[5] and representation[6].

Design constraint has been used in space-planning tasks to select and arrange elements in a two- or three-dimensional space[7], and to study the interaction of problem-defining and problem-solving[8], or problem-structuring and problem-solving[9]. Simon indicates that design constraint is one of the influential factors on style in design[10]. But no case study has been done, nor any experiment conducted, to observe and to support such a hypothesis. This paper does not intend to formally prove this hypothesis at the present time, but to provide a foundation for a further study on style. Hence, the cognitive mechanisms involved and the role that design constraints play in architectural design problem solving are the concentration area of this research. The basic assumption in this study is that humans are processors of information, and that human thinking can be explained by means of the information processing theory. Human beings, therefore, are treated as information processing systems[11].

The basic concept in this research is built upon the observations made by Eastman[12,13], Simon[2,10], and Akin [14,15] about the cognitive processes in design. Some theories from cognitive psychology are applied to the construction of a conceptual framework. Among them, schema theory is applied to knowledge representation, and problem solving theory is applied to describing the design processes. The study also includes an experiment conducted for protocol analysis. Based on the protocol transcriptions, production systems are made to simulate the control structure. Thus, the developed framework is actually a cognitive model which is capable of mapping the whole design process. Results obtained are expected to verify the model and to illustrate the cognitive activities and thinking phenomenon occurring within the design process.

ARCHITECTURAL DESIGN PROBLEM SOLVING

According to Newell and Simon[11],

> a person (is) confronted with a problem when he wants something and does not know immediately what series of actions he can perform to get it.

Usually, a problem contains the initial situation of the problem solver and is referred to as the initial state. A goal state is the stage at which the problem has been resolved. The process of problem solving from initial state to the goal state can be modelled as a series of transformations generating a sequence of problem states. A problem state is a particular stage in which a problem solver knows a set of things, and is referred to as a knowledge state. The various states that the problem solver can achieve are called problem spaces. The various ways of changing one state into another are operators*.

In architectural design, the problem space includes the following components.

- A set of design units which is either given initially by clients as the design programme or brief, or is generated by the designer at any intermediate problem state. The design units are all physical elements of building components that are considered or manipulated during problem solving. For example, a design unit may be a living room, a dining room or a bathroom in a residential design.
- A set of operators which is not specified by clients but is a part of the designer's knowledge base. The operator is anything that changes the knowledge state. It can be arithmetic rules for numerical calculation, or a set of rules for allocating or generating a design unit.
- A set of design constraints that is specified by clients or generated by the designer. For example, a design constraint can be the limitation of total floor area.
- A goal in which the designer finds an object satisfying all of the constraints.

Based on the definition, a problem space can be formulated as:

$$\text{Problem space} = \{\{\text{goal}\}, \{\text{design unit}\}, \{\text{operator}\}, \{\text{constraint}\}\}$$

The knowledge state in design problem solving is a stage in which the designer knows the design unit, design constraint and applied rules. Then a state can be moved forward by applying the rules which satisfy the set of constraints.

*These definitions are taken from Anderson[16], and Newell and Simon[11].

A COGNITIVE MODEL

When a designer works on a problem space and searches for solutions, the involved cognitive activities can be modelled as follows.

A design task can be broken down into a sequence of goals. The generation of goals derives either from a goal plan that is stored in memory or from a perceptual-test. The means of selecting a goal to work on is referred to as the control strategy. The goal plan contains a sequence of goals that the designer must know in order to process the design task, and must achieve in order to get the design problem into the final goal state. In accomplishing a goal, the designer manipulates a set of design units. A package of knowledge about the design unit called a schema, which contains associated design constraints and rules for application, is stored in a knowledge base as a part of the designer's long-term memory. By taking a set of design units and retrieving its associated schemata, design solutions for a particular goal are generated and tested. This process can be illustrated within a simplified diagram in Figure 1. By repeating the process (taking a goal, activating a design unit, retrieving a set of associated schemata, applying a rule to search for a solution and then testing the solution), the design problem gradually moves toward the final goal.

This model contains several key components: knowledge base, control strategy, design constraints and search. The following sections will briefly discuss each.

Architectural knowledge base

Knowledge has been categorized as declarative know-

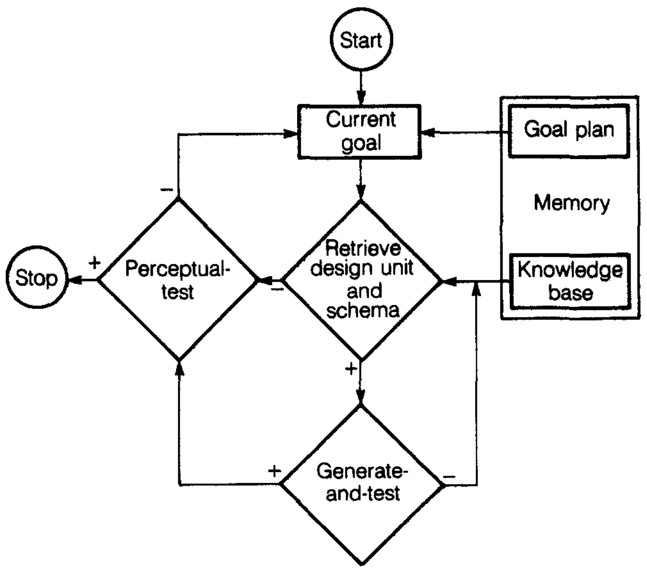


Figure 1. A general model of the design process

ledge, which comprises the facts we know, and procedural knowledge that comprises the knowledge of how to perform[16–18]. Declarative knowledge is concerned with knowledge as static information. Its representation has been proposed by semantic network theory[19,20]. Procedural knowledge is the knowledge of knowing how to perform a task. In performing a task, the declarative information is transformed into a procedural form. The procedural knowledge is commonly represented by production systems[11], an extensively developed and a widely accepted representation of human cognitive skills.

Schema theory

Rumelhart and Ortony[21] have argued that semantic network theory is not able to account for the ability to organize, summarize and retrieve information about connected sequences of events. They suggest that schema theory, which stems from semantic network theory, more appropriately represents the knowledge in general. The concept of schema is that all knowledge is grouped into units and these units are schemata. Embedded in these groups of knowledge is, in addition to the knowledge itself, information about how this knowledge is to be used[21,22]. Hence, a schema holds both declarative knowledge and procedural knowledge. Other characteristics of schemata include:

- schemata have variables and associated knowledge about the variables and their interrelationships
- schemata can be embedded, one within another; a schema is a network of subschemata
- schemata represent knowledge at all levels of abstraction
- schemata represent knowledge rather than definitions[21]

These characteristics explain that the schema can be seen as a data structure representing the generic concepts stored in memory.

In architectural design, knowledge can be represented by a hierarchical semantic network[14,15]. Since a designer must handle design units during the process of design, design units are subjects of the processing of design information. Therefore, it is appropriate to represent nodes in the semantic network by design units. Design units in the network are grouped according to related architectural functional relationships. A designer must have knowledge of the general components (design units) of a building as well as generic knowledge of what they are and how to design them. Therefore, by the application of schema theory, it is assumed that a set of schemata which contains a large amount of information (design knowledge) is associated with each design unit in the semantic network. A schema in the net consists of variables (design unit), the value of the variable (attributes) and knowledge about how to use it. All the pieces of knowledge associated with design units are hierarchically organized and the whole structure is called the knowledge base.

Design constraints

As described previously, a rich set of schemata associated with a design unit is embedded in the knowledge base. Among them is a set of schemata called design constraint schemata which are assumed to be the most important ones used in the course of design. Before discussing the importance of schemata to the design task, the definition of design constraint and related notions are explained.

Definition

The design constraint is analogous to the problem constraints defined by Reitman as 'attributes of objects'[23]. In Simon's definition, constraints are as 'bounds on the magnitude of certain variables'[10]. In this research, the design constraint is defined as certain requirements that must be fulfilled in order to design a design unit or a group of design units. This definition is similar to that used by Eastman[13], or design parameters used by Akin[9,14].

Design constraint schemata

A design constraint schema which is bound to a design unit contains the following components.

- *An identifier.* The identifier is the name tag of the constraint.
- *A variable.* A schema may have one or more variables. The variable is a design unit. If there are two or more variables, then a cross-reference among the variables is developed.
- *A set of rules.* These rules apply to knowing how to satisfy the constraint or how to find out the value of the constraint. Design knowledge is embedded in this part and can be represented by a set of production systems.
- *A value of the variable.* The value results from the application of rules and is bound to the design unit. The value of the variable may be a number, a list structure representing facts, or the topological and geometric magnitude of the design unit.

Whenever a design constraint schema is evoked, a copy of the schema is made in order to bind a particular configuration of values to a particular configuration of variables at a particular moment in time. This is the notion of an instantiation of a schema[22]. The value of a schema is generated by the evaluation of the associated set of rules in order to fit the existing situation, and this value is then bound back to the variable. Such a schema (which has an updated value) is called an instantiated schema. New knowledge is therefore accumulated by the instantiation of schemata.

Categories of constraints in design processes

Design constraints can be classified into two kinds. The first kind includes global constraints applicable to a group of design units. They may be given by clients or generated by the designer at an early design stage, and are carried through the whole design process. The second kind is local constraints or what Simon has called autonomous constraints[10]. The autonomous constraints are neither implied by the initial problem requirements nor provided by the client. They are bound to a unique design unit and are a rich set of design repertoire stored in the designer's long-term memory. Only a small subset of autonomous constraints may come into active use in the course of designing any single object. They will be evoked by particular situations that arise in the course of the design, retrieved from memory, and then applied.

One of the characteristics of an ill-defined problem is its huge problem space. In order to save the searching effort in the large problem space, a designer would introduce a design constraint to reduce the problem space for solution generation. In order to find an optimal solution, the designer would also introduce other constraints, which have been described as criteria by Akin *et al.*[9], to test the generated solution. For these reasons, design constraints are crucial in design problem solving.

Control strategy

When the problem space becomes more complex, problem solvers are more likely to have a plan developed beforehand. The plan has been defined by Miller *et al.* as a hierarchical process that controls the order in which a sequence of operations is to be performed[24].

Goal plan

Described in the model, goals are developed from two sources. One is from a goal plan, the other is from perceptual-test. In architectural design, designers have a general design method, as described by Heath[25], stored in their long-term memory called a general goal plan. This goal plan consists of a sequence of general goals which the designer must accomplish that will guide the design process. During the process, the designer may perceive a potential problem that he or she must solve at a particular knowledge state; a subgoal is formed accordingly. The operational relationships between a subgoal and the goal that a designer has reached is better explained by the concept of goal stack.

Goal stack

The short-term memory is assumed to have the form of a stack which contains goals[26]. The goal plan in long-term memory holds a list of symbols representing goals. The first symbol at the goal plan is activated and therefore held in short-term memory as the current working goal. If a goal cannot be accomplished, a new subgoal will be developed and activated on short-term memory and the previous goal is pushed back to the goal stack and stored.

Perceptual-test

The design solution accumulates from state to state, and information presented in external display is changing accordingly. A designer must gather information about the problem situation from time to time and this is done by perception. Research studies on perception in problem solving have dealt with the perception of chess positions[27,28], or solving the Tower of Hanoi puzzle[26]. The perception has been formulated by production systems to describe the function of its mechanism, and is referred to as perceptual-test[26].

The condition of perceptual-test involves a series of tests, while the action involves a sequence of elementary actions. The test in the condition part is usually a test of the presence or absence of a particular kind of symbol in the goal stack to determine the appropriate step. The action may be a motor act of drawing, data input, solution generation or solution testing. The result will generate new information and change the contents of short-term memory.

The concept that the perceptual-test serves as a pointer connecting to the nodes in the knowledge base has been proposed by Larkin *et al.*[29]. Whenever a design unit is presented, it is perceived by the system. Then the perceptual-test serves as an index to access the information stored in the knowledge base. Other mechanisms of perceptual-test are assumed to be able to:

- test whether the current goal has been achieved
- test whether the generated solution satisfies global constraints
- perceive which design unit is lacking at the current design stage
- perceive the problem context to determine the appropriate step to follow

By using these mechanisms, perceptual-test presumably serves the following functions.

- The test of goal state will guarantee that the system is always in progress and that the process always moves toward a goal. Thus, if the current goal has been achieved, then the system will retrieve the next goal from the goal plan or the goal stack. Otherwise, the perceptual-test will perceive which is the next candidate design unit in order to continue accomplishing the current goal.
- The test of global constraints will make sure that the generated solution is optimal. If the generated solution satisfies all the constraints, then the system will proceed to the next design unit under the current goal. Otherwise, a new goal is set up.
- If a design unit is presented in short-term memory and

a set of constraint schemata is evoked, the perceptual-test will recognize that such a design unit must be solved in order to process the next one. Thus, a subgoal is developed to solve the problem being presented.
- The perceptual-test will perceive what happens at the current state and will determine an appropriate step to process.

When a goal is generated from a goal plan, it is goal-driven. If a goal is developed from perceptual-test, it is perceptual-driven or stimulus-driven. The way of selecting a goal or the way of structuring a solution path is referred to as control strategy. The control strategy will provide clues to how a designer structures the solution path.

Search

Simon indicates that problem solving activity can be described as a search through the problem space, until a state is reached that provides the solution to the problem[26]. Thus, the whole process is a search through the knowledge states guided by information accumulated during the search. There are many search methods discussed in AI literature. Only three basic categories classified by Newell and Simon are used here, as these methods provide primary and fundamental explanations to the cognitive processes[11].

Recognition

The recognition method is defined as knowing the answer. It happens when the problem is reduced to a point at which a known procedure or model can be applied to the remaining stages. The known procedure or model has been described by Simon as 'prefabricated solutions', which provide answers to subproblems that arise repeatedly in different contexts[10]. It has also been described as 'presolution model' by Foz[30]. The retrieval of the presolution model is done by perception, which gets access to the index of information in the knowledge base, and thus is called the recognition method.

Means–end analysis

The means–end analysis requires a known goal, the identification of differences that exist between the current state and the goal state and the selection of operators that will reduce these differences.

Generate-and-test

The generate-and-test method includes a generator and a test[2]. The generator will take design units and a set of corresponding schemata to generate objects that are candidate solutions or components of solutions. The test will determine whether a candidate satisfies a set of constraints. By using the test information, the generator produces a new knowledge state by modifying a state produced previously in the search. This dependency of generation upon the test outcome characterizes the heuristic search method. Any object generated that satisfies the test process is guaranteed to satisfy all the design requirements. In a generate-and-test process, the design is assembled component by component. Each generated component is added to the previous component and a new assembly is created and tested. If the test succeeds, the process continues; if it fails, the new component is discarded and another one generated. The design process contains a series of generate-and-test cycles.

A LABORATORY EXPERIMENT AND DATA COLLECTION

The cognitive model delineates design problem solving processes in general. For the purpose of justifying the model and of observing the cognitive activities in design, a laboratory experiment was designed next.

Task and subject

The task was adapted from the design project documentation used for Design Level 2 Studio in Spring 1986, Department of Architecture, Carnegie Mellon University. The original design instruction had been simplified to fit the experimental purposes. The task was to design a three-bedroom dwelling for a single family on a large property in the northern campus. Design units included a living room, dining room and two bedrooms for a son and a daughter. The total floor area was limited to 2 200 square feet. The client was a professional architectural perspective draftsman. Two image units*, a Doric column and a bay window, were required to be included in this residential design. The accurately scaled floor plan, elevation and section drawing of the image units were also provided. The purpose of having image units was to observe when and how a designer deals with image part.

A professional workshop, which is not common in a residential dwelling, was also required to observe how a designer processes an unfamiliar design unit. The design information was reduced to a minimum to discern the kind of knowledge that can be retrieved from memory. The subject was a PhD in Architecture student enrolled at Carnegie Mellon University. At the time of the study, he had eight years of design experience and had worked for a professional firm for approximately two years.

*An image unit is defined as a specific architectural form that is developed by the client. A designer will perceive such a form and develop an image code in his or her long-term memory in order to process the design task[31].

Procedure

This experiment was conducted at the Design and Information Processing Laboratory in the Department of Architecture. It began right after the subject finished reading the task instruction. Drawing paper and marker pens were provided. The subject could draw anything he wished, but a final site plan, floor plan and facade drawing were required to be finished at the end of the experiment. There was no time limitation. The subject was asked to speak aloud at all times while he worked, and his verbalizations and drawings were recorded on video-tape. The experiment lasted for about four hours (232 minutes).

Method of data analysis

Methods of data analysis required five sequential steps.

Transferring data into protocol transcription

The raw data was transferred into a protocol transcription (a list in written form of the subject's verbalizations), referred to as statements in the experiment. Statements were segmented by any pause greater than 4 seconds. This method is different from the 2 second pause time used by Byrne[32] because this experiment involved visual perception of drawings. Simon indicates that a few hundred milliseconds to a couple of seconds are needed to retrieve information from memory[33]. Posner states that memory can hold visually perceived information for 2 seconds[34]. Taking the upper bounds, the reaction time of holding a visually perceived item in short-term memory plus retrieving information from memory is estimated to be 4 seconds. A pause time greater than 4 seconds indicates that the successive statement probably provides information about a new perceived item. According to this method, this protocol contains 604 statements.

Identifying episodes

After the protocol transcription had been completed, it was classified into episodes. An episode is defined by Newell and Simon as 'a succinctly describable segment of behaviour associated with attaining a goal'[11]. Each episode contained a unique goal that was to be achieved, and was treated as one unit episode. The goal in an episode was identified:

- by the verbal information in protocol transcription
- by tracing a series of actions which attempted to solve one design unit
- by a particular recognizable intention under which a group of design unit was to be resolved

The purposes of developing episodes and identifying goals were to observe the mechanism that determined goals, and to find out how goals were initiated and terminated. There were 22 episodes in this experiment.

Identifying knowledge states

After episodes had been identified, knowledge states in episodes were clarified. Specifically, a knowledge state is a stage of knowledge in which some pieces of information are activated in short-term memory. Any change of knowledge state symbolizes a move, and also marks an application of an operator. Thus, the trace of a move of knowledge state is based on any changing information occurring in the statements. In this study, the purposes were to understand what kind of knowledge appears in a knowledge state, under which design unit it was considered, and what sort of operators caused the move.

Problem behaviour graph development

The knowledge state and its move only provide fragmental information. In order to understand the whole sequential moves in achieving a goal, a problem behaviour graph[11] is used. The problem behaviour graph is a concise expression of moves of knowledge states. Nodes represent knowledge states and lines symbolize transformations in the graph.

The partial problem behaviour graph of the subject is shown in Figure 6. It is coded according to the taxonomy given in Table 12. This problem behaviour graph should be read from left to right, then down. The design unit that is being considered is shown on top of the line. The operations that were used for state transformation are shown below the line. The question mark represents data input from the experimenter. GC represents given constraints. RC means retrieving constraints from memory. NC means newly generated constraints. RCXX/G stands for generating a solution by applying the rules that are associated to the constraint XX. RCXX/T stands for applying the retrieved constraint XX to test the result. The far left vertical line symbolizes goal stack.

The purposes of constructing the problem behaviour graph were:

- to observe how a goal was achieved
- to understand the pattern of moves
- to detect how search methods were implemented

The back-up of a knowledge state does not mean that the knowledge has been abandoned. Rather, it signifies the change of knowlege state that corresponds to either goal development or searching effort.

Discovering the invariant structure

From observing the typical processing pattern exhibited in the problem behaviour graph, and from fitting data into the cognitive model to observe the consistency of process, the invariant cognitive structure which represents the system's behaviour was able to be detected.

Reliability of data collection

In testing the reliability of data collection, a portion of the protocol transcription (the process of solving Doric column and bay window), was given to a third person, an architect, for the purpose of coding the problem behaviour graph. He was provided with a set of specified procedures to be followed. Although he did not have an insight understanding of the subject matter in this research, the patterns displayed in his results show many similarities to the experimenter's. In particular, the four generate-and-test search cycles on solving the Doric column and bay window were detected by both graphs (see Figure 6). This shows that the method developed for observing the change of knowledge state to explore the search effort is pertinent. Also, his identified operators together with the discerned knowledge states were in agreement with our results under the application of the same method. This indicates that, by following the same method, the discrimination of knowledge states and operators that cause the move can be made explicitly. Therefore, the method being developed provides an exact norm for gathering data about design processes.

RESULTS OF THE PROTOCOL ANALYSIS

Results obtained from collecting and analysing data according to the methods described are explained in the following sections.

Knowledge base

The design units which were retrieved and used by the subject in this experiment are shown in terms of a tree structure (Figure 2). This is the proposed knowledge representation of the subject in this design. Nodes are either design units or a class of design units headed by an abstract name. Arranging this tree structure required tracing the order of appearance of design units in the protocol and then grouping the design units together by categories. Cross-references among nodes are not shown in this figure.

This knowledge base provides the designer with an organized network of information that is applicable to design. It also provides a clue as to which design unit should be processed next. Because of this organized representation, an efficient search is possible. However, this diagram should not be construed as a literal model of the internal data structure being accessed, although it may serve to suggest some properties of these structures. For example, the upper nodes only appear at the early design stage. When design proceeds into later stages, lower nodes gradually appear in more detailed drawings. Such an information retrieval reveals a top-down process.

It is assumed that there is a prototypical representation for a building type stored in the long-term memory. When a design task is assigned, a designer must develop a new representation to fit the problem at hand. Such a development is stepwise and gradual. The whole representation will be robust and concrete after the designer is familiar with the nature of the problem. For example, the

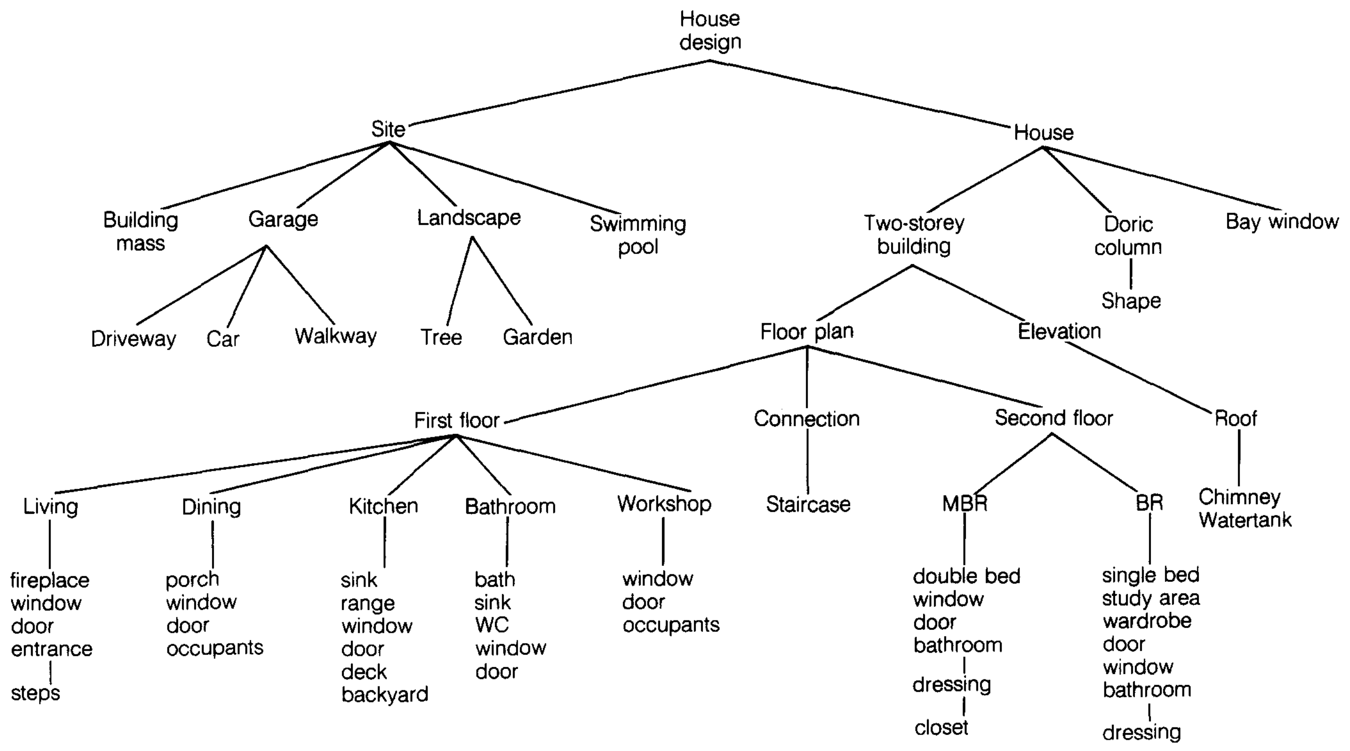


Figure 2. The representation of knowledge base

professional workshop in this case was unusual to common residential design and was therefore an unfamiliar design unit. It had only a loose tie to the rest of the design units in the original prototypical knowledge representation and therefore required time and effort to establish relationships with the rest of the units. This experiment required a considerable amount of effort on the part of the subject, who retrieved a large number of constraints to solve the location and the roof shape of the workshop.

Goal plan

Research into well-defined problems has determined that information contained in the knowledge state can be used to guide the generation of new knowledge states, so that the search through the problem space can be selective rather than random[35]. But in an architectural design problem (ill-defined problem), the generation of a new knowledge state sometimes does not have any correlation to the previous knowledge state. It has been shown in the protocol that when a goal is satisfied, the subject has a clear idea about what the next goal state is. And the new state is discrete from its parent state. The protocol statements cited in Table 1 indicate such phenomena. These provide evidence of the existence of a goal plan in the subject's memory.

The trace of goal plan

One characteristic of episodes is that each starts from a developed goal and ends when a goal is achieved. If a goal cannot be achieved, then a subgoal is developed, and the current goal is pushed back to the goal stack. By studying the context of goals in episodes as shown in the right part of Table 2, and by excluding the subgoals, it is possible to discover an overall goal plan as shown in the left part of Table 2. The discovery of the existence of a goal plan supports Akin's early finding that a designer employs his knowledge in a goal-directed way[36]. Moreover, the subject's goal plan reveals the following characteristics.

- A goal plan is a sequence of abstractions ranging from the schematic level to the detailed level that produces the solution.
- A goal plan is constructed in the fashion of a series of stepwise refinements. In other words, the abstract upper levels are done first and details are gradually added later. In referring to the knowledge representation shown in Figure 2, this goal plan also shows a top-down process.
- A goal plan is developed from previously learned knowledge about processing a design task.

The goal plan is a general goal plan that a designer must know in order to solve a particular building type and is learned by experience. When designers encounter a new architectural problem which they have not solved before, a new method (plan) is needed. Heath indicates that such a method might be individual and personal. It is possible that each new architectural problem requires a new and different method, and that methods might be specific to building types[25].

Table 1. Protocol statement of goal transformation

Goal: Site organization	#15:	The thing that I am trying to think of right now is in terms of the site. (04:49)
Goal: Two-storey house	#27:	See, the things that I am trying to get are some kind of image. One [previous episode] is the site organization, where the major function goes, which side the workshop should go and act as a buffer for the winter breeze, and also vertically [new episode], in terms of if I am going to divide the scheme to fit in more than one floor, that is permitted, right? Everything has to be in one floor? (08:22)
Goal: Room size	#145:	OK, maybe later I would try again the workshop. (71:05)
	#146:	Now, I would try to get into the sizes, first to get some rough size for each of these spaces. (71:10)
Goal: Accurate first floor layout	#225:	The way I am thinking here, that you have the [Doric] column on one floor, and then on top of that [Doric column], you support the bay window . . . on the top level, so this bay window is a part of the bedroom. (101:41)
	#226:	Now, it seems, uhm, possibly, I would like to try to get into a little more accurate scale. (102:30)
	#227:	And see . . . how it looks on the site plan. (102:49)
Goal: Accurate second floor layout	#302:	. . . So if I have to draw a plan. (124:58)
	#303:	then [place a paper over the first floor plan]. (125:34)
Goal: Elevation generation	#350:	This is how the plan is solved, worked out, except for [upper part of workshop], I don't know, I still have to spend some time in resolving that, how that place is going to work out. (137:54)
	#351:	As this elevation in the front, this is what I am trying to get right now. (138:08)
Goal: First floor final plan	#457:	OK, now, what I would like to do is to draw it up a little bit more clear, all the plans. (174:48)
	#458:	And then I can place the sheet on the site plan, so to give an exact location. Maybe I can draw it on the site plan. (175:10)

Table 2. Representation of goal plan and goal stack

Initial goal stack which represents goal plan	Goals appear in goal stack
1 Task understanding	1 Task understanding
2 Site organization	2 Site organization
3 Two-storey house (develop scenario)	3 Two-storey house
	4 Image units
4 Initial space layout	5 Initial space layout
	6 Location of workshop
	7 Initial space layout résumé
	8 Form of workshop
5 Room size	9 Room size
6 Space generation	10 Space generation
7 Accurate first floor layout	11 Accurate first floor layout
8 Accurate second floor layout	12 Accurate second floor layout
	13 Decide three bathrooms' space
	14 Accurate second floor layout résumé
9 Elevation generation	15 Elevation generation
	16 Workshop roof
	17 Fireplace
10 First floor final plan	18 First floor final plan
11 Second floor final plan	19 Second floor final plan
12 Final elevation	20 Final elevation
13 Site development	21 Site development
14 Evaluation	22 Evaluation

Control strategy

The way of selecting goals or developing solution paths has been referred to as control strategy. The control strategy reveals the manoeuvre by which a designer chooses to attack the final goal. The subject used four strategies in this experiment.

- *Scenario development.* At the beginning of episode 3, the subject said

  > See, the things that I am trying to get are some kind of image. One [previous episode] is the site organization, where the major function goes, which side the workshop should go and act as a buffer for the winter breeze, and also vertically [new episode], in terms of if I am going to divide the scheme to fit in more than one floor; that is permitted, right? Everything has to be in one floor? (08:22)

  This protocol statement gives information about goal specifications for how to work on site organization in the previous episode. It also shows that a new design guideline or a new problem structure, which was to develop a two-storey house, was under development. A design guideline can be understood as a specific goal plan which differs from a general goal plan. The general goal plan only provides a general goal sequence for a design task, while the specific goal plan specifies goals and is different from task to task. Under the guidance of the specific goal plan, a design problem can be well structured and the search for the solution path is possible. It is because of the specific goal plan that the design problem can be converted into a well-structured problem versus an ill-structured problem*. Such a phenomenon is analogous to Akin's finding of the usage of scenarios for problem structuring[9].
- *Image units.* According to this strategy, when there is an image unit, the image unit is solved first. At the end of episode 3, when the subject worked on the two-storey house image, he encountered the Doric column and bay window, and recognized them as important units. The subject immediately developed a new episode (a new subgoal) to solve these image units until a solution came up. His solution was to use a Doric column to support the bay window on top.
- *Error correction.* This strategy occurred when the subject discovered that a design unit had been mistakenly interpreted, and he resolved it right away. At episode 6, when the subject realized that the workshop was a professional workshop instead of hobby workshop, he concentrated on the workshop until a satisfactory result had been achieved.
- *Back-up strategy.* This strategy occurred when the subject faced a problem which could not be solved at this time. He tried to solve other problems instead and came back to solve the initial problem later. This happened when the subject had difficulty solving the junction of the bay window and the window opening at the master bedroom facade. His strategy was to switch to solving the roof shape first. Then his intention to match the bay window roof produced a solution for the unsolved junction part. His final solution was to make a cut in the wall as an offset to separate the bay window element. To interpret the strategy, when a designer encounters a problem without having an available schema rule, the system keeps perceiving other design units until available rules for the earlier unit are indexed and evoked.

Functions of perceptual-tests

The above strategies explain how goals are developed to guide the design behaviour. In the following, the fundamental functions of perceptual-test are verified by examining the protocol data.

- The first function is to determine whether the current goal has been accomplished. In the protocol, there was no statement indicating any satisfaction at the end of each goal state. The subject simply switched to a new goal. These silent transformations imply that unless a goal is achieved, it is impossible to develop a new goal.
- The second function is to test the generated solution for perceiving and determining the solution path. Two examples in the protocol explain it well.
  - *Change problem space.* When a solution does not satisfy a global constraint, then the system will change the problem space. At the end of episode 4,

*An ill-structured problem is a well-defined one, but lacking the structure required to apply powerful or algorithmic search strategies[37].

after the combined image unit was generated and tested, the subject retrieved a new constraint which was that the structural elements must dictate how things correspond inside. It seemed to the subject that what he had been working on was half from the inside (function) and half from the outside (form). Although the goal, which was to solve the image units, had been achieved, the generated solution did not satisfy the new constraint. So, he switched the problem space to the next goal, which was to solve the functional layout.

- *Critical problem situation.* This happens at the selection of solutions. For instance, in this experiment there were four cycles of generate-and-test in solving the Doric column and bay window. In each cycle, one solution had been generated by one constraint and tested by another constraint. For the second alternative solution, which was to use a Doric column as an interior single element supporting the ceiling, the subject indicated two things: (1) such a form must match a classical vault, which would change the character of the house; (2) he was not keen on doing a historical revival. Therefore, this solution was abandoned.

  Although the second reason suggests a personal preference, the first one indicates that the subject perceived a critical problem situation at the time when a solution was generated. The critical problem situation refers to the possibility of changing the problem structure* or solution path. In other words, the critical problem situation is the state of affairs or position that will lead to a possible restructuring of the problem. In this example, the subject perceived that the solution would cause the change of the interior form, material, structure and the character of space. Such changes may possibly have led him to restructive knowledge representation or to change the goal plan, since historical elements were involved. Therefore, the selection of solution was made upon the perception of the problem situation and to avoid having a major change of the problem structure.

- The third function is to perceive what is lacking at the present stage. The system searches for a design unit to work on next. Table 3 gives an example cited from the protocol statement. In this example, after the subject had solved the Doric column and bay window, he searched for a new design unit to work on. Glazing was the evoked new design unit in this instance.
- The fourth function is to perceive the problem context and solution context to determine the next step or the next action.
  - *Problem context.* The perception of the problem context is to perceive the problem structure and determine the goal sequence. Two examples of this were found (see Table 4). The first example shows that the subject perceived the size of the building mass as a small one, so he decided to do site development later. It turned out that the site development appeared at a later stage in the protocol. The second example shows that the subject intended to determine the scale of drawing by perception.
  - *Solution context.* The perception of solution context function is to perceive the solution path and determine the next solution generation. For example, the choice of a symmetry constraint was developed originally from the development of the Doric column and the bay window. The development of the Doric column and bay window had gone through four generate-and-test circles to satisfy several constraints. The subject was satisfied with the final result of the centralized column which supported the bay window, so a symmetric centralization aesthetic principle was created accordingly. Following the same aesthetic principle, the symmetry constraint was again selected to solve the living room layout. As the subject indicated

    Since it [Doric column] is going to be something striking as an element like that [Doric column], at least here I am trying to keep this [living room] spaces and try to maintain the symmetric disposition. (113:40)

    At this state, the subject perceived the solution

*The problem structure means the format of knowledge representation, goal plan and constraint establishment, and is the result of problem structuring.

Table 3. Perceptual-test searches for new design units

#72:	I am trying to see, in terms of the plan, like here [previous floor plan drawing] I am trying to see in terms of section what is going to have, and I am trying to see what other things could be attached to this column as an element. (28:50)
#73	One thing is that, you may call it some kind of glazing, [draw a horizontal line across the column in the floor plan] in which the column. (29:15)
#74	Really is a free-standing element, visually. (29:30)

Table 4. Perceptual-test determines goal sequences

Example 1	
#33:	Thirty feet is . . . what? It is a tiny house on a property like this! It is a very tiny house. (11:33)
#34:	And it is going to be so tiny. (11:46)
#35:	Then except for the major orientation, it doesn't matter where, one could place it later on on the site. Because there is so much of land around it. (11:56)
Example 2	
#42:	I am trying to get roughly that initial scale and gives me some idea of scale. (15:21)
#43	So, it is approximately so much [draw a horizontal line]. (15:37)
#44	I am trying to see whether I am going to work on this scale, or I could work on just schematic on this scale, and then going into more details before I make any more . . . spatial organization. (15:52)

context and selected the next solution which fit the solution context. The solution context means that the occurrence of solution B is related to solution A, or that the result of solution A leads to the cause of solution B. In this instance, the result of solution A (Doric column and bay window) resulted in the selection of symmetry for solving the living room layout.

The discovered control strategy and these functions of perceptual-test confirm the prediction of the model. Moreover, this evidence demonstrates that perceptual-test is the primary mechanism which determines goal sequences and thus constructs the solution path.

Design constraints

The problem behaviour graph shows that whenever a design solution is generated or tested, there is at least one design constraint involved and a rule is verbalized. Whenever a designer works on a design unit, a set of constraints is evoked from memory and the associated rules are applied.

Global constraint and local constraint

Data show that the global constraints are most likely developed during the first episode (8 out of 12), which is the task-understanding stage (see Figure 6). Then they reappear at lower level nodes. In this experiment, the most distinct global constraint was climate. The climate factor influenced the space organization (more abstract level), and also affected the location of the window opening and the glazing size (more detail level). The global constraints used by the subject are listed in Table 5. On the left of the table are constraints given by the task instruction, and those on the right were retrieved by the subject. These constraints reflect the following characteristics:

- they were mostly evoked at the first episode
- they were applicable to a group or to all design units
- they were able to be used in different design tasks

There are 47 local constraints in this protocol. For the workshop, the global constraints were land slope, climate and natural light; whereas the local constraints were location on basement, visitor accessibility, noise and ventilation. A careful study shows that global constraints appeared at both upper and lower level nodes, but local constraints only appeared at the two lowest levels in the knowledge tree. This indicates that global constraints guide the whole design process and are used to generate upper level design units. When the upper design units have been solved, the system proceeds to the detail level and local constraints are evoked.

Rules in schemata

When a design unit is presented on working memory, a constraint schema is retrieved. Rules in schema are then used for solution generation or testing. These rules, which contain domain-specific knowledge (design knowledge), can be represented by production systems. For example, there are three protocol statements in Table 6. These statements associate with the design unit of building mass under the constraint of site privacy. Used by the subject to come up with a solution, these rules are written in production systems as shown in the same table.

In rule 1, the knowledge of fact or declarative knowledge is knowing the private corner in the site is embedded at the left-hand side of the production. The procedural knowledge, which is to put the building at the private corner, is at the right-hand side. The action side is critical in design problem solving, for it contains domain-specific knowledge which is heavily relied upon by the designer. The data analyses show that there was a tremendous amount of design-specific knowledge embedded at the action part. In order to find the fact in rule 1, the system will sequentially instantiate rule 2 and 3. Thus, rules in schemata are applied for solution generation.

Data input

Data input happens at the time when a designer needs more information about a design constraint. For instance, after the semi-open workshop space had been generated, the subject used noise constraint for a test. In order to find out the level of noisiness, he relied on data input. This shows that when a constraint schema is evoked and the value of the constraint is uncertain, or there is no rule to find the value, then data input is needed.

Table 5. Global constraints found in protocol data

Given		Retrieved	
Climate	(LR, window, workshop, BR) (building mass)	Light	(window opening)
		Privacy	(LR, building mass)
Total floor area	(building mass)	Near access road	(building mass)
Land slope	(building mass)	Common bathroom	(bathrooms)
Access road	(building mass)	Bedroom with attached bathroom	(bedroom)
Site area	(site)	Symmetrical disposition	(floor plan, elevation)
		Room dimension	(rooms)

Table 6. Protocol and rules in schema

Protocol statements

#18: Now, somehow, it seems that this [northeast] corner here, seems more private. Because these two edges [west and south] are bound by outside roads. (06:07)

#19: And there is a property on this [north] side and a private property on this [east] side. (06:21)

#20: So, things will be better, if I place things along this [northeastern corner] side. (06:28)

Rule representation

Rule 1: If there is a private corner
then put building at private corner

Rule 2: If edge A is private
and edge B is private
and A and B are adjacent
then the corner formed by A and B is a private corner

Rule 3: If edge is next to a property
then this edge is a private edge

Operator

Anything that causes the state transformation is called an operator. The different kinds of state transformation found are categorized in Figure 3. Rules found for binding value to variable are arithmetic rules, assertions or logical inductions. Evidence also shows that if the subject is uncertain about the rule, then no action is executed.

Conflict of constraints

If a generated solution is in conflict with the global constraint, the system will change the problem space. This is a part of control strategy as described before. If a generated solution is in conflict with local constraints, the system is supposed to modify the solution. However, in this experiment the subject tended to sacrifice the original constraint, and only the new retrieved constraints were taken into account. For example, the centralized kitchen door, which opened to the dining room, was generated by observing the internal symmetric disposition constraint. But the result was in conflict with the visual access from the living room, and with the location of the swing of the kitchen door. The subject said,

> Uhm, two things, one is . . . because this kitchen door is just in a wrong place. I am trying to get to smaller details, the reason being that, is absolutely central, like anyone who is sitting there [living room] is straight looking into the kitchen. Another reason is that, where is this swing of the [kitchen] door goes? I don't have a position for this. You open the door and that panel of the door that is going to swing them [user in the kitchen], which is not very sensible. (121:19)

The final solution, which was not consistent with symmetric disposition, was to move the kitchen door two feet to the right. This can be interpreted as meaning that the constraint may have priority status, and the solution is subject to the top priority. In this case, the visual access (the second constraint) had priority over symmetry. Therefore, the symmetry constraint was released to compromise the solution.

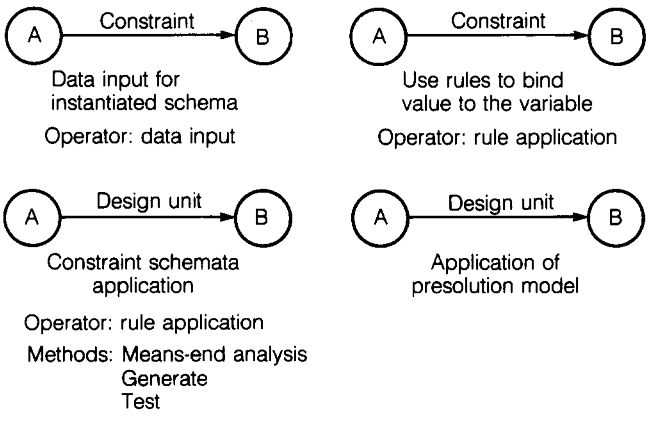

A
Constraint
B
Data input for instantiated schema
Operator: data input
A
Constraint
B
Use rules to bind value to the variable
Operator: rule application
A
Design unit
B
Constraint schemata application
Operator: rule application
Methods: Means-end analysis
Generate
Test
A
Design unit
B
Application of presolution model

Figure 3. Causes of state transformation

Search

Recognition and presolution models

When knowledge is repeatedly used in design, the skill becomes more and more rapid and automatic. When a skill gets to an automatic level, it requires less attention and the person who is using the skill may lose ability to describe the skill verbally. This is the phenomenon of retrieving a presolution model for recognition search. Since the information is a part of a prefabricated solution, whenever a stimulus presents, the answer is immediately found and sketched. There are nine presolution models shown in Table 7. As in Table 7, seven out of nine presolution models are iconic images, a kind of internal representation. In other words, the subject can draw them immediately using a very simple sketch. Thus, when a design unit is activated in short-term memory, the subject can retrieve the image quickly. This suggests that a solution for a design unit is constructed in advance and can be repeatedly adapted to the needs of

Table 7. Presolution models used by the subject

Design unit	Presolution model	Type
Building mass	A two-storey building	Concept
Doric column	A setting on a stage, a free-standing element	Image
Doric column	A column supports ceiling	Image
Bathroom, staircase	A rectangular pattern	Image
Workshop	A sunken courtyard	Concept
Porch, workshop roof	Pitch roof	Image
Garage	The shape of garage is a shade like	Image
Staircase	A prototypical image of staircase plan	Image
Workshop window	A prototypical image of clerestory elev.	Image

new circumstances, and that the image is encoded in memory by a very special code[31]. The rest of the models are used for designing new objects. The underlying thought process is to retrieve a presolution model first, then to evoke the associated design rules for the completion of the solution. These rules are initially embedded at the moment when the presolution model was fabricated in the past experience. Thus, a presolution model is a part of the design schemata, and has the iconic image nature.

Generate-and-test

Solutions of a design unit are generated by introducing a design constraint. Two phenomena have been identified in the generate-and-test cycle.

- If a solution has been generated but no new design constraint is evoked, then no test is called. The system moves to the perceptual-test, looking for the next design unit.
- If a solution has been generated and a new design constraint is activated, then a test follows. If the test succeeds, then the system proceeds to the next design unit. If the test fails, then the system will search for new constraints to generate a new solution on the next generate-and-test cycle. A clear example is during episode 4 under the goal of solving the Doric column and the bay window. The subject applied the presolution model to generate a solution and used a constraint to test the result. Such a generate-and-test cycle was repeated twice for two alternatives. Finally, the subject pulled three design constraints (visual focus, support for another element, location of the column outside) to generate a final result. This suggests that the generate-and-test method is a search cycle.

An interesting question, which has not been posed up to this point, is what happens to the system if more than one solution is generated? Theoretically, solutions will be tested by other constraints to determine the feasible one. However, that was not the case given the data from this experiment. For example, the subject retrieved the constraint that 'a fireplace is a place where people sit around'. Two alternative solutions satisfied the constraint. The subject used the test constraint (circulation) to test only one solution and reached the final decision. The data does not show that the subject tested the other solution. A better interpretation for this example is that after the generation of solutions, perceptual-test also perceives the potential of the solution and determines the solution path. In this case, the perceptual-test perceived that one solution was more promising, so that the other solution was abandoned without test.

Means–end analysis

Besides the recognition and generate-and-test search methods, the subject also used means–end analysis in this experiment. An example shown in Table 8 is cited from episode 5. In this example, the goal of the episode was to develop an initial space layout. The design unit was a staircase. The subject indicated that he was looking for a kind of geometry (subgoal) to fit into the service bay of the staircase and the bathroom. After the subgoal had been identified, he first used a presolution model to generate a solution, which failed when tested. Then he searched for seven rules (operators) for sequential moves in order to arrive at the goal state, which is the typical method of means–end analysis. From observing the protocol data, it is reasonable to say that the generate-and-test search method was used to search for one rule for the generator and another rule for the test, while the means–end analysis was used to search for the whole set of rules.

Initial problem space and task instruction

Newell and Simon suspect that the initial problem space is either ready-made or is constructed from elements in the long-term memory of the problem solver[11]. The first episode of this experiment revealed that prototypical constraint schemata were instantiated from memory. Since the value of the instantiated constraint schema varied from task to task, data input was needed to fill up the slot. This explains why data input occurred throughout the first episode (see problem behaviour graph).

The design programme provides a cover story which implicitly or explicitly specifies what the designer must consider. However, a problem solver may have difficulty understanding a task presented through nature language instruction[38]. The information extracted from the task-understanding period determines the representation that a designer has. For example, the workshop in this experiment was to be a professional workshop. While the instruction indicated the occupation of the client, it did not specify the nature of the workshop. Thus, the subject mistakenly perceived it in his own way and represented it as a hobby workshop.

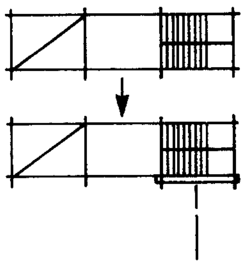
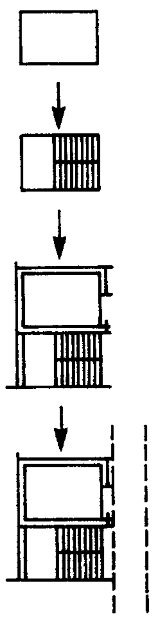
INVARIANT STRUCTURE AND SIMULATION MODEL

Invariant structure

Results discussed in previous sections confirm the existence of a goal stack, the design schemata, the perceptual-test and the search methods used by the subject. Based on the findings, an invariant structure of cognitive processes is mapped into the proposed model and shown in Figure 4. This cognitive model can be briefly described as follows.

When a goal is developed, a design unit and the associated schema are retrieved from long-term memory. The retrieved schemata specify the current goal. This is the problem structuring stage. Then the system searches

Table 8. Example of means–end analysis search method

Protocol transcript	Search methods	Drawing result
#97: Uhm, I am trying to think of some kind of factor, of geometry, [draw a rectangular, divide it into three parts, draw the right end part as staircase], which gives me some basic organization, in which it could fit in . . . services such like a service bay that could fit in, which also gives, (47:08)	Subgoal: develop a geometry Goal specification: the geometry should give some basic organization in which a service bay could fit Design units: bathroom and staircase	
#98: The entrance [draw a line below the staircase] and the house, I don't want this solution, this, (47:52)	*Recognition + Generate-and-test* Presolution model: a rectangular form for service bay Generate: a geometry of three bays, one for staircase Test constraint: the distance to entrance. Test fails	
#99: One other thing is that typical, just put the service bay, basically the bathroom and staircase, in the same bay, and that might give the division, let the living and living, dining, and kitchen, and workshop outside, and then you want to keep these things together, kitchen and dining and then possibly, dining and living together. (48:06)	*Means–end analysis* Rule 1: put the service bay, the bathroom and staircase in the same bay Rule 2: leave living, dining, kitchen and workshop outside	
#100: The thing that I am trying to do is . . . to (48:37)	Rule 3: keep kitchen and dining together, keep dining and living together	
#101: get the bathroom, staircase and the kitchen. Because these are basically the service spaces. (48:48)	Rule 4: get the service spaces which are bathroom, staircase and kitchen	
#102: And a clear organization, which would also give the kind of division, that I want for living and dining, and these things are not only for this floor, like I said before, if I am going to place bedrooms on upper floor, on the second floor, then these things have to continue on the second floor. So it is not only in this floor that I am thinking about, but is also, where the spaces that is going to be fitted on the upper floor layout as well, that I could continue this same bathroom upstairs, the staircase obviously will be going one more floor, so that inside to leave the same volume open up on upper floors. So together with rooms and spaces on the ground floor, they also going to be at least partially carried forward on upper floors, so these both things are that I am going to match. (49.03)	Rule 5: if bedrooms are placed on upper floor, then bathroom and staircase have to be continued on second floor Rule 6: open both volume of bathroom and staircase up on upper floor	
#103: Another possibility is in which I place . . . [draw a rectangular] (50:28)		
#104: Staircase and . . . [draw staircase inside the rectangular] (50:40)		
#105: Service for the bathroom [draw another block on top], take these [staircase and bathroom] as very important [space] (51:04)		
#106: Which I have to leave, (51:28)		
#107: And . . . that leaves another circulation zone that has to be carried forward at least until here [draw a vertical line next to the block], and you also need a door, that goes . . . (51:42)	Rule 7: leave a circulation zone Subgoal achieved, problem solved	

for rules embedded at the schema for solution generation. If the solution is generated and another schema can be evoked, the test proceeds. This is the problem solving stage. The perceptual-test will control the system whenever the failure in memory retrieval or in search occurs.

This cognitive model has a goal-driven but perceptual-test oriented nature.

Simulation model

In order to test the accuracy of the model being discovered, a framework for simulating the whole cognitive process is proposed below. This simulation model is simplified on a conceptual diagram in Figure 5. The fundamental structure contains long-term memory, short-term memory, perceptual-test and the search

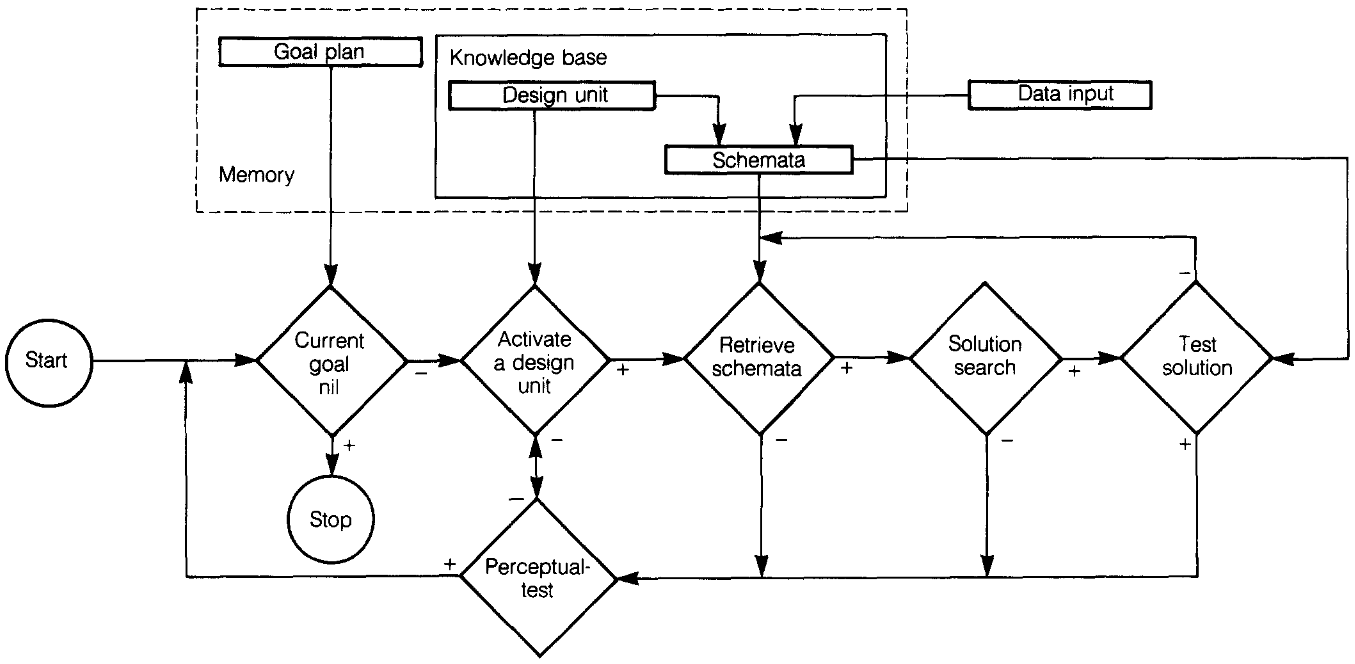


Figure 4. Cognitive model of design problem solving

mechanism. The long-term memory is represented by a list of schemata and an internal representation of a knowledge net. The working memory represents short-term memory. The block of production system includes the control strategy and search mechanism. This model intends to provide the capability of representing the invariant cognitive structure and to fit the protocol data as well.

Schema representation

In the model, the long-term memory is represented by two elements, the internal representation and the schemata. A schema contains identifier, arguments and rules. The template of a schema is shown as:

```
  <Identifier> <Arguments>
     Rules: If . . .
            then . . .
```

The identifier is the name tag of the schema. The argument is the design unit. The rule can be coded by an

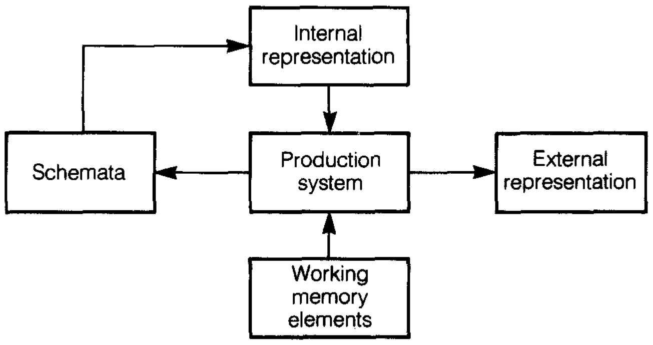


Figure 5. Conceptual diagram of simulation model

if–then clause where a body of design knowledge is embedded. This if–then clause also has the nature of a production. Therefore, the condition side consists of declarative knowledge and the action side has the procedural knowledge. In referring to Table 6, for example, the protocol statements can be converted into rule representation and in turn can be represented by schemata (see Table 9). Each schema can also be converted into a LISP function. In doing so, the identifier is the name of the function, and the argument is the variable. Whenever a function is evoked, it is evaluated. The value of the function is then returned and bound to the internal representation. The internal representation is a list of literalized declarations specifying the relations of design units.

Control strategy

The production system controls the progress of the system. A firing of a production depends upon the element that appears in working memory. The working memory contains a set of literalized declarations of goal and constraint. The declaration of a goal consists of the identification name and a vector of design units that are manipulated under the goal. The declaration of the constraint consists of the name of the constraint and the name of the design unit to which it is bound. Whenever the working memory element matches the condition side of the production, the corresponding schema is evoked and evaluated. When a solution is generated, then it is drawn as an external representation.

In other research studies, the production system is used to encode both the declarative knowledge and procedural knowledge. For example, the same set of

Table 9. Schema representation and LISP function

Schema representation

```
<Site-privacy> (<Building>)
   Rule = If there is a <Private-corner>
          then put building at <Private-corner>

<Private-corner>
   Rule = If <Private-edge> (<A>) is private
          and <Private-edge> (<B>) is private
          and <A> and <B> are adjacent
          then the corner formed by A and B is a private corner

<Private-edge> (<x>)
   Rule = If x = next-to-property
          then it is private
```

LISP representation

```
(putprop East 'private-property 'next-to)
(putprop North 'private-property 'next-to)
(putprop West 'road 'next-to)
(putprop South 'road 'next-to)

(defun private-edge (edge)
   (cond ((equal (get edge 'next-to) 'private-property) 'private)))

(defun private-corner ()
   (prog (corner)
         (setq corner '((North East) (South East) (North West) (South West)))
   loop
         (cond ((null (caar corner)) (return nil))
               ((and (equal (private-edge (caar corner)) 'private)
               (equal (private-edge (cadar corner)) 'private))
               (return (car corner))))
         (setq corner (cdr corner))
         (go loop)))

(defun site-privacy (building)
   (setq building (private-corner)))
```

protocols of schemata is converted directly into 'production rules', or 'if–then' rules[39]. To do so, a simple set of conversion rules can be used, such as when the protocols manifest an if–then, if–when or when–then structure. This transformation is quite straightforward and covers a majority of the protocol data. In this research, the production system was mainly used to map the control system. By doing so, the domain knowledge was able to be differentiated. The purpose was to clarify the cognitive structure and the domain knowledge, so that the model could simulate the cognitive model well. The developed production system is shown in Table 10.

SPECIFIC FINDINGS AND CONCLUSION

There are 286 total moves in the problem behaviour graph. The collected data of operators and the number of corresponding moves are listed in Table 11. The category of draw action in this table means that the subject traced the old drawings and that it was a mechanical motor action. The empty move happened when the subject verbalized either the name of the design unit or the name of the constraint, but no decision had been made. This was interpreted as the subject tried to scan through the knowledge base to evoke appropriate information. Extracting the number of moves of these two categories from the total, there were 210 moves involved with the application of classified operations. Within them, the applications of identified constraint schemata constitute 171 moves. In other words, 81.4% of the moves were caused by the application of constraint schemata. This result suggests that design constraint is a major means for design problem solving.

Presumably, the design problem solving ability is decided by the factor of the number of constraints, associated rules, and presolution models stored in long-term memory. Thus, a larger number of these factors will enhance the design skill. The contents of schemata have been used by Chi *et al.* to study the differences between experts and novices in solving physics problems[40]. They found that the schemata of the experts contain more procedural knowledge. This explains why rules in schemata determine the design ability. The design ability is also determined by the following two factors.

- *The ability of selecting rules in constraint schemata.* If the rules in a schema are insufficient for solution generation, then other rules must be selected. For example, climate was a global constraint which had been instantiated early at the first episode and was used for deciding the window location on surfaces. As shown in the protocol, the subject did not have an appropriate rule for deciding which orientation should be used

Table 10. Template of production systems

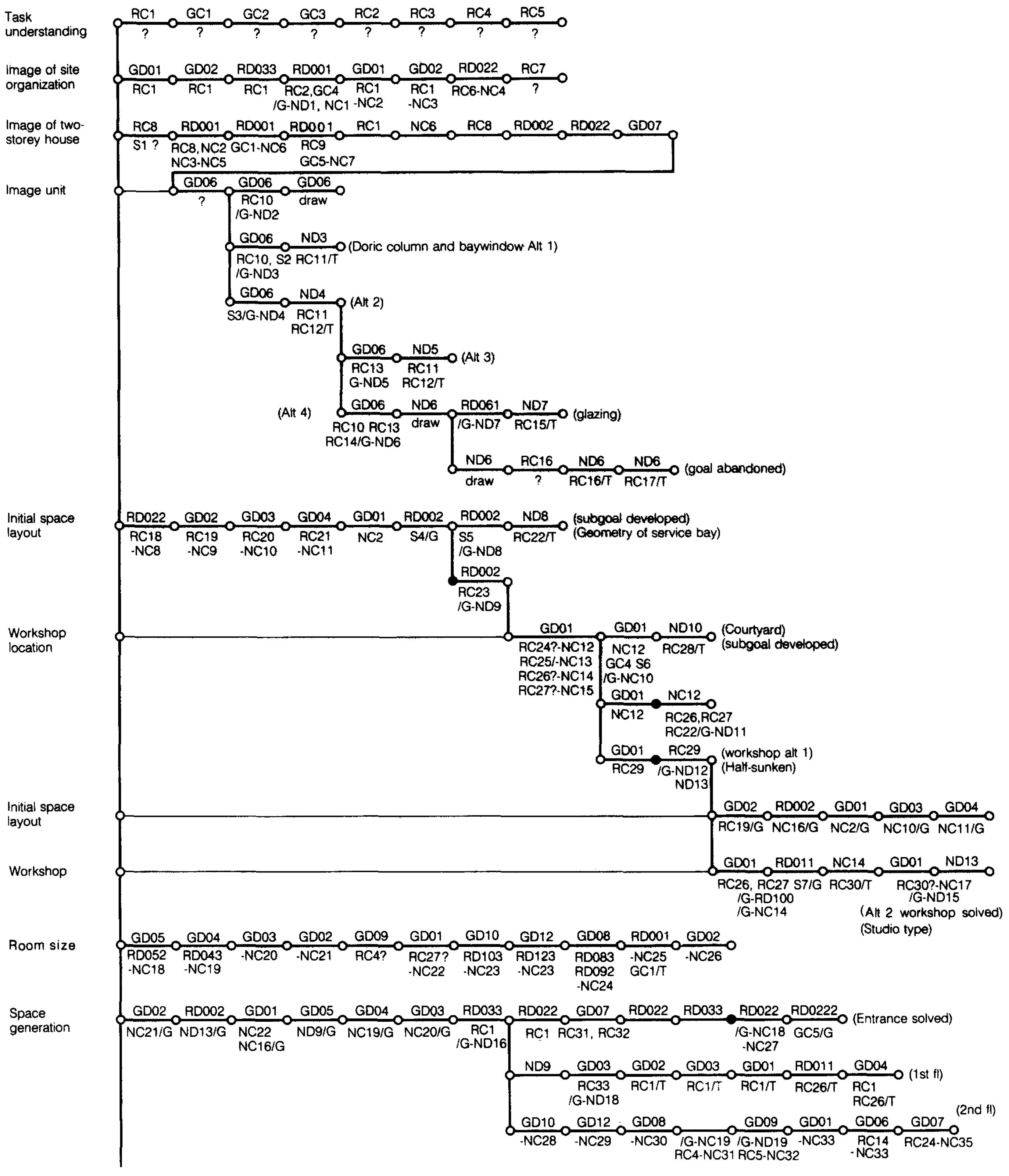
```
P1:  System halts
     If current goal = nil
        and first (goal) = nil
        and state = done
     then halt;
P2:  System driver
     If current goal <> nil
        and first (goal) <> nil
        and state = done
     then current goal = pop (first (goal))
        and state = nil;
P3:  Initial system
     If current goal = nil
        and first (goal) <> nil
        and state = nil
     then current goal = pop (first (goal));
P4:  Instantiate schema
     If current goal = G
        and design unit = D
     then retrieve constraint schema;
P5:  Schema rule application
     If current goal = G
        and design unit =D
        and constraint schema = Cl
        and condition (Cl) = nil
     then retrieve other subschema to fill up cond (Cl);
P6:  Data input
     If current goal = G
        and design unit = D
        and constraint schema = Cl
        and condition (Cl) = nil
        and no other schema is available
     then subgoal = data input;
P7:  Application of presolution model
     If current goal = G
        and design unit = D
        and constraint schema = Cl
        and there is a presolution model
     then apply the presolution model for D;
P8:  Constraint schema application for generation
     If current goal = G
        and design unit = D
        and constraint schema = Cl
     then value (Cl) = evaluate Cl
        and bound value to D;
P9:  Solution testing
     If current goal = G
        and design unit = D
        and solution (D)
        and there is constraint schema = C2
        and solution (D) is conflict to rule (C2)
     then test fails
        and solution (D) is abandoned;
P10: Image unit
     If current goal = G
        and design unit = D
        and D is an image unit
     then subgoal = solve D
        and push (current goal (goal))
        and current goal = subgoal;
P11: Re-evaluate a design unit
     If current goal = G
        and design unit = D
        and D has been mistakenly interpreted
     then subgoal = solve D
        and push (current goal (goal))
        and current goal = subgoal;
P12: Subgoal development
     If current goal = G
        and design unit = D
        and solution (D)
        and global constraint (D) = GC1
        and solution (D) is conflict to rule (GC1)
     then subgoal = solve D
        and push (current goal (goal))
        and current goal = subgoal;
P13: Process to next design unit
     If current goal = G
        and design unit = D1
        and solution (D1)
     then design unit = retrieve next design unit D2;
P14: Termination of current goal state
     If current goal = G
        and design unit = nil
        and constraint schema = nil
     then state = done;
```

Table 11. Number of moves by operator category

Operator category	Number of moves
Data input for design unit	3
Data input for schema instantiation	18
Rule application for schema instantiation	12
Application of constraint schemata	129
Application of instantiated schema value	12
Presolution model	9
Draw action	57
Empty move	19
Application of unidentified schema	8
Missing data of schema application	19
Total:	286

to place the window and which surface should be avoided having glazing. Hence, the window location of the living room and kitchen were not resolved. However, the subject was able to retrieve another rule, which was to reduce the window and the glazing size, and an alternative solution was reached.

- *The ability of developing new constraints for the test of a newly generated design unit.* A newly generated design unit may not be the one stored in the knowledge base. If it is a new form, then the ability of associating existing schemata to it, or developing a new schema for it, would strengthen the design skill. Such a skill is especially important for testing the solution being generated. Several examples occurred during this experiment.



Task understanding
RC1 GC1 GC2 GC3 RC2 RC3 RC4 RC5
? ? ? ? ? ? ? ?
Image of site organization
GD01 GD02 RD033 RD001 GD01 GD02 RD022 RC7
RC1 RC1 RC1 RC2,GC4 /G-ND1, NC1 RC1 -NC2 RC1 -NC3 RC6-NC4 ?
Image of two-storey house
RC8 RD001 RD001 RD001 RC1 NC6 RC8 RD002 RD022 GD07
S1 ? RC8,NC2 NC3-NC5 GC1-NC6 RC9 GC5-NC7
Image unit
GD06 GD06 GD06
? RC10 /G-ND2 draw
GD06 ND3 (Doric column and baywindow Alt 1)
RC10, S2 RC11/T /G-ND3
GD06 ND4 (Alt 2)
S3/G-ND4 RC11 RC12/T
GD06 ND5 (Alt 3)
RC13 G-ND5 RC11 RC12/T
(Alt 4)
GD06 ND6 RD061 ND7 (glazing)
RC10 RC13 RC14/G-ND6 draw /G-ND7 RC15/T
ND6 RC16 ND6 ND6 (goal abandoned)
draw ? RC16/T RC17/T
Initial space layout
RD022 GD02 GD03 GD04 GD01 RD002 RD002 ND8 (subgoal developed) (Geometry of service bay)
RC18 -NC8 RC19 -NC9 RC20 -NC10 RC21 -NC11 NC2 S4/G S5 /G-ND8 RC22/T
RD002
RC23 /G-ND9
Workshop location
GD01 GD01 ND10 (Courtyard) (subgoal developed)
RC24?-NC12 RC25/-NC13 RC26?-NC14 RC27?-NC15
NC12 GC4 S6 /G-NC10 RC28/T
GD01 NC12
NC12 RC26,RC27 RC22/G-ND11
GD01 RC29 (workshop alt 1) (Half-sunken)
RC29 /G-ND12 ND13
Initial space layout
GD02 RD002 GD01 GD03 GD04
RC19/G NC16/G NC2/G NC10/G NC11/G
Workshop
GD01 RD011 NC14 GD01 ND13
RC26, RC27 /G-RD100 /G-NC14 S7/G RC30/T RC30?-NC17 /G-ND15
(Alt 2 workshop solved) (Studio type)
Room size
GD05 GD04 GD03 GD02 GD09 GD01 GD10 GD12 GD08 RD001 GD02
RD052 -NC18 RD043 -NC19 -NC20 -NC21 RC4? RC27? -NC22 RD103 -NC23 RD123 -NC23 RD083 RD092 -NC24 -NC25 GC1/T -NC26
Space generation
GD02 RD002 GD01 GD05 GD04 GD03 RD033 RD022 GD07 RD022 RD033 RD022 RD0222 (Entrance solved)
NC21/G ND13/G NC22 NC16/G ND9/G NC19/G NC20/G RC1 /G-ND16 RC1 RC31, RC32 /G-NC18 -NC27 GC5/G
ND9 GD03 GD02 GD03 GD01 RD011 GD04 (1st fl)
RC33 /G-ND18 RC1/T RC1/T RC1/T RC26/T RC1 RC26/T
GD10 GD12 GD08 GD09 GD01 GD06 GD07 (2nd fl)
-NC28 -NC29 -NC30 /G-NC19 RC4-NC31 /G-ND19 RC5-NC32 -NC33 RC14 -NC33 RC24-NC35

Figure 6. An example of problem behaviour graph

- After the subject developed a combined image of the Doric column and the bay window, he became aware that simple shape and construction expenses were constraints for the new image.
- When the subject combined the constraints of having visitors with the basement location for the workshop, he generated a half-level sunken workshop and was able to develop light and noise constraints for the new design unit.
- After the symmetrical disposition constraint had been applied, a centralized kitchen door was generated. However, the centralized kitchen door was also subjected to two constraints: the kitchen door should not be visually accessible from the living room, and the position of swing of the door should not disturb the user.

In conclusion, this study confirms the existence of goal

Table 12. Index of design unit for problem behaviour graph

First floor

Index of GD	Design units	Index of RD
	Building mass	RD001
GD01	Workshop	
	window	RD011
	wall	RD012
	roof	RD013
GD02	Living room	
	wall	RD020
	window	RD021
	entrance	RD022
	entrance door	RD0221
	steps	RD0222
	fireplace	RD023
	chimney flue	RD0231
	chimney	RD0232
GD03	Dining room	
	wall	RD030
	window	RD031
	door	RD032
	porch	RD033
	porch door	RD0331
GD04	Kitchen	
	wall	RD040
	window	RD041
	door	RD042
	range, sink	RD043
GD05	Bathroom	
	wall	RD050
	door	RD051
	tub, sink, WC	RD052
GD06	Doric column	
	glazing	RD061
	Staircase	RD002
	landing	RD0021
GD07	Bay window	
	glazing	RD071

Second floor

Index of GD	Design units	Index of RD
GD08	Master bedroom	
	wall	RD080
	window	RD081
	door	RD082
	double bed	RD083
GD09	MB bathroom	
	wall	RD090
	door	RD091
	dressing room	RD092
	wall	RD0920
	door	RD0921
GD10	Bedroom A	
	wall	RD100
	window	RD101
	door	RD102
	bed, study, wardrobe	RD103
GD11	Bathroom A	
	wall	RD110
	window	RD1101
	door	RD111
	Dressing room A	RD112
	wall	RD1120
	door	RD1121
GD12	Bedroom B	
	wall	RD120
	window	RD121
	door	RD122
	bed, study, wardrobe	RD123
GD13	Bathroom B	
	wall	RD130
	door	RD131
	Dressing room B	RD132
	wall	RD1320
	door	RD1321
	Roof	RD003
	watertank	RD0031
	Garage	RD004
	wall	RD0040
	door	RD0041
	walkway	RD0042
	car	RD005
	driveway	RD0051
	tree	RD006
	swimming pool	RD007

Note:
GD stands for given design units
RD stands for retrieved design units

plan, the different cognitive search methods used in design. It also explains how perceptual-test controls the progress of the system. The most important phenomenon is that the knowledge contained in constraint schemata provides resources for solution generation and testing. Therefore, the constraint schemata can be seen as tools for design problem solving. The ability of organizing and applying schemata determines a designer's design ability. These findings not only describe the nature of design process in detail, but have three additional meanings as well:

- that together they provide a basic framework for the novice designer to use in understanding the science of design and in developing his own design ability
- that the proposed model of schema representation furnishes a potential for representing the design domain-specific knowledge
- that the method used in discerning the knowledge state to compose the solution path also provides an opportunity for the study of style in design

ACKNOWLEDGEMENTS

The data was collected in the spring of 1987. I am indebted to Professors Herbert A. Simon, Omer Akin and John R. Hayes for their guidance on this research. Without their comments and encouragement, this paper could not have been completed.

REFERENCES

1 **Reitman, W** *Cognition and thought*, Wiley, New York (1965)

2 **Simon, H A** 'The structure of ill-structured problems' *Artificial Intelligence* Vol 4 (1973) pp 181–201

3 **Thomas, J C and Carroll, J M** 'The psychological study of design' *Design Studies* Vol 1 (1979) pp 5–11

4 **Akin, O and Baykan, C** *Problem structuring in architectural design*, Carnegie-Mellon University, Pittsburgh, PA (1984) in preparation

5 **Carroll, J M, Thomas, J C, Miller, A M and Friedman, H P** 'Aspects of solution structure in design problem solving' *American Journal of Psychology* Vol 93 No 2 (1980) pp 269–284

6 **Carroll, J M, Thomas, J C and Malhotra, A** 'Presentation and representation in design problem-solving' *British Journal of Psychology* Vol 71 (1980) pp 143–153

7 **Cinar, U** 'Facilities planning: a system analysis and space allocation approach' in *Spatial synthesis in computer-aided building design* (Ed. C M Eastman), Wiley, New York (1975) pp 19–40

8 **Henrion, M** 'Automatic space-planning: a postmortem?' in *Artificial intelligence and pattern recognition in computer aided design*, (Ed. J C Latombe), North-Holland, NY (1978) pp 175–192

9 **Akin, O, Chen, C C, Dave, B and Pithavadian, S** 'A schematic representation of the designers' logic' in *CAD and robotics in architecture and construction*, Hermes, Paris (1986) pp 46–58

10 **Simon, H A** 'Style in design' in *Proceedings of the 2nd Annual Environmental Design Research Association Conference* (Eds. J Archea and C Eastman), Dowden, Hutchinson and Ross, Stroudsbury, PA (1970) pp 1–10

11 **Newell, A and Simon, H A** *Human problem solving*, Prentice-Hall, Englewood Cliffs, NJ (1972)

12 **Eastman, C M** *Cognitive processes and ill-defined problems: a case study from design*, Proceeding of IJCAI, Washington, D.C. (1969)

13 **Eastman, C M** 'On the analysis of intuitive design processes' in *Emerging methods in environmental design and planning* (Ed. G T Moore), MIT Press, Cambridge, MA (1970) pp 21–37

14 **Akin, O** 'How do architects design?' in *Artificial intelligence and pattern recognition in computer aided design*, op. cit. pp 65–104

15 **Akin, O** 'Models of archictectural knowledge: an information processing view of architectural design' *PhD dissertation*, Carnegie-Mellon University (June 1979)

16 **Anderson, J R** *Cognitive psychology and its implications*, Freeman, San Francisco, CA (1980)

17 **Posner, M I** *Cognition: an introduction*, Scott, Foresman, Glenview, IL (1973)

18 **Winograd, T W** 'Frame representations and the declarative-procedural controversy' in *Representation and understanding: studies in cognitive science* (Eds. D G Bobrow and A Collins), Academic, New York (1975)

19 **Quillian, M R** 'Semantic memory' in *Semantic information processing* (Ed. M Minsky), MIT Press, Cambridge, MA (1968)

20 **Anderson, J R and Bower, G** *Human associative memory*, Winston, Washington, D.C. (1973)

21 **Rumelhart, D E and Ortony, A** 'The representation of knowledge in memory' in *Schooling and the acquisition of knowledge* (Eds. R C Anderson, R J Spiro and W E Montague), Lawrence Erlbaum Associates, Hillsdale, NJ (1977)

22 **Rumelhart, D E** ' "Schemata": the building blocks of cognition' in *Theoretical issues in reading comprehension* (Eds. R J Sprio, B C Bruce and W F Brewer), Lawrence Erlbaum Associates, Hillsdale, NJ (1980)

23 **Reitman, W R** 'Heuristic decision procedures, open constraints, and the structure of ill-defined problems' in *Human judgments and optimality* (Eds. M W Shelley and G L Bryan), Wiley, New York (1964) pp 282–315

24 **Miller, G A, Galanter, E and Pribram, K** *Plans and the structure of behavior*, Holt, Rinehart and Winston, New York (1960)

25 **Heath, T** *Method in architecture*, Wiley, New York (1984)

26 **Simon, H A** 'The functional equivalence of problem solving skills' *Cognitive Psychology* Vol 7 (1975) pp 268–288

27 **De Groot, A D** 'Perception and memory versus thought: some old ideas and recent findings' in *Problem solving* (Ed. B Kleinmuntz), Wiley, New York (1966) pp 19–50

28 **Simon, H A and Barenfeld M** 'Information processing analysis of perceptual processes in problem solving' *Psychological Review* Vol 76 (1969) pp 473–483

29 **Larkin, J, McDermott, J, Simon, D P and Simon, H A** 'Expert and novice performance in solving physics problems' *Science* Vol 208 (June 1980) pp 1335–1342

30 **Foz, A T K** 'Observations on designer behavior in the parti' *Master's thesis*, Massachusetts Institute of Technology (June 1972)

31 **Chan, C S** *Mental image and internal representation* Carnegie-Mellon University, Pittsburgh, PA (1988) in preparation

32 **Byrne, R W** 'Mental cookery: an illustration of fact retrieval from plans' *Quarterly Journal of Experimental Psychology* Vol 33A (1981) pp 31–37

33 **Simon, H A** 'How big is a chunk?' *Science* Vol 183 (1974) pp 482–488

34 **Posner, M I** 'Abstraction and the process of recognition' in *The psychology of learning and motivation* (Ed. G H Bower), Academic Press, New York, Vol 3 (1969)

35 **Simon, H A and Lea, G** 'Problem solving and rule induction: a unified view' in *Knowledge and cognition* (Ed. L W Gregg), Lawrence Erlbaum Associates, Potomac, MD (1974) pp 105–127

36 **Akin, O** 'An exploration of the design process' *Design Methods and Theories* Vol 13 No 3/4 (1979) pp 115–119

37 **Newell, A** 'Heuristic programming: ill-structured problems' in *Progress in operations research* (Ed. J Aronofsky), Wiley, New York, Vol 3 (1969)

38 **Hayes, J R and Simon, H A** 'Understanding written problem instructions' in *Knowledge and cognition*, op. cit. pp 167–200

39 **Newell, A** 'Production systems: models of control structures' in *Visual information processing* (Ed. W G Chase) Academic Press, New York (1973) pp 463–526

40 **Chi, M T H, Glaser, R and Rees, E** 'Expertise in problem solving', in *Advances in the psychology of human intelligence* (Ed. R Sternberg), Lawrence Erlbaum Associates, Hillsdale, NJ, Vol 1 (1982) pp 7–75