Omer AKIN
Chen-Cheng CHEN
Bharat DAVE
Shakunthala PITHAVADIAN
Department of Architecture
Carnegie-Mellon University
Pittsburgh

A schematic Representation of the Designers' Logic

Resume

Nos recherches actuelles visent a developper un modele operationnel permettant de decrire la demarche des designers en architecture. Nous comptons representer cette demarche par deux taches essentielles: la decomposition du probleme, c'est a dire l'etablissement des parametres descriptifs du probleme de design, et sa resolution, a partir des parametres definis precedemment. Nous allons decrire dans cet article une representation formelle de la demarche des designers, basee sur les resultats d'experiences d'analyse du comportement. Tout d'abord nous allons brievement decrire et analyser les experiences realisees. Ensuite nous presenterons le modele que nous avons developpe a partir des donnees recueillies lors de ces experiences. Ce modele repose sur plusieurs parties qui representent les differents types de connaissance employes par les designers dans les problemes de design architectural. Enfin nous decrirons un programme en cours de developpement, qui, nous l'esperons, permettra de verifier la justesse du modele.

Abstract

The aim of our research is to develop an operational model of the behavior of designers in the domain of architecture. We hope to account for the designers' behavior in terms of two major functionalities: Problem Structuring- establishing and transforming the parameters of the design problem; and Problem Solving- seeking a solution based on the known parameters of the design problem. In this paper, we present a formal paradigm of the designers' behavior that is based on protocol experiments. First, the protocol experiments are briefly described and analyzed. Next, the paradigm that we have developed based on the protocol data is presented. The paradigm is elaborated in distinct categories of representation to reflect the diverse nature of knowledge employed by human designers in the architectural design problems. This is followed by a brief description of a computer program under development that we hope will operationally demonstrate the efficacy of our paradigm.

1. Introduction

In the past few years, many researchers have attempted to explain the design process within the framework of information processing[5] theory. Based on protocol experiments, it has been shown that design expertise consists of retrieving structured information from memory and selecting representations that are appropriate for the design problem at hand[3]. Expertise in design grows with experience and provides the designer with 'pre-solution models'[4]. This form of pre-compiled knowledge guides a designer in his search for a solution. In most cases, the design problems are defined only tangentially and the designer transforms the 'ill-structured' problems into 'well-structured' component tasks[7] in the very process of design. It also has been observed that there may exist a large number of ways to traverse a solution space and select one solution, and hence the designers may tend to look for a 'satisficing'[6] solution rather than an optimal one.

A designer employs his knowledge in a goal-directed way[1] to search for a solution that fulfills a set of criteria that are either given to or established by him. To arrive at such a solution, the designer imposes a generative strategy, searches for solutions and evaluates them against the relevant criteria. This process does not entail exhaustive generation of solutions but it is heuristic in nature[2].

1.1. Our Goals

The ultimate goal of our work is to develop an operational model that accounts for the designers' behavior in terms of two functionalities: Problem Structuring- establishing or transforming the parameters of the problem, and Problem Solving- seeking a solution that satisfies the established parameters. A more immediate goal of our work is to model the behavior of an expert and a non-expert designer solving a non-trivial design task. The goal of this paper is to describe our research approach and to present a formal paradigm that captures the behavior of these designers.

2. Approach

Building models of phenomena has always relied on empirical evidence gathered from observations. Such models are then represented in abstract terms and validated by showing how they account for, explain or predict similar phenomena. Protocol studies provide a useful tool for collecting such observations. And we resorted to protocol experiments as the first step in our work.

2.1. Protocol Experiments

Subjects were given two different tasks: a bin-packing problem and a space-planning problem. The first problem requires reaching a spatial arrangement that meets with specified criteria. The second problem involves establishing the criteria to facilitate reaching of the spatial arrangements that meet those criteria. As design expertise involves skills in both these functionalities, our aim was to discover similarities and differences adopted by subjects with different skills.

Two different subjects were chosen for these tasks: an operations research specialist (S1) and an experienced architect (S2). By choosing these two subjects, we were hoping to see if there was any correlation between the problems and the strategies adopted by two different experts with different domain knowledge.

As part of the bin-packing task, the subjects were given 23 rectangular pieces to fit (without overlap) in a specified rectangular area. In the task of space-planning, the subjects were asked to design a layout of an office for a small engineering firm. They were given a list of personnel and furniture to be accomodated. Subjects were instructed to verbalize their thoughts while they were working on both the tasks. Entire session was video-taped and it was later transcribed as a sequential dialogue, supplemented with sketches marking the stages in the development of solutions.

2.2. Protocol Analysis

2.2.1. Patterns of Decisions

Two major decision patterns emerge from the protocol analysis. One of them involves establishing the relevant parameters of the problem at hand. We refer to this activity as Problem Structuring. Following example illustrates this concept:[1]

```
S2:23. These are two doors here?
S2:24. Which is the primary one?
       Experimenter : Doesn't matter.
S2:25. One becomes primary though.
S2:26. What I've got to do is invent a scenario and design to it.
S2:27. If there'd be time, I'd design to a different scenario.
       Experimenter : How would you use that scenario?
S2:28. There are certain constants that'd come in like someone is going to be
       coming in from time to time and the engineers wouldn't want to be
       disturbed so the secretary would be an interceptor.
```

In this episode, S2 is trying to establish a spatial relationship : primary entrance, need for controlling it, and locating secretary as a desirable controller. With this structural information, now all that S2 needs is to assign a primary entrance and to locate secretary in reference to that. Thereafter, the design process can move onto a more focused task referred to as Problem Solving, e.g.:

```
S2:84. This is the way people come in allowing a separation.
S2:85. We'll put the secretary someplace ...
```

Such information can be applied for generating a solution (as in the above example) or for testing it:

```
S2:200. I can't break the front door relationship with the secretary.
        It is the one rule not to break.
S2:201. If I break that one, then one of those people functions as
        receptionist and that's a no-no.
```

2.2.2. Predicates

Once the major patterns of design decisions were identified, underlying relationships became apparent. Some of these relationships between elements of design have to be maintained while generating a solution and others have to be tested at the end. In other words, there are relationships among elements of design which, depending on the context, the subjects selectively applied as either generative rules, i.e. constraints or evaluative rules, i.e. criteria.

```
S2:41. The conventional one, which is hierarchical.
S2:42. Which means that the chief engineer wants to be separated from the others,
S2:43. wants to have direct access to the secretary
S2:44. and yet wants to be able to bypass the secretary for direct access to
       the other engineers.
```

[1]From now on, all the examples from protocol are preceded by S1 or S2, and a number indicating the subject and the statement as numbered in the transcript for the space-planning task.

In this example a relationship is identified and its design implications (i.e 'employer needing privacy') are used to generate a solution. This assertion or predicate, in turn, leads to another predicate about 'spatial privacy', defined in terms of spatial location and degree of enclosure provided. Such forms of reasoning commonly seen in designers' behavior can go to arbitrary depths of processing for obtaining specificity, only bounded by the scope of the subject's past experiences to make such inferences.

2.2.3. Representation

Predicates can be differentiated on the basis of the kind of information they represent. Furthermore they are instrumental in transforming information (relationships) from one representation to another so as to affect a change in the current design state or to provide inferences about it. A brief categorization of such representations and possible transformations is discussed next.

Problem-dependent relationships are global and diffuse in character. Usually they provide the point of departure for design e.g.- *the layout for an office*. This, on one hand, eliminates other building types from consideration, yet on the other hand, it opens up a next level of decisions, e.g. *will it be an open-plan office or a territorialized office with partitions*. And the designer has to know or choose which option to adopt based on his domain and problem-specific knowledge. Similarly, the kinds of spaces and the objects (furniture) to be incorporated in the design solution provide evidence which supports one choice or another based on the relationships to be maintained in the design.

Some other relationships derive from the general domain knowledge of architecture and they are invoked in almost all design tasks e.g. *North light is better for working surfaces*. These relationships can be thought of as the 'common-sense' knowledge on the part of designers.

All these relationships can ultimately be translated into topological and geometric attributes of the objects under consideration, thereby resulting in physical representations of the solutions. The relationships dependent on topology of elements or objects are subtle.

```
S2:151. I can't get over the fact that when I go into offices, people sit with
        their backs to light.
S2:152. I like to work with the light coming over the left side because I am
        right handed.
S2:153. I am impressed on how people use it as the ceremonial thing, the
        formality. I am boss and you sit there.
```

In this example, many topological notions- front, back, left, right, proximity, are called into play. These assertions refer to many relationships that have to be finally expressed through physical objects and their locations, and that is when their geometric attributes become evident.

To sum up, there seems to exist a tightly meshed network of relationships. The subject, whenever he does not have a specific and externally defined relationship, draws upon his personal expertise to infer or generate missing information. The process seems to proceed from problem-specific information, generating scenarios of relationships, asserting them in terms of topological and geometric attributes and testing these relationships for accepting or rejecting a solution.

3. The Paradigm

In the process of design generation, we observed two basic functionalities: Problem Structuring and Problem Solving. Both these functionalities draw upon various representational forms of knowledge.

In the following sections, various components of these functionalities are discussed. And later in this section, a comprehensive discussion brings them together.

3.1. Problem Structuring

Given a problem statement, the designer's behavior can be characterized as a series of transformations of information starting with functional issues and transforming them eventually into formal manipulations of objects on the drawing board. Initially, the designer expresses functional ideas that the final solution has to satisfy in the form of scenarios. These scenarios, in turn, provide the decomposition of the design elements or units into a hierarchic organization coupled with access or adjacency relationships among the component units. Both these relationships establish problem dependent parameters of the final solution. In addition to these, a set of generic relationships is to be satisfied independent of the problem at hand. This latter category establishes parameters that are considered in most if not all design problems (i.e. windows should not be bisected by walls, doors are needed for access through walls, etc.).

Establishing the parameters which define a solution space consisting of design units and the predicates stating the desired relationships among these units, has been called Problem Structuring. These parameters and solution space may be restructured at a later stage in design as a function of preceding attempts at finding a solution; this we will call Problem Re-structuring.

3.1.1. Scenarios

Scenarios are constructs which represent behavioral patterns of people in spatial environments. They are derived from our daily experiences and provide useful abstractions for capturing and expressing large chunks of functional knowledge.

```
S2:30. That is why I was looking to see if it was a highly participatory
       office regardless of rank or if wasn't.
```

In this example, S2 attempts to infer behavioral pattern that is relevant to a design solution under consideration. Such information may be given as part of the problem statement (design brief) or the designer may have to retrieve and assert relevant information from his domain experience. It serves a useful purpose in that the design inquiry becomes more focused for subsequent stages.

The ability to encapsulate and utilize broad features of the design situation typically distinguishes an expert from a novice. A novice designer is more likely to start with partial details and progress towards a behavioral construct which may or may not be explicitly stated. In our experiments, S1 started with individual design requirements and additively, generated a solution. He did not start with an initial overall behavioral conception like S2 but satisfied local constraints as he progressed.

This suggests two different approaches to design. One in which, based on the problem statement and scenarios, operational and access relationships among component design units (spaces) are established. Problem solving subsequently is based on these relationships and their satisfaction. In the second case, the designer starts from a priori relationships known to be necessary (from experience) and builds towards general behavioral scenario through specific solutions. The first akin to top-down search characterizes the expert's behavior while the second akin to bottom-up search characterizes the novice's behavior.

3.1.2. Access Relationships

Based on the scenarios, access relationships represent interactions in terms proximity implied or required by the design problem among design units, e.g.:

```
S2:42. Which means that the chief engineer wants to be separated from the others,
S2:43. wants to have direct access to secretary
S2:44. and yet wants to be able to bypass the secretary for direct access to
       the other engineers.
```

Primarily, in our space planning experiments, access relationships are interpreted in three distinct meanings: direct, easy and indirect. Directly accessible implies that a space is adjacent to another space and is linked through a door. If a space is linked to another space through a circulation space, it is called easily accessible. Indirect access means that two spaces are linked through a chain of two or more circulation spaces. Depending on the nature of the design task, these access requirements are asserted although it is not unusual to have more than one of them as being acceptable.

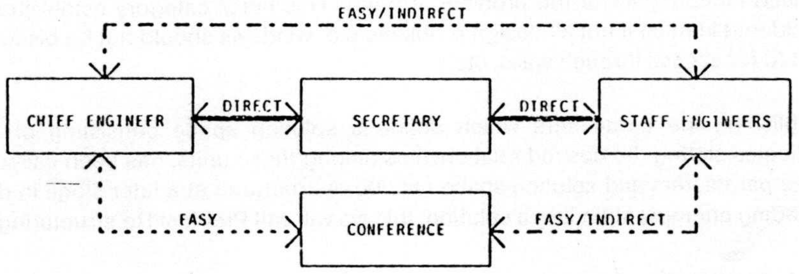

Figure 1: Access Relationships : S1

Access relationships just establish desired proximity among design units and depending on how these constraints are resolved, the result could be represented as adjacency relationships. However, adjacency does not always mean access and vice versa. The need for these two nearly similar representations stems from the fact that access relationships are closer to how designers think, they recognize that constraints are not immutable and that every design solution invariably has incidental or circulation spaces that are not specified a priori. In this sense, access relationships are only partial descriptions to which edges are added in the process of design, finally expressing adjacency relationships among the design units.

3.1.3. Design Units Hierarchy

A design product like a building is an assembly of parts (spaces). Each of these parts, in turn, are made up of smaller parts and so on until we reach the smallest units that can be usefully described in terms of their relevant attributes. In our experiments, hierarchy of spaces and objects they comprise of is shown in Fig.2. This hierarchy is one of the universal relationships assumed by all of our subjects.

As a representation, this hierarchy embodies only part-of relationship and is useful in the sense that it reduces the attention of the designer to the level at which he is mediating currently. Based on furniture that a spatial unit is comprised of, its dimensions can be ascertained. When such units are determined in our experiments, initially the designers tend to think in terms of spaces rather than paying attention to smaller details like furniture. Once the layout of rooms is decided, however

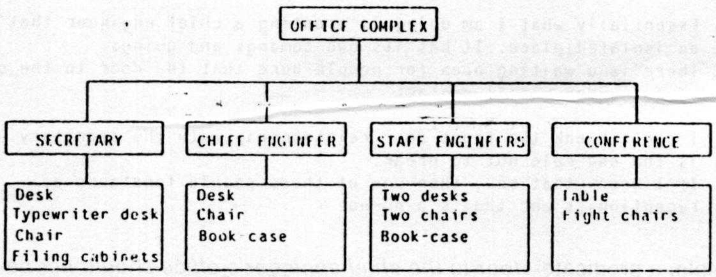

Figure 2: Design Units Hierarchy : S1

attention shifts to furniture arrangements and this informs any refinements necessary for the room layouts.

3.1.4. Generic Relationships

In addition to the design units hierarchy that is brought to bear on the problem from the designer's long term knowledge and experience, the designer also brings to the problem knowledge about how things generally ought be. These constraints embody generic relationships among the design elements often independent of the specific problem at hand and they are observed in the behavior of most designers regardless of their expertise, e.g.- *Each space should be connected to the main entrance via other spaces or circulation areas; Partition walls should not divide existing windows.*

These generic relationships state what is desirable but their application for generation or evaluation of a design solution is discretionary based on the design context. In other words, these constraints may be added or dropped to the existing set of constraints to reconfigure solution space. Since these relationships are part of the domain knowledge, an expert designer would use a much larger range of information in this category than a novice. In our experiments, S1 did not seem to be aware of many such requirements and in fact, he violated a number of such requirements for example, he did place a partition wall that divided the windows and a radiator.

3.2. Problem Solving

Having structured the problem parameters and thereby defining the solution space, the designer's task is to find a specific solution. Problem solving functionality, in this sense, attempts to search for a solution that satisfies these multiple requirements or constraints. The process of search for a solution is a purposeful one. To this end, problem solving can be described in terms of a control structure, predicates that are used as generators which make assignments of design units in the problem domain and as testors that evaluate these assignments. These problem solving components, based on the current information, transform the current design state into a successive one, progressively getting closer to the final solution. Problem Solving also involves sending feedback to higher level operations which is often used in restructuring the solution space when the need arises.

3.2.1. Predicates

Predicates which are asserted at this stage are derived from problem specific and generic information used in structuring the problem. These can be viewed as relationships which have to be fulfilled in order to solve the problem as it is structured. Following two examples illustrate these concepts :

```
S2:138. Essentially what I am doing is creating a chief engineer that's
        an isolated place. It has its own comings and goings.
S2:139. There is a waiting area for people here that the door to the office
        is not under visual contact.

S2:200. I can't break the front door relationship with the secretary. It
        is the one rule not to break.
S2:201. If I break that one, then one of those people functions as
        receptionist and that's a no-no.
```

In the first example, a predicate- *door to the chief engineer's office should not be in the direct line of sight from the waiting area*, is used as a test for the generated locations and a contingent assignment- *location of door*, is made. In the second case, a predicate- *secretary should be placed close to the front door*, is used to reject a solution.

The predicates can be expressed as either constraints- relationships based on which design solutions are to be generated, or criteria- relationships which are used as testors for acceptability of design solutions. Some of the most basic operations that are carried out as part of asserting these predicates involve operations on design units in terms of assignment or deassignment of locations or attributes, dimensional reduction or enlargement, checking for overlap, etc.

3.2.2. Control Structure

Control structure directs the process of solution generation based on the relationships mandated through the problem structuring function. It is characterized by two basic operations: generating possible locations for a design unit and testing which of these satisfies other contingent requirements. Control structure determines, out of a set of possible operations, the sequence of generate and test operations to apply in the current design state. This implies strategic knowledge about promising paths of inquiry which may be: satisfying the most constrained tasks first, bottom-up approach in which constraints are satisfied one at a time or top-down approach wherein more global constraints are attended to first.

Our notion of the control structures needed in problem solving is presently evolving. However, at the moment we assume a generate and test paradigm as the producer of possible locational assignments for design units. A typical ordering mechanism for the generator found in the protocols, is to start with the most constrained component of the design problem. In our experiments, both subjects establish different access requirements but in both the cases, placement of secretary turns out to be the most constrained task and the key to their solutions. And both the subjects proceed with this assignment first.

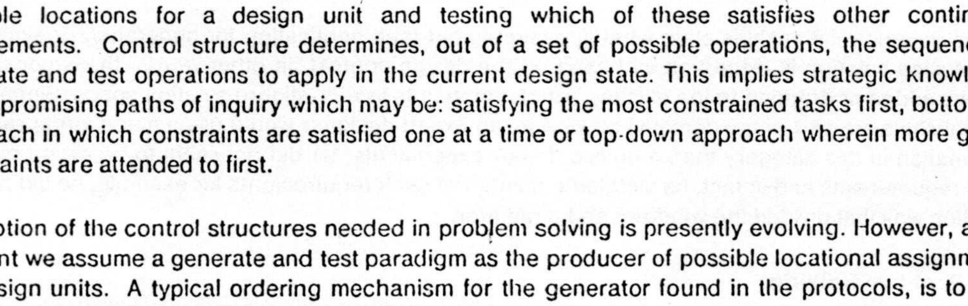

Figure 3: Conflict Resolution : S1

In case a current solution space fails to produce an acceptable solution, a conflict is detected and pertinent resolution operation is undertaken. As an extended example, S1 started with the assignment sequence as shown in Fig.3. At this stage, one more design unit (conference) had to be accommodated. Since, it was not feasible given the locational assignments of other units, he reassigned the geometric locations, maintaining the access relationships that were already satisfied.

3.3. Review of Protocols and Paradigm

Earlier discussion covered various components of the paradigm that we have developed based on the protocol studies. There exist interdependencies among various components as shown in Fig.4. In this section, we will try to point out major differences observed in the approach of both subjects.

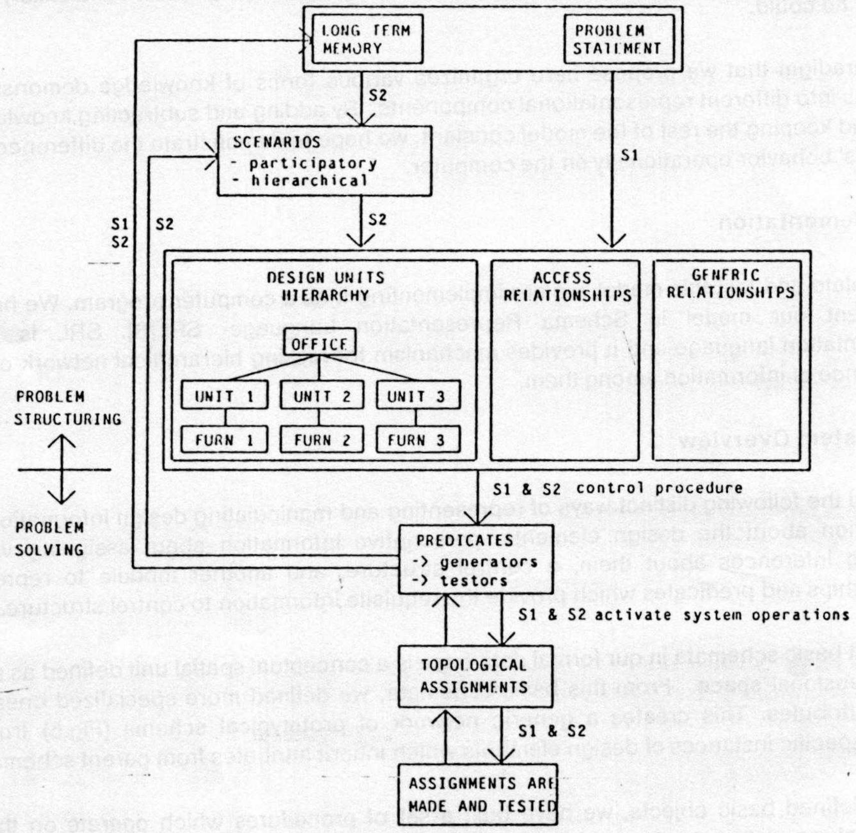

Figure 4: The Paradigm

S1 as a novice designer did not use any overall strategy or global scenario in generating the final solution. Instead, he started out with a singular design unit i.e. the secretarial unit and added other units in relation to it. Such an approach could be characterized as an additive, bottom-up processing that lacks a more holistic overview of the design task. Although S1 did not start with a scenario, he did show some conception of design units hierarchy and access requirements among them, based on which he ultimately arrived at a solution. But at the same time, he failed to take account of more generic relationships among the design units and violated some of the previously satisfied constraints in the later stages of design generation.

On the other hand, S2, being an expert designer, generated several scenarios, from which he deduced alternative access relationships. These, together with generic relationships among the design units which had to be satisfied a priori, formed the core of information based on which he generated design solutions. He asserted more problem specific predicates, some of which clearly forced him to redefine the predicates that were established previously. To a major extent, the differences in the approach of both the subjects can be accounted in terms of domain knowledge. The expert relied on a larger set of domain specific information, thus restructuring the problem several times in his search for the best tradeoffs between alternative problem structures. The novice on the other hand defined the problem once at the onset and then proceeded to satisfy its parameters as best he could.

The paradigm that we propose here organizes various forms of knowledge demonstrated by both subjects into different representational components. By adding and subtracting knowledge at the top level and keeping the rest of the model constant, we hope to demonstrate the differences between the subjects' behavior operationally on the computer.

4. Implementation

To simulate and test this model, we are implementing it as a computer program. We have chosen to implement our model in Schema Representation Language- SRL[8]. SRL is a knowledge representation language and it provides mechanism for making hierarchical network of schema and inheritance of information among them.

4.1. System Overview

We need the following distinct ways of representing and manipulating design information: descriptive information about the design elements, prescriptive information about assigning values to and returning inferences about them, a control structure, and another module to represent various relationships and predicates which provide the requisite information to control structure.

The most basic schemata in our formal data-base is a conceptual spatial unit defined as a rectangle in two-dimensional space. From this basic schemata, we defined more specialized ones which have added attributes. This creates a generic network of prototypical schema (Fig.5) from which we defined specific instances of design elements which inherit attributes from parent schema.

Having defined basic objects, we have built a set of procedures which operate on them to either assign values, retrieve values and return inferences, to be used subsequently. These include checking and updating adjacencies; assigning or returning edge (wall) attributes; locating furniture patterns as a function of orientation and area of spatial units; enlarging or reducing a spatial unit to cover residual areas or resolve overlap conflicts; etc. This module together with data-base is already encoded at present.

These procedures are to be driven by a control structure which, in turn, operates on the basis of higher level information in the form of design scenarios, access and generic relationships, etc. Currently we are working on the representation of relational predicates and the control structure as well as identification of higher level heuristics that would assert design scenarios.

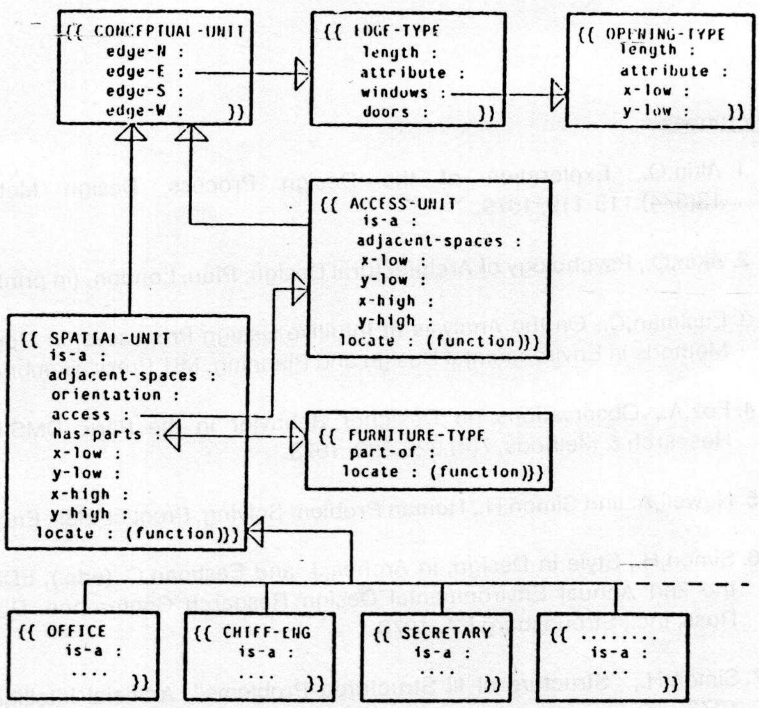

Figure 5: Network of Schema

5. Conclusion

Our analysis of the designers' behavior in solving space-planning problems has led to some important distinctions between expert and non-expert designers. A primary distinction is the use of scenarios (primarily used by experts) as the organizers of problems and problem solving behavior. Another distinction is the relative richness of the expert's domain behavior which exhibits itself in the number of predicates to be satisfied and number of restructurings the problem goes through.

Some notable similarities also exist. Design at both levels is primarily a goal directed generate-and-test behavior driven by general purpose heuristics, such as, most constrained first and least effort principle.

Through the articulation of components responsible for Problem Solving, Problem Structuring and scenarios, the similarities as well as differences can be expressed in an operational sense in the architecture of a system under development. We hope to show, through this system, the causal relationships between knowledge and expert-like or non-expert-like behavior.

Acknowledgement

This research is funded by NSF Grant No: CEE-8411632.

References

1. Akin,O., Exploration of the Design Process, Design Methods and Theories, 13(3/4):115-119, 1979.

2. Akin,O., Psychology of Architectural Design, Pion, London, (in print).

3. Eastman,C., On the Analysis of Intuitive Design Processes, in Moore,G. (ed.), Emerging Methods in Environmental Design and Planning, MIT Press, Cambridge, 1970.

4. Foz,A., Observations on Designer Behavior in the Parti, DMS-DRS Journal: Design Research & Methods, 7(4):320-323, 1973.

5. Newell,A. and Simon,H., Human Problem Solving, Prentice Hall, Englewood Cliffs, 1972.

6. Simon,H., Style in Design, in Archea,J. and Eastman,C. (eds.), EDRA II: Proceedings of the 2nd Annual Environmental Design Research Conference, Dowden, Hutchinson & Ross, Inc., Stroudsburg PA, 1970.

7. Simon,H., "Structure of Ill Structured Problems", Artificial Intelligence, 4(3-4):181-201, 1973.

8. Wright,J. and Fox, M., SRL/1.5 User Manual, The Robotics Institute, Carnegie-Mellon Uni., Pittsburgh, 1.5 ed., 1983

Authors' Name Omer Akin
Chen-Cheng Chen
Bharat Dave
Shakunthala Pithavadian

Address Department of Architecture
Carnegie-Mellon University
Pittsburgh PA 15213 USA

Phone (412) 268 2356 **Telex** 469002