Formal Representation of Design Knowledge and Process

Omer Akin and Bharat Dave

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

KEYWORDS

architectural design, domain knowledge, heuristics, formal representation, simulation

ABSTRACT

Our work focuses on the processes of search and inference involved in architectural design. This requires identification and representation of knowledge that an architect brings to bear upon the design task. Based on a protocol analysis of an expert architect solving a design problem, we describe a paradigm that has been developed to simulate these processes into a computer program. The paradigm organizes various representations used by the designer in distinct knowledge categories representing geometric, topological and contextual relationships. These, in turn, are driven by design propositions in the generation of design solutions and their evaluation. This paper broadly describes features of the paradigm that have been implemented as a computer program.

Representation Formelle des Connaissances et Processus lies a la Conception

Omer Akin et Bharat Dave

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

MOTS-CLES

design architectural, domaine de connaissances, heuristiques, representation formelle, simulation

RESUME

Nos travaux portent sur les processus de recherche et d'inference mis en jeu lors du design architectural. Cela requiert l'identification et la representation des connaissances qu'un architecte utilise lors de la phase de design. En se basant sur l'analyse de la demarche d'un architecte expert confronte a un probleme de design, nous decrivons un modele destine a simuler ces processus sur ordinateur. Ce modele classe les diverses representations utilisees par le designer dans differentes categories de connaissances representant les relations geometriques, topologiques et contextuelles. Celles ci sont ensuite activees par les propositions de design pour engendrer les solutions et leur evaluation. Cet article decrit les caracteristiques du modele tel qu'il a ete implemente sur ordinateur.

1. BACKGROUND

In the past few years, many researchers have focused their attention on understanding design expertise. Many of these efforts view design process as a problem-solving activity within the framework of information processing (1) theory.

Based on protocol analysis, it has been shown (2) that design expertise is reflected in selecting appropriate representations and being able to retrieve structured information from memory rather than external cures. Expertise accrues over experience and provides a designer with 'pre-solution models' (3). Such an approach restricts and directs search for solutions depending on specificity of the design problem. In most cases, a designer does not have explicit specifications for the solution but renders them explicit within the process of design, transforming 'ill-structured' problems into 'well-structured' component tasks (4). Over the years, an expert develops a repertoire of promising strategies for effectively utilizing his expertise in a goal-directed way (5). Since design problems usually have a potentially large number of ways to traverse the solution space and select one solution, it has been also proposed that designers tend to 'satisfice' rather than 'optimize' solutions (6). These ideas provide some of the major characterizations of design expertise.

A designer applies his expertise in a purposeful fashion, searching for solutions that are subject to a set of criteria. Most of the criteria are derived from domain knowledge rather than being given explicitly as part of the problem definition. Solutions are generated on the basis of what is known at each stage and are evaluated against applicable criteria. These processes of search and inference in design are not exhaustive or algorithmic in nature. Instead, designers employ heuristic methods (7) to guide their search.

1.2 OUR GOALS

Our ultimate goal is to develop an operational model of the designer's behavior in terms of two functionalities (8): inference making (problem solving)- seeking a solution on the basis of known parameters of the problem; and search (problem structuring)- an activity that transforms parameters of the problem. A more immediate goal of our work is to model the behavior of an expert architect solving a non-trivial design problem. In this paper, we will describe our research approach and findings so far and general properties of the model that we have developed.

2. APPROACH

Building models of phenomena has always relied on empirical evidence gathered from observations. By collecting all pertinent observations, underlying patterns that explain phenomena have been elucidated in various disciplines. Protocol studies, in this sense, provide a useful tool for these purposes. We adopted this tool to address several issues discussed below.

An expert designer has at his disposal a set of well-proven or promising actions for a given context. These actions or schemata represent structured relationships among and information about the elements of design. Our immediate concern has been to discover these relationships and information; and how and when such information is brought to bear upon the development of design.

2.1. PROTOCOL EXPERIMENTS

To uncover these issues, subjects were given two different tasks; a bin-packing problem and a space-planning problem. In the first problem, the subjects were given 23 rectangular pieces to fit (without overlap or gap) in a specified rectangular area. In the task of space-planning, the subjects were asked to design a layout for an office for which a list of personnel and furniture to be accommodated was provided. For both the tasks, the subjects were provided with cardboard pieces representing rectangles or furniture, to be fitted in a delimited boundary on a board.

Two different subjects were chosen for these tasks: an operations research specialist (S1) and an experienced architect (S2). They were instructed to verbalize their thoughts while solving the tasks. The entire session was video-taped with sound on the same tape. Following the recording, the session was transcribed in a sequential list of sentences, supplemented with sketches marking the stages of design development. (A complete discussion of experiments and differences in behavior of both the subjects is included in Baykan (9).)

2.2 PATTERNS OF DECISIONS

Two major patterns of decisions 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. An example of this is[1]:

23. These are two doors here?
24. Which is the primary one?
Experimenter: Doesn't matter.
25. One becomes primary though.
26. What I've got to do is invent a scenario and design to it.
27. If there'd be time, I'd design to a different scenario.
Experimenter: How would you use that scenario?
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 door, need for control, location of secretary. With this structural information, now all S2 needs is assigning a primary door and locating secretary in reference to that. The design process moves onto a more focused problem solving task.

[1]numbers refer to line numbers of S2's protocol in solving the space-planning problem

84. This is the way people come in allowing a separation.
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:

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

First we identified such statements in the protocol transcript, providing us with some broad design strategies used by S2. These statements, in essence, reflect desired relationships among design objects. These relationships are selectively applied by S2 as either generative constraints or criteria of acceptability or rejection of a solution. These predicates of relationships are large chunks of domain knowledge which can be represented as component expressions with varying degrees of specificity.

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

In this example a relationship is identified and its design implications (i.e. chief engineer needs privacy) are used to generate solutions. This predicate, in turn, leads to another predicate 'spatial privacy', defined in terms of spatial location and degree of enclosure provided. This can go to arbitrary depths of processing for obtaining specificity, although the subject relies on his past experiences to make such inferences. At the bottom most level of such predicates, we see what are perceptual assertions which don't seem to be further atomized.

48. Because corners are more valuable than centers because they are bounded, which means no intrusions can come in.

To sum up, the processing sequence seems to flow through 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 to generate missing information. The process consists of identifying 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.

These relationships can be differentiated on the basis of the kind of information they represent. The important feature of such representations is to transform information (relationships) from one to another. Following such an analysis, we developed a paradigm that can be encoded and tested as a computer program.

3. THE PARADIGM

3.1 PROBLEM-SOLVER AND PROBLEM-STRUCTURER

As observed in the protocol and its subsequent analysis, the design activity can be seen in terms of two functions: establish parameters of the problem (problem structuring) and find a solution within those parameters (problem solving). As part of the first, the designers identify "legal" states (8) ensuring that the final solution is likely to be found within that search space. If the problem still eludes an acceptable solution, some of the parameters- criteria are restructured so as to reconfigure the boundaries of the search space.

Problem solving procedures take given problem spaces and generate solutions based on known parameters. Just as in restructuring a problem, if a solution seems to satisfy criteria (based on which it was generated) but violates some other, then a new solution is to be generated that satisfies the new criteria added with the previous ones.

3.2 PROBLEM-SOLVER

These design operations can be represented as transformations of relationships as observed in the patterns of decisions in our protocol analysis. Briefly, they can be elaborated under the following representation domains: geometric, topological, tautological (perceptual), scenario-driven.

Geometric representations are the primary values by which objects are assigned 'physical' substance. These can be seen as the dimensions, planar equations or the location of an object in space.

Topological representations provide definitions of shapes. Primarily, they explicate planar invariants of an object. These are used as templates for defining a shape to which are attached the geometric attributes mentioned earlier.

Tautological representations encompass concepts which are largely perceptual, e.g. spatial privacy. They could surely be made manifest or manipulated in terms of topological or geometric attributes but yet are distinct from them.

Scenario-driven representations rely heavily on the external information about the problem as well as personal expertise of the designer. In other words, they can be seen as the top-level relationships which permeate down in other representational domains and attain specificity. This implies that all these representation domains are not independent of each other and tend to fuse together or to be defined in terms of each other.

To illustrate the adequacy of such data representation, we can refer to the following episode in the protocol:

34. The first thing I was working on is a scenario with no walls because there is no reason for walls.
35. The purpose of the walls would be if they needed to hang things up, or

if they needed visual or acoustical privacy.
36. But visual privacy can be achieved without walls just turning desks back to back, so the direction you are looking into gives you the privacy. Strategy of people looking in different ways.

Here S2 deliberates on a possible scenario- not using the walls, and debates its implications on functioning of the office. This derives from his domain knowledge and that in turn, brings up the notion of privacy- a tautological predicate. It could be achieved by translating it in terms of topological patterns- placement of desks facing away from each other. And at this level of detail, it is possible to assign geometric values to satisfy all these constraints.

Although we have elaborated these concepts as distinct representations, usually they are manipulated as composite entities (constructs). The site that was given to the subjects in the experiment is an example of constructs. It was given as a volume of space as part of an existing building. This defines its 'interior' and 'exterior' spaces. Being a rectangular volume, it has six bounding planes, two of which are 'solid' walls, one contains 'windows', another 'doors'. And each of the elements in this assembly of primitives has geometric values assigned to it.

3.3 PROBLEM-STRUCTURER

Such representations by themselves do not say anything about design development which depends on the given problem context as defined or interpreted by the designer. They are invoked and manipulated in response to different design contexts. To restate an earlier example- if the layout is to be hierarchical, functions should be placed in the order of their status, contains a condition. These assertions derive from the domain knowledge about what is desirable for a given design context. If a solution is not feasible- e.g. within the given site, if the functions can't be placed as desired; then domain heuristics is exercised to assert an alternate path of inquiry. In the end, these predicates of knowledge can be thought of as the agents of control that invoke and manipulate constructs or primitives discussed above.

4. IMPLEMENTATION

To formally test and compare this paradigm, we are implementing it in a computer program in SRL (10). SRL is a knowledge representation language and it provides mechanism for making hierarchical networks of schema and inheritance of information among them.

4.1. TEMPLATES OF OBJECTS AND RELATIONS

As shown in the analysis of the protocol, we need three ways of representing and manipulating design information. Descriptive information about the design elements, prescriptive information about assigning values to them and another module to trigger and control the processing sequence. In essence, we intend to implement design information in three tiers: objects and their known attributes are represented as schema, their relational data (e.g. locations) are to be computed as functions of the known attributes. These,

in turn, are to be driven by conditional predicates to be represented as functions or productions (e.g. IF-THEN statements).

Most basic schema are conceptual spatial units. They have two major attributes: space that is occupied by the unit, and space that is required for functionally accessing that unit. From this basic schema, we defined more specialized ones which have added attributes. This created a generic network from which we can create specific instances of design elements.

Having defined basic objects, we have built a set of procedures which operate on them to assign and/or 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. These are to be augmented by procedures to invoke design scenarios and consequent actions.

These procedures are the workhorses of the program. But they have to be driven by higher level domain heuristics, providing the problem structuring functionality to the program. That is the next module to make the program operational and test it against behavior of the designer observed in the protocol. This module is not implemented yet.

4.2 FUNCTIONAL BEHAVIOR

Basic object schemata to be manipulated in the design task are stored in the database. But these are prototypes and only specific instances of them are to be manipulated in different design contexts; the instances would not have locational information at the start of the system. For these instantiations to occur, the system needs to assert particular predicates based on design strategies and scenarios.

Once these top-level predicates are asserted, they invoke actions. These actions are the implications of the predicates, invoked as procedures that are already defined (e.g. for assigning location to a spatial unit or evaluating if two spatial units are in visual proximity of each other). These functions operate on the values already present in the object schema and either change, add or simply retrieve necessary data and provide inferences.

Currently, we have encoded the basic descriptions of the objects, functions that manipulate these data, and compute and generate locational values for different objects. The scenario-procedures to trigger specific series of actions for a design context are being developed. At the same time, we are also working on identifying heuristics that would assert these scenarios and thereby provide the top-level control.

In its present state, there is much that remains to be implemented in the computer program to enable us to draw concrete conclusions but its partial performance seems encouraging enough to substantiate our paradigm.

ACKNOWLEDGEMENT

This work is funded by NSF Grant No: CEE-8411632. We would also like to acknowledge collaboration of Can Baykan during the early phase of this project.

REFERENCES

1. A. Newell and H.A. Simon, Human Problem Solving, (Prentice Hall, Englewood Cliffs, 1972).

2. C. Eastman, "On the Analysis of Intuitive Design Processes", Emerging Methods in Environmental Design and Planning, edited by G.T. Moore (MIT Press, Cambridge, 1970), Chap. 3, pp. 21-37.

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

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

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

6. H.A. Simon, "Style in Design", EDRA II: Proceedings of the 2nd Annual Environmental Design Research Conference, edited by J. Archea and C. Eastman (Dowden, Hutchinson & Ross, Inc., Stroudsburg, PA, 1970), pp. 1-10.

7. O. Akin, "Models of Architectural Knowledge: An Information Processing View of Architectural Design", (Carnegie-Mellon University, Pittsburgh, 1979), PhD thesis.

8. O. Akin, "A Formalism for Problem Restructuring and Resolution in Design", (Dept. of Arch, Carnegie-Mellon University, Pittsburgh, 1985), manuscript.

9. C. Baykan, "Heuristic Methods for Structuring Architectural Design Problems", (Dept. of Arch, Carnegie-Mellon University, Pittsburgh, 1984), manuscript.

10. J. M. Wright and M.S. Fox, SRL/1.5 User Manual, (The Robotics Institute, Carnegie-Mellon University, Pittsburgh, 1983), 1.5 ed..