KAAD: Knowledge-based Assistance for Architectural Design

G. Carrara and G. Novembri
Department of Building and Environmental Control Technologies
Rome University 'La Sapienza', Rome. Italy.

ARSTRACT

The research being conducted at the CABD LAB at the Department of Building and Environmental Control Technologies is geared to the production of an Expert System for architectural design, which is able to perform interactive design tasks and help to provide an accurate and complete description of the buildings in question.

The Expert System will control the design process, continually ensuring consistency between the definitions of the designer and a given set of constraints.

Accordingly, the System will be able to determine the effects of choices taken at different stages of definition, performing the necessary calculations and checks.

The System is based on a general representation of the building objects, from individual components to the whole building defined in terms of a number of hierarchical, topological and functional relational structures resulting from earlier research conducted into the automatic management of architectural design since 1975.

1. INTRODUCTION

In recent years, professionals engaged in the building process have resorted increasingly to dedicated architectural design software, such as drafting and management systems, data bases, calculation and evaluation programs.

However, at the same time, it has become increasingly necessary to integrate these tools into a single system that will help the designer through the various design phases.

To meet this need, a research project is being conducted at the CABD LAB of the Department of Building and Environmental Control Technologies to create a dedicated architectural design expert system.

The proposed system will serve as an interface between the designer and the conventional software, and can perform the following tasks:

- interacting with the designer in a simple and natural manner using a high level language;
- checking the consistency of the data input and ensuring that they match a given system of constraints;
- managing the conventional software tools required in each of the design phases.

These tasks are performed by the software subsystems that comprise KAAD: the Parser, the knowledge base, and the data base of the procedures for interfacing with the conventional software.

These subsystems are explained briefly below.

2. THE LANGUAGE: LISP

LISP, especially its most common dialect COMMON LISP, has been chosen for the system's knowledge base, using Frame Formalism.
There are two main reasons for choosing this language. First, the Frame-based system can be simply and efficiently handled; second, it can be used to create the Procedural Attachment.
A small number of elementary instructions in this language can be used to implement a Frame-based system since the formal structure of Frames and of LISP are virtually identical.
A Frame, like a LISP program, can be formally defined as a generalized list, each element of which can be recursively a list of elements.
The Procedural Attachment is possible because the data and calculation programs have the same presentation: this makes it possible to model even complex relations between building objects using calculation procedures linked to the Frames that represent them.
These procedures are performed automatically, operating on the knowledge base, when these aspects related to the building objects in question are involved.

3. THE PARSER

The Parser is the software subsystem which interprets the commands given to the system by the operator. These systems have been developed to high reliability and complexity levels, and are used as the interface for many commercial database management systems.
The KAAD system's Parser will be created using a very simple formalism called ATT (Argumented Transition Trees). An ATT is a tree composed of nodes linked by labelled arches. The command is interpreted word by word to select the appropriate branch, following a path through the tree providing an interpretation of the command.

4. THE KNOWLEDGE BASE

When searching for the solution to a given design problem, the representation used plays a fundamental role (2),(19),(25),(26).
To build the knowledge base of the KAAD system, particular care was therefore taken to define the structure of the representation of the building objects, and the choice of the formalism used.

4.1. The representation of building objects

The Design Process needs to continually cheek the consistency of the design choices against a given set of goals to be attained: this check is generally performed by matching abstract representations of design goals against those of the real building objects (R.B.O.) being sought, resulting from complex intellectual actions closely related to the designer's own culture and environment.

This paper defines a possible formalization of such representations for the goals and the R.B.O. that are usually considered in the architectural design process by our culture in our environment.
The representation of the design goals is performed by expressing their objective aspects (requirements) and by defining their allowable values (performance specifications). The resulting system of requirements defines the set of allowable building objects (B.O.) consisting of the set of characteristics (attributes and relations) which are considered relevant to represent the particular kind of R.B.O. with respect to the consistency check against the design goals. The values related to these characteristics define the performances of the R.B.O., while their set establishes its behaviour.
Generally speaking, there is no single object corresponding to an abstract representation but the whole class of the R.B.O. that are equivalent with respect to the values assumed by the considered characteristics. The more we increase their number and their specifications, the smaller becomes the membership of the class, until it coincides with one single real object - given that the assessed specifications are fully consistent.
Moreover, the corresponding representation evolves up to the total prefiguration of the R.B.O.
It is not possible, therefore, to completely define a B.O. deductively in advance, since it is inferred from the considered goals and is itself a result of the design process. All that can be established in advance in that set of characteristics assumed to represent any R.B.O. are hierarchical, topological, geometrical and functional relations between the parts of the object - at any level of aggregation, components, space units, building units, the whole building - which we define as Representation Structures.
In other words, the Representation Structures, by superposition and interaction, form the abstract representation that matches the design goals.

a. Hierarchical structure
Building objects, as complex systems, can be represented by means of their parts, which, in turn, are conceivable as sets of lower level parts, and so on, to the desired specification level.
That means defining on the B.O. set an order based on the transitive, reflecting an asymmetrical relation 'To-be-part-of' ( ).
An algebraic structure called a semilattice corresponds to the set B ( ) ordered by relation.
The hierarchical structure is the most general of the representation structures and defines the set of the allowable organization of the B.O. parts, whose reciprocal dispositions are analyzed and defined by the topological structure.

b. Topological structure

The B.O. expressed in the hierarchic structure, viewed as subsets of a space define on it the base of a topology T. In fact, for any point X <- W there is an element B T such as x B and given two elements B_1 and B_2 T, $B_1 \cap B_2 \neq \emptyset$ that is an element of T.

The set of existing or requested relations on T defines the topological structure of the investigated B.O.. Such relations describe the reciprocal disposition of the considered spaces and physical elements, the access and communication conditions among space units and the specific relation that is considered.

A topological structure can be represented by means of generalized graphs $G(B_i)$ so defined: $G(B_i) = (B_i, T_i, R_i)$ where B_i, T_i and R_i are respectively the B.O., its subset family, and the specific relation that is considered.

c. Geometric structure

While the hierarchical and topological structures represent the constitution of B.O. and the reciprocal disposition of their parts, the geometric structure represents the shape and the absolute positions of the physical elements of B.O. together with the form of the space system.

To this end, there are many geometrical representation systems almost all belonging to the following two main classes: CSG (Constructive Solid Geometry) and B-Rep (Boundary Representation). In CSG representation, a complex volume results from Boolean operations on elementary volumes: in the B-Rep representation, volumes are defined by their bounding surface. But in both, volumes are defined in a space that is not an element of the representation.

On account of the relevance that void space has in architectural design as a result of the definitions of solid volumes, these schemata need to be overcome: the definition space of volumes is entirely filled by void and solid shapes; every shape surface will thus separate two different kinds of volumes, the void and the solid ones, independently of the kind of representation adopted.

d. Functional structure

The characteristics of a functional element, beyond the kind of B.O. that it concurs to define, its interactions with other B.O. and its particular shape, result from the tasks it has in defining and classifying the building space, to guarantee safety and stability and establishing required environmental comfort levels.

Such characteristics will be represented on the one hand by identifying every F.E. as an element of a functional class that can be specified at will, according to the requirements of the representation (e.g. from enclosure to: external envelope, internal division; from external envelope to vertical walls and floors; from vertical walls to walls and windows, and so on).

The F.E. will inherit all the functions of the class it belongs to, besides its own functions.
Moreover, the aforementioned characteristics will be represented by means of calculation schemes that simulate the behaviour of the F.E. and of the system it belongs to.

4.2 The Frame Formalism

The Frame Formalism proposed in 1975 by Marvin Minsky broke the dichotomy between two clashing representation philosophies in Knowledge Engineering: the declaratory and the procedural (2).
The procedural approach tends to represent an individual's knowledge by codifying it in terms of the procedures used to perform tasks. In other words, knowledge is viewed essentially as 'know-how'.
The declaratory approach represents knowledge by codifying it in terms of two distinct components: a set of facts relating to a domain of knowledge and a system of general inferential procedures which can operate and draw conclusions using these facts.
It is impossible to say which is the best technique, because both approaches have their merits and drawbacks. This is why we have chosen the Frame Formalism for the knowledge base of the KAAD system, because it has many of the merits of both approaches.

The main advantages of the declaratory representation approach are:
- Flexibility, economy.
The facts are codified without reference to the way in which they may be used in inferential procedures.
This means that each one can be used in any deductive process, making the resultant representation extremely flexible and small.
- Transparency of representation
Since the internal codification of knowledge uses first-order predicate calculus propositions, it is directly comprehensible.
This means that the knowledge stored can easily be increased or altered.

The main advantages of the procedural representation approach are:
-Modelling
Many things we know can be represented naturally in terms of the procedures needed to obtain a result instead of being in the declaratory form.
Since the procedural representation is based on calculus procedures, this kind of knowledge can be codified easily.
- Second-order knowledge
One particularly important aspect of knowledge is knowing about what is known, or meta-knowledge.

In any given situation, knowing the most relevant facts and the strategies to adopt can help reach a solution that would otherwise be difficult.
The procedural approach provides an easy way of representing this type of knowledge by explicit links between groups of information, and of ranking the information hierarchically.
The Frame Formalism combines many of the positive aspects of the two philosophies mentioned above, and even though there is not yet any commonly agreed theoretical definition of it, applications made so far have proven extremely effective.
A frame may be defined as a structured set of features called SLOTS, which define an abstract representation of a real object. It may be organized at any number of hierarchical levels, so that if one feature requires further specification it can be expanded by using a further frame.
Every slot is associated with a set of values preceded by a FACET which defines both the type and the domain of variation.
Each slot can be associated with expected values preceded by the facet, known as DEFAULTS which define a set of expectations about the object represented.
A frame may be used as a prototype of a class of objects, and when a set of expected values is replaced by real values, an INSTANCE is created, which inherits all the general features of the class to which it belongs.
Each slot can be associated with calculus procedures called DEMONS, using appropriate FACETS. Through these, procedural knowledge is codified, so that the system can determine the value of the slot to which they are attached.
This possibility, known as procedural attachment, makes it possible to create complex relations between objects in the knowledge base and to simplify the processes performed on it, suggesting strategies to be used as a result of the status of the representation. In formal terms, a frame may be defined as a generalized list, which is a list in which every element can also be a list.

4.3 The structure of the knowledge base

The KAAD system's knowledge base has been created by implementing the representation structures through a set of prototypes, slots and calculating procedures. The system of prototypes is a basic component of a frame-based knowledge representation system because it guides the inferential process of the system through the slot filling process, which we shall examine shortly.
The fundamental prototypes that will be used stem from the previous definition of the representation structure.
Below is a brief description of the most important prototypes used in the KAAD system.

a. The building unit prototype

This represents the highest level of aggregation of the building objects. A B.U. is a structured set of space units. The form which represents the B.U. prototype comprises a single slot: IMP (Immediate Predecessor) and the indication of the type of values which it can have. As a result of the definition of the hierarchical structures, these values are yet more B.U. or Space Units. Using the B.U. prototype, it is possible to insert more instances of specific B.U., filling in the frames of the further slots which are required. b. The space unit prototype

A S.U. is a class of equivalent environments used for the same group of activities, requiring the same environmental characteristics and the same equipment.

The S.U. prototype comprises a set of slots which represent the fundamental features, such as the elements which comprise it, the elements to whose definition it contributes, the topological relations between it and the other S.U., and its geometric forms.

The main slots in a S.U. are:

- the IMP: as with the B.U., the IMP contains the type of hierarchically subordinate building objects which contribute to the definition of a S.U., namely the functional elements;
- the IMS (Immediate Successor): this contains the building objects on the immediately higher hierarchical level, to whose definition the S.U. may contribute, namely, the B.U..
- ADJ (Adjacent): this establishes an adjacent relationship between a S.U. and those contained in the slots. Its content may have consequences on one or more S.U.
- COM (communication): this establishes the existence of a path between S.U.
- SHAPE: this defines the geometric form of the S.U., and its content may only be an object of the SUSHA type. SUSHA (Space Unit Shape) is another prototype of the system which establishes the way in which the S.U. is represented in terms of the type of geometric model used.

c. The functional element prototype

A functional element (F.E.) is a class of equivalences of complex physical elements, used for the same functions to delimit space, to assure an appropriate level of safety and environment comfort. The prototype that the F.E. represents is structurally similar to the one which represents an S.U. The IMS and IMP slots contain the names of the S.U. which the F.E. helps to define, and the F.E. and/or constructive elements which comprise it, respectively. The ADJ slot indicates the names of the F.E. to which it is deemed to be adjacent. The SHAPE slot, as with the S.U., will contain a FESHA object.

d. The constructive element prototype

A constructive element is a class of equivalences of physical elements with a specific value in the definition of the F.E. and their functional operations.

The C.E. prototype is wholly similar to the functional element prototype. The IMP and IMS slots can only contain constructive elements and functional elements, respectively. The ADJ slot will contain elements of a more complex type, also prototypes, which describe the variations of their mutual positions in the space of the constructive element in terms of the functional elements that they contribute to define. The SHAPE slot contains a FESHA type object.

e. Constraints

In the KAAD system's knowledge base, two types of constraints can be defined: natural constraints (N.C.) and design constraints (D.C.).

The natural constraints determine the limits on the variation of certain features of the building objects in order to guarantee the internal consistency of the knowledge base.

The second type of constraints stem from the need to comply with the demands of the designer. The N.C. are represented in the knowledge base by defining appropriate facets for the characteristics to which they refer. In this way, they are propagated to the instances as they are created and hence there is an ongoing consistency check on the input data.

The D.C. are represented by appropriate frames whose prototype REQ (requirement) is the slot domain which defines the object(s) which constitute(s) the domain of application of the requirement. The requirement's instances are completed one after the other with the specification of the slot(s) to be constrained and of the accepted variation limits.

The variation domains of the constrained features, whatever the type of constraint involved, are defined using the facets listed below:

Range: this contains two values which define a continuous variation interval;

Set: this identifies a set of discrete values which the slot may take on;

Classes: this comprises pairs of elements which identify intervals of variation, in terms of which the value attributed to one feature can be classified;

Max, Min: these identify acceptance thresholds;

Value: this is the facet which contains the value supplied by the instance in respect of a feature, and which must be matched against those defined by the preceding facets.

5. THE PROCEDURAL DATA BASE

One of the tasks of the KAAD system is to create a high level interface between the designer and the conventional software systems. This has made it necessary to define another software component of the system; the procedural data base (P.D.B.). The P.D.B. is a set of calculating procedures, whose task is to receive data from the knowledge base to be used to execute the programs as required. The number and type of these procedures depend in the type of program with which the KAAD system has to interface, and are therefore determined on an experimental basis.

For the Drafting systems, procedures will have to be designed to move from two-dimensional to three-dimensional representations, and for each type of representation it will be necessary to define the way in which the logic elements and the operations allowed by these types of programs can be used to represent the building objects under consideration.

Furthermore, an analysis will be made of the operating procedures used by the designer to represent information on technological, economic and production aspects using the logic available in the present DBMS systems (hierarchical, networking, relational), and indications will be provided on the way in which these may be related to the graphic representations of the building objects.

Lastly, for the calculation and checking procedures, we shall define the operational sequence of information required derived from graphic representations or technological descriptions.

The procedures comprising the output of the experimental period will subsequently be translated into calculation programs and used to provide the System with the required procedural knowledge to enable it to perform the operations described above.

6. THE OPERATION OF THE KAAD SYSTEM

The KAAD system operates on the basis of a general deductive process which can exploit the hierarchical relationship established by the frame formalism between the instances and the knowledge base prototypes.

This process, called inheritance, has been created by means of a calculation procedure which uses two slots called ISA and AKO (is a) and (a kind of). The ISA slot establishes a hierarchical relation between a prototype and the instances which depend on it. The AKO slot relates the instance in question to the class to which the prototype it is linked to belongs.

Whenever a new instance is defined in the knowledge base, the system uses the inherit procedure to trace back the hierarchy of information defined by ISA and AKO, thereby removing the structure of the instance to be created from the prototypes it finds.

The system then tries to get the operator to define the values for the slots in question, in default of which it triggers off another inherit procedure to search for the default values or procedures able to calculate the desired values. Since all the information is memorized in the knowledge base by appropriate prototypes, the inheritance procedure is used in the system to manage interaction with the user, to interface with the conventional software tools, to check the consistency of the data input, and match them against the constraints defined by the system.

---

1. A. Baer, C. Estman and M. Henrion, Geometric modeling: a Survey, Computer Aided Design, II, pp. 253-271 (5 September 1979)
2. Representation and Understanding, edited by D.G. Bobrow, A. Collins, Academic Press (New York 1975).
3. D.G. Bibrow et al., G.U.S. A Frame-driven Dialog System, Artificial Intelligence, B(2) pp. 155-173 (1977).
4. G. Carrara, P. Cocomello. C. Gori Giorgi, A. Paoluzzi, A systems approach in building component design, Proc. of IFA Conference, Pergamon Press (New York, 1978).
5. G. Carrara, A. Paoluzzi, System approach to building program planning, Istituto di Architettura, Edilizia e Tecnica Urbanistica (1980)
6. G. Carrara, The building object and its representation. Forthcoming.
7. G. Carrara, G. Novembri, Constraint-bounded Design Search. Proceedings of CAAD Futures International Conference on Computer-Aided Architectural Design, Delft, 1985.
8. V. Checcucci, A. Tonioli, E. Vesentini, Lezioni di topologia generale, January 1977, Feltrinelli (1977).
9. C. Girard, Presque Half Planes: towards a general representation scheme, Computer Aided Design, 16, no.1 (January 1984).
10. D.R. Hofstadter, Godel, Escher, Bach: An Eternal Golden Braid, Basic Books (New York 1979).
11. H.J. Lavesque, Foundation of a Function Approach to Knowledge Representation, Artificial Intelligence 23 (1984) pp. 155-212.
12. J.C. Latombe, Artificial Intelligence in Computer Aided Design, CAD Systems, edited by JJ Allen, North Holland (Amsterdam 1977).
13. P.T. Lee, F.P. Preparata, Computational Geometry. A Survey, IEEE Transactions on Computers, 33, 12 (December 1984)
14. M. Mantyla, A note on the modelling space of Euler operators, Computer Graphics and Image Processing.
15. M. Mantyla and R. Sulonen, GWB a solid modeler with Eucler operator, IEEE Comp. Graph. Appl. 2 (7), 1982, pp. 17-31.
16. D. McDermott, Planning routes through uncertain territory, Artificial Intelligence 22 (1984), pp. 107-156.

17. Machine Learning: an Artificial Intelligence Approach, edited by R.S. Michalski, J.G. Caubonell, T.M. Mitchell, Springer Verlag (1984).
18. J.D. Monk, Introduction to Set Theory. McGraw-Hill 1969.
19. N.J. Nillson, Problem-solving Methods in Artificial Intelligence, McGraw-Hill, (New York 1971).
20. N.J. Nillson, Principles of Artificial Intelligence, Springer Verlag (1982).
21. F. Preparata, T. Yeh, Introduction to Discrete Structures for Computer Science and Engineering, Addison Wensley (1963).
22. A.A.G. Requicha, H.B. Voelcker, Solid Modeling. a Historical Summary and Contemporary Assessment, IEEE Comp. Graph. Appl. 2 no.2, (March 1982) pp. 9-24.
23. A.A.G. Requicha, H.B. Voelcker, Representation for Rigid Solids: Theory, Methods and Systems, Computing Surveys, 12 no.4, (December 1980), pp. 437-464.
24. A.A.G. Requicha, H.B. Voelcker, Boolean Operations in Solid Modeling Boundary Evaluation and Merging Algorithms, Proceedings of the IEEE, 75 no.1, (January 1985), pp. 30-34
25. R. Schenk, R. Abelson, Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Associates, Publishers (Hillsdale, New Jersey, 1977)
26. H.A. Simon, The Sciences of the Artificial, MIT Press 1969.
27. M.I. Stefik, Planning with Constraint, Stanford University (1980).
28. P.H. Winston, Artificial Intelligence, Addison Wesley (1984)
29. P.H. Winston and B.K.P. Horn, LISP, Addison Wesley (1984).