Knowledge-based Stair Design

Osman Ataman
State University of New York at Buffalo

The application of computer-based techniques to support architectural design has often concentrated on matters of representation. Typically, this means computer-aided drafting, and less frequently, computer-aided modeling and visualization. The promise of new computer-based tools to support the process of design has thus far failed to produce any significant tool that has had a widespread impact on the architectural profession. Most developments remain in university-based research labs where they are used as teaching instruments in CAD courses or less often in design studios. While there are many reasons for this lack of dissemination, including a reluctance on the part of the architectural profession itself, the primary obstacles deal with difficulties in explicating design knowledge, representing this knowledge in a manner that can be used for design, and providing an intuitive and effective user interface, allowing the designer to easily use the tool for its intended purpose.

This study describes a system that has been developed to address a number of these issues. Based on research findings from the field of Artificial Intelligence which expounds on the need for multiple techniques to represent any complex area of knowledge, we have selected a particular approach that focuses on multiple techniques for design representation. We review this approach in depth by considering its many facets necessary when implementing a knowledge-based system. We then partially test the viability of this approach through a small case study, implementing a knowledge-based system for designing stairs. While this effort only deals with a small part of the total design process, it does explore a number of significant issues facing the development of computer-based design assistants, and suggests several techniques for addressing these concerns.

Background

A number of techniques and systems have been studied as the appropriate means for representing design knowledge. They include: constraints (Carrara and Novembri 1985), (Gross, et al 1987), shape-grammars (Stiny 1980); prototypes (Oxman and Gero 1988), (Oxman 1990); expert systems (Gero 1985), (Oxman and Gero 1987); components (Harfmann and Majkowski 1992), (Harfmann and Chen 1989); and multi-modal techniques (Carrara, et al. 1992). Research in Artificial Intelligence has found that domains requiring complex knowledge often require multiple techniques to adequately capture the many facets of this knowledge (Reichgelt 1991). In this work, we have selected an approach discussed by Carrara, Kalay and Novembri that recognizes the multi-modal nature of design knowledge (Carrara, et al. 1992). In their approach, design knowledge is composed of three distinct modalities:

- Descriptive knowledge representing objects and concepts of a domain, the relationships between these parts, and how they function.
- Normative knowledge representing the intent of the design process and the constraints that shape the emerging design solution.
- Operational knowledge which describes the processes by which goals are attained through the selection and manipulation of objects.

They propose that descriptive knowledge can best be represented using semantic networks and frames; normative knowledge using rules and constraints; and operational knowledge using methods or procedures. Although recommending a number of significant options and leaving room for other representational techniques as well, it is not clear how all the components come together in a working system. We therefore consider this approach from...
an implementation perspective, to begin to validate
and verify the ability of the approach to represent
real design concerns and processes.

Although this study is based on the approach of
Carrara, Kalay and Novemhi (Carrara, et al 1992),
there are several differences:

- The classification of design knowledge
- The representation of goals

In our work, knowledge is classified according to its
structure and specificity, exploiting the inherent hi-
erarchical structure common in many areas of design
knowledge. Our current implementation further con-
centrates on the descriptive knowledge required,
considering normative and operational knowledge
in an implicit manner. Goals and constraints, viewed
as control mechanisms, are not represented as a dis-
tinct entity, but rather as a part of the general know-
ledge of the domain. While this is workable for the
domain of stair design which is relatively con-
strained, larger design concerns may require explicit
representation. Further work will continue the im-
plementation of the other modalities of this approach.

Design Knowledge

A knowledge base is composed of knowledge in a
particular domain, including such things as facts,
methods (procedures), heuristics, and strategies for
solving problems in the domain. Architectural
knowledge is of many different types. This study
classifies architectural knowledge into three types:

1. Knowledge of building structures, including
   materials, construction methods, facilities, and how
   buildings interact with the environment.

2. Knowledge of building functions that should com-
   ply with the requirements of our social and daily life.

3. Knowledge of architectural symbols as the repre-
   sentation of ideology, including historical meanings.

For example, the knowledge that a beam should be
attached to a column is the first kind of knowledge;
the knowledge that a kitchen and a dining room
should be located closely together belongs to the
second kind; and the knowledge that a flat roof
building’s facade contains a cornice is the third kind
of knowledge.

Following the work of Robert and Rivka Oxman
(Oxman and Oxman 1990), each of these knowledge
areas can be further categorized according to the level
of specificity:

1. Domain-independent or general knowledge
   which can be applied to most buildings.

2. Domain-specific knowledge of architectural
design pertaining to certain building types or space
functions, or related to a particular era or architect.

The Oxmans define general knowledge as
paradigmatic knowledge which is “a form of
higher-level knowledge which derives from a class
of architectural problems and is relevant to all
members of the class regardless of the particular
building type” (Oxman and Oxman 1990). General
knowledge is equally applicable to many different
circumstances or individual instances (Akin 1986).
Akin further suggests that both general knowledge
about things, as well as knowledge how to adapt
them to specific conditions, are part of the designer’s
general knowledge. The knowledge structure of the
proposed model is based on the idea that
representation of specific knowledge is dependent
upon the representation of the general knowledge,
and that the derivation of this knowledge is a
fundamental problem in developing a knowledge
based system for architectural design. In this view,
for example, specific knowledge of wood structure
is based on the general knowledge of building struc-
tures and the knowledge of style is based on the gen-
eral knowledge of architectural symbols (Figure 1).

<table>
<thead>
<tr>
<th>KNOWLEDGE OF STRUCTURE</th>
<th>KNOWLEDGE OF FUNCTION</th>
<th>KNOWLEDGE OF ARCHITECTURAL SYMBOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL ARCHITECTURAL KNOWLEDGE</td>
<td>BEAM AND COLUMN BUILDING SYSTEM</td>
<td>COLORING ARCHITECTURAL MATERIAL</td>
</tr>
<tr>
<td>SPECIFIC ARCHITECTURAL KNOWLEDGE</td>
<td>STRUCTURE OF WOODEN STAIRS</td>
<td>PARTICULAR STYLE</td>
</tr>
<tr>
<td>KNOWLEDGE OF EDUCATIONAL FACILITIES HOSPITAL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1
Representing design knowledge

Following the approach suggested by Carrara, et al., we use represent combination of descriptive and procedural knowledge, using three techniques: rules, semantic networks and frames.

Rules

Production systems, a technique employing “if condition-then conclusion” rules, have proved useful in the implementation of expert systems and have interested cognitive scientists as explanations of human behavior (Young 1979), (Anderson 1983). Production systems are composed of a set of rules (called productions) where individual rules may “fire” when matches occur with features of the project environment. The system allows the addition, deletion and modification of rules by the system itself and by the user, and provides an easy to understand and modular means of representing knowledge. However, as this technique focuses primarily on the relationship between objects, it has not been universally popular. Problems occur when the technique is applied to large rule bases due to rule conflicts or interactions, the lack of a clear structure of knowledge, and the inability to represent certain types of knowledge (such as negative attributes). Numerous techniques have been developed to overcome these problems, often reducing the modularity and intuitiveness of the basic approach.

Semantic Networks

In the semantic network approach, knowledge is represented in a distributed manner as a collection of nodes and links. Nodes represent concepts or objects, while the links indicate binary relations between the nodes. The links describe particular relations between nodes, including class membership or instantiation (INST links), hierarchical structuring of objects through sub-class relationships (SUB links), part-whole relationships (PART-OF links) and properties (HAS links). Semantic networks lend themselves to graphic presentations, providing a means that is descriptive and comprehensive. The network is easily modified, allowing the addition, subtraction and modification of knowledge as required. This method also provides a convenient way of dealing with hierarchies, as the links provide a natural connection between classes and subclasses. However, there are a number of problems. There are no specific semantics associated with semantic networks. Rather, the meaning of a semantic network depends on the reasoning procedures associated with it. The technique works well for small problems but quickly becomes unmanageable for realistic models, where search and inference techniques become unworkable.

Frames

Frames, first described by Minsky (Minsky 1975), provide a means of representing knowledge about objects or concepts, recognizing that knowledge often occurs in chunks larger than rules. In this technique, knowledge is represented as a collection of slot-facet pairs. The slot defines the property of the object in question, while the facet or filler represents the value associated with the slot. It is important to note that the property is fixed for the object, but the value is not. Frames also provide a means of specifying default values for fillers, ranges of acceptable values, and attaching procedures (demons) for calculating the value of a specific filler, based on values in other fillers. Using this technique, the geometric description is a filler for the slot shape. Frames can also be used to represent non-geometric information, such as material, manufacture, cost, etc. Frames are a convenient means of providing an ordered and succinct representation for large amounts of information, and provide techniques for incorporating knowledge manifested in rule based and semantic network approaches. Frames, however, are somewhat limited in representing certain types of knowledge, particularly certain forms of incomplete knowledge or disjunctions between slots.

The frame system used in this model has some characteristic slots. They include: PART-OF, INST (instance) and SUB (subclass).

PART-OF

The slot PART-OF indicates hierarchical part-whole relations among frames. This is a prototype frame for a class and contains the list of slots applicable to the class along with the possible default values for these slots. For example, the material frame is available under the frame of “stair”, but we have to descend to more detailed sub-frames when we design the wood-stair frame itself. In this case, the system can move through the slot of PART-OF into the sub-frame wood. (Figure 2)
way, we deduce facts through certain links. For instance, we deduce that darkened-oak is a material because we know darkened-oak is an oak, oak is a wood and wood is a subclass of material (figure 4).

The frame formalism supports reasoning with incomplete data by using default values, restricting the range of values when a specific value is not known, and providing attached procedures to acquire or calculate data that can not be specified a priori.

**Figure 2**

**INST/Instance**

Instantiation represents a specific occurrence of a class, with specific attributes and location. Inheritance occurs at the class level and allows attributes to be shared but are not stored in the instance or subclass. For example, when the system encounters the frame "darkened-oak" in a wood-frame situation, and if there is a frame of "oak" in the existing knowledge base, the system assumes that the "darkened-oak" frame inherits attribute values of the "oak" frame (Figure 3).

**Figure 3**

**SUB/subclass**

When the frame does not match a given problem, the system has to search for an adaptable frame. This search can begin with the SUB slot, allowing the inheritance of values from general classes to more specific subclasses. For example, if we want to know darkened-oak color, we match against the frame darkened-oak and retrieve the value. If we want to know fire_prop property of darkened-oak, we check darkened-oak frame first and then, since there is no fire_prop slot, we follow the INST, SUB slots to the wood frame, and then to the material frame. In this way, we deduce facts through certain links. For instance, we deduce that darkened-oak is a material because we know darkened-oak is an oak, oak is a wood and wood is a subclass of material (figure 4).

**Figure 4**

**Implementation of the Model: Stair Design**

Humans use vertical circulation elements in order to access points or spaces at different levels. These elements, which include ramps, stairs, escalators, and
elevators, possess certain functional requirements such as safety, comfort, and aesthetics. Since physical environments are different, it is often unclear how these vertical circulation elements can satisfy these characteristics, and how to determine the type, size, and physical utility of these elements. Even among conflicting functional requirements, safety is the most important constraint to be considered. The necessity and importance of efficient vertical circulation often strongly impacts design intent, making it a major design component in most architectural endeavors.

Although the significance of vertical circulation in architecture may be clear, the selection of stair design for the study of knowledge-based systems may not. Stair design was considered because the structure of individual element composition is clearly arranged in such a way that the size and position of each element can be determined using established procedures and formulas. For example, when the length and/or width of a single tread is determined, the stair structure dimensions are defined and many other proportions of members can be calculated. If no parameters are specified then the predefined values (or attributes) are used as default. For example, if only floor-floor height is defined, then according to the information collected so far (location, usage of the stair, etc.), a default stair design proposal is produced. Furthermore, there are distinctive part-whole relations including type, construction method and finishing which are part of the knowledge organization of stairs. Stair design is manageable and suitable for examining the general knowledge of building structures, because there is established declarative and procedural knowledge available to support its implementation.

**Declarative knowledge**

There are three kinds of frames involved in the knowledge representation of the stair elements. The first frame represents the structure of elements, including sub-frames as its parts. These are prototype frames that have been hierarchically structured as sub-frames, and which may "point" to other sub-frames. Examples include: "Structure", "Type", and "Material" sub-frames. The predicate representing this part-whole structure of frames is PART-OF. The second frame is the instantiation frame of a prototype class frame. For example, the frame of single_run and turn_stair_type is abstracted into the frame of general_stair_type frame through SUB relations. The bottom of this hierarchical structure consists of the frames of specific stair elements. These frames acquire attributes from class frames through the INST slot. These relations are diagrammed in Figure 5.

**Procedural knowledge**

Procedures are often used to represent process knowledge—knowledge about how to perform some operation. In the area of stair design, procedures are used to determine the size and position of various...

---

**Figure 5**

---

**ACADIA 1993**
elements. Standard measurements and intervals are
often used, or the size of a stair element may be
expressed in terms of the distance between elements
or as a multiple of another elements’ size. The
positions of each element may be determined relative
to the positions or standard measurement of other
elements. For instance, the height of the stair can be
defined as n times riser dimension or riser dimension
times the number of risers. In this study, four kinds
of procedures are used to determine the size and
position of stair elements:

1. Deriving from a standard element value

This process derives the size or position of one
element based on the size or position of other
elements. For example:

\[ H = T \times N \]

where

- \( H \) represents the height of the stair
- \( T \) represents the height of the riser
- \( N \) represents the number of stair risers = treads+1

2. Deriving from a standard interval

This process derives the value of one element from
the distance between two other elements. In some
cases, the distance between the elements is defined
by the measurement from center to center, while in
some cases it is defined by the inside measurement
from surface to surface. In this study, only inside
measurement between two elements is used, because
the definition of inside measurement is easy and is
frequently used. For example, the distance between
handrails from center to center is defined as the inside
measurement plus one handrail size.

3. Deriving from multiple operations

This process uses the measurements determined by
the above mentioned processes and multiplies them
by some factor, e.g., \( N \) times the handrail size.

4. Requesting user input

These procedure requires the definition of input of
values based on input from the user. The mechanism
supplies default values which the user may accept if
they are unsure of the exact input value.

The Reasoning Process

The structure of the system has two distinct parts:
the knowledge-base and the inference mechanism.
These react via a user interface with the end user as
shown in Figure 6.

```
Figure 6

The designer is not required to start with a specific
constraint level such as the function of the stair or
budget. Instead the user can start in any level with
any constraint. Since the range of solutions reduces,
the achievement of more specific solutions is
dependent on the knowledge-base and the inference
mechanism has a very important role
supporting the concept of this model. The inference
involved in determining the construction of elements
begins with the choice of the part to be designed and
its sub-frames as input. Then we inquire of the system
about the size or position of any sub-frame or
element. The reasoning process starts by activating
the procedures to determine the size or position of
some frame. This procedure requires the value of the
size or position of another part, and these are
referred one after another until finally the
procedure reaches the defined part and obtains a
fixed value. Then the procedure derives the size or
position of the original frame from this fixed value.

The advantage of this inference system is that the
system requires a minimum number of inputs to
determine a particular value and can begin from any
frame or sub-frame. Several example predicates,
written in Prolog, for the design of stairs are shown
below:

relative_size (Object, RelativeSize) :-
    value (Object, Size, ObjectSize),
    value (Object, is_a, ObjectClass),
    value (ObjectClass, size, ClassSize),
    RelativeSize is ObjectSize * ClassSize.

ACADIA 1993
In order to prevent the arguments Object and RelativeSize from getting lost in communication between frames, we also stated them as part of relative_size slot information. The contents of this slot are combined into a single term:

\[ \text{trigger(relative\_size)} (\text{Object}, \text{RelativeSize}) \]

The relative_size slot of frame stair is then specified by:

\[ \text{stair(relative\_size, trigger(relative\_size)} (\text{Obj, Value1, Obj, Value1)}) \]

Through its inference process, the system inquires some values which are not available within the established facts and procedures in the knowledge base. Of the many subtleties of inference among frames, we have not fully addressed the question of multiple inheritance. This problem arises when a frame has more than one parent frame, according to the relation PART-OF, SUB or INST. In this case, an inherited slot value may potentially come from more than one parent frame and the question of which one to adopt arises. In the current implementation, the first value encountered is selected, found by a depth-first search among the frames that potentially can supply the value. However, other strategies or tie-breaking rules may be more appropriate.

A sample run of the interactive process is shown in Figure 7. The program is a straightforward implementation of basic ideas embodied in the multimodal approach, with little regard for efficiency at this time. The main structure of the program is
written in Quintus Prolog ProXT is used as an interface tool for the graphic user interface component written in C. The "VOGLE" C library is used to support the graphic output.

The program described here was implemented on Sun workstations. User interface windows interact with the user by asking questions and evaluating the answers. The system checks consistencies for each answer and infers the stored information. The user can save the information s/he supplied and start the next program with the stored information for a re-run to improve the design.

Conclusion

A conceptual framework for an architectural knowledge-based system and techniques for representing architectural knowledge have been explored. The functionality of this knowledge was demonstrated by using stair design as a case study. The methodology presented here integrates different techniques for representing the descriptive components of design knowledge, effectively integrating both declarative and procedural knowledge.

The implementation seems to verify the multi-modal approach as configured at the outset of this work. However, as the domain of stair design is relatively restricted and the implementation has not fully addressed the normative and operational modalities of design, further verification is indicated.

The implementation as completed does question the use of semantic networks and frames for representing descriptive knowledge. Based on the implementation, it seems apparent that the frames paradigm offers sufficient flexibility to accommodate the representational needs of semantic networks, without requiring a separate model. It is also felt that the frame paradigm, which can incorporate facets of many other representational techniques, also may offer opportunities for representing the normative and operational modalities.

The real merit of this study lies not in what it actually achieves, but rather in its "pointing out" various issues that need to be addressed in the design and development of a knowledge-based stair design system. However, even with that restricted domain, many design issues are not addressed, such as "non-standard" layouts, landing shapes, and handrail treatment at turns. The author realizes that the choice of a representation is dependent on the nature of the knowledge being represented. Future work will include exploring and implementing the normative and operational modalities of design knowledge, and the possible implementation of the system in a more complex and encompassing design task.

Acknowledgements

The author wishes to acknowledge the guidance and feedback by Stuart Chen, Yehuda Kalay, Bruce Majkowski, Edward Steinfield, and Ian Taberner during the development of this work.
References


ACADIA 1993