A Computerized Fire Safety Evaluation System for Business Occupancies

Filiz Ozel

Architectural Studies Program
University of Nevada, Las Vegas
Las Vegas, NV 89154 USA

The development of computer-based code compliance checking programs has been the focus of many studies (Ozel, 1984, 1992; Delis & Delis, 1991; Coyne et al., 1990; New York State, 1990). While some of these investigated the procedural aspects of building codes, others focused more on their rule base. On the other hand, due to the complexity of the codes, the process of identifying which sections apply to a given problem, and in which order to access them requires a meta-knowledge structuring system. National Fire Protection Association (NFPA) 101M, Alternative Approaches to Life Safety *(1992) provides a framework through which code sections can be systematically accessed by means of a set of checklists.*

The study presented here primarily focuses on the development of a computer based fire safety code checking system called ARCHCode/Business for business occupancies following the guidelines and the methodology described in Chapter 7 of NFPA 101M.

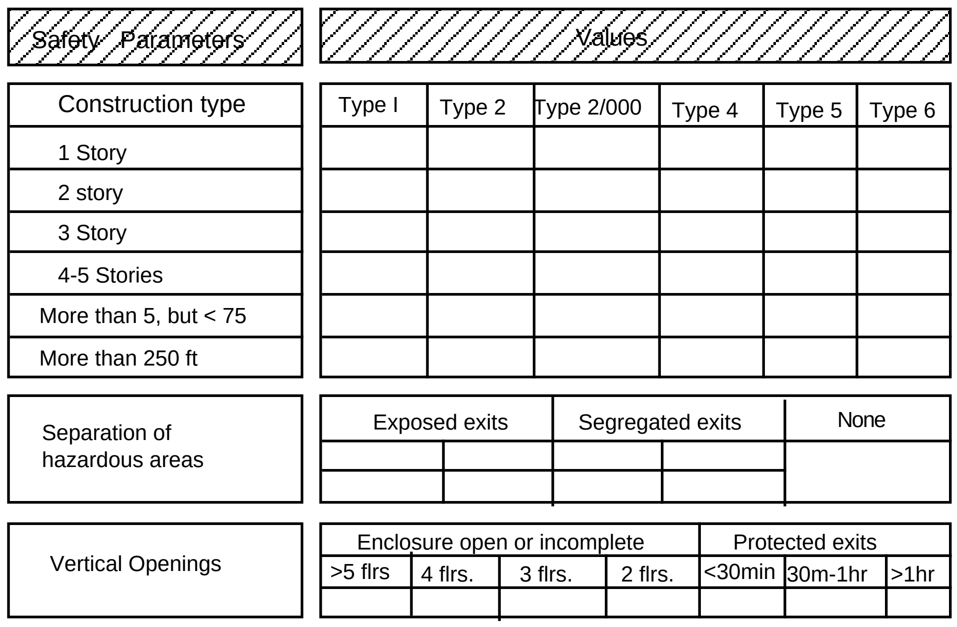
Keywords: fire safety expert system, business occupancies, CAD interface.

1 Overview

Among the difficulties architects encounter in finalizing a design project is the process of identifying the sections of a building code that are applicable to a particular design problem. Code compliance expert systems can potentially assist designers during the design process by allowing quick code checking, especially in those cases where a large number of buildings must be checked for compliance. On the other hand, fire and life safety codes constitute a major portion of building codes, and thus have been the focus of several computer applications .

Some of these systems (Kim, 1986; Coyne et al., 1990) primarily focused on the rule base of the codes and did not incorporate an “object-graphic entity” modeler, thus geometric reasoning about objects was not possible. On the other hand, programs such as ARCH:Firesafety (Ozel, 1984) mainly addressed the geometric reasoning needs of the problem without an extensive rule base. Clearly there is a need to incorporate both features into a comprehensive code checking program. More recent studies (Ozel, 1992) focused on the data modeling needs of code compliance applications through a CAD system. Furthermore, identifying the code sections that are applicable to a given problem requires meta-knowledge and a systematic way of accessing such code sections. The present study represents an attempt to implement a meta-knowledge structuring system that allows the assessment of tradeoffs between different aspects of fire safety in buildings. Here the word “tradeoff” is used within the context of focusing on alternative ways of achieving a desired level of fire safety by manipulating the level of safety afforded by different systems in a building. For example, there will be a tradeoff between providing a sprinkling system throughout the building, thus allowing longer travel distances versus having only a partial sprinkling system and reducing travel distances by introducing additional protected egress routes.

Fire safety code compliance checking programs must not only focus on the architectural features of a building (passive precautions), but also on fire protection and prevention systems (active precautions). The system presented here addresses both areas by generating scores for different aspects of fire safety in a building, which are in turn propagated through the system to generate a single score for compliance/noncompliance. Breaking the problem into parts and generating a score of compliance for each part enables the user to backtrack deficiencies in the fire safety system of a building if noncompliance is inferred (Figure 1). Twelve categories of safety parameters are identified in NFPA 101M (1992). The full list of categories can be seen in the final evaluation sheet of ARCH-Code/Business (Figure 5).



Safety Parameters	Values					
Construction type	Type I	Type 2	Type 2/000	Type 4	Type 5	Type 6
1 Story						
2 story						
3 Story						
4-5 Stories						
More than 5, but < 75						
More than 250 ft						

Separation of hazardous areas	Exposed exits		Segregated exits		None

Vertical Openings	Enclosure open or incomplete				Protected exits		
	>5 flrs	4 flrs.	3 flrs.	2 flrs.	<30min	30m-1hr	>1hr

Figure 1. Some of the safety parameters for business occupancies. Each value box in this display is attached to an attribute of an object in Level5Obj. Thus, while the program is running, the resulting values are automatically reflected to the worksheet (adapted from NFPA 101M, Table 7-1).

Expert system principles such as representing expert knowledge through a rule base and using inferencing mechanisms for compliance checking can help to address the large volume of codes that must be accessed during code compliance checking (MacKellar and Ozel, 1991). Furthermore, geometric and spatial reasoning methods that are necessary for code checking can be associated with objects more easily and built into the rule base of such an application. CAD integration is important in providing a geometric data base for such reasoning purposes. For example, in deciding whether a given vertical opening is fully enclosed or not (see item 3 in Figure 1), the program must find all of the walls that bound the space (room) where the vertical opening is located, and assess their fire rating. Such a process clearly requires spatial data and a geometric interface.

Issues related to architectural expert systems with CAD integration can be summarized as follows (Figure 2). The CAD system must provide:

- an architectural object base;
- a fire safety object base;
- a graphic entity data base.

The expert system must provide:

- a production rule language;
- an architectural object data base;
- a fire safety object base such as fire zones, smoke areas, fire walls, alarms, etc.;
- a graphic entity data base;
- methods that allow geometric reasoning;
- rules that recognize the relationship between architectural objects and graphic objects
- system objects that serve operational purposes and do not necessarily have real world counterparts. For example, in the fire safety evaluation system summarized here “egress requirements” and “fire protection requirements” of a building are defined as objects. This allows the comparison of the attributes of a given building with the attributes of the “Code” object.

2 Methodology

NFPA 101M (1992) contains alternative methods of evaluating the life safety performance of a building, and can be used in place of NFPA 101, Life Safety Code (Lathrop, 1988). A set of worksheets are provided in NFPA 101M, where the results of each sheet are progressively propagated to other sheets, finally leading to a single value for compliance/non compliance. Each worksheet addresses a certain aspect of the fire safety problem. Breaking down the items further in each worksheet helps to comprehensively cover all aspects of the problem at hand.

Furthermore, this alternative handbook for code compliance provides definitions for a range of life and fire safety objects and then lists the rules that are to be used in conjunction with these objects, related tables and worksheets. These rules can be readily represented in an expert system and the results reflected to worksheets, while the objects defined in the Code can be modeled through the data structure of the expert system.

In NFPA 101M (1992), a single value representing minimum acceptable score is specified for each of the following areas for both new and existing buildings:

- Fire control requirement;
- Egress requirement;
- General fire safety.

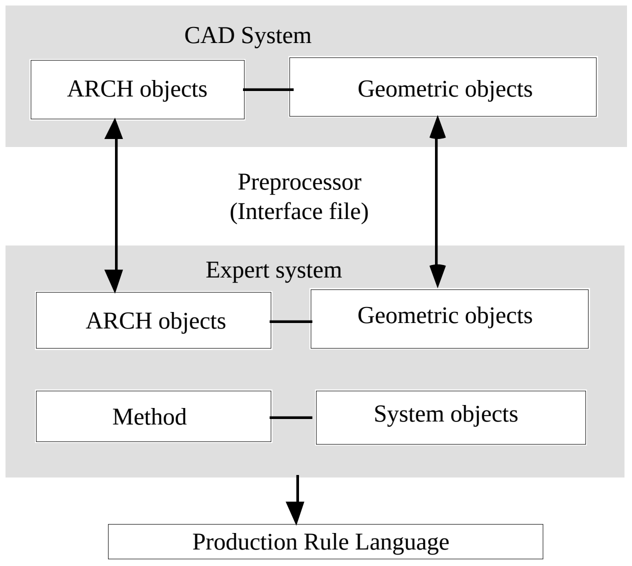


Figure 2. CAD-expert system interaction.

For example, the score for the egress requirement parameter of a new building that is taller than 150ft should be a minimum of 7.5. When the features of a building are assessed through the evaluation tables and the results are totaled, if this minimum score is not achieved, then it can be inferred that the building does not provide the required level of life safety. Furthermore, generating a single final value for compliance and knowing the components that lead to this single value enable the architect or the fire protection engineer to modify the most deficient parameter or to initiate a change in the most cost effective parameter to achieve the required level of life safety.

Therefore, the user should be able to trace the parameters that contribute to a compliance or noncompliance score, and decide at which fire safety features improvements must be made. Furthermore, the program can suggest areas of improvement by checking the lowest level of safety among the parameters that were propagated through the system.

3 Implementation

Level5 Object expert system (Information Builders Inc., 1992) runs under Microsoft Windows environment and provides a set of tools ranging from backward and forward chaining processes to object-oriented data structures to a graphic user interface.

The software provides the following set of tools and menus.

3.1 Object Editor

The definition of classes of objects as well their instantiation can be performed through this function. The programmer has the option to save the objects and their attributes for a specific application as a shell. Although objects can be instantiated through Level5obj, more typically they are instantiated through an external data file. In an architectural expert system application, this data file can be an interface file to a CAD system or to a 3D-modeling software. In addition to classes of objects and their attributes, class inheritance can also be defined through the Object editor. Furthermore, when a reference to an undeclared object is made in a rule or a demon, the system automatically creates a new object or a new attribute of an object by parsing the syntax of the new rule.

The program also provides a set of system classes such as dialog boxes, value boxes, time objects, picture boxes, prompt boxes etc., which can be attached to user objects allowing a fully controlled user interface environment.

3.2 Rule and Demon Editor

Level5Obj provides a rule and demon editor that enables the programmer to define the rules that govern the behavior of objects, or the methods (demons) that are triggered during forward chaining process. Rules, on the other hand, control the backward chaining process.

3.3 Agenda Editor

A backward chaining process can only be invoked if a goal is set through the Agenda editor. During the execution of the program, goals trigger the inferencing mechanism by processing the rules that are related to a given goal. While goals can be set and invoked individually, it is also possible to create a hierarchy of goals, where a sibling goal is invoked only if the parent goal is satisfied.

3.4 Knowledge Tree

The relationship between objects, rules, demons, goals in the agenda, and windows (which are themselves default system objects) that help to define the user interface can be displayed as a knowledge tree. For example, in Figure 3 a portion of the Knowledge tree for ARCHCode/Business can be seen. This is mainly a debugging tool, and helps the programmer track the inferencing system he/she has created through the rules. The system automatically identifies all possible paths for existing rules and demons. The programmer has the option to trace all reasoning paths or just the session paths taken by the program.

3.5 Methods

Methods can be associated with the objects in the system through "When Needed For" or "When Changed For" functions, depending on whether a backward or a forward reasoning system is being built. They allow the calculation of the values for the attributes of an object and replace other options such as asking the user to interactively input a value or reading it from a data file. For example, fire severity must be calculated to decide if a potential fire in a given space is structurally endangering. Equation (1) defines fire severity:

$$t = 4.9\frac{L}{\sqrt{A \cdot A_s}} \qquad (1)$$

where t = fire severity in minutes
L = total fire load (lbs of wood equivalent)
A = area of opening (ft^2)
A_S = surface area of walls, floors, and ceiling of room (ft^2)

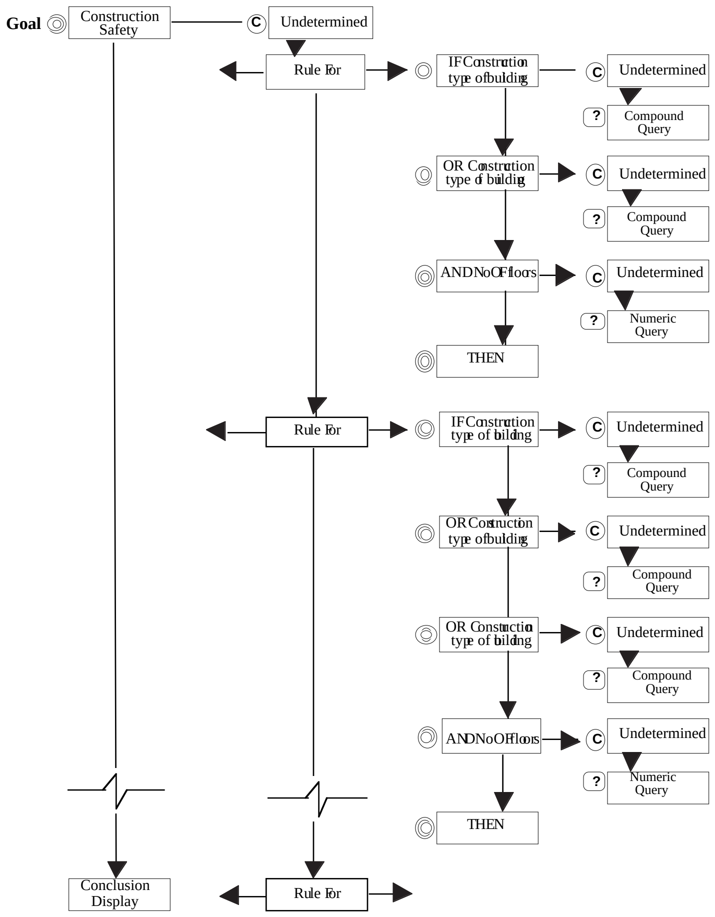


Figure 3. Knowledge tree showing reasoning paths in a sample session.

The method "When Needed For: Fire severity OF room" is defined by using the formula above, which is invoked automatically whenever fire severity is undefined and a value is required. Furthermore, the parameters of this method, such as "Area OF Opening" can be associated with an additional "When Needed For: Area OF Opening OF room" method. The knowledge base of the system provides a framework for multiple levels (Figure 3) of sophistication, where if a value is not associated with an internal method, then it displays a query. Thus, the programmer can start testing incomplete models and refine the system progressively.

Level5Obj provides two different program file formats. Knowledge base (KNB) files can only be displayed through the graphic interface provided by Level5Obj, whereas a PRL (Production Rule Language) file is essentially a program file in ASCII form. One can think of the KNB file as an object file, whereas PRL file as a source file (Table 1).

4 The Interface To a CAD System

Typically, researchers who build prototypes of architectural expert systems create the instances of objects by directly inputting them through the expert system. In most cases this is not practical for large scale applications. A CAD system such as AutoCAD (Autodesk Inc., 1992) can be used to create an interface file that will instantiate the object classes in such an expert system. Defining data structures that accommodate room-edge relationships, space aggregates such as smoke areas, fire zones etc. through AutoCAD were addressed by this author in an earlier paper (Ozel, 1992). The interaction of AutoCAD and Level5Obj is possible through a number of methods:

1. Outputting a Level5Obj program file in PRL syntax through AutoCAD. This can provide the flexibility of creating as well as instantiating objects through AutoCAD. In this case almost all of the calculations such as those needed for spatial and geometric reasoning must be done through AutoLISP, and the data structuring tools available in expert systems will not be available.
2. Creating a database extraction file through the attribute extraction feature of AutoCAD. Level5Obj recognizes DB3 database file structure, but the relational database interaction of Level5Obj is rigidly defined and only provides standard functions such as search, append, delete or insert a record. Furthermore, to take advantage of the object oriented nature of the data structuring and rule based reasoning system of Level5Obj, one must further establish new class relationships and hierarchies, thus this method does not necessarily serve the needs of the application developed here.
3. Generating a data file that contains architectural, fire safety and graphic entity information. *Class hierarchies* must essentially be identified through the expert system, and the *data* needed for the instantiation of classes is imported into the expert system from a file created through the CAD system. This process has the greatest potential in reflecting the graphic data base of CAD to the rule base of the expert system. Thus the decision to use this method was made.

Graphic objects such as lines, surfaces, volumes which can be associated with architectural objects must also be available in the expert system. As part of this project, graphic object classes and hierarchies were created in Level5Obj. Methods, such as in_-polygon, polygon travel distance, polygon adjacency that allow geometric reasoning must be developed in conjunction with graphic objects.

Multiple inheritance allows one to establish class relationships among architectural objects such as rooms and their boundaries as well as between architectural objects and graphic entities such as lines, polygons, surfaces etc. The current implementation addresses only two dimensional graphic entity objects, since most of the geometric reasoning needs of Life Safety code applications require the manipulation of 2d entities. Future expansion should provide the inclusion of 3D objects into the expert system and as well as the tools that will allow geometric reasoning using them.

5 ARCHCode/Business

5.1 Class Definitions

In the present application, an object class called BUILDING was created at the top of the class hierarchy. Among the attributes of this object are NoOFfloors, Height, Construction type, Fire severity (hazard type and level), Level of Exit Protection etc. (Table 1). This table is also a demonstration of the syntax of Production Rule Language (PRL) of Level5Obj.

Table 1. Class Definitions in PRL Syntax

```
CLASS Building
        WITH Age COMPOUND
                New
                Existing
        WITH NoOFfloors NUMERIC
     WITH Construction COMPOUND
                NoncombTYPEIORII      |
                NoncombTYPEII111      |
                NoncombTYPEII000      |
                CombTYPEIII211        |        Possible values
                CombTYPEIII200        |
                CombTYPEIV            |
                CombTYPEVIII          |
                CombTYPEV000          |

        WITH Hazardtype COMPOUND
                Exposed
                Segregated from exits
                None
        WITH Hazardlevel COMPOUND
                Double
                Single
        WITH Height NUMERIC

CLASS Floor
        WITH name STRING
        WITH Height NUMERIC
ETC.
```

Among other objects created in ARCHCode/Business are safety parameters with attributes as seen in Figure 4, the CODE object, architectural and spatial objects such as rooms and their boundaries, and graphic objects such as lines, surfaces etc. As one generates a rule, a demon or a method, the reference to the attributes of objects are made through the *OF* keyword, such as Height OF Building or Height OF Floor, therefore the same attribute name can be used within the context of different object classes.

Many of these attributes, such as the *Construction* OF BUILDING are defined as having a compound data type, since they can assume only one value from a limited number of values. Construction type of a building can be identified as one of eight different types, which are provided as the values for this attribute in ARCHCode/Business (Table 1).

5.2 Rules and Demons

The “rule and demon editor” of the system was used to define a number of rules for reasoning through backward chaining. For example, to identify the fire safety parameter associated with the type of construction, the number of floors of a building and its height, 22 rules were defined. Some of these rules can be seen in Table 2. The height and the number of floors are extracted from the interface file. One should also be able to modify these parameters through the expert system, since a designer may want to test some global decisions such as the selection of the construction type of a building quickly, without necessarily going through full design and implementation by means of a CAD system.

Table 2. Rule Base of Fire Safety Parameter for Building Construction

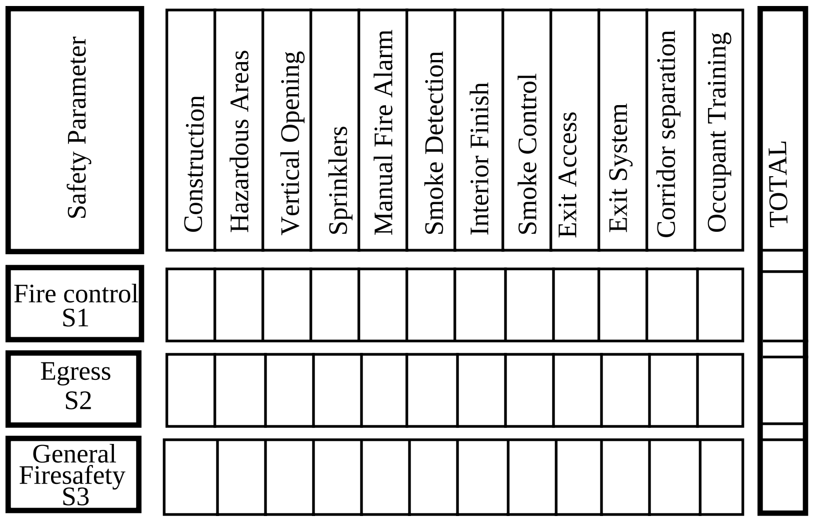
```
RULE for construction parameter
        IF Construction OF Building IS NoncombTYPEIORII
        OR Construction OF Building IS NoncombTYPEII111
        OR Construction OF Building IS NoncombTYPEII000
AND NoOFfloors OF Building = 1
THEN Construction OF Safety Items :=0

RULE for Construction parameter1
        IF NoOFfloors OF Building =2
AND Construction OF Building IS NoncombTYPEII111
THEN Construction OF Safety Items := 2
ETC.
```

Along with these rules, a goal called “Construction parameter OF Safety Items” was defined through the Agenda Editor, through which a backward chaining process can be triggered. The resulting value indicating the safety level of the construction type of a building is propagated to the item called “Fire Control of Construction” in the checklist “Individual Safety Evaluation” (Figure 4).

5.3 Agenda

In ArchCode/Business, fire safety parameters of the building were identified as goals through the agenda editor. For example “safety parameter for hazardous areas” is a goal which must be satisfied through the rules defined in the system, thus the rules that are related to this goal are automatically invoked and tested for their truth values. In fact, in the system multiple goals are defined and related rules tested before a proposed building can be assumed to comply with the requirements of the code.

Furthermore, items in each worksheet (from NFPA 101M) were represented as subgoals. For example, the Safety Parameter for Vertical Openings is identified as a subgoal (Figure 1), which is propagated to the Fire Control of Vertical Openings as seen in Figure 4. Thus, the Fire Control of Vertical Openings is higher in the hierarchy of goals. The final objective is to find the compliance scores for the Fire Control (S1), Egress Provision (S2), and the General Firesafety (S3) of a BUILDING (Figure 4). Then these parameters of the BUILDING object must be compared with the similar parameters of the CODE object, i.e., the requirements of the code, with the assumption that BUILDING object meets the needs of the CODE object if the minimum score is achieved for each of these parameters.



Safety Parameter	Construction	Hazardous Areas	Vertical Opening	Sprinklers	Manual Fire Alarm	Smoke Detection	Interior Finish	Smoke Control	Exit Access	Exit System	Corridor separation	Occupant Training	TOTAL
Fire control S1													
Egress S2													
General Firesafety S3													

Figure 4. Individual Safety Evaluation screen (adapted from NFPA 101M, Table 7-2).

6 Summary

Assessing the code compliance of a building requires expert knowledge and the ability to structure a very wide range of information included in the codes. Such an assessment can be performed through a computer application that uses expert system programming principles. While there is a need to represent code sections in the form of rules in such an expert system, there is also a need to structure and identify the order within which these rules will be followed in assessing a given building plan. By taking a systems approach where different aspects of fire safety in buildings are assigned weights through a set of checklists and are ultimately summed to generate a single value for compliance/noncompliance, it is possible to structure the order within which a proposed building will be evaluated. NFPA 101M, *Alternative Approaches to Fire Safety* provides several such guidelines. Furthermore, such an expert system can identify specific areas of deficiency in case of noncompliance and indicate tradeoffs.

On the other hand, architectural expert systems must include a geometric and graphic data base, which is critical in developing geometric reasoning methods that are assignable to architectural objects. Not only architectural objects but also graphic objects such as lines, polygons, surfaces must be created in such an expert system. In the future more detailed geometric reasoning algorithms specifically designed to solve reasoning problems encountered during the fire safety evaluation of a given building must be developed, which is recognized as the objective of the next phase of this study. Among these must be algorithms that identify the type of a spatial (e.g., atrium) object through its geometric configuration, or algorithms that use non-spatial attributes (e.g., fire resistance of the material of a wall) to define spatial clusters such as fire zones.

References

Autodesk Inc., 1992. *AutoCAD Manual*.

Autodesk Inc., 1992. *AutoLISP Manual*.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M. and Gero, J.S., 1990. *Knowledge Based Design Systems*. Reading, MA: Addison Wesley.

Delis, E. and Delis, A., 1991. "Algorithmic Support for Intelligent Fire-Code Checking," in *Proceedings*, IEEE International Conference on Tools for Artificial Intelligence, San Jose, CA.

Information Builders, Inc., 1992. *Level5 Object User's Manual*.

Kim, U., 1986. "Model for Integrated Design Evaluation System Using Knowledge Bases," in *Proceedings*, ACADIA '92 Workshop, pp. 203-215.

Lathrop, J., 1988. *Life Safety Code Handbook*, NFPA, Quincy, MA.

MacKellar, B. and Ozel, F., 1991. "An Integrity Constraint Approach to Verifying Building Designs," in *Proceedings*, First International Conference on Artificial Intelligence in Design, Edinburgh, UK.

National Fire Protection Association, 1992. *Alternative Approaches to Life Safety*. NFPA 101M, Quincy, MA.

New York State, 1990. *Office of Mental Health Bidding Manual.*

Ozel, F., 1984. "ARCH:Firesafety Code checking program," University of Michigan, College of Architecture and Urban Planning.

Ozel, F., 1992. "Data Modeling Needs of Life Safety Code (LSC) Compliance Applications," in *Proceedings*, ACADIA '92 Conference, Charleston, SC.