A knowledge-based approach to the automatic verification of designs from CAD databases

M. Balachandran, M. A. Rosenman and J. S. Gero

Design Computing Unit
Department of Architectural and Design Science
University of Sydney NSW 2006 Australia

Abstract. This paper commences by discussing some of the difficulties involved in the process of automatic verification of designs from CAD databases. The main focus of the paper is to describe and demonstrate an approach to the development of a knowledge-based integrated system which is capable of verifying a design through its description in a CAD database for conformance with some specified requirements. IPEXCAD, an early version of such an integrated environment, is described in terms of its architecture, implementation and operation issues. The system is illustrated by an application in building design.

INTRODUCTION

In the various disciplines of design, such as engineering, architecture, drawings are the medium through which decisions are communicated. A designer uses drawings to represent symbolically designed objects and their configurations in space. Furthermore, graphical representations are used to illustrate the different alternatives available at any stage of a decision making process and to aid in the evaluation of the outcome of these decisions. Commercially available computer-aided drafting (CAD) packages are widely used by design professionals. Such computer-aided drafting packages provide designers with powerful and flexible facilities for creating and modifying a graphical representation of their design descriptions. A CAD system stores a design description produced by a designer in its database in terms of drawing elements. Such a database only contains syntactic relationships among those drawing elements. Therefore, one major limitation of such CAD systems is that they do not possess the capabilities to interpret the contents of a drawing as human designers do.

The verification of a design is the process of checking the completeness, consistency and correctness of the design against given sets of requirements. Such requirements may be of many different types, such as those given by design codes and may come in the form of clauses, heuristic rules, tables, diagrams, graphs and mathematical equations. Verification of a design from its computer representation in the form of a CAD database is a knowledge intensive task which requires semantic interpretation of the CAD database and the representation and manipulation of the design verification knowledge. Advances in artificial intelligence have led to the emergence of expert systems which explicitly represent the knowledge of human experts in some specific domain and apply this expertise to the problem solving process (Bobrow et al., 1986; Waterman, 1986). Expert systems are capable of representing and using the design verification knowledge in an explicit manner. The semantic interpretation of a CAD database is the process of transforming the syntactic relationships in the CAD database in accordance with the semantic relationships in the expert system's knowledge base.

One major problem involved in this approach is the mapping of syntactic information stored in the CAD system's database to the semantic information contained in the expert system's knowledge base. The syntax to semantics mapping of a CAD database is a tedious and domain dependant task which requires extensive domain knowledge. We use the notion of design prototypes to represent the domain knowledge as a set of abstractions which are used by the expert system for conceptual knowledge and by the CAD database interpreter to search for information from the drawing database. Therefore in order to verify a design from its graphical representation, a system must be able to perform interpretations and knowledge processing tasks.

Based on the needs and the nature of the design verification process, a conceptual architecture of an integrated knowledge-based CAD environment has been developed. The remainder of this paper describes the motivation, issues and design of the system with an emphasis on the problems encountered and techniques used.

BACKGROUND

Interactive computer graphics has attained considerable importance in computer-aided design and manufacture. Designers make their decisions with the help of computer presentations in the forms of perspectives, plans, elevations, and so on. The task of preparing and modifying such drawings has been considerably simplified by various packages commercially available for computer-aided drafting. However, the major drawback with most such systems lies in the interpretation of the graphical entities created by the designer or their incorporation into other stages of the design process. Currently, in most systems, the mapping between the graphical representation and the designed artefact is made by a human designer. In such systems this is partly because the systems have no representation of the artefact created by the user other than its graphical manifestation on the screen and its syntactic model in a data structure in the database of the system. In order to expand the effectiveness of the relationship between the user and the computer, we need to incorporate knowledge about the underlying artefact so that the system can use that knowledge to derive meanings of design elements represented graphically. For instance, in building design, a building plan is graphically represented by a series of straight lines and some symbols. If the knowledge required to interpret that set of lines and symbols as walls, windows, doors, etc. were represented in the system, the system could enforce other properties such as topology and geometry during any operation.

In the past significant efforts have been expended in developing integrated computer systems for design applications. For example Lee (1977) describes an integrated system, called ARK-2, for architectural design. Hoskins (1977) presents an integrated interactive computer-aided design system, called OXSYS, for building design. More recently many design researchers have carried out work aimed at developing integrated systems for design applications utilising knowledge-based approaches. Rehak et al. (1985) and Fenves et al. (1990) present some approaches to developing integrated knowledge-based software for design applications. Jain and Maher (1987) discuss a number of possible ways of combining expert systems and computer-aided drafting techniques. Dym et al. (1988) presents a knowledge-based system for checking the Life Safety Code of the fire regulations. Tyugu (1987) describes the importance of merging conceptual and expert knowledge in CAD. Balachandran and Gero (1988) describe a model for building knowledge-based graphical interfaces.

So far, existing integrated systems have been implemented using a tightly-coupled approach (Hoskins, 1977; Jain and Maher, 1987). That is, the systems know exactly what entities to expect from the graphics and the graphics system is tailored exactly to the knowledge in the system. Rehak and Howard (1985) put forward KADBASE as an approach to the integration of a distributed CAD system containing a variety of heterogeneous systems and design databases. This approach of advocating a flexible interface in which multiple expert systems and a variety of design databases communicate within an integrated system is the one pursued in this paper.

Current CAD Systems

Today's CAD systems can be broadly classified into three categories, namely general drafting systems, general modeling systems and domain-specific modeling systems. Drafting systems merely produce a graphic representation of whatever views of the design are drawn. There is no relation between elements in one view and those in another view. General modeling systems allow for the modeling of elements and assemblies of elements. The graphical representations are merely views of the one model. Information regarding the model can be entered or modified via any view (although some systems restrict this to the plan view only). Elements are created through drawings or macro-languages and are accessed through name-labels attached to them. In some systems other non-graphical properties, such as material, colour, and catalogue numbers may be associated with an element. In domain-specific systems the system has knowledge about a variety of types of elements within its domain as libraries of elements with various attributes. These elements may be of fixed geometry or parametric in nature.

Expert Systems in Design

In the past decade several expert systems have been demonstrated for design applications (Kostem and Maher, 1986; Gero, 1987a; Rosenman, 1990). While the advantages of expert systems are well documented for symbolic processing in logical inferences, they are not well suited to the computational processing involved in mathematics or graphics. They are also criticised for their shallowness and inability to reason beyond very limited scope. An important area of application of expert systems in design is that of engineering and building codes (Rosenman et al., 1986a; Garrett and Fenves, 1987; Sharpe et al., 1989). Complex design codes are ideal application areas for expert systems since both users and developers often have difficulties with ensuring correct interpretations.

THE VERIFICATION TASK

Let us have a look at the problems, processes and knowledge involved in the verification task. In order to understand this task better we first describe the main properties of designed artefacts, namely function, behaviour and structure properties.

Function, Behaviour, Structure

A designed artefact may be described in terms of its function, behaviour, and structure. the structure description is a description of what the artefact is; the behaviour description, a description of what it does and how it does what it does; and the function description, a description of what it is for, i.e. its purpose. For example, the structure of a clock includes the parts from which it is made, their material, shape, dimensions, how they are joined, etc.; its behaviour (if it is an analog clock) is that its hands rotate with a periodicity matching the elapsing of time, pointing to marks on the dial to display this; and its function is to mark the time.

Structure properties describe the form and physical properties of an artefact in terms of its geometrical and physical attributes, such as shape, dimensions, material, colour, etc. The behaviour properties of an artefact are emergent properties of the artefact given its existence and can be directly derived from its structure properties given that we have the necessary knowledge regarding these relationships. The function of an object, however, is a human determined property related to some human needs. There is no clear link from behaviour to function except through a human interpretation. Having said that, design is about the fulfilment of required functions through the creation of structures which exhibit behaviours which are recognized to be beneficial in producing these functions.

Not only are these function, behaviour and structure attributes important in the description of designs, but also their relationships. Relationships exist between function and behaviour, between behaviour and structure, between function and structure and among the attributes themselves.

Design Analysis and Evaluation

Artefacts, through their existence, exhibit a multitude of behaviours, not all of which are relevant to some particular context. For example, we are not generally interested in the electrical resistance of a pencil or the fact that it makes a sound when we strike another object. However, we may indeed be interested in the latter behaviour in the context of tapping out a message. The selection of which behaviours to focus on depends on the context defined by the function to be satisfied.

Given that we are operating in a certain functional context we will be interested in certain behaviours of any proposed design and that these behaviours achieve satisfactory levels of performance. These required levels of performance of the behaviours may be derived from the function, if we have such relationships or else they may be directly prescribed by codes, rule of design, etc. The process of interpreting a design description to derive levels of performance of behaviour is the process of *design analysis*. The process of comparing these derived performance levels to the required levels is the process of *design evaluation* while the process of determining the cause of an unsatisfactory behaviour is the process of *diagnosis* and the process in which structures are derived from behaviours and functions is the process of *design synthesis*. Design verification includes the processes of design analysis and design evaluation and, this paper will concentrate on these processes as we are interested in evaluating the performance of a proposed design whose description is given by modeling it graphically using a CAD system.

The Verification Task—An Example

Let us have a look at the sort of verification task we may want to do. As an example let us take a verification task dealing with compliance with a building code. The building code in question is the Building Code of Australia, referred to as the BCA Code (AUBRCC, 1988). We wish to ensure that a design, described graphically using a CAD system, complies with the provisions of this code. Let us take an excerpt from this code, namely some of the provisions dealing with masonry construction. Following is an excerpt from the BCA Code itself:

B2.3 **MASONRY CONSTRUCTION**

B2.3(1) **External wall thickness: Interpretation** - For the purposes of this clause the combined thickness of the inner and outer leaves of a cavity wall shall be deemed to be the thickness of the wall.

B2.3(2) **Minimum thickness of external walls** - The external walls of a building, if of masonry construction, shall be not less than 200 mm thick, except in the case of:

(a) a Class 7 or 8 building;
(b) a single storey building or the topmost storey of a multi-storey building where cavity wall construction is used and the combined thickness of the inner and outer leaves is not less than 190 mm;
(c) a Class 10 building or garage, laundry, tool shed, closet or the like forming part of a building of another Class; or
(d) a building subject to subclause B2.3(7).

As we can see, Clause B2.3 applies to masonry construction; Subclause B2.3(1) is an interpretive provision explaining how to arrive at the thickness of cavity walls (double walls); while Subclause B2.3(2) deals with the required minimum thickness of external walls. Except for specified situations this minimum thickness is required to be 200 mm.

Having a design of a building, our verification task consists of checking whether the above provisions apply, and if so whether our building complies. That is we must first check that our building contains elements of masonry construction, then Clause B2.3 is applicable; that our external walls are of masonry construction, then Subclause B2.3(2) is applicable and finally that all such external walls comply with Subclause B2.3(2).

The first problem that arises after having provided a graphical description of our design (and having determined that compliance with the BCA Code is required) is to determine which provisions of this code are actually applicable. This depends on the nature of the design. For example, a one storey building does not have stairs and therefore all provisions dealing with stairs are not applicable. We could thus start with the elements of the design and search the BCA Code for all relevant provisions. This entails three problems. The first problem is that of recognizing the elements of the design from the graphic database; the second problem is that of matching the description derived from the database to the descriptions used in the code; and the third problem is that of interpreting the BCA Code to ascertain the relevancies of certain references in the BCA Code to the elements found. For example, the above provisions deal with masonry construction. Most probably none of the elements interpreted from the CAD database will be identified as of 'masonry construction' but perhaps of brick, concrete, etc. In order to determine that Clause B2.3 is in fact applicable, there will have to be some recognition, that, a wall of brick or a floor of concrete does constitute masonry construction. So that, if we were to start from the elements themselves as obtained from the CAD database we could not directly identify the applicable provisions in the BCA Code.

Alternatively, we could start with the provisions of the BCA Code and determine their applicability. This would mean interpreting each provision so as to determine its relevancy vis-a-vis the elements derived from the CAD database. Because of the structural organization of the BCA Code (and most such codes) it is not necessary to examine exhaustively every provision but an implicit enumeration approach makes it possible to prune those portions of the BCA Code not relevant to the situation at hand. For example, since the BCA Code is structured into Parts; each Part into Clauses; and each Clause into Subclauses it will be possible to determine firstly if a Part is applicable. If not, then there is no need to investigate any of its provisions. Similarly if a Clause is found not to be applicable then none of its Subclauses need be investigated.

We do not want to design a system in which the description of a designed artefact is tailored to a particular description used in any particular application program. Therefore, the representations and the descriptions derived from a CAD system will be independent of the various application programs or knowledge systems which are needed to carry out the verification task. We will assume, however, a common domain, the domain of building designs, for example, so that both the descriptions in the CAD system and in the verification system refer to the same domain. We will continue using the BCA Code as an example of a verification application. It is obvious that both the descriptions of elements in the CAD system and those in the BCA Code are based on a much wider domain knowledge than the information stated. There are many assumptions made in the BCA Code about the relationships of building elements to each other, e.g. that columns are structural elements supporting such elements as beams, etc; about how to determine various information required, e.g. distance between exits. It is assumed that users of the BCA Code have a good deal of understanding of buildings. Similarly, when we interpret drawings we bring to bear this wide domain knowledge, e.g. walls connect to each other, they enclose spaces (together with ceilings and floors); doors and windows are located in or between walls, they form openings in walls, etc.

Therefore, an essential part of any integrated system for the verification of designs from

CAD databases will have to incorporate a domain knowledge base wherein knowledge exists about the elements in the domain and the relationships between these domains. It is not sufficient to include descriptions which are concerned only with the physical and geometrical aspects of these elements, the descriptions must include functional and behavioural aspects since a verification task involves the evaluation of the performance of designs. This domain knowledge base, while recognizing that CAD systems and verification applications will refer to it, needs to be independent of the particular CAD system and application program. It requires a representation suitable for the description of this domain knowledge in a generic fashion so that it can be useful in a wide variety of situations. However, in order to effect the necessary communications between the CAD database and this domain knowledge, there will be a need for a graphic database interpreter. Similarly, in order to effect the necessary communications between the verification program and this domain knowledge, there will be a need for an interpreter , which in the case of verification programs in the form of expert systems will be a knowledge base interpreter. Both these interpreters will, of necessity, be specific to the systems being used.

Interpretation

Graphical images are usually interpreted semantically by humans without any concern for the process used to produce the image. One of the main difficulties in using traditional CAD systems is that they store their models syntactically in the form of data and procedures. However, as was shown in the example above, verification programs deal in terms which require interpretation of the syntax produced by the CAD system. It means that a computable semantic model of the screen graphics needs to be generated in order to be used in other design activities. Such syntax to semantics transformations are fundamental to any automated design verification system using graphical representations.

Syntax and semantics in design

Syntax is the vocabulary and rules governing the composition or structure of such vocabulary. Semantics is the meaning derived from some syntactical description. The vocabulary varies according to the domain and the application. For literature, the syntax consists of words and the rules of grammar. The semantics consists of the ideas expressed by this composition. The collection of these ideas is called a 'book'. At a different level, the syntax consists of books and the rules governing their arrangement. The semantics is the interpretation of this collection as a 'library'.

Design is also concerned with vocabulary elements, such as gears, rods, transistors, struts and windows and how these elements are put together (Coyne et al., 1990). Design descriptions are in terms of syntax. The interpretation of design descriptions is an issue of semantics. Semantics is concerned with derived descriptions, either in terms of conceptual identification or in terms of performance. For example, a design description of four walls, a floor and a ceiling arranged in a given structure is interpreted to mean a 'room'; this same design description (given dimensions) can also yield such interpretations as: the area of the room is such and such; the size of the room is large; and the proportions are pleasing. The notions of syntax and semantics are not absolute. What is semantics at one level is syntax at another. For example in a geometrical context, given a syntax comprising points and their locations, the semantic interpretation may be that of a set of lines. Taking the lines and their arrangement as syntax, at a higher level of description, the semantic interpretation may be that of some shape, e.g. a rectangle.

Type of knowledge inside a CAD system: syntax
Drawings are graphical representations of design descriptions usually representing only topological and geometrical properties of a design although other properties may be attached to the elements of the design. Drawings on paper are merely marks on paper while on a computer screen they are merely an arrangement of pixels. Drawings are therefore syntactical descriptions of a design. Interpretations of the meaning of such drawings is done by the human observer. Certain lines mean elements such as walls or doors; groupings of these elements mean rooms, spaces, etc.; groupings of these spaces mean buildings. Working drawings specify the type of elements, their material, dimensions and location. This is all that is necessary for contractors (or CAM systems) to assemble the artefact. There is no need for them to know about the intent of the designer as to the function of the artefact or the sort of effects the designers are trying to create as long as the syntactic descriptions are complete and unambiguous. Of course when this is not so a knowledge of intent can help resolve such problems.

As mentioned previously, there are three general types of CAD systems: drafting systems; modeling systems and domain specific modeling systems. Drafting systems, such as AutoCAD (Autodesk, 1988), store information as vectors of points or lines and/or may have a limited capability for representing 2D shapes and creating symbols to which element labels can be attached. There is no relationship between different views of the same object and such relationships have to be computed if required. A modeling system, such as EAGLE (Carbs, 1985), allows for the modeling of objects through graphic (and textual) input. These objects are identified by labels or names at the time of modeling. The modeling system stores information about objects—the database is organized as a series of objects indexed by their name and possessing certain properties, such as geometric and labels as to material. Drawings can be generated from any view and relate to the same object or set of objects. Elements such as walls which are represented by parallel lines at certain fixed distances apart but which may vary in length (and height) have to be interpreted as such. Elements such as rooms and other spaces must also be interpreted as these are not modeled explicitly but 'emerge' from the arrangement of their constituent elements. A domain-specific modeling system, such as ArchiCAD (Graphisoft, 1989), provides menus of parametric elements (such as walls, roofs, floors) about which the system has certain knowledge regarding their topology and representation. Users select the appropriate symbol representing the required element and locate it by defining its position as well as its dimensions in three dimensions. These elements usually come with certain default values for various properties, such as thickness, material and representation. Users may change these default values at any time and these values are then associated with all instances of that element. In addition, there usually exist libraries of domain elements, such as doors, windows, fixtures and fittings as well as the ability for users to create their own.

CAD databases usually have special formats that could be read and used by other computer systems. Two of the most common such file formats are:

(a) DXF (Drawing eXchange Format) and
(b) IGES (Initial Graphics Exchange Standard)

A DXF or IGES file contains both graphical and non-graphical information about a drawing formatted to be easily read by the user or a computer. Almost every CAD system provides some sort of facilities for reading and writing DXF or IGES files. An example of a DXF representation of an object inserted into a drawing is shown below. The indented numbers 0, 8, 2, etc. are some of the group code numbers used by the DXF format.

```
  0                                            0
INSERT   Entity type                         ATTRIB      Entity type
  8                                            1
FLOOR    Layer name                          JOHN        The value of the attribute
  2                                            2
ROOM     Block name                          OCCUPANT    The name of the attribute
  10                                          ...
5.0      The X coordinate of the insertion
  20
10.0     The Y coordinate of the insertion
```

Type of knowledge inside verification applications: semantics
The knowledge in verification applications, as was shown in the above example drawn from the BCA Code, deals with required behaviours of elements in various situations. That is, the knowledge deals with semantic information and requires the interpretation of the current situation so as to match the intentions specified in the Code. For example, the meaning of the term 'exit' is that of a doorway leading to a road or open space or a pathway leading from spaces in the building to the exterior through passageways, stairs or ramps. This needs to be interpreted from the available syntax.

DEVELOPMENT OF A SYSTEM FOR THE AUTOMATIC VERIFICATION OF DESIGNS FROM CAD DATABASES

Domain Knowledge—Design Prototypes

Knowledge about a particular domain needs be represented in a generic way so that many situations pertaining to that domain can relate to this knowledge. The type of knowledge required is knowledge about artefacts within the domain from the point of view of their structure, behaviour, function and relationships between these and also to other artefacts. Such knowledge provides a deep structure useful for reasoning within the domain. This knowledge should be sufficiently comprehensive to allow for a wide variety of situations which may arise within the particular domain and for a variety of verification tasks which may be deemed necessary. Furthermore, this knowledge must be able to be augmented or modified as the need arises since it is practically impossible to determine every need that may arise except for a limited set of applications. In general, all the objects that need to be described using the CAD system must exist in the domain knowledge for meaningful instances to be created which will be capable of being evaluated.

Previous systems for representing global domain knowledge have represented this knowledge using frames (Rehak and Howard, 1985; Dym et al., 1988). In the CODE system (Rosenman et al., 1986b), the model of the artifact described was extracted from the rule base of an expert system dealing with a building code and represented as frames. The EAGLE system was used to describe graphically an instance of a building and the elements thus described were modeled as instances of the model frames. The disadvantages in frames is that all slots representing attributes are equivalent in type. There is no capability to distinguish between the structure, behaviour and function properties of an object. Moreover, the frames describe only the attributes themselves and not the knowledge regarding the relationships between these attributes.

Design prototypes have been proposed as a means of representing design knowledge comprehensively (Gero, 1987b; Gero and Rosenman, 1989). Design prototypes represent a class of design elements and embody all the knowledge necessary to produce a particular instance of the class or to evaluate the performance of a given instance. Design prototypes are structured along function, behaviour and structure properties as well as relationships between these. The design prototype schema allows the syntactic and semantic knowledge about the above classes of properties and how they relate to each other to be represented and manipulated efficiently. Figure 1 shows the model of a design prototype. The knowledge in a design prototype allows the derivation of values for structure and behaviour attributes, and the interpretation of structure descriptions to derive behaviour and function properties. In addition, there exists knowledge about the relationships to other design prototypes, and knowledge about function and structure constraints.

Design prototypes may be at various levels of abstraction and may be related to each other by typological or structure properties. For example, a design prototype may have links 'a_type_of', 'an_element_of', etc. with some other design prototype, thus forming a hierarchy of design prototypes both from a taxonomic and an elemental view. Instances may inherit properties from other more generic design prototypes through appropriate links. The prototype base contains all the prototypes deemed necessary for a given domain. This prototype base must be able to be augmented and/or modified as necessary. A design prototype engine performs the tasks of manipulating the knowledge in the prototype base, creating an instance of a specific design prototype and deriving values to specific variables of the instances as needed.

It has been shown that an expert system based on such design prototypes has the capability to represent experience as a set of general concepts in which semantic and syntactical relationships are explicitly defined (Rosenman et al., 1989). This can form the basis of an expert system where different knowledge bases dealing with these concepts can be implemented to interpret structure descriptions to derive behaviour, to evaluate performances or to derive structure descriptions from function descriptions. Moreover, this representation allows for other applications, as well as graphical modeling systems to provide and receive information which can be interpreted as necessary.

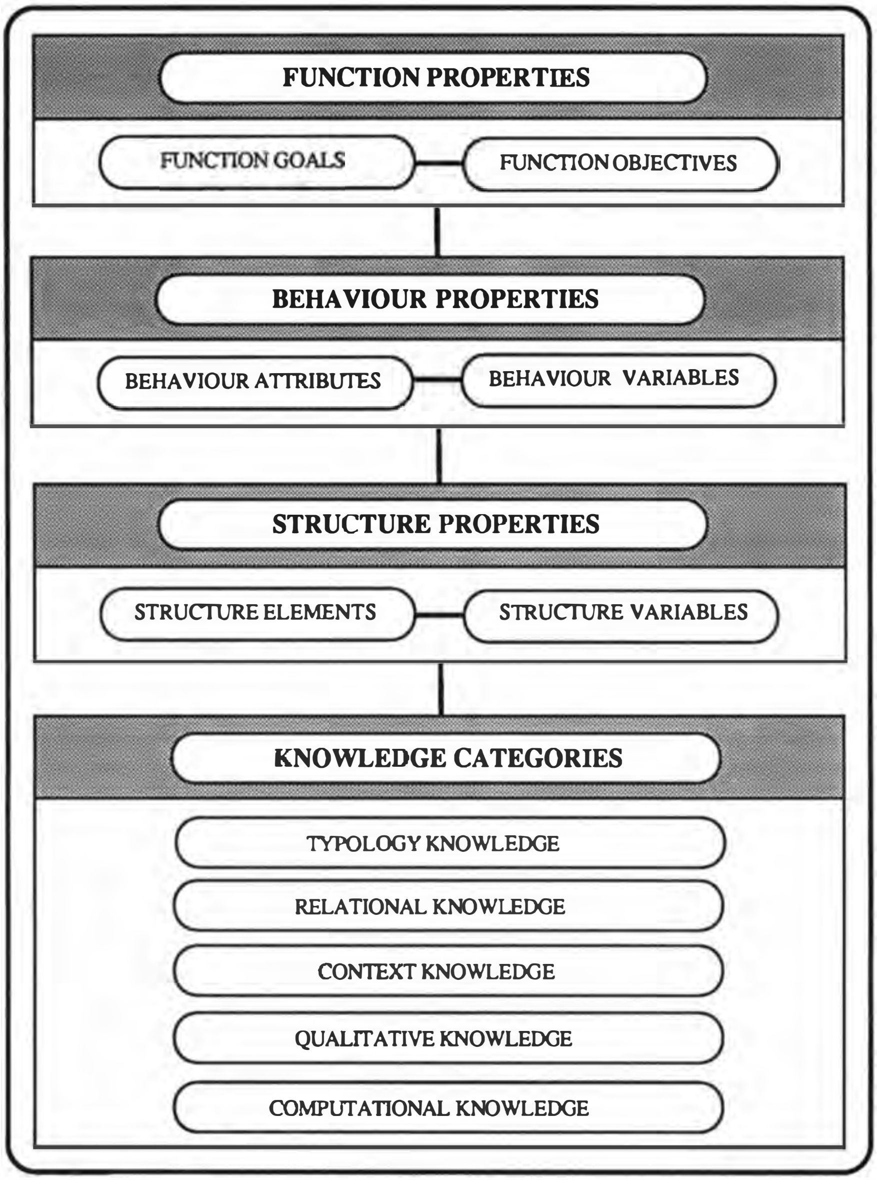


Figure 1. The model of a design prototype

Verification Knowledge—Production Rules and Procedures

The verification tasks may be quite varied depending on the domain, the particular design and also the design context. Moreover, the verification knowledge may be in different forms, e.g. procedures, tables, rules of thumb, prescriptive clauses etc. Rule-based expert systems have shown themselves especially capable of modeling situational and causal knowledge in a form which is easily comprehensible and relatively easy to formulate and maintain. In addition, they have the capability of providing explanations during the implementation process. However, for such systems, to incorporate some meaningful descriptions of the objects in their domain, an object-attribute-value representation is necessary to enable the extraction of these descriptions. This object-attribute-value representation thus allows for the mapping between the objects and their attributes within the scope of the expert system and that of the design prototypes. This knowledge representation allows for a wide variety of verification applications to be modelled in a uniform manner.

For example some of the rules formed from the Code example of Section 3.3 are as follows:

```
R50  IF           element OF building IS E
          AND     type of construction OF E IS masonry construction
     THEN         applicability OF clause B2.B2.3 IS determined.

R60  IF           applicability OF clause B2.3 IS determined
          AND     FOR_ALL subclause OF clause B2.3 IS SCL
          AND     applicability OF SCL IS determined
          AND     compliance WITH SCL IS satisfactory
     THEN         compliance WITH clause B2.3 IS satisfactory.

R70  IF           applicability OF clause B2.3 IS determined
          AND     FOR_ANY type of construction OF external wall IS masonry construction
     THEN         applicability OF subclause B2.3(2) IS determined.

R80  IF           applicability OF subclause B2.3(2) IS determined
          AND     FOR_ALL type of construction OF external wall IS masonry construction
          AND     NOT exemption FROM subclause B2.3(2) IS applicable
          AND     thickness OF external wall >= 200
     THEN         compliance WITH subclause B2.3(2) IS satisfactory.

R90  IF           classification OF building IS class 7 OR class 8 OR class 10
          OR_IF   number of storeys OF building IS 1
     THEN         exemption FROM subclause B2.3(2) IS applicable.
```

In general, the above rules take the form of:

IF $\quad$ condition_1 AND ... AND condition_m

THEN $\quad$ consequence_1 AND ... AND consequence_n

where condition$_i$ and consequence$_j$ take the form:

<attribute> <PREP> <object> <VERB> <value>

where <PREP> and <VERB> are any user defined terms

The rules are formulated as a representation of the particular verification knowledge and, as such, follow the terminology used in such applications. This may not match exactly to terms used in the graphic description or existing in the domain knowledge. To be useful, this knowledge will have to be interpreted. Further, there may be knowledge regarding some aspect of the design both in the expert system knowledge base and in the design prototype base. Since the expert system knowledge is more specific to the problem at hand it takes precedence over the general domain knowledge.

Not all knowledge can be formulated as rules. For example, calculations may be required or table lookups required. In such cases procedures will be required. The necessary information will have to be passed to these procedures and results obtained in a form compatible with the rest of the knowledge.

Knowledge Base Interpreter

The role of the knowledge base interpreter (KBI) is to interpret the rules in the knowledge base and make explicit the terms used therein. Using the object-attribute-value format the KBI is able to extract the objects, attributes and range of possible values within the scope of the expert knowledge. Each object is then represented as a frame, its attributes as slots and facets provide for descriptions of various types of values for the attributes, including in which rules and which part of the rules the attribute is to be found. The KBI notes whether matches exist between the objects found in the expert system knowledge base and those in the domain knowledge.

Not all the knowledge present in the knowledge base of the expert system can be interpreted by the KBI automatically. The knowledge engineers formulating the verification knowledge rule base will, determine matches and non-matches between the objects constructed and those in the domain knowledge. They may modify the terminology in the rules to conform with the terminology in the design prototypes or they may modify the design prototype base to incorporate new information where it is deemed appropriate. This may include modifying existing design prototypes and/or adding new design prototypes. Else, they have the option to add links between frame objects and design prototypes. For example, instead of changing the term 'stairway' it is possible to add a descriptor ' same_as' as in 'same_as: stair' to notify the system that 'stairway' and 'stair' are synonyms.

In most cases, extra interpretative knowledge is required to make the necessary linkage between the knowledge base objects and the design prototypes. This knowledge will usually include procedures for deciding which information is required to identify such associations. To illustrate this kind of necessary interpretative knowledge, let us have a look at another example taken from the BCA Code:

C3.5 TYPE A CONSTRUCTION

C3.5(1) **Requirements** - In a building required to be of Type A construction, each part mentioned in Table C3.5, and any beam or column incorporated in it, shall (subject to the modifications set out in this clause and clause C4.2) -

(a) ...
(b) have an FRL not less than that listed in the Table, for the particular Class of building concerned; and
(c) ...

A simplified form of this knowledge formulated as rules and procedures could be as follows:

```
R200  IF          applicability OF Part C3 IS determined
         AND      type of construction OF building IS type A
      THEN        applicability OF clause C3.5 IS determined.

R210  IF          applicability OF clause C3.5 IS determined
         AND      FOR_ALL element OF Table C3.5 IS E
         AND      DO identify-element(E)
         AND      DO table_lookup(C3.5, E, FRL, T)
         AND      FRL OF E IS_REQD_TO_BE T
         AND      FRL OF E >= T
      THEN        compliance OF clause C3.5 IS satisfactory.
```

Once clause C3.5 is applicable, i.e. we are dealing with a building of type A construction, we check every element in Table C3.5 to see if it exists in the building. If so, we look up the table to get its required FRL and compare that to its actual FRL. The 'DO' keyword in the rules makes calls to the function following. The elements in Table C3.5 are described in terms that will not match directly to the design objects in the domain and interpretation is required. A procedure to identify one of the elements mentioned in Table C3.5, e.g. 'loadbearing internal wall bounding a public corridor' would have the following form:

```
identify_element(loadbearing internal wall bounding public corridor)
    if  find        instance I
        such that   I is an instance of internal wall
        and         I is an instance of loadbearing wall
        and         I bounds S
        where       S is an instance of public corridor
```

The above procedure will succeed if it finds an instance with the following type of information:

```
wall1                                           hall2
    instance_of:   internal wall                instance of: corridor
    instance_of:   loadbearing wall             ...
bounds:            hall2
    ...
```

In addition, since the domain knowledge will not have the concept 'public corridor' this will have to be defined in the following manner:

```
public corridor
    a_type_of:               corridor
    membership_criteria:     dependent_on R43
```

where rule R43 may make a call to a procedure for determining whether an instance (of a corridor or hall) is in fact an instance of a 'public corridor'. In this case this means determining such things as whether the instance 'hall2' provides means of egress from a part of a storey to a required exit.

Knowledge Base Objects—Frames

It can be seen from the example rule base that there are two types of objects referred to in the rules. The first type are objects that refer to design objects. The frames corresponding to these objects are shown below:

```
building (prototype)                                external wall (prototype)
    element                                             type of construction
        affects:    [BCA_R50]                               options:  masonry construction, ...
    classification                                          affects:  [BCA_R80]
        options:    class7, class8, class 10,...        thickness
        affects:    [BCA_R90]                               affects:  [BCA_R90]
        category:   unknown
    number of storeys
        affects:    [BCA_R90]
```

In this case, the above objects could be reasonably expected to exist in the domain knowledge as design prototypes. When a match is found in the design prototype base this is noted by adding '(prototype)' to the object frame's name. Not all design objects mentioned in the expert knowledge base will have corresponding matches in the design prototype base. This may be because this object is specific to the expert knowledge, e.g. the object 'exit' or because of terminology used, e.g' stairway' as against 'stair'. It is also possible for attributes in an object matched to a design prototype not to exist in the domain knowledge as it is specific to the particular expert knowledge, e.g. the attribute 'classification' of the object 'building'. In this case the attribute is marked with the value 'unknown' in the facet 'category' as the system has no way of telling the type of such attributes, i.e. if function, behaviour or structure.

The other type of objects are objects which are not design objects but are objects to the expert system. These are also formulated as frames but, obviously, there will be no reference to any design prototype. Examples of such frames are shown below:

```
clause                                        subclause
    applicability                                 applicability
        options:   determined                         options:  determined
        affects:   [BCA_R60, BCA_R70]                 affects:  [BCA_R80]
    subclause                                         d e p e n d e n t _ o n :
[BCA_R70]
        affects:   [BCA_R60]                      compliance
    compliance                                        options:  satisfactory
        options:   satisfactory                       adependent_on:
[BCA_R80]
        dependent_on:                         [BCA_R60]
```

The frame part of the system makes use of the usual properties of frame systems.

Design Description—CAD Database

Here we will assume a level of information in the CAD database corresponding to objects in the design. That is, we are not concerned with the interpretation of arrangement of pure geometrical features, such as lines, to derive features or objects as is the work of Nnaji and Kang (1990). We are therefore assuming that the CAD system allows the users to create 'objects'. We have also previously stated that all objects to be described using the CAD system must exist as design prototypes. Nevertheless, the representation of such objects in the CAD database will be in a format incompatible with that of the domain knowledge and must be interpreted to produce descriptions which are meaningful to the rest of the system.

CAD Database Interpreter

In the process of integrating CAD systems with expert systems, the information presented graphically must be converted to a form that can be understood by the expert system, that is to a form commensurate with that of the design prototype format. For this purpose a CAD database interpreter (DBI) must be used. Where the CAD system database uses a standard graphic database format such as IGES or DXF or higher level format the interpreter must convert these formats to that of the design prototype format while if the CAD system database uses some non-standard format the interpreter must be written specifically for the particular CAD system used. CAD systems consist of a graphical interface and a database. The database of a CAD system contains representations of drawing elements and numeric or alphanumeric information associated with those elements. The syntactical information, namely, in the form of dimensions, locations, shapes, etc. can be mapped onto the structure properties of a design prototype. The graphic database interpreter consults the appropriate design prototypes and converts the information in the CAD system's database to the appropriate instances of design prototypes. Where required, the design prototype will have the necessary information for recognizing elements in the CAD database and forming the appropriate instances. For example, the design prototype 'room' will have the required cells to procedures for recognizing that a 'space' with a label bounded by walls (and/or windows, partitions, etc) is a room of a certain type. Most current CAD systems provide special database utilities to access and manipulate their databases. For example, AutoCAD and EAGLE have programming languages that let users write programs to manage and manipulate both graphic and nongraphic data.

Instance Base

All processes concerning an actual entity deal with an instance of that entity. Entities are generated through their description using the CAD system. The instance base contains the instances created during the running of an application. The instance base thus constitutes the working environment of the system. Instances are descriptions of elements derived from the CAD database interpreter or of knowledge base objects derived from the knowledge base interpreter with their relevant attributes instantiated to derived values.

System Implementation

A system named IPEXCAD (Integrated Prototype-based EXpert CAD environment) for the verification of designs from CAD drawings has been implemented on SUN workstations and is described below.

One of the major concepts utilised in the development of the system is a clear separation between the generic expert system module, which performs reasoning processes, the generic domain knowledge and the CAD package, which performs drawing operations. The system consists of seven subsystems. They are:

(a) a user interface;
(b) a stand-alone expert system shell, EXBUILD (Balachandran and Rosenman, 1990);
(c) a stand-alone CAD system that provides standard drafting and modelling features;
(d) a stand-alone design prototype system, PROTOKIT;
(e) a knowledge base interpreter which interprets EXBUILD's knowledge base and allows the knowledge engineers to provide necessary interpretations;
(f) a graphic database interpreter which interprets queries from PROTOKIT and directs them to the CAD database; and
(g) the working memory.

Figure 2 illustrates the system architecture diagrammatically. IPEXCAD uses a Macintosh-like interface under the Sunview window system. The user interface provides facilities for creating, modifying, displaying and saving information associated with design prototypes, instances and knowledge bases. A mouse-based text editing facility allows any text to be modified easily and conveniently. An appropriate CAD system may be invoked by selecting from a sub-menu which is displayed when clicking on the CAD icon from the main menu.

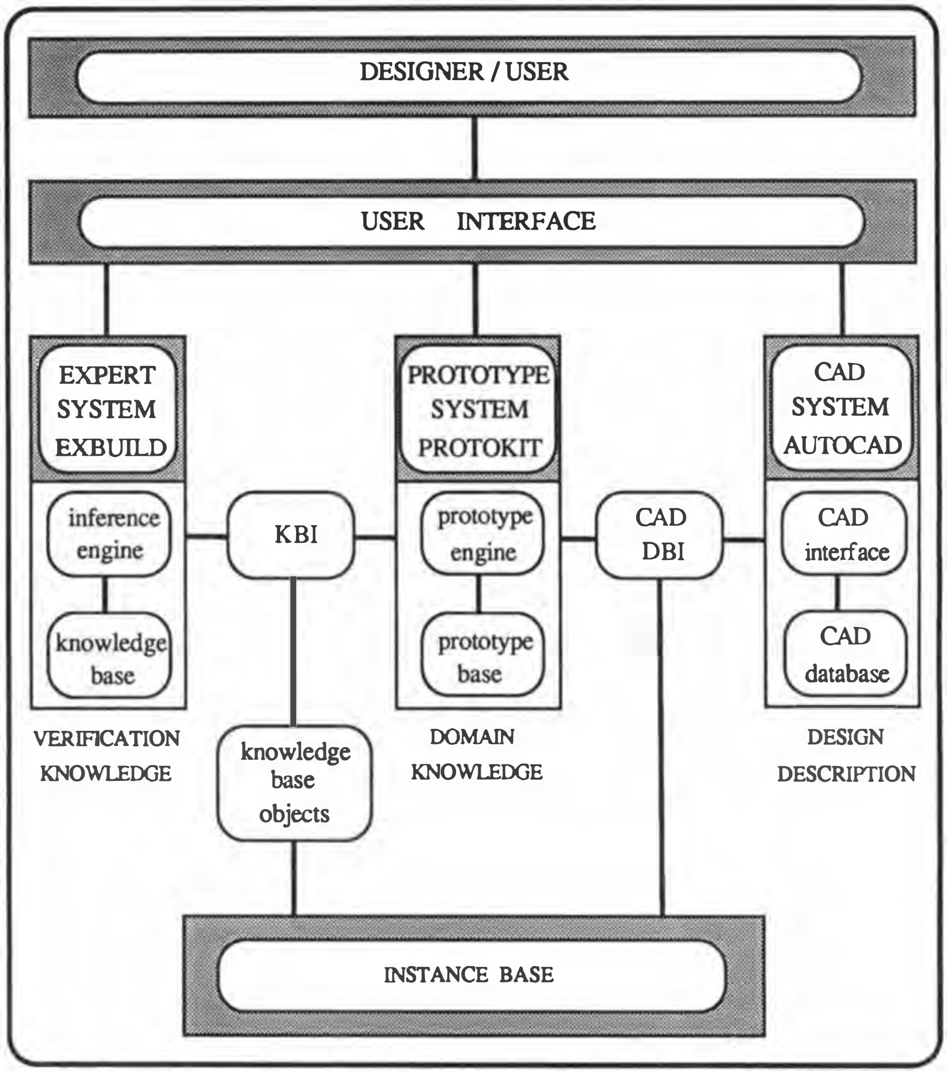


Figure 2. The system architecture of an integrated system for the verification of designs from CAD databases

The verification process is initiated by the users selecting the 'verify' icon. If no CAD database is loaded (e.g. no design description exists) the users will be asked to describe their design either by generating a new description or by loading an existing one. Similarly, if no knowledge is loaded the users will be asked to load one or more such bases. The process will then pose the necessary queries to each such knowledge base to activate the verification process. Alternatively, users may load knowledge bases and pose selective verification queries to IPEXCAD.

The expert system used is EXBUILD (Balachandran and Rosenman, 1990), a hybrid expert system development tool written in C. EXBUILD uses both rule-based and frame-based representations.

In this project we have chosen two of the popular CAD packages, namely EAGLE (Carbs Ltd, 1985) and AutoCAD (Autodesk, 1988). AutoCAD provides two ways to access and manipulate databases. The first is with AutoLISP, a programming language within AutoCAD that lets us write programs that will manage and manipulate graphical and non-graphical data. The second way is to extract the desired data from the DXF file using other external programs.

EAGLE is a general purpose three-dimensional CAD modelling system which includes database management facilities. EAGLE is capable of exchanging graphical and non-graphical data through a vast array of data-exchange formats including DXF and IGES file formats.

The design prototype system developed is PROTOKIT. PROTOKIT contains the design prototype manager, the design prototype base and the design prototype engine. The design prototype manager allows the user to create, modify, display, print or delete prototypes and instances. The design prototype system is capable of accessing information through inheritance using links such as 'a_type_of', 'an_instance_of', 'an_element_of' and 'same_as'.

The KBI performs the necessary interpretations on EXBUILD's knowledge bases and creates the knowledge base objects according to the format of PROTOKIT.

Currently a CAD database interpreter is being developed that performs the necessary mappings between AutoCAD's database and PROTOKIT.

The working memory contains all the instances generated by the CAD database interpreter during a session. The expert system is capable of accessing information from the working memory through the knowledge-base interpreter. It is also capable to post inferred information to the appropriate instances in the working memory.

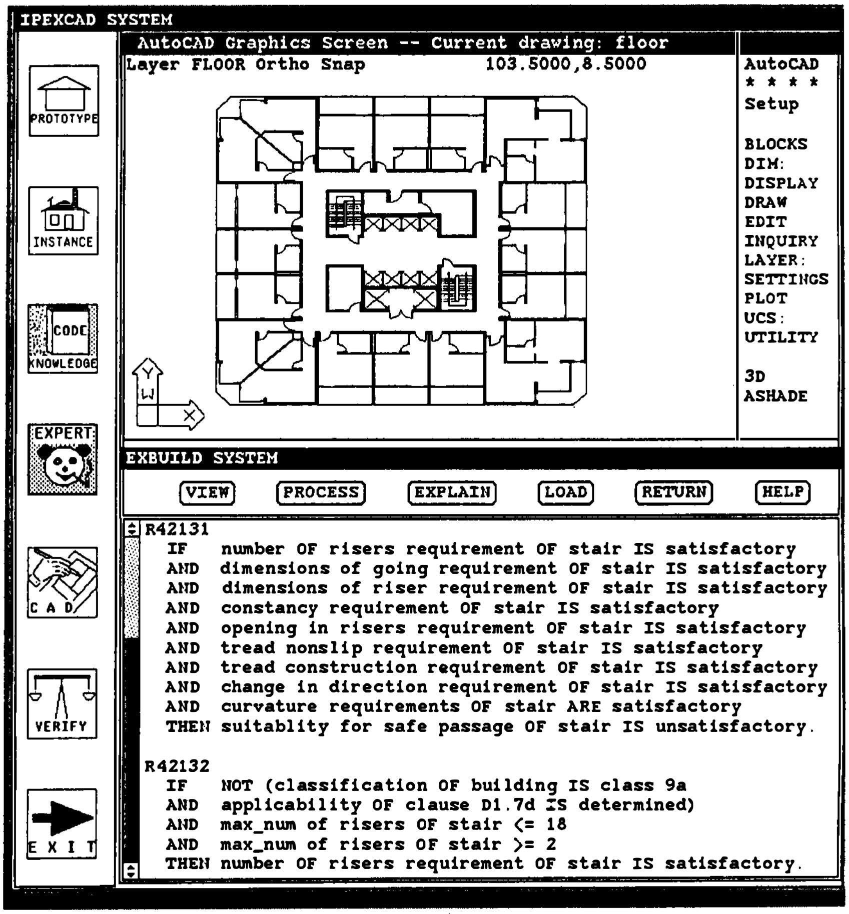
APPLICATION TO BUILDING DESIGN

Building Design Representation

The AutoCAD drawing system is used to model and represent building designs in this example. Although AutoCAD is primarily used for 2-D drafting it provides features that are useful in modeling objects. A set of associated objects can be placed on layers or grouped together to form complex objects that can be manipulated as a whole. AutoCAD stores the locations, sizes and colours of the objects we draw for subsequent retrieval, analysis and manipulation. For example, objects such as walls, floors, stairs, windows and doors can be explicitly modelled and manipulated. The top window in Figure 3 shows a plan of an office building modeled using AutoCAD.

Building Code Representation

As noted previously, the requirements of the building code are represented in the form of IF/THEN rules. The major advantage of using the rule-based approach is that it allows us to keep the representation of any particular requirement at the same high level of abstraction appearing in the original code. The representation of a part of the BCA Code, namely the Clause D2.13 is shown in the bottom window of Figure 3.



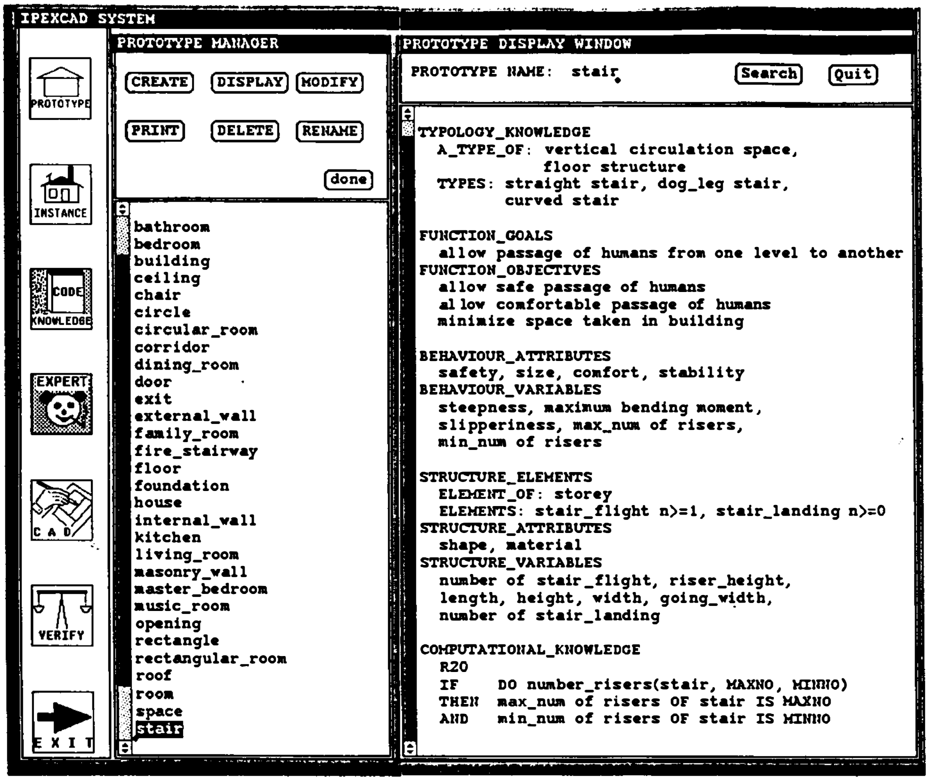
```
R42131
  IF    number OF risers requirement OF stair IS satisfactory
  AND   dimensions of going requirement OF stair IS satisfactory
  AND   dimensions of riser requirement OF stair IS satisfactory
  AND   constancy requirement OF stair IS satisfactory
  AND   opening in risers requirement OF stair IS satisfactory
  AND   tread nonslip requirement OF stair IS satisfactory
  AND   tread construction requirement OF stair IS satisfactory
  AND   change in direction requirement OF stair IS satisfactory
  AND   curvature requirements OF stair ARE satisfactory
  THEN  suitablity for safe passage OF stair IS unsatisfactory.

R42132
  IF    NOT (classification OF building IS class 9a
  AND   applicability OF clause D1.7d IS determined)
  AND   max_num of risers OF stair <= 18
  AND   max_num of risers OF stair >= 2
  THEN  number OF risers requirement OF stair IS satisfactory.
```

Figure 3. Example of a design description and verification knowledge

The rules shown are concerned with suitability and safety of stairways and deal with verifying the compliance with the given requirements as to provide safe passage.

Domain Knowledge Representation

As described previously, the domain knowledge is represented as design prototypes. The process of representing a design domain using the design prototype schema requires a large number of design prototypes in a network of hierarchies. Figure 4 illustrates the interface of the PROTOKIT system and the right window displays the prototype, named 'stair'.



```
bathroom
bedroom
building
ceiling
chair
circle
circular_room
corridor
dining_room
door
exit
external_wall
family_room
fire_stairway
floor
foundation
house
internal_wall
kitchen
living_room
masonry_wall
master_bedroom
music_room
opening
rectangle
rectangular_room
roof
room
space
stair
```

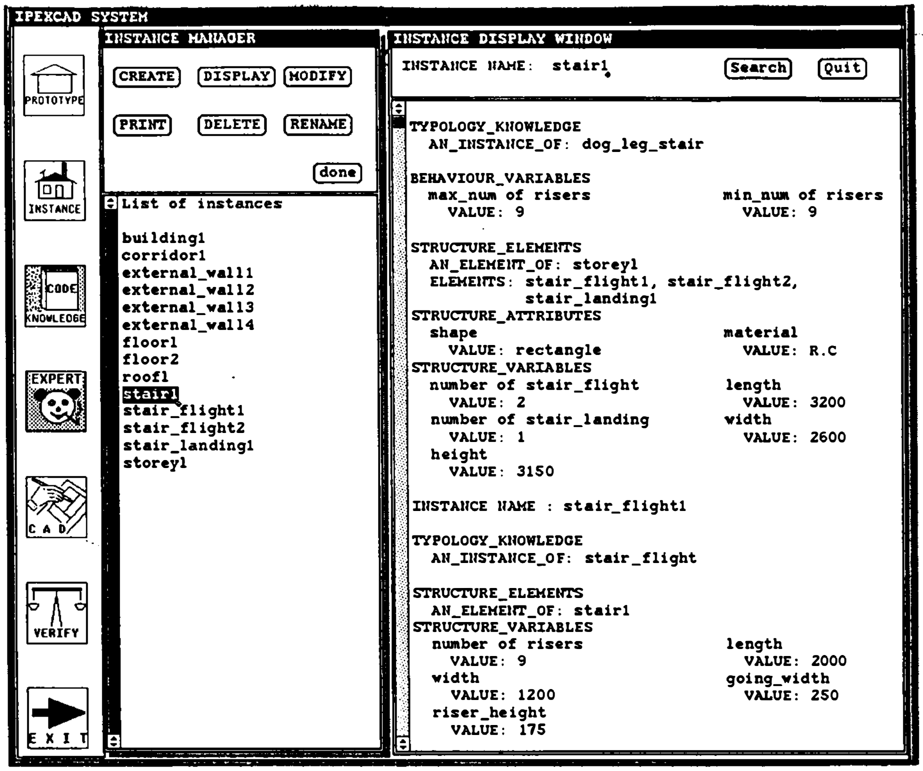
```
PROTOTYPE NAME:  stair

TYPOLOGY_KNOWLEDGE
  A_TYPE_OF: vertical circulation space,
             floor structure
  TYPES: straight stair, dog_leg stair,
         curved stair

FUNCTION_GOALS
  allow passage of humans from one level to another
FUNCTION_OBJECTIVES
  allow safe passage of humans
  allow comfortable passage of humans
  minimize space taken in building

BEHAVIOUR_ATTRIBUTES
  safety, size, comfort, stability
BEHAVIOUR_VARIABLES
  steepness, maximum bending moment,
  slipperiness, max_num of risers,
  min_num of risers

STRUCTURE_ELEMENTS
  ELEMENT_OF: storey
  ELEMENTS: stair_flight n>=1, stair_landing n>=0
STRUCTURE_ATTRIBUTES
  shape, material
STRUCTURE_VARIABLES
  number of stair_flight, riser_height,
  length, height, width, going_width,
  number of stair_landing

COMPUTATIONAL_KNOWLEDGE
  R20
  IF     DO number_risers(stair, MAXNO, MINNO)
  THEN   max_num of risers OF stair IS MAXNO
  AND    min_num of risers OF stair IS MINNO
```

Figure 4. Example of domain knowledge represented as prototypes

Verification Process

We will now follow through the verification process for the example given in Figure 3. The rules are structured so that, firstly, the applicability of each clause is determined and, if found applicable, then its compliance is checked. A part of the verification knowledge used to check stairs for their compliance is shown in Figure 3. Rule R42131 states all the conditions necessary for a stair to allow safe passage while rule R42132 states the conditions necessary for the first condition of rule R42131, i.e. the requirement on the number of risers, to be satisfied (for certain buildings).

According to rule R42132, given that our building is a Class 5 building and the first condition is satisfied, each stair must have a maximum number of risers of 18 and a minimum number of risers of 2. The design prototype description of a stair, illustrated in Figure 4, shows that the values for the behaviour variables 'max_num of risers' and 'min_num of risers' can be determined from the computational knowledge in the design prototype, namely rule R20. Note that the computational knowledge of the design prototypes in the form of rules should not be confused with the rules of the expert system representing the specific verification knowledge. Rule R20 makes a call to a procedure 'number_risers' which requires a given instance of a stair and returns the maximum and minimum number of risers for that instance. In this example, the stair under question is the stair instance 'stair1' as shown in Figure 5. Figure 5 also shows the instance 'stair_flight1' which is an element of 'stair1'. The information regarding the number of risers in 'stair_flight1', namely 9, is derived from the graphical database of the CAD system. The procedure 'number_risers' uses this information and instantiates the values of 9 and 9 for the maximum and minimum number of risers of 'stair1', since all the stair flights are equal. The expert system accesses the required information from the appropriate instances and updates the instances with all the facts inferred during the verification process.



```
List of instances

building1
corridor1
external_wall1
external_wall2
external_wall3
external_wall4
floor1
floor2
roof1
stair1
stair_flight1
stair_flight2
stair_landing1
storey1
```

```
INSTANCE NAME:  stair1

TYPOLOGY_KNOWLEDGE
  AN_INSTANCE_OF: dog_leg_stair

BEHAVIOUR_VARIABLES
  max_num of risers                     min_num of risers
    VALUE: 9                              VALUE: 9

STRUCTURE_ELEMENTS
  AN_ELEMENT_OF: storey1
  ELEMENTS: stair_flight1, stair_flight2,
            stair_landing1
STRUCTURE_ATTRIBUTES
  shape                                 material
    VALUE: rectangle                      VALUE: R.C
STRUCTURE_VARIABLES
  number of stair_flight                length
    VALUE: 2                              VALUE: 3200
  number of stair_landing               width
    VALUE: 1                              VALUE: 2600
  height
    VALUE: 3150

INSTANCE NAME : stair_flight1

TYPOLOGY_KNOWLEDGE
  AN_INSTANCE_OF: stair_flight

STRUCTURE_ELEMENTS
  AN_ELEMENT_OF: stair1
STRUCTURE_VARIABLES
  number of risers                      length
    VALUE: 9                              VALUE: 2000
  width                                 going_width
    VALUE: 1200                           VALUE: 250
  riser_height
    VALUE: 175
```

Figure 5. Example of stair instances generated during the verification process

DISCUSSION AND CONCLUSIONS

This paper has described an approach to the development of a knowledge-based system capable of checking a design through its description in a CAD database for conformance with some requirements. Conformance with the building code, BCA Code has been used as an example of this process. Design prototypes have been used to represent the domain knowledge necessary to provide the necessary interpretations from semantics to syntax. The expert system, the domain knowledge and the CAD system are independent systems using knowledge representations suitable to their application. Communication is through the knowledge base interpreter and the CAD database interpreter. The domain knowledge is central in providing the unifying model through which communication is possible. The role of the knowledge engineer is to provide the specific CAD database interpreter depending on the particular CAD system used and providing the necessary interpretations to allow the expert system to access the relevant domain knowledge.

This flexible coupling of the elements in the system allows for a wide variety of CAD systems and expert knowledge applications to be used. Though this paper has used the BCA Code as an example of verification knowledge and AutoCAD as an example of a CAD system, in general, various other knowledge bases and CAD systems can be used. For each such different knowledge base and CAD system the task of integration will be to provide the appropriate interpreters.

While this paper advocates the production of a heterogeneous system, it is envisaged that in the future such systems will become more homogeneous as CAD databases become more standardised and especially as CAD systems begin to contain more domain knowledge, i.e. become more 'intelligent'.

ACKNOWLEDGEMENTS

This work is supported by continuing grants from the Australian Research Council. EAGLE is provided by courtesy of CADCOM Pty Ltd, and AutoCAD by courtesy of AutoDesk (Australia) Pty Ltd.

REFERENCES

AUBRCC (1988). *Building Code of Australia*, Australian Uniform Building Regulations Co-ordinating Council, Department of Industry, Technology and Commerce, Canberra.

Autodesk (1988). *AutoCAD Reference Manual*, Autodesk, Inc, Sausalito, CA.

Balachandran, M. and Gero, J.S. (1988). A model for knowledge-based graphical interfaces, *in* J.S. Gero and R. Stanton (eds), *Artificial Intelligence Developments and Applications*, North-Holland, Amsterdam, pp.147-163.

Balachandran, M. and Rosenman, M.A. (1990) *EXBUILD, Expert System Shell Users Manual*, Design Computing Unit, Department of Architectural and Design Science, University of Sydney, Australia.

Bobrow, D. G., Mittal, S. and Stefik, M. J. (1986). Expert systems: Perils and promise, *Communications of the ACM*, **29**(9): 880-894.

Carbs Ltd (1985). *EAGLE, 3D Modelling System Command Reference Manual*, Clwyd, UK.

Coyne, R. D., Rosenman, M. A., Radford, A. D., Balachandran, M. and Gero, J. S. (1990).*Knowledge-Based Design Systems*, Addison-Wesley, Reading, Mass.

Dym, C.L, Henchey, R.P., Delis, E.A. and Gonick, S. (1988). A knowledge-based system for automated architectural code-checking, *Computer-Aided Design*, **20**(3):137-145.

Fenves, S. J., Flemming, U., Hendrickson, C., Maher, M. L. and Schmitt, G. (1990). Integrated software environment for building design and construction, *Computer-Aided Design*, **22**(1):27-36.

Garrett, J. H., Jnr and Fenves, S. J. (1987). A knowledge-based standards processor for structural component design, *Engineering with Computers*, **2**:219-238.

Gero, J. S.(ed.) (1987a). *Expert Systems in Computer-Aided Design*, North-Holland, Amsterdam.

Gero, J. S. (1987b). Prototypes: a new schema for knowledge-based design, *Working Paper*, Architectural Computing Unit, Department of Architectural Science, University of sydney.

Gero, J. S. and Rosenman, M. A. (1989). A conceptual framework for knowledge-based design research at Sydney University's Design Computing Unit, *in* J.S.Gero (ed), *Artificial Intelligence in Design*, CMP/Springer Verlag, Southampton and Berlin, pp.361-380.

Graphisoft (1989). *ArchiCAD, Version 3.4, User's Manual*, Graphisoft.

Hoskins, E.M. (1977). The OXSYS system, *in* J.S. Gero (ed), *Computer Applications in Architecture*, Applied Science, London, pp.343-391.

Jain, D. and Maher, M.L. (1987). Combining expert systems and CAD techniques, *in* J.S. Gero and R. Stanton (eds), *Artificial Intelligence Developments and Applications*, North-Holland, Amsterdam, pp.65-81.

Kostem, E and Maher, M.L. (eds) (1986). *Expert Systems in Civil Engineering*, ASCE, New York.

Lee, K. (1977). The ARK-2 system, *in* J.S. Gero (ed), *Computer Applications in Architecture*, Applied Science Publications, London, pp.312-342.

Nnaji, B. O. and Kang, T. S. (1990). Interpretation of CAD models through neutral geometric knowledge, *AI EDAM*, 4(1):15-45.

Rehak, D. R. and Howard, H. C. (1985). Interfacing expert systems with design databases in integrated CAD systems, *Computer-Aided Design* **17**(9): 443-454.

Rehak, D.R., Howard, H.C. and Sriram, D. (1985). Architecture of an integrated knowledge-based environment for structural engineering applications, *in* J.S.Gero (ed), *Knowledge Engineering in Computer-Aided Design*, North Holland, Amsterdam, pp.89-117.

Rosenman, M. A. (1990). Application of expert systems to building design analysis and evaluation, *Building and Environment*, **25**:(3):221-233.

Rosenman, M.A., Gero, J.S. and Oxman, R. (1986a). An expert system for design codes and design rules, *in* D. Sriram and R. Adey (eds) , *Applications of Artificial Intelligence in Engineering Problems*, Springer-Verlag, Berlin, pp. 745-758.

Rosenman, M.A., Manago, C and Gero, J.S. (1986b). A model-based expert system shell, *IAAI'86*, pp.c:1:1-15.

Rosenman, M.A., Balachandran, M. and Gero, J.S. (1989). Prototype-based expert systems, *in* Gero J. S. and Sudweeks, F. (eds), *Expert Systems in Engineering, Architecture and Construction*, University of Sydney, Sydney, pp.179-200.

Sharpe, R., Marksjo, B. and Ho, F. (1989). Wind loading and building code expert systems, *in* Gero J. S. and Sudweeks, F. (eds), *Expert Systems in Engineering, Architecture and Construction*, University of Sydney, pp.223-242.

Tyugu, E. (1987). Merging conceptual and expert knowledge in CAD, *in* J.S. Gero, (ed.) *Expert Systems in Computer-Aided Design*, North-Holland, Amsterdam,pp.423-431.

Waterman, D. (1986). *A Guide to Expert Systems*, Addison-Wesley, Reading, MA.

Balachandran, M., Rosenman, M. A. and Gero, J. S. (1991). A knowledge-based approach to the automatic verification of designs from CAD databases, *in* J. S. Gero (ed.), *Artificial Intelligence in Design '91*, Butterworth-Heinemann, Oxford, pp. 757-781