

*Presented at the Fourth New Zealand Conference on Expert Systems, NZIS 90,  
Massey University, Palmerston North, November 1990*

# **An Augmented Frame Representation for Building Designs**

**Robert Amor, Lindsay Groves**  
Computer Science Department  
Victoria University of Wellington  
Wellington  
New Zealand

**Mike Donn**  
School of Architecture  
Victoria University of Wellington  
Wellington  
New Zealand

## **Abstract**

We are developing an intelligent interface for the many design evaluation programs becoming available in architecture. As the central part of this project we have developed a model of buildings, capable of holding all the information needed to describe a building to a number of design tools commonly used by architects. The model also contains knowledge about components in a building and their inter-relationships. This paper examines the applicability of frame systems for representing the different types of information and structures that exist in a building, including declarative knowledge about buildings and heuristics about building design. An extension to the frames system is described which allows the designer to simultaneously model, compare and contrast many design alternatives for a single building. A frame system of this type has been devised for this project. Its low level structure and the methods used to encapsulate declarative and heuristic knowledge about buildings are described. In the conclusion the fit between our augmented frames system models and the knowledge required to represent buildings is evaluated.

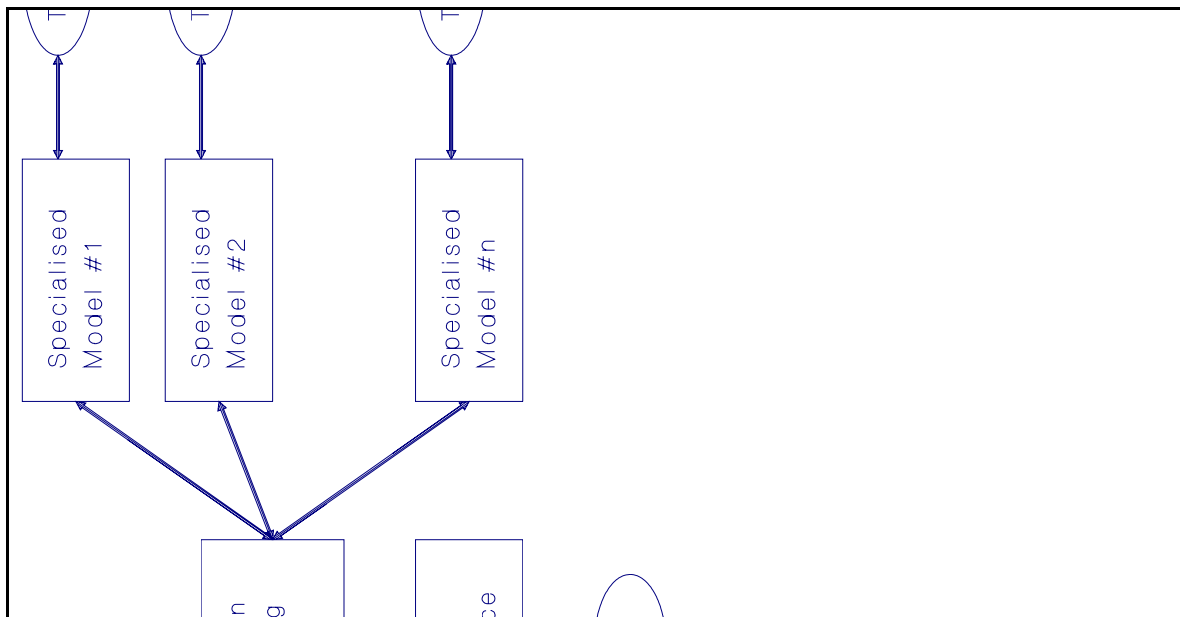
## **1. Introduction**

In a world that expects its buildings to be one-off designs with increasing performance and reliability, the building design team is under great pressure to use as many of the available design analysis tools as possible. Numerous CAD systems, simulation programs, and, more recently, expert systems are being marketed to meet this demand. Each of these tools requires a detailed description of the building or structure being modeled. Constructing a description of a building for entry into a simulation program from existing plans can take several days. When the same building is to be analyzed by several different expert systems or simulators, similar base data have to be entered into each tool, with considerable duplication of effort and many opportunities for error.

Each tool is generally designed to perform one particular type of analysis or simulation. Few are designed with a view to interfacing with other tools. This leads to a situation where many tools exist in isolation and there is little or no transfer of information between tools in different areas. The exception to this is in CAD systems, where it is now realized that some sort of exchange protocol is needed to allow data to be moved from one system to another. This has led to several different protocols from the major vendors, and only recently has an ISO standard been defined for the exchange of data between CAD systems [IGES86].

Further problems arise from the "black box" nature of many analytical programs. While it is relatively simple to encapsulate the mathematical expertise of a structural engineer in a program to calculate the load on a beam, it is far from simple to encode an engineer's expertise in the application of the mathematics. This means that a naive user requires assistance, not only to interpret the results, but also to enter the initial building data accurately.

The problems outlined above suggest that there is a need for a system that allows a user to create building descriptions for different tools without having to learn the command syntax and other idiosyncrasies of each tool. The user should be able to transfer relevant data about a building from one system to another, instead of having to duplicate the effort. This situation is illustrated in 1, where all tools can exchange information through a common building model. These goals seem to be achievable with the current state of technology and we intend to show that such a system can be created.



**Figure 1.** Design tools: The future

Internationally there has been considerable interest in projects of this nature, and we have gained from the experiences of [BJOR89A], [BJOR89B], [CLAR89], [HOWA86], [LEVA88], [REHA84] and [SELK86].

The system we are developing will allow a single description of a building design to be constructed and used to provide the necessary data for a variety of design tools. To do this, we need a framework for representing a building design that can accommodate all the information required by the tools we are considering, and a method to represent this information. In this paper

we look at what types of information are needed to represent a building, we look at why frames were chosen, the structure of the frames system and their facets and the addition of worlds to the frames system.

## 2. What to Represent

To help us decide which representation to use for our building model we looked at the types of information necessary to describe a building, and determined which of these could be implemented by the various representations.

### 2.1 Information and Knowledge Classes

We have identified nine classes of information that are necessary to represent the information about a building. They fall into the following categories:

**Structured objects:** Describe the properties of an object in terms of geometric data, numerical data, codes and text, e.g. some of the attributes of the class 'wall' will be 'shape', 'height', 'width', 'total thickness', 'total conductance' and 'solar transfer'.

**Relationships:** Describe the connections between different objects comprising the building. Indicate which objects are sub-components of a higher order object, e.g. the sub-components of 'window #35' may be 'frame #17' with 'pane #6' and 'pane #7'.

**Dependencies:** Describe the conditions under which certain objects can exist, e.g. windows can only exist within a wall, a roof, or a door.

**Hierarchies:** Describe hierarchically structured domains.

**Classes and Inheritance:** Describe generic classes of objects, and under them sub-classes which inherit all the knowledge about the generic class but also have additional knowledge specific to the subclass, e.g. have a generic class 'wall' with subclasses 'interior wall', 'exterior wall', 'trombe wall', 'partition', etc.

**Formulas:** Calculate an attribute's value from existing attributes using standard procedures, e.g. calculate total thermal resistance of a wall from the resistances of the individual wall layers.

**Building Codes and Standards:** Incorporate the relevant constraints imposed by local codes and standards of the country and the region, e.g. if a door is a fire door, its width must be greater than one metre.

**Conventions:** Represent personal preferences or local conventions for design. These are traits that may have no logical explanation, but are the way in which people design, e.g. if the building is a sky-scraper then let it have a flat roof.

**Principles:** The first principles of physical properties, so if there is no formula for calculating an attribute's value directly then it may be possible to derive the value from first principles, e.g. calculate the total thermal resistance of a wall from the properties of the materials comprising the layers of the wall.

On top of this there are considerations of the design process, which can be affected by the method in which this information is represented. There are two considerations that should be satisfied by a representation, these are:

**Free Design:** The representation should not influence the design process in any way. Designers must be free to work at any level they choose, and not forced to design in a predefined manner.

**Alternatives:** One of the major functions needed in design evaluation is the ability to look at several variations to the initial design. This means there must be some mechanism to signify design alternatives which replace a certain attribute in the system.

### 3. Why Frames?

With a clear idea about what types of information we have to represent it was necessary to examine the different methods commonly used to represent information about buildings, to ascertain which was most suited to the task. To do this we examined the strengths and weaknesses of the following commonly used representation methods: Computer Aided Design (CAD) packages; database systems; Hypercard; Object-Oriented database systems; and Frame systems ([AUTO88], [ULLM82], [CORK87], [GOOD87], [KIMW89], [MINS75], [WINS77]). This comparison resulted in the chart shown in 2 which indicates the capabilities of each of the representations.

|                         | CAD  | DBMS | Hypercard | OO-DBMS |
|-------------------------|------|------|-----------|---------|
| Objects                 | Part | Yes  | Yes       | Yes     |
| Links                   |      | Yes  | Yes       | Yes     |
| Attributes              |      | Yes  | Yes       | Yes     |
| Methods                 |      | Yes  | Yes       | Yes     |
| Classes and Inheritance |      |      | Part      | Yes     |
| Code and Standards      |      |      | Yes       | Yes     |
| Names                   |      |      |           | Part    |
| Dimensions              |      |      |           |         |
| Colors                  | Yes  |      |           | Yes     |
| Views                   | Yes  |      |           |         |

**Figure 2.** Representations capabilities

As can be seen clearly from 2 there were only two contenders for representing buildings: object-oriented database systems and frame systems. The deciding factors in the choice of the

frames representation were the availability of software (the frames system is fairly easily implemented in Prolog), the slight edge in what it could represent and the freedom for expansion or modification offered by a system implemented on site.

### **3.1 Frame Structure**

Having made the decision of the representation method we looked at what relationships needed to be implemented in our frames system. In the implementation of our system, we define nine separate sets of attributes and relationships that hold true, and split all attributes among these levels:

**Object-Cardinality:** Defines the number of this object that can exist in the system. Used mainly in tools' frames to record restrictions on objects posed by the maximum number of objects the tool can handle, e.g. maximum number of walls allowed in a particular design tool is 40.

**Comment:** Describes the particular class or object being represented. Used to provide help to the user about the classes that are defined. When used with instances of an object, it can describe the general properties of the instance, e.g. aluminium sliding door with twin inset glass panels.

**Values:** The attributes describing the physical properties of this class or object, e.g. door height, door width, door thickness, door's thermal resistance. These attributes do not contain relationships between objects, as defined previously in the frames system.

**Parts:** The constituent parts of this object, e.g. the parts of WALL#29 are DOOR#15, WINDOW#2, and WINDOW#17.

**Ako:** Describes the hierarchical structure among classes of objects. Each class in the ako chain inherits the attributes of all classes above it and can introduce additional attributes. This is useful where common attributes are shared by several classes of objects. A good example of this hierarchy arises in the description of walls, where the classes further down the hierarchy from the generic wall are distinct wall types (e.g. exterior wall, interior wall, structural wall, trombe wall, underground wall, and partition; all of which inherit the general attributes of a wall, such as position, height, width, tilt, surface coefficient, and solar absorptance fraction.

**Subclasses:** Defines all the specialisations of a class, this is used as a programming convenience when asserting some property of a class. This is the inverse of the Ako relationship, e.g. the subclasses of the class wall might be trombe wall, exterior wall, interior wall.

**Is-a:** Describes the relationship between classes of objects and instances of those classes. An instance of a class inherits the attributes of that class. The values of inherited attributes are also inherited, except where these are explicitly overridden.

**Part-of:** Describes the relationship between an object (or class of objects) and its components. These links correspond to the links in an abstraction hierarchy e.g. a window and a door are part-of a wall.

**Connected-to:** Describes connections between objects at the same level in the abstraction hierarchy, e.g. a wall is connected-to two spaces, one of which may be the exterior.

### 3.2 Facet Structure

When deciding on the attributes needed for the various relationships defined in the frames system, we examined what was needed to describe each attribute. Attached to each attribute defined in the building model are a set of facets that describe the properties of the attribute. These properties can be used to determine the values or restrictions on the attribute. The facets defined for our frame system are:

**Value:** The actual value this attribute takes, described in the same way as the other facets to make the description of the frame and extraction of information as easy as possible.

**Type:** The type of the attribute, e.g. strings, reals, integers or lists of any of the above.

**Description:** The function of this attribute for the particular object. This value is used mainly for user help, when the user does not know what part the attribute plays in describing the object.

**Asserted-by:** Defines where the value for this attribute came from. This lets us record whether the value came from the user, is a default value from a specific tool, a default value from the building representation, or calculated from some procedure.

**Default:** The default value for the attribute. This is used to determine the value of the attribute if no value is supplied and the user requests the attribute's value, e.g. the default value for the attribute 'height' for the class 'doors' might be 2.2 meters.

**Units:** The units in which the attribute is represented. For attributes in the common building representation, these are SI units. For attributes in the tool's frames, the unit is whatever unit the tool uses to represent the attribute in its representation. The mapping system provides automatic conversion of values between the units in the common building representation and the tool's representation.

**Constraints:** The constraints on the values that can be taken by the attribute. Constraints can be of the form:

*In-Range:* The values must lie within the defined range.

*Member-Of:* The values must be members of the defined set.

*Factor-Of:* The values must be divisible by the defined value.

*Instance-Of:* The values must be instances of the given classes.

*Name-Constraint:* The value must be of a certain form.

*Name-Restriction:* The value must not be of the form described.

*In-Library:* The value must exist in the named library.

**Attribute-Cardinality:** The maximum and minimum number of values this attribute can take. This facet doubles as a required facet by specifying the minimum cardinality to be greater

than zero. This is mainly used in tool's frames to specify restrictions on object numbers imposed by the tool.

#### 4. Implementation

From the description languages for the tools we studied, and from constructing a data base of attributes of the objects in the various tools, it was apparent that we should describe a building in a hierarchical manner, as a collection of systems, each consisting of a number of subsystems, etc. This hierarchy should provide classes of objects, corresponding to the systems and subsystems in buildings, each of which is described by a set of attributes.



**Figure 3.** Abstraction Hierarchy

We have defined an abstraction hierarchy that breaks the description of a building into five levels: building, system, subsystem, object, and part (see 3).

**Building level:** This is the top level of the data structure. There is only one building object for each building. It contains only building-specific data, such as the location, orientation, number of floors and gross area, height, etc.



**System level:** This level breaks up a building into its main functionally distinct systems. For our purposes these are: the spaces in a building, the structure, Heating, Ventilating and Air-Conditioning (HVAC) system, lighting system, lifts, power, communications, etc.

**Subsystem level:** This level is used to break a system into its major functional components. Thus far, we have used this level mainly in our description of the HVAC system that is broken up into plant, distribution, and terminal. These are further subdivided to represent all the major functions of the components in an HVAC system. For example, the plant subsystem is further divided into heat production, heat removal, humidity control, and storage.

**Object level:** This level represents each physical object in its own generic class. All definable objects are listed on this level. The types of objects are walls, doors, windows, columns, boilers, chillers, fans, diffusers etc.

**Part level:** This level is used to represent the constituent parts of components at the object level. For instance, a window at the object level is represented as a frame and panes at the part level. The same applies for some plant equipment, e.g. fans, condensers, and cooling coils are parts of a chiller.

This hierarchy turns out to be very similar to the one proposed for the RATAS project in Finland ([BJOR89A], [BJOR89B], [ENKO88], [HANN87]) although their hierarchy represents different objects at different levels due to the goals of their project.


With the structures mapped out above we have constructed a building model that contains just under one thousand attributes spread over more than eighty classes of objects.

## 5. Worlds

The only concern we had with the frames system described above was its inability to represent design alternatives, which is one of the major considerations for a building representation. To solve this problem we have implemented a system very similar to the worlds defined in [KEES88].

The worlds system takes the concept of inheritance one step further than the inheritance of attributes between objects, to the inheritance of whole sets of objects, in this case comprising a whole building. The worlds system is based on a hierarchical tree of worlds, with each world inheriting all the modifications made further up the tree right back to the root world which is the original description of the building being examined (see 4).

The example in 4 shows four worlds. The base world (world 1) describes the whole building, which for the purposes of this example has a coal fired boiler and single glazed windows. After analysing the base building, the designer might like to examine the effects of two changes. The first (world 2), replaces the coal boiler with a gas fired boiler. The second (world 3), makes all exterior windows double glazed and replaces the coal fired boiler with an electric boiler. After further analysis of these two modifications the designer may decide to test a gas fired boiler instead of the electric boiler in world 3 (world 4). Worlds 2 through 4 demonstrate the power of the worlds system. With very little effort the designer is able to construct a new building based on a base design without duplication of any objects except the changed systems.



**Figure 4.** Worlds

The actual implementation of the worlds system required remarkably little effort. Representing an attribute's value for different worlds required only a small modification to the value facet of the frames structure. By changing the value of the value facet to a list of values each denoted by a world number, any number of worlds can be accommodated. The problem of adding a new object in a higher level world requires that the attributes of the new object are all labelled for that world level, and there is no root world value to stop other worlds inheriting the object. An example of the workings of the worlds system is the modification of a value in a value facet. When an attribute's value is to be changed, the system searches through the list of values denoted by world numbers. If it finds a value for the current world, then it is replaced with the new value. If there is no value for the current world, then the modified value is entered into the value list for the current world. The only inconvenience this representation presents is when we want to remove an object from the current world. To do this, the value slot of all pointers to this object must be modified to a slot with no pointers, or a list of all possible pointers to objects except the one not required.

The inheritance path of the different worlds was represented by constructing a hierarchical tree detailing the level of the various worlds for a specific building. Using this structure when looking for a value for an attribute, the frames system will look for a value for the current world. If there is no value for that specific world it will recursively find the previous world in the hierarchy tree and search for a value in that world, until it reaches the root world. If there is no value in the root world then it will fail.

As an example, in 4, if the user requests information about the exterior windows while working in world 4, then the system will look for an exterior window in world 4, as there are no window definitions in world 4 it will recurse up to world 3. Here it will find a modification to the exterior window definition changing them to double glazed windows. Similarly, if the user looks for

information about doors in world 4 then the system will recurse right up to the root world, world 1, before it finds any door definitions.

## **6. Conclusion**

This paper shows how the use of a frames system augmented with the worlds concept provides a powerful yet flexible representation method for use in representing the variety of knowledge about a building. The building model building that has been constructed has proved capable of holding the majority of information needed by a range of design tools from many disparate areas of design. This building model shows that it is possible to construct a useable system that can model more than just the plain attributes used to describe buildings in product data models. The added power of the frames approach over other methods of modelling buildings currently in use, seems to indicate that more effort should be put into exploring this avenue of research.

## References

- [AUTO88]AutoCAD (1988). Autodesk, Inc, Sausalito, CA 94965
- [BJOR89A]Björk, B-C. and Penttilä, H. (1989). A scenario for the development and implementation of a building product model standard, Current research and development in integrated design, Construction and Facility Management, CIFE, Stanford University, California, 28-29 March, 18p.
- [BJOR89B]Björk, B-C. (1989). Basic structure of a proposed building product model, Computer-Aided Design, 21(2), March, pp. 71-78.
- [CLAR89]Clarke, J. A., Rutherford, J. H. and MacRandal, D. (1989). An intelligent front-end for computer-aided building design, University of Strathclyde, Scotland.
- [CONK87]Conklin, Jeff (1987). Hypertext: An Introduction and Survey.
- [ENKO88]Enkovaara, E., Salmi, M. and Sarja, A. (1988). RATAS Project - Computer-aided design for construction, Building Information Institute, Helsinki, Finland, 62p.
- [GOOD87]Goodman, Danny (1987). The Complete HyperCard Handbook, Bantam Computer Books.
- [HANN87]Hannus, M. (1987). Object oriented modelling in AEC applications, Proc NBS/DATA Nordic Seminar, Neste generations datasystem i byggebransjen, 24 September, pp. 100-115.
- [HOWA86]Howard, H. C. and Rehak, D. R. (1986). Expert systems and CAD databases, Knowledge engineering and computer modelling in CAD, September, pp 236-248.
- [IGES86]IGES (1986). Initial graphics exchange specification 3.0, National Bureau of Standards, Gaithersburg, MD 20899, USA.
- [KEES88]Knowledge Engineering Environment (KEE), KEE Software Development System User's manual (1988). Unisys.
- [KIMW89]Kim, Won and Lochovsky, Frederick H. (1989). Object-Oriented Concepts, Databases, and Applications, ACM Press, New York.
- [LEVA88]Levary, R. and Lin, C. (1988). Hybrid expert simulation system (HESS), Expert Systems, 5(2), May, pp. 120-129.
- [MINS75]Minsky, A. (1975). A framework for representing knowledge, The psychology of computer vision, McGraw-Hill, pp 211-277.
- [REHA84]Rehak, D. R., Howard, H. C. and Sriram, D. (1984). Architecture of an integrated knowledge based environment for structural engineering applications, Knowledge engineering in computer-aided design, IFIP WG5.2 Conf., Budapest, Hungary, September, pp. 89-117.
- [SELK86]Selkowitz, S. E., Papamichael, K. M. and Wilde, G. M. (1986). A concept for an advanced computer-based building envelope design tool, International Daylighting Conference, November, Long Beach, CA, pp.496-502.
- [ULLM82]Ullman, Jeffrey D. (1982). Principles of Database Systems, Second Edition, Computer Science Press.
- [WINS77]Winston, Patrick Henry (1977). Artificial Intelligence, Addison Wesley.