MODELING AND MANIPULATING BUILDING DESIGNS

R. Burnett[†1], K.F. Li[†], and N. Smith[‡]

† Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
‡ Natalie C. Smith, Architect, Victoria, BC, Canada

ABSTRACT

This paper presents some preliminary work done on the Architectural Design Assistant. The function and structure of the Design Assistant are presented along with some of the motivation behind its development. A building design model, as well as its manipulation are discussed.

The role of the National Building Code of Canada is outlined and the latter sections of the paper present an example of how this code is incorporated into the Design Assistant. One of the most important aspects of the Design Assistant — the use of high level abstraction — is highlighted as part of the example.

1 INTRODUCTION

1.1 The Architectural Design Assistant

The primary objective of this work is the definition of a framework leading to the development of an automated architectural design assistant. The Design Assistant is an intelligent tool that can work in concert with a building designer and perform some of the more mundane tasks that are involved in the design and evaluation of a building.

The Design Assistant that we envision consists of two distinct components: a high level independent design model, and a set of extensible reasoning modules which can manipulate the model developed by the designer in a variety of ways with a variety of goals (see Figure 1).

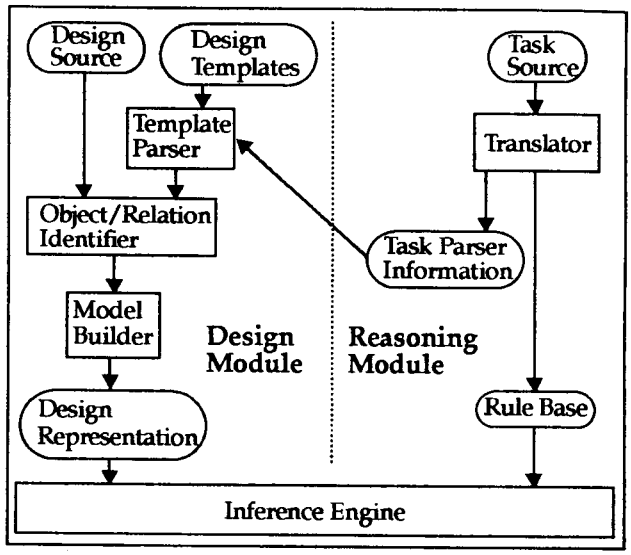

Figure 1 - Design Assistant

The core of the Design Assistant, and the focus of the present work, is a set of generic building design templates which can be instantiated and be manipulated by a designer when producing a design model. The templates specify a set of object classes, properties of the objects and relationships between them.

The classes are general enough to facilitate a number of reasoning tasks. These reasoning tasks are implemented as rule base modules, any number of which can be developed to operate on the design representation. The goal is to have a completely extensible system whereby the rule base and parser information for each specialized task can be developed independently and added into the system as need dictates.

1. R. Burnett is supported in part by NSERC and ASI.

1.2 The National Building Code of Canada

The reasoning tasks that can be performed with the system are quite varied but one of the most relevant is design analysis based on a codified set of constraints. In Canada, the most relevant of such sets of constraints is the National Building Code of Canada (the NBCC) [1]. All provincial building codes are derivations of the NBCC with minor local adaptations.

As a set of guidelines defined by the National Research Council of Canada, the NBCC outlines the minimum provisions respecting the safety of buildings with reference to public health, fire protection, and structural sufficiency. The NBCC is closely followed and referred to by architects, building contractors and engineers throughout the design and construction phases of new building development. Before a building construction permit is issued by the municipal government, the building plan is subject to rigorous checks to determine whether the intended development complies with the NBCC, or the provincial equivalent. The task of this verification is tedious, lengthy, error-prone, and more importantly, not a pleasant job to be performed by a human expert.

However, the NBCC lends itself nicely to translation into an intelligent rule base system. The NBCC, and other similar codes represent the combined knowledge of many domain experts captured in highly structured and available text in the form of constraints. A simple syntactical translation is all that is required to convert these texts into functional rule bases ready to integrate into the Design Assistant.

2 HIGH LEVEL ABSTRACTION

2.1 Relationships at a High Level

The present method for modelling a building design is as a series of architectural drawings or working drawings which convey all information required for the construction in compliance to the building code. This format provides a series of geometric forms and accompanying textual material to represent the various components of a building. This representation, however, requires a great deal of spatial and language interpretation and is highly inappropriate for efficient computer manipulation.

With the increasing popularity of CADD systems designs are starting out as a computer based representation and ideally, the need for extensive interpretation should be reduced. However, today's CADD systems are still mostly drawing tools dealing primarily with the connection of geometric primitives in a graphical representation. Certain higher level information can be attached to the underlying representation but basic geometry still drives the definition of objects [2].

When we look at the NBCC or other reasoning sources, objects are not defined by simple geometries, and basic geometric relationships are not of primary importance. Objects are more often defined by function than spatial information and the relationships between objects that must be satisfied are far more abstract than can be easily dealt with using only a series of X, Y, Z coordinates.

This has tremendous implications in the management of any rule base module in the Design Assistant. Keeping the rule syntax on the same abstract level as the source text makes rule development much faster and, more importantly, greatly eases maintenance if the source text changes. The history of expert systems research suggests that the maintainability of a rule base has a tremendous impact on the success of a system [3]. Therefore, templates in the Design Assistant are quite abstract using objects and relationship definitions as similar as possible to those that might be referenced in the reasoning sources.

2.2 Abstraction in the NBCC

The best way to demonstrate the need for abstract relationship handling is by way of an example. Shown below are two sentences extracted from the Residential Occupancy section of the NBCC. These sentences are typical of the type of objects and relationships that need to be addressed when doing code verification.

> 3.3.4.2.(1) Suites of residential occupancies shall be separated from each other and the remainder of the building by a fire separation having a fire-resistance rating of at least 1 h, except that a $^{3}/_{4}$ h fire-resistance rating is permitted where the fire-resistance rating of the floor assembly is not required to exceed $^{3}/_{4}$ h.[1]
>
> 3.3.4.2.(3) Storage rooms not contained within a suite, for the use of tenants in residential occupancies, shall be

sprinklered and separated from the remainder of the building by a fire separation having a fire-resistance rating of at least 1 h, except that a $^3/_4$ h fire-resistance rating is permitted where the fire-resistance rating of the floor assembly is not required to exceed $^3/_4$ h.[1]

The objects that are of concern in these sentences are: suites, residential occupancies, buildings, fire separations, floor assemblies, and storage rooms. Although any complete design representation must include the geometric specification of these objects, they are not distinguished in this instance by geometry, but by function. Even the relationships such as *separated* and *within*, which have a spatial connotation to them, are very difficult to extract from simple positonal data.

2.2.1 Rule Syntax

Shown in Figure 2 is the translation of sentence 3.3.4.2.(1) of the NBCC into a Design Assistant rule. Represented in italics are the relevant object classes, properties, and relationships that are defined in the design templates.

IF

there exists any objects of class *suite* which are of class *residential_occupancy*

THEN

make all such objects with a *fire_resistance_rating* less than $^3/_4$ h members of class *non_conforming_objects*

make all such objects with a *fire_resistance_rating* between $^3/_4$ h and 1 h which are related to an object of class *floor_assembly* via the *consists_of* relationship which has a *fire_resistance_rating* greater than $^3/_4$ h members of class *questionable_objects*

Figure 2 - Rule for NBCC 3.3.4.2.(1)

Of interest in the syntax of this rule is the high level relationship *consists_of*. Although the relationship is not mentioned explicitly in the original text, the NBCC deals with objects as being composites of other well defined objects. To represent this we have developed the complementary abstract relationships *part_of* and *consists_of*. Although there is a spatial component to these two relationships, the concepts that they represent are not strictly geometric, and to derive these concepts from geometric data requires a tremendous amount of intelligent interpretation. The syntax and semantics of the rule would be almost unmanageable if it had to include information on how to infer the high level relation from geometric data. It is far more effective to represent the objects in question complete with the high level relationships intact.

It can be seen from this rule that aside from stated exceptions, the code sentence translates simply into rule syntax. However, exceptions specified in the code are less restrictive than the clauses that they are meant to modify. In order to avoid making a conclusion regarding an object and then retracting that conclusion, constraints represented by the exception are dealt with first. In the rule shown, the case of the less restrictive $^3/_4$ hour rating is dealt with before the 1 hour case, even though the text does not represent the two constraints in this order.

Figure 3 shows the translation of NBCC sentence 3.3.4.2.(3) into a Design Assistant rule. This sentence is very similar to 3.3.4.2.(1) except there is a high level relationship explicitly mentioned within the text — the *within* relationship. It is this type of explicit relationship that we most want to handle at a high level. By maintaining the abstraction and wording of the rule base source — the NBCC in our example — design and maintenance of the rule base itself becomes a much simpler task. As much as possible, there is a one-to-one mapping between words within the source text, and objects, properties, and relationships within the rule base.

2.2.2 Questionable Objects

An important concept in these rules is the idea of "questionable objects". Again this addresses the ease of construction and maintenance of rule modules within the Design Assistant framework.

In order to fully determine the conformity of the objects of interest in either of these rules we must first determine the conformity of the appropriate floor assembly objects. However, building codes are of such size and complexity that while determining the non-conformance of an object is a relatively simple matter, determining its conformance involves a exhaustive application of the constraints. The simplest thing to do is assume that any related objects

IF

there exists any objects of class *residential_occupancy* and there exists any objects of class *storage_room* which are not related to an object of class *suite* via the *within* relationship

THEN

make all such objects which are not sprinklered members of the class *non_conforming_objects*

make all such objects with a *fire_resistance_rating* less than $^3/_4$ h members of class *non_conforming_objects*

make all such objects with a *fire_resistance_rating* between $^3/_4$ h and 1 h which are related to an object of class *floor_assembly* via the *consists_of* relationship which has a *fire_resistance_rating* greater than $^3/_4$ h members of class *questionable_objects*

Figure 3 - Rule for NBCC 3.3.4.2.(3)

are acceptable as designed (if not they will be flagged by some other rule).

If we assume that any floor assembly of fire resistance rating $^3/_4$ hour or less are valid, then related storage rooms with a rating of $^3/_4$ hour are fine. However, if a storage room's floor assembly is greater than $^3/_4$ hour then the acceptability of the storage room is not known. Either the floor assembly is required to be greater than $^3/_4$ hour and the storage room is not valid, or the floor assembly is over designed and the storage room is perfectly acceptable. Since we do not know which until the entire rule base has been examined, we simply flag the storage room as "questionable" and leave it up to the designer to decide if a design change is necessary.

STATUS AND CONCLUSIONS

The present research involves implementing certain aspects of the Design Assistant. The definition of a generic class hierarchy capable of facilitating building design related reasoning tasks has begun. The two rules shown here have been implemented using Nexpert Object™ and custom C routines. Work is underway to implement the full Residential Occupancy section of the NBCC.

Progress to date suggests that the general, extensible Design Assistant discussed above is viable. Most of the components in Figure 1 are based on well established concepts and there are few implementation problems.

The one aspect of the system that remains at all questionable is the form of the design source and the object identification. Some preliminary work was done on using existing paper or CADD based geometric representations and performing automated feature extraction in order to obtain the higher level objects and relationships [4]. This did not prove to be a feasible solution.

It is far preferable to incorporate the high level abstractions into a CADD system and work at a more abstract level right from the beginning stages of design. This concept has been recognized in the CADD industry and some work is being done with an eye toward more abstract representations [5].

REFERENCES

1. National Research Council of Canada - Associate Committee on the National Building Code, *National Building Code of Canada 1985*, Ottawa, 1985, Second Printing, includes first and second errata.

2. C. M. Eastman, "Architectural CAD: A Ten Year Assessment of the State of the Art", *Computer-Aided Design*, vol. 21, no. 5, June, 1989.

3. S. Polit, "R1 and Beyond: AI Technology Transfer At DEC", *AI Magazine*, vol. V, no. 4, Winter 1985, pp. 76-79.

4. R. Burnett, *Shape Recognition: A symmetry Based Approach*, University of Victoria, January, 1991.

5. T. Smithers, "AI-Based Design Versus Geometry-Based Design or Why Design Cannot Be Supported by Geometry Alone", *Computer-Aided Design*, vol. 21, no. 3, April, 1989.