Research at the Advanced Construction Technology Laboratory

Thomas, J. R.; Cornick, S. M.; Leishman, D. A.; Vanier, D. J.

Research at The Advanced Construction Technology Laboratory

R. Thomas, S.M. Cornick, D.A. Leishman, D.J. Vanier
National Research Council Canada, Institute for Research in Constrution, Advanced Construction Technology Laboratory, 6815, 8 Street North East, Calgary, Alberta, T2E 7H7, Canada

Abstract

The Advanced Construction Technology Laboratory of the National Research Council of Canada was set up to conduct research into the application of computer techniques within the construction industry. This research has focused upon improving the flow and integration of knowledge and information throughout a structure's lifecycle. Central to this process is the explication of models of the construction process as a mechanism for representing and exchanging the knowledge within the industry. Current research in the laboratory focuses on the development of tools and techniques which support the use of building codes and the design process. These tools and techniques include the development of hypertext building code documents, minicode generators, systems for training building inspectors, plans compliance checking systems and techniques to support the reuse of designs.

Keywords: Analogy, Codes, Construction, Compliance, Design, Design Reuse, Hypertext, Knowledge Representation, Modeling, Regulations, SGML, Training Systems.

1 Introduction

The Advanced Construction Technology Laboratory (ACTL) of the National Research Council (NRC) of Canada's role is to support the construction industry by conducting research into improving the flow of knowledge and information throughout a structure's lifecycle. The design and manufacture of artifacts, which include constructed objects such as buildings, is moving towards an integrated and decentralized approach. Construction has always been decentralized, having many different players and companies involved in the process. However, unlike the design and manufacture of artifacts such as automobiles, the construction process has yet to be integrated. If we wish to establish a version of Computer Integrated Manufacturing for the construction industry, Computer Integrated Building (CIB), the focus of current research should be on knowledge and information technologies.

The penalties for the improper transfer of information and the lack of critical or timely knowledge are prohibitive. Lack of understanding, ambiguity, and errors can cause cost overruns and construction delays. Towards this end, the development of knowledge and information models is paramount if appropriate or timely information is to be transferred from a designer to a contractor to a building code official and ultimately to a worker or robot. This requirement for better knowledge and information models has recently been realized by many organizations around the world. The number of research groups working on developing Building Product Models and information interchange standards such as PDES and STEP (Warthen, 1991), and the efforts to computerize regulations in many countries is testimony to recognition of a need for better ways of handling knowledge and information.

ACTL is supporting the trend towards developing knowledge and information based systems for the construction industry in three ways. The first is by supporting and developing information based tools for building regulations which make them more accessible and usable by designers and codes officials. These tools provide designers and building officials with more flexible and efficient means of accessing and applying regulations. They also provide a way of integrating the body of codes and standards into a single information space. This type of research improves the quality of regulations by restructuring them, ensuring the consistency of content and terminology, and guaranteeing completeness. The second way ACTL is supporting the push to better knowledge and information representation is through the generation of Building Product Models. Better knowledge and information models allows the development of information-based products for training people, such as code officials and designers. The third method of support is utilization of the building product models as the basis for design tools such as automated compliance checkers and design generation tools.

This paper outlines the ACT laboratory's current research activities. The first topic deals with the building codes. This section discusses some of the building code related activities including the low level representations used within the code documents. The section continues by showing how these representations facilitate access systems such as hypertext versions of the codes, minicode documents and training systems. Design issues including design evaluation and generation, is the second topic. The need for and the method of producing building product models is explained first. Following this, we outline how these models can be used to produce automated compliance checkers and design reuse systems.

2 Details of Activities at ACT-Lab

2.1 Building Code Related Activities

2.1.1 SGML

Much of the information used by the construction industry is based upon textual and graphical sources. The ability to access and exchange this information has become an important factor in the efficiency of the industry. Although this was first apparent in the manufacturing sector, it is now becoming increasingly important to all industrial sectors. One of the major international standards for the interchange of text based information is the "Standard Generalized Markup Language" (SGML) (ISO 8879-1986). SGML is a metalanguage for describing the internal structure of document types and for tagging the structural elements of documents.

In addition, SGML has also been adopted by many of the world's largest publishers and by many governments around the world, including both the U.S. Government and the European Parliament. SGML is used widely for industrial documentation, particularly in those organisations where there are especially difficult document management problems, such as the military, aerospace and pharmaceutical industries.

NRC has recently converted all of its building code documents into SGML. The Canadian Standards Association is also converting many of its construction related documents to this standard. In part, the stimulus for this activity has been the need to provide the base documents in a format which can be used in support both traditional paper based and electronic formats. The advantages of adopting the SGML approach are that it provides a mechanism for enriching the information content within the document and provides the ability to treat the document as an information database.

A number of the projects currently being undertaking by ACTL involve either the adding of additional information, such as classification information for specific clauses of the documents, or the inter linking of various sections of the document to provide hypertext capabilities. For example, using the functionality provided by SGML, ACTL is able to encode the Provincial variants of the Prototype National Code Documents within a single document structure. In the future, ACTL will also be able to provide the capability of viewing temporal versions of the documents (previous versions), again within the context of the base document.

In addition, ACTL is developing facilities to allow individual users of the documents to add annotations to their own versions of the documents with the ability to display or hide these annotations. This type of facility linked with hypertext and "free text" search capability provides users with the ability to locate information either on the basis of their own annotation or on the documents content.

As part of this activity, ACTL has developed a number of tools for supporting the code developers in their work. These have been based upon the use of hypertext versions of the documents containing sophisticated search capabilities. One important system that ACTL has been working on provides a parallel view of both the English and French versions of the documents. This allows the translators (English to French) to search the documents in either language to see examples of how they previously translated similar terms in other sections of the documents. This supports them in maintaining consistency between and within the two language versions of the documents.

ACTL is also looking at a variety of software tools to increase the usability of the document. These include the use of a more structured form of the document and the use of a "simpler" language style/scope. Concurrently, ACTL is augmenting the search capabilities with the use of imbedded construction thesauruses (Davidson, 1978) within the search mechanisms, these will provide access to synonyms and both broader and narrower terms for the search source.

2.1.2 **Hypertext**

Information sources such as the National Code Documents and their referenced Standards represent a highly complex and interlinked knowledge base. The need to search and navigate around these large information sources require the development of sophisticated search and navigation systems. Traditional texts, both paper based and computer based, are sequential documents in which one moves through them in a linear sequence. In contrast, hypertext documents are designed to be used in a nonsequential fashion and a user is able to follow one of an infinite number of paths through the documents (Nielsen, 1990). Some of the current development of hypertext around the world are based upon the use of SGML to provide structural and informational tagging within the hypertext documents (cf. ISO HyTime Standard for multimedia systems).

Our original work in this area was based upon the use of early hypertext tools which had limited capabilities. This work took place in the mid 80's and investigated the use of hypertext for the presentation of the Canadian Building Code Documents. Towards the end of the 80's, this work moved to more powerful hardware platforms and the use of more generic hypertext tools.

Using the Macintosh HyperCard system, ACTL has been able to develop a number of sophisticated hypertext versions of the Code Documents (cf. Vanier, 1990). These incorporate a number of different methods for accessing information and navigating through highly complex documents. This work has enabled the interlinkage between documents and thereby provides direct access to a number of referenced documents (Standards, etc.).

Techniques utilized by ACTL for the hypertext system include the use of thesauri to support search using synonyms and broader terms. This provides a very powerful search mechanism. In addition, rather than using hard links between sections of the documents, the system locates the target material by parsing the selected text and uses structural information to locate the target information. Providing appropriate user interfaces to these systems is important in ensuring the acceptance and usability of the systems. For example, tables and diagrams are stored both as text and as graphic entities so that the search mechanisms are able to locate information contained within them and yet the graphic files are used for presenting the material in a familiar format. Within the documents, defined terms are visually indicated by the use of italics (as in the paper versions of these documents) and the selection of a term by the user will present the definition of the term in a separate window. Finally, a history mechanism provides the user with the ability to return immediately to any point in the documents that they have previously visited.

Our current hypertext systems are being moved to an IBM platform to ensure wider dissemination of the technology. At the same time ACTL is investigating the development of a version of the Hypertext Code Documents on one of the new pen based computer systems. As further experience is gained with the SGML encoded versions of the Code Documents, enhancements to the Hypertext systems will utilise the tagged information to provide new functionality.

2.1.3 MiniCodes

Many users of the Code Documents are only interested in specific subsets of the total documents. For example, a house builder is only interested in the sections of the documents which refer to residential buildings. When a building official examines a set of building plans she/he only needs those sections of the total building code documents which relate to the task at hand. There have been a number of attempts to develop more specialized paper based systems such as "Housing Codes" but even these are still rather generic documents.

The development of computer-based "MiniCodes" which apply to a closely defined class or prototypical building are currently underway. This system, given a high level description of the building project, will identify the relevant subset (clauses) of the code documents which apply. Although the system is initially designed to deal with a closely defined class of building, further development of the system to produce "Custom Codes" for individual buildings is envisaged. Much of this work is based upon the tagging of the code documents (using SGML) with detailed classification information and the development of a selection engine.

2.1.4 Training Systems

As part of ACTL's objective to support the use of the National Code Documents, a prototype training system for building inspectors has been developed. Many existing traditional computer-based training systems emphasise rote learning of the contents of the code documents and the procedural application of the codes. In contrast , the system developed by ACTL focuses upon training the inspector to identify the existence of a violation first and then provides them with the tools (hypertext versions of the code documents, etc.) to identify the correct citation.

The prototype system focuses upon a typical residential house inspection. As an inspector's task is primarily visual, this system presents a number of scenarios based upon pictorial representations of various stages of the construction process. In each scene, the student is presented with a typical construction detail/scene and asked to identify if it contains any violation of the code. At all stages the system provides the student with access to the necessary information including the house documents (site plan, grade slip, etc.) and hypertext versions of the code documents. If the student indicates that there is a violation of the code in the scene, they are required to indicate which clause of the code which has been violated. Once the student has inspected all the scenes associated with a specific stage of the construction (foundation, framing, fireplace or final inspection), they are required to submit their report for that stage. The report is reviewed by the system and the student is invited to correct any inaccurate code citations. If they fail to identify the presence of a violation, they are invited to re-inspect the relevant scenes. If they are still unable to correctly identify the violations, they are given corrective feedback.

The prototype system provides a correct and incorrect version of a scene for each of the four items at each of the four stages of the construction inspection. On the first occasion that a student uses the system, it randomly presents correct or in-correct versions of the scenes. On subsequent occasions it presents the student with one of the remaining permutations of the correct/in-correct scenes. In the future the system will be extended to provide a greater selection of scenarios and to integrate with an existing text based codes tutoring system. Although this system was initially developed for Building Inspectors, there are plans to extend the system to cover Plans Checkers and to develop a similar system for Fire Inspectors.

2.2 Design Evaluation and Generation

Researchers in the field of design can be roughly divided into two groups: those seeking to understand the design process and those developing tools that make designers more efficient. This later group concentrates on automating routine design tasks and simplifying more complex tasks. At ACTL, there are two on going projects in the design field, one focusing on design evaluation and the other on design generation. Both projects are intended to enhance, automate, and understand portions of the design process.

2.2.1 **Modeling**

The current trend towards computerizing the design and manufacture of artifacts has led to a push towards generating standard ways of representing knowledge. Knowledge includes such things as geometric data, topological data, attribute information, and reasons for design decisions. Product models, which combine various types of knowledge including attribute information and topological data, can also be categorized as knowledge. Buildings are properly viewed as products and consequently are not exempt from the move towards product definition standards. In fact the construction process is so involved and complicated that it is imperative that product models be developed before computerization can significantly impact the construction industry.

2.2.1.1 Building Product Models

Computer integrated building, the equivalent of computer integrated manufacturing for the construction industry, will require the development of building product models in order to provide neutral data exchange. De Waard (1991), for example, describes two models to be used in computer-aided compliance checking, an knowledge model of a building and an knowledge model of a building regulation. In fact it has been shown that building regulations themselves contain an knowledge model of a building (Cornick, Leishman, and Thomas, 1990a).

A code of practice, such as the National Building Code of Canada, can be viewed as comprising a set of models describing the domain of applicability, buildings for example, and a set of constraints that define an envelop within which a design must fall. However, regulations can only partially define a set of models since a considerable amount of background knowledge is required. Only a portion of a document such as the Code defines the domain. The remainder of the National Building Code of Canada is a complicated set of constraints on the domain it describes.

Currently, models are constructed by analysing the document under consideration, the National Building Code of Canada for example, and building models from the text of the document. At an abstract level these models are represented as concepts and relations. Models are simply concepts linked by relations. In addition, concepts are organized in type hierarchies and the relations are defined in a catalogue of relations. Taken together the models, type hierarchies and relations form a semantic net which forms the basis for reasoning about a particular code (Cornick, Leishman, and Thomas, 1990b; Cornick, Leishman, and Thomas, 1991).

2.2.1.2 Knowledge Representation

A new generation of design systems has recently become available to designers, for example, Design++ and Wisdom (Weinstock, 1990). These systems combine geometric modelers with an object-based programming language. Usually they also include a constraint propagation language, a truth maintenance system, and some tools to create and organize objects. The appearance of such design systems has made it possible to begin the development of advanced design tools such as compliance checkers. In order to implement these tools however another level of representation is needed.

Tasks such as compliance checking require the ability to define object type hierarchies and recognize objects using those hierarchies, as well as the ability to define and reason about aggregate objects and their parts. To perform these high level reasoning tasks knowledge models are required. Current design systems do not provide a level of representation adequate for constructing knowledge models. Another layer of representation must be added to express the necessary concepts and relations.

At an abstract level a knowledge model can be viewed as a semantic net. The net consists of concepts (nodes) linked together by relations (arcs). Many knowledge representation schemes have been developed from this model. For example, conceptual graphs provide a way of representing the information in a semantic net by using directed acyclic graphs (Sowa 1984). Our initial approach to automating the compliance checking process was to represent the models derived from the building code as conceptual graphs. A type hierarchy of concepts was defined as well as a catalogue of relations. Models were constructed from these elements and subsequently incorporated into the type hierarchy. The advantages of using conceptual graphs are: 1) parts of the compliance checking process can be done with graph operations, 2) the recognition of objects can be done using graph operations and the type hierarchy, and 3) selectional constraints can provide type checking (Cornick et al, 1991).

Initially models were built manually in an ad hoc fashion. We found this approach unsatisfactory because it did not lend itself to generating consistent models, primarily because of the flexibility of the representation. Consequently our approach has shifted towards using tools that, while still based on semantic nets, are more specialized in their representation. However building models manually did allow for a critical examination of the contents and structure of the National Building Code of Canada. This analysis revealed the modal nature of the National Building Code of Canada and the fact that large parts of it could be modeled using exception hierarchies.

Subsequent attempts at building models have used KSSn (Gaines, 1991), a variant of CLASSIC (Brachman, McGuinness, Patel-Schneider, Resnick, Aplerin & Borgida, 1991), a KL-ONE type knowledge representation language. KSSn was used to construct models from the building regulations. KSSn, like .ts ancestors has a uniform, compositional language, with term forming operators for creating descriptions of concepts and individuals. The tool provides a more formal method for defining type hierarchies and reasoning in a constrained environment as well providing tractable reasoning mechanisms, unlike conceptual graphs. The trade off is that KSSn is not as expressive as conceptual graphs.

An important development in knowledge representation for building product models has been the growing acceptance of NIAM as a representation scheme. NIAM (Meersman, 1988) was originally developed as a mechanism to model databases. There is a similarity between NIAM and the conceptual graph representation. Many researchers are beginning to look towards NIAM as a basis for building information models (Turner, 1988; de Waard, 1991; Vanier, 1991). There is also a possibility of combining both conceptual graphs and NIAM into an ISO standard on information modeling. We plan to continue using a conceptual graph approach and incorporating more specialized representational schemes such as KSSn. Future models will also borrow some of the elements defined by other building product models, such as those defined using NIAM.

2.2.2 Plans Compliance Checking

The main objective of the plans compliance checking project at the Advanced Construction Technology Laboratory is to develop strategies and mechanisms to enable the implementation of computerized code checkers The intent is not to replace plans checkers but to remove the burden of the routine tasks. The overall goal is to reduce the time taken during the plans checking process as well as to reduce the number of errors and inconsistencies in interpretation.

2.2.2.1 The Compliance Checking Process

Building codes and standards define an envelop within which designers must create proposed designs. Compliance checking is the process of determining whether or not the design falls within the envelop. A design, or more specifically a set of architectural drawings, represents a model of a proposed design. The process of compliance checking consists of comparing the designers model with the models derived from the building regulation in question. A model, derived from a building code, will have a set of constraints associated with it. If the constraints associated with that particular model are satisfied by the design then the design is said to comply.

The designers model and building code model should overlap to some degree. The first task in compliance checking involves determining from the designers model which constraints apply. This task involves recognizing objects in the drawing and determining which building code model it most closely resembles. If objects in a design are labeled then the recognition task is straight forward. A more difficult task is where the building code contains some part of the design which is not likely to be labeled, such as a *means of egress*. Building code models are structural descriptions of some real world objects and contain slots for attributes and role playing information. Once an object has been recognized it can be expanded by filling in the slots obtained from the building code model. For example, if an object in a drawing is labeled as a building then the following slots become associated with that object: *area*, *height*, *occupancy*, and *class*.

The values for *height* and *occupancy* may be given but the values for *area* and *class* will have to be derived. Once an appropriate model is recognized then the set of constraints that must be satisfied can be identified. The constraints are checked against the actual data or derived data obtained from the designers model (the drawing). (Cornick et al, 1991; Coyne, Rosenman, Radford, Balachandran and Gero, 1990).

A model for the compliance checking process is:

1. Identify the building object in question, *building*.
2. Fill in any slots now associated with the object, *occupancy, height, sprinklered, number of streets facing*.
3. Derive data when required, *area, building class*.
4. Constrain attributes of the model, *construction type, fire-resistance rating*.
5. Compare the constrained attributes against the real data.
6. Flag any inconsistencies.
7. Look for permissions that override obligatory requirements.

2.2.2.2. Constraints

The function of a code of practice is to ensure life safety and health. A code also serves to describe models in the domain of applicability and imposes constraints on those models. The constraints are usually provisions or limitations that provide for a degree of life safety and health. There are several types of constraints imposed on the models defined by these codes. A large number of constraints in a building code deal with specifying or limiting the values that a particular attribute or characteristic can have. A building regulation, however, may also specify that certain relations must hold (or conversely a relation must not exist) between various concepts. Many of these types of constraints are topological in nature, such as adjacency constraints of various occupancy types. Finally a constraint may specify the addition or deletion of a particular concept that comprises a model.

The representation of constraints brings to focus the issue of how they are to be incorporated into the semantic net and in a system used for compliance checking. There are three possibilities, 1) incorporating constraints directly into the models, 2) letting them live somewhere outside of the concepts and semantic net, or 3) attaching them to the models as rules. Our attempts at using a model and constraint based approach to compliance checking has shown that a combination of the three ways of representing constraints is needed.

2.2.2.3 Exception Hierarchies and Deontic Logic

A large number of requirements in the National Building Code of Canada are expressed in the form of antecedent/consequent rules and have the following form:

```
if <conditions>
then <requirements>
except<exception conditions>
then <exception requirements>
```

The models derived from the National Building Code of Canada are encapsulated to a certain degree. Constraints that deal specifically with a concept, such as a firewall, are pointed to or attached to the specific object type. If the rules are organized into exception hierarchies then the number of rules can be reduced and the reasoning process somewhat simplified.

Exception hierarchies were initially developed to represent generalized knowledge (Lange 1987). For example suppose that in most cases a firewall is required to have a 2 hour fire-resistance rating.

```
rule 1
if <type> is-a firewall
then <type> has-a <frr> = 2 hours.
```

There is however an exception to this general rule, namely that if a building contains a high hazard occupancy then the rating should be 4 hours.

```
rule 2
if <type> is-a firewall
    <building> contains <occupancy> type <high hazard>
then <type> has-a <frr> = 4 hours
```

In an exception hierarchy the rules are linked together.

```
rule 1                          rule 2
if <conditions>                 if <conditions>
then <requirements>             then <requirements>
except <rule 2>                 exceptions <>
```

In most cases the most general rule holds and the 2 hour requirement is asserted. However, if it becomes known that a building contains a high hazard occupancy then the two hour requirement must be retracted and the 4 hour requirement asserted. This is a kind of a default reasoning that states unless otherwise known then assume the general case. Lange (1987) claims this form of reasoning to be more representative of how experts reason.

Regulatory documents also tend to contain a significant amount of modality (Moulin, 1990). The National Building Code of Canada is no exception. Deontic logic introduces permission and obligation operators to account for *may* and *ought* modalities. A quick reading of the National Building Code of Canada reveals that many of the requirements contain these types of modals, expressed as **X** shall be **Y** or **X** may be **Y**. The requirements that contain exceptions exhibit a definite pattern. They tend to be in the form of obligations whereas the exceptions tend to give permission to the designer to ignore the obligation. An updated form of a typical requirement is shown below:

```
if <conditions>
then <requirements> (obligation)
except <exception conditions>
then <exception requirements> (permission)
```

By putting together conceptual graphs, exception hierarchies, and deontic logic, a framework, within which some methods for automated compliance checking can be tested, has been constructed. The conceptual graphs provide a mechanism for recognizing objects, deriving information, and constructing and completing objects, essentially performing steps 1 through 4 in the compliance checking process described above. Rules and exception hierarchies provide a means of reasoning about parts of designs and models as well as posting and checking constraints (steps 5 and 6). Modal operators, such as *ought* and *may*, provide a mechanism for reasoning about conflicts between the building code models, the designs, and constraints that have been posted by the rules (step 7). To date we are continuing to experiment with defining models of the compliance checking process and trying them out in the framework we have developed.

2.2.3 **Design Reuse**

Design generation can be defined as finding an artifact description that satisfies various types of constraints (Mostow, 1989). Design is also a goal oriented activity that involves search in a potentially infinite space. Neither this process of search nor the space within which the search proceeds is well understood. How to aid designers in making this search efficient while producing "good" designs is the challenge for future computer aided design systems. One way to aid designers in their task is to reuse past designs and histories of design decisions. This reuse has the potential to reduce search complexity and has been shown to be part of a designers expertise (Akin, 1988). In this way, designs and parts of designs that have worked in the past can be reused under similar situations in the future. As well, there is the potential to alleviate duplication of past failures if records are kept and are easily accessible.

Reuse as described above, can be characterized as a form of analogical reasoning which Broadbent states is a central mechanism used by designers (Broadbent, 1973). Analogies can be drawn between two different types of models of design in order to facilitate reuse; 1) analogies between process models, and 2) analogies between structural models of designs.

2.2.3.1 Process Models

Analogies drawn between design processes makes use of decisions made during previous designs to aid synthesis of new designs or for redesign. Psychological studies have shown the extent to which analogy is used by architects in tasks such as room layout and states that the use of analogy can often result in better quality designs that more completely satisfy constraints (Akin, 1988).

Derivational analogy is one area of design that has dealt explicitly with drawing analogies between processes as a way to gain efficiency (Mostow, 1989; Blumenthal, 1990). This type of analogy gains power from utilizing past decisions about design actions as well as the action sequences of a design process (Carbonell, 1986). These include how certain constraints were satisfied or how some constraints were traded-off against others. In this way, decisions made and actions taken in past designs which are appropriate for a current design are reused to save doing the same work from scratch.

Another approach taken to the reuse of the design process is that of design rationale capture (Gruber, Boose, Baudin and Weber, 1991). Design rationale refers to the knowledge or reasoning underlying a design. Researchers in this area see the problem as a knowledge acquisition task and center on issues such as relationships between formality and expressiveness of representations and kinds of automated support for elicitation and analysis of knowledge.

2.2.3.2 Structural Models

Use of structural models refers to the use of descriptions of the actual artifacts that have been produced rather than the process of producing the artifact. Typically these will be CAD drawings of designs which are subsequently built. Analogies are often drawn between an artifact to be produced and a previously produced successful artifact or case. One area of research into reuse of structural models consists of analogical reasoning from prototypes (Gero and Rosenman, 1989). Here prototypical examples of buildings and their sub-parts are used to set up expectations and templates of designs which are then modified to fit the current design situation. Much of the work in Case-Based Reasoning in design can also be characterized as drawing analogies between artifacts (Navinchandra, 1991). Most case-based systems emphasize how best to store cases in memory to be able to retrieve only relevant cases when needed. The cases are parameterized versions of a design and reasoning from cases means changing the parameters to fit the current design situation.

2.2.3.3 Design Reuse in Spatial Layout Tasks

One of the projects being pursued at the ACT laboratory is research into the reuse of designs and design histories as described above. More specifically, the research is focussing on the reuse of spatial layouts of rooms in a building. This particular domain was chosen because layout tasks are central to much of design in general. For example, in the layout of rooms, electrical wiring, and plumbing. Other domains outside of building also deal with spatial layout such as the field of VLSI design where layout of components on circuit boards is a central task. As well, this particular domain was chosen because a body of work already exists dealing with the automated layout of rooms (Coyne, 1990; Coyne et al, 90). In this way room layout using analogical techniques can be compared to other more algorithmic methods which tend to work fine on small problems but often have difficulty scaling up to larger problems.

The approach taken so far in this project is to look at analogical reasoning over both the structural and process models of design and at how the two can be combined. Along these lines we are concentrating on derivational analogy techniques for reuse of the processes of design and case based techniques for the reuse of structural models of designs. It is felt that both structure and process need to be considered when deciding what parts of past designs can be reused as well as how they are to be reused. The important issues that need to be addressed are those noted by Mostow (Mostow, 1989) in his overview paper on derivational analogy techniques including:

1) **Representation** - What information about the original design decisions and structure is needed in order to reuse them.
2) **Acquisition** - How can this information be captured?
3) **Retrieval** - Given a problem, how can relevant previous designs be found?
4) **Correspondence** - Which objects, constraints, etc. in the new design correspond to which ones in the old design?
5) **Appropriateness** - When should a part be reuse?
6) **Adaptation** - How can a previous design be altered to fit a new problem?
7) **Partial reuse** - Which parts of a design can be reused by themselves?

This research is just beginning but is meant to utilize the same structural building models consisting of objects, relationships and constraints as described in section 2.2.1 above. In this way it is felt that design synthesis can be aided by reuse of past designs which can then be checked for compliance by the automated system also described above in section 2.2.2. Both design aids and compliance checkers can then be related to object based CAD packages through these same structural models.

3 Conclusions

The ability to utilize the knowledge and information that is created by and exists within the construction industry will be critical in the coming decades. Much of the work that we have outlined above represents some of the firsts critical steps in this endeavor. Without good models of the construction process the industry will be unable to take full advantage of the new information based technologies. Providing access to information is important but it is equally important to provide the tools to ensure that the information is correctly interpreted and applied. During the design stage this involves the ability to help the designer "reuse" previous successful design solutions and the provision of code verification tools. In situations such as building inspections where we are dependent on the inspectors visual skills we need to support the human inspector with tools such as training systems and hypertext documentation.

There is clearly a lot of work to be undertaken in the development of tools and techniques to enable the construction industry to take full advantage of the "Information Age". We have presented just a few of those first steps that we are currently undertaking within our laboratory.

References

Akin, Omer. (1988) Expertise of the Architect. Chapter 7. In Expert Systems for Engineering Design. Edited by Michael D. Rychener. Academic Press, Inc. Boston: USA.

Blumenthal, Brad. (1990) Empirical Comparisons of Some Design Replay Algorithms. Proceedings of AAAI 90. pp. 902-907.

Brachman, R., J., McGuinness, D., L., Patel-Schneider, P., Resnick, F., Aplerin, L., & Borgida, A. (1991). Living with CLASSIC: When and How to Use a KL-ONE Type Language. In J. F. Sowa (Eds.), Principles of Semantic Networks: Explorations in the Representation of Knowledge (pp. 401-456). San Mateo, CA: Morgan Kaufmann.

Broadbent, Geoffrey. (1973). Design in Architecture. John Wiley and Sons, London.

Carbonell, Jaime. (1986). Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. Chapter 14 in Machine Learning Volume II. Edited by Ryszard Michalski, Jaime Carbonell, and Tom Mitchell. Morgan Kaufmann Publishers, Los Altos.

Cornick, S. M., Leishman, D. A., & Thomas J. R. (1991). Integrating Building Codes into Design Systems. In First International Symposium: Building Systems Automation-Integration, (pp. 4.7.1-4.7.26). Madison, Wisconsin:

Cornick, S. M., Leishman, D. L. & Thomas, J. R. (1990a). Incorporating Building Regulations in Design Systems: An Object Oriented Approach. In ASHRAE Transactions, 96 St Louis, Missouri: American Society for Heating, Refrigerating, and Air-Conditioning Engineers, Inc.

Cornick, S. M., Leishman, D. L. & Thomas, J. R. (1990b). The Integration of Regulatory Codes with Design Systems. In A. Gulliver (Ed.), Canadian Conference on Electrical and Computer Engineering: Ten Years to 2000, 1 (pp. 14.4.1-14.4.5). Ottawa, Ontario: Canadian Society for Electrical and Computer Engineering.

Coyne, R. D.. (1990). Logic of Design Actions. Knowledge-Based Systems. V3 (4). pp. 242-257.

Coyne, R. D.,Rosenman, M. A.,Radford, A. D.,Balachandran, M., & Gero, J. S. (1990). Knowledge-Based Design Systems. Sydney, Australia: Addison-Wesley.

Davidson, C. (1978) Canadian Thesaurus of Construction Science and Technology, Industry Science and Technology Canada (Formerly Department of Industry Trade and Commerce Canada), Ottawa, Canada.

de Waard, M., & Tolman, F. (1991). Modeling of Building Regulations. In K. Kahkonen & B.-C. Bjork (Eds.), Computers and building regulations: VTT Symposium 125, (pp. 195-209). Espoo, Finland:

Gaines, B. R. (1991). An Interactive Visual Language for Term Subsumption Languages. In IJCAI'91: Proceedings of the 12th International Joint Conference on Artificial Intelligence, Vol. 2 (pp. 817-823). Sydney, Australia: Morgan Kaufmann.

Gero, J. and Rosenman, M.A. (1991) A Conceptual Framework for Knowledge-Based Design Research at Sydney University's Design Computing Unit. In Artificial Intelligence in Design. Edited by D.T. Pham. Springer-Verlag, Berlin.

Gruber, T., Boose, J., Baudin, C., and Weber, J.(1991) Design Rationale Capture as Knowledge Acquisition: Tradeoffs in the Design of Interactive Tools. Machine Learning Proceedings: Eighth International Workshop. Evanston, Illinois.

International Standards Organization (1986). ISO 8879: Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). Geneva, Switzerland.

Lange, R. (1987). Exception Hierarchies as a Knowledge Representation for Expert Systems. In D. Sriram & R. A. Adey (Ed.), International Conference on Applications of Artificial Intelligence in Engineering Problems (2nd), (pp. 1-10). Cambridge, Mass: Computational Mechanics Publications, Boston, USA.

Meersman, R. A. (1998). Towards Models for Practical Reasoning About Conceptual Database Design. In R. A. Meersman & A. C. Sernadas (Eds.), Data and Knowledge (DS-2), (pp. 245-263). North Holland.

Mostow, J. (1989). Design by Derivational Analogy: Issues in the Automated Replay of Design Plans. Artificial Intelligence 40. pp. 119-184.

Moulin, B., & Rousseau, D. (1990). Designing deontic knowledge bases from regulation texts. Knowledge-Based Systems, Vol. 3 (2), pp. 108-120.

Navinchandra, D. (1991). Exploration and Innovation in Design. Springer-Verlag, New York.

Nielsen, J. (1990). Hypertext and Hypermedia. Academic Press, San Diego, CA.

Sowa, J. F. (1984). Conceptual Structures: Information Processing in Mind and Machine. Don Mills, Ontario: Addison-Wesley.

Turner, J. A. (1988). A systems approach to the conceptual modeling of buildings. In P. Christiansson & H. Karlsson (Ed.), Conceptual Modeling of Buildings: CIB Proceedings, (pp. 179-187). Lund, Sweden: International Council for Building Research Studies and Documentation.

Vanier, D.J. (1990) "Hypertext - A computer tool to assist building design", In, The Electronic Design Studio, Eds. MIT Press, Cambridge MA.

Vanier, D., J. (1991). A Parsimonious Classification System to Extract Project-Specific Building Codes. In K. Kahkonen & B.-C. Bjork (Ed.), Computers and building regulations: VTT Symposium 125, (pp. 134-145). Espoo, Finland: VTT.

Warthen, B. (1991). PDES/STEP: The coming cornerstone of CAD/CAM Integration. In First International Symposium: Building Systems Automation-Integration, . Madison, Wisconsin:

Weinstock, N. (1990). The Complete Directory of Automated Design Software. New York: Simon & Shuster.

Applications of Artificial Intelligence in Engineering VII

Editors: D.E. Grierson, University of Waterloo, Canada
G. Rzevski, The Open University, U.K.
R.A. Adey, Wessex Institute of Technology, University of Portsmouth, U.K.

Computational Mechanics Publications
Southampton Boston

Co-published by

Elsevier Applied Science
London New York