A REVIEW OF COMPUTERISED STANDARDS SUPPORT IN NEW ZEALAND

ABSTRACT: The contributions made to the field of computerised standards checking by New Zealand institutions and the services offered by commercial providers are detailed in this paper. Historically the work has been research oriented, with collaborations between university and industry associations concentrating on requisite frameworks to support code compliance checkers. In more recent times commercial interest in the area has grown, leading to fully commercial products and services being offered to handle uniquely New Zealand standards. This is balanced with on-going research into frameworks for the integration of multiple compliance checkers through common models of buildings and systems.

Keywords: standard compliance, integrated environment, computerised checking.

INTRODUCTION

As part of this special issue on standards processing, we present a review of national activities in research and practice in New Zealand. Work on computerised support for standards and standards processing in the building and construction industry in New Zealand has proceeded for approximately 10 years. The major thrust of this work has been research oriented, but the advent of a new building code together with wide-spread access to cheaper computers and networking facilities has created opportunities for larger scale commercial development of tools.

We commence with a discussion of the earlier research work, which has focussed on frameworks for the development of code conformance applications. Current work is then described. Current research has concentrated on support for integrating multiple code conformance (and other) tools into a design support environment, while commercial development has concentrated on cost effective delivery. We conclude with a description of likely future activity in both research and commercial sectors.

EARLIER RESEARCH

Previous work in computerised standards processing in New Zealand has principally been conducted by the University of Auckland Department of Computer Science (UACS), to which the authors belong, in collaboration with the Building Research Association of New Zealand (BRANZ). This work has been predominantly research oriented, focussing on the design and implementation of a framework for developing code conformance applications (Hosking et al, 1987; Hosking et al, 1989; Mugridge and Hosking, 1989; Hosking et al, 1990; Amor et al, 1992). This framework has developed out of the experience in constructing a number of such applications including:

- *FireCode*: a system for checking building designs for conformance with the means of escape provisions of a draft Fire Safety Code (Hosking et al, 1987). This code of practice is quite prescriptive.
- *Seismic*: which assists in checking that a design meets earthquake and wind loading requirements of a rather more performance-based code of practice

than the Fire Safety Code (Hosking et al, 1989). This system incorporates heuristic advice obtained via interviews with a domain expert.

- *WallBrace*: (Mugridge and Hosking, 1989) which assists in checking and designing wall bracing to meet the requirements of a prescriptive loading code for light timber framed buildings (SANZ, 1990). Despite the prescriptive nature of the code, WallBrace required additional expert-derived heuristic information to allow it to cope with designs that fall within the scope of the code of practice, but which were not adequately covered by its provisions. Variants for floor and roof bracing have also been developed by BRANZ (Hosking et al, 1990).
- *ThermalDesigner*: (Amor et al, 1992) which assists in checking conformance of residential dwelling designs against a Thermal Insulation Code (SANZ, 1977). The code itself is performance based, but ThermalDesigner encodes a previously developed paper based methodology for checking compliance (Bassett et al, 1990).

In the following, we describe the framework and its rationale in a reasonable amount of detail, as it is the cornerstone of current research activities.

Representational model

The development of the Standards Analysis, Synthesis, and Expression (SASE) methodology and tool set (Fenves et al, 1987) has been influential in most subsequent work in code of practice representation. The SASE approach formalised the *provision* as the basic building block of codes of practice and code conformance systems, and argued for a functional interpretation of provisions. Tools provided with SASE permitted a provision to be multiply categorised and indexed.

A diverse range of programming techniques has been used in the implementation of provisions for conformance checking, including procedural code (Turk, 1991), rule-based approaches (Rosenman, 1985), logic programming (Sergot et al, 1986), decision tables (Fenves et al, 1987) and the authors' approach of functional programming described below. An alternative to SASE's functional interpretation of provisions is to treat them as multi-directional constraints (Cornick et al, 1991). This latter approach may be appropriate for automated design optimisation, but it is not necessary for conformance checking where a design has already been provided. A general constraint satisfaction approach can also have serious practical difficulties (Leler, 1988). For these reasons, a functional interpretation of provisions is more common.

> The seasonal heat loss of a building is the air heat loss plus the sum of the floor, wall, window, and roof heat losses for each space in the building
>
> The heat loss of a floor is the floor area times the annual loss factor of the floor divided by the thermal resistance of the floor.
>
> The thermal resistance of a suspended floor is the subfloor thermal resistance plus the floor components thermal resistance plus the floor covering thermal resistance.
>
> The thermal resistance of a slab floor is the ground plus slab thermal resistance plus the floor covering thermal resistance.

Figure 1: Example thermal code provision

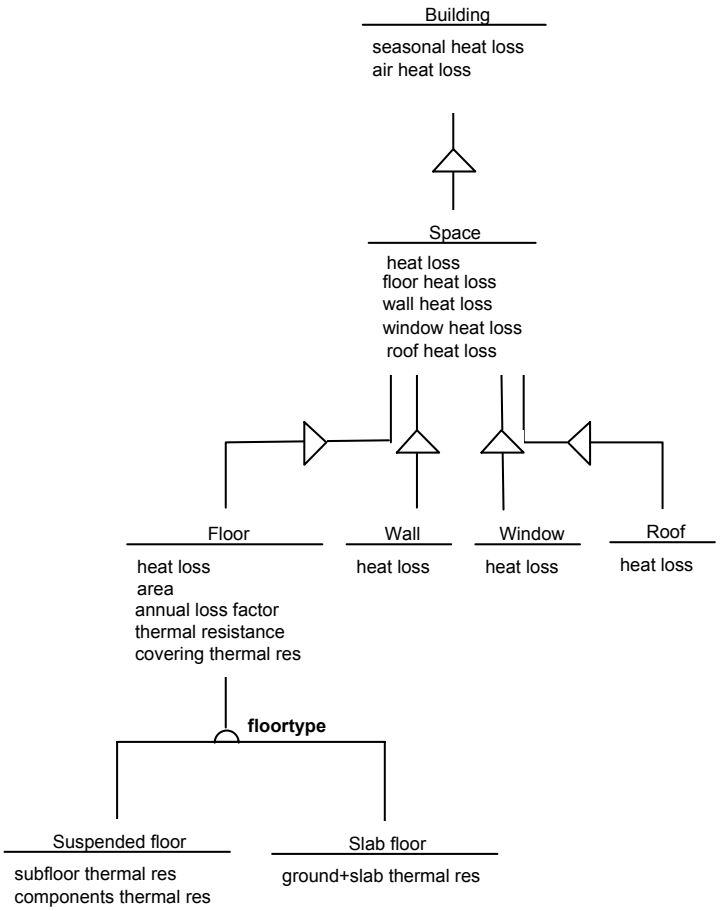

Figure 2: Object-oriented model from analysis of Fig. 1

In addition to provision interpretation, a representation is needed for the building being checked for conformance. An object-oriented approach to building representation is the de facto standard (de Waard, 1992; Turk, 1991). As an example of how such a model can be developed, consider the provision in Fig. 1, taken from ThermalDesigner, which specifies the seasonal heat loss of a building. Fig. 2 shows an object-oriented model constructed from the significant objects/classes and attributes mentioned in the provision, together with their relationships. The representation uses standard object oriented analysis constructs, and follows the OOA notation of Coad and Yourdon (1991).

Representing provisions within the object-oriented model

While an object-oriented building model is becoming standard, opinions differ on the most appropriate way of integrating provisions with a building model. Some advocate a separate "rule-base" of provisions with free access to the entire building model (Dym et al, 1988). We argue that provisions must be applied within a context (eg the thermal resistance of the north facing wall of the kitchen of my house) and thus it is better to first model the context in which provisions are applied and then embellish the model with a suitable encoding of provisions. Thus in our approach, provisions are represented as functions organised and

distributed according to the components of the building that they affect. To support this approach, Kea, a functional object-oriented language, has been developed (Hosking et al, 1990, 1991). Fig. 3 shows the Kea implementation of the provision of Fig. 1 within the object-oriented model of Fig. 2. The following features of the implementation are of note:

- Inheritance is used to implement generalisation-specialisation links. Thus SuspendedFloor and SlabFloor inherit from Floor.
- Lists are used to implement one-to-many whole-part relationships. Thus Space includes list features for its floors, walls, windows and roofs.
- Attributes, corresponding to provisions and the data items they use, are coded as functions.
- Information hiding provides control over the interaction between objects.
- Functions may be inherited or overridden. Thus SuspendedFloors and SlabFloors both inherit Floor's function for evaluating heat_loss. While Floors have a thermal_resistance, SuspendedFloors and SlabFloors have their own methods for calculating it.
- Placing provisions as attributes in the object-oriented model often simplifies them due to an assumed context (Hamer et al, 1989). For example, as the thermal_resistance function in class SuspendedFloor can assume a SuspendedFloor object is being checked, this need not be explicitly stated in the function.
- Kea is strongly typed, reducing the number of possible errors in an application and the manual effort involved in the validation of an application.
- Classifiers are built into Kea. Class Floor has a classifier floortype which can take a value of SuspendedFloor or SlabFloor. Classifiers have both a static interpretation as constraints on inheritance and a dynamic interpretation (Hamer et al, 1992). The latter allows the class membership of an object to be elaborated at run time. Thus, an object initially created as a Floor may be refined to be a Suspended (or Slab) Floor, as more is known about or required of the object.
- Functions are evaluated lazily. Thus the spaces of a Building are only constructed as needed, such as to calculate the Building's seasonal_heat_loss. Classification is also lazy. Thus if the thermal_resistance of a Floor is needed, the floortype classifier is first evaluated, causing classification of the Floor to SuspendedFloor or SlabFloor. The thermal_resistance function in the chosen subclass is then executed. Laziness means that only parts of the model necessary for satisfying a particular task need elaboration, reducing computational costs and data entry requirements.

```
class Building
  (* parts *)
  spaces: list Space.
  (* attributes *)
  public seasonal_heat_loss: float
            := air_heat_loss + sum(collect(s in spaces, s^heat_loss)).
  air_heat_loss: float := ...
end Building.

class Space
  (* parts *)
```

```
  floors: list Floor.
  walls: list Wall.
  windows: list Window.
  roofs: list Roof.
  (* attributes *)
  public heat_loss: float
      := floor_heat_loss + wall_heat_loss + window_heat_loss + roof_heat_loss.
  floor_heat_loss: float        := sum(collect(f in floors, f^heat_loss)).
  wall_heat_loss: float         := sum(collect(w in walls, w^heat_loss)).
  window_heat_loss: float       := sum(collect(w in windows, w^heat_loss)).
  roof_heat_loss: float         := sum(collect(r in roofs, r^heat_loss)).
end Space.

class Floor
  classifier
     floortype: [SuspendedFloor, SlabFloor] := ...
  public heat_loss: float := area * annual_loss_factor / thermal_resistance.
  annual_loss_factor: float := ...
  thermal_resistance: float.
  covering_thermal_res: float := ...
end Floor

class SuspendedFloor
  inherits Floor.
  thermal_resistance :=
     subfloor_thermal_res + components_thermal_res + covering_thermal_res.
  subfloor_thermal_res: float := ...
  components_thermal_res: float := ...
end SuspendedFloor.

class SlabFloor
  inherits Floor.
  thermal_resistance: float :=
     ground_plus_slab_thermal_res + covering_thermal_res.
  ground_plus_slab_thermal_res: float := ...
end Sleeping.
```

Figure 3: Kea implementation of the provision of Fig. 1

User interaction

Users must be able to construct and interact with instances of the building model and be supplied with code compliance results. This requires i/o and model navigation facilities. The Kea-based code conformance architecture includes three components for interacting with the object-oriented building model:

- A CADrafting front end to allow users to enter plan details. This is a generic, X-window based package for entry of simple orthogonal building plan information, particularly the geometry of the external envelope.
- An X-window based forms interface. Forms allow users to provide additional non-geometric information as well as to obtain information about the model and its conformance to the code of practice. Buttons on forms also provide a hyper-text like navigation facility permitting the user to browse the model and treat the system more as a set of services than as a function that calculates a single result.
- Textual output for production of summary reports.

Figure 4 shows an example of the ThermalDesigner system, constructed using the Kea architecture, in use.

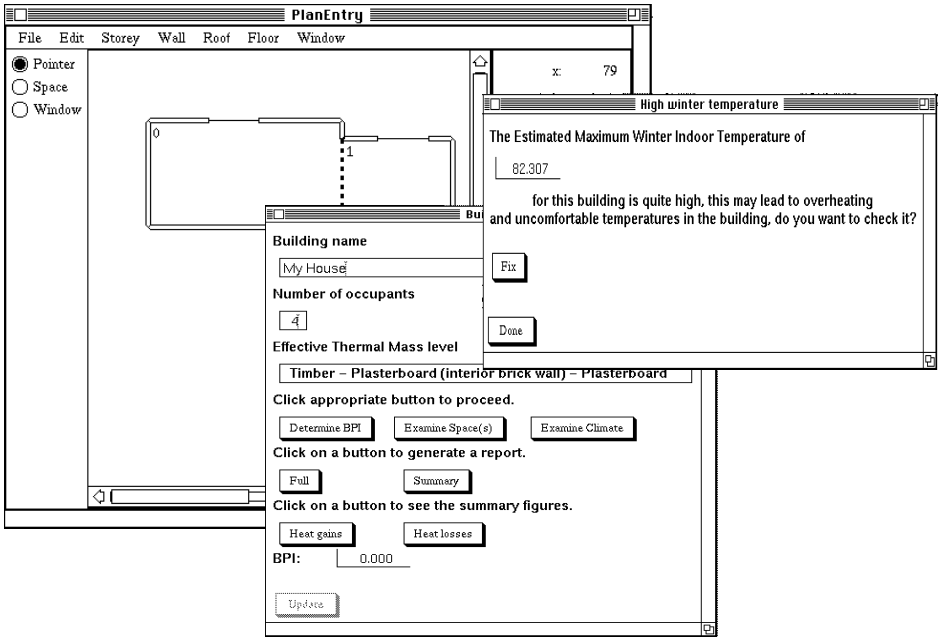

Figure 4: ThermalDesigner system in operation

Consistency management

Facilities for modification of the model are also necessary to allow users to alter building designs, experiment with design alternatives, and correct design flaws. In the Kea framework this is handled incrementally, i.e. changes are immediately propagated through the system and results modified appropriately. Providing a mechanism to handle incremental modifications usually involves explicit code in the design system to handle the changes. An important feature of the Kea approach to modelling is that the programmer programs as if the building is described once by the user with no subsequent modifications. Thus there is no explicit code in a Kea application to handle modifications to the building design. This approach simplifies the application code considerably. The user may modify *any* input and Kea will *automatically* propagate the effects of the change. The state following the design change is as if the application had the modified value as its initial entry.

For example, modifying plan elements, such as resizing a space, causes Kea's consistency manager to propagate the effects of the change to the corresponding object attributes, ensuring displayed results are consistent with the plan modifications. Thus a provision will be reinterpreted whenever any value it depends on is changed, either due to a direct input change (from plan entry or a form) or indirectly due to the reinterpretation of another provision on which it depends. Provisions that check for conformance with the code may signal violations through the display of additional forms (with suggested changes to achieve conformance). The top-most form in Fig. 4 is an example. Such error forms are subsequently eliminated if an input change causes the violated provision to be satisfied.

Experience

The results of the Kea-based project have been encouraging. However, they have not directly led to commercial products for a number of reasons. One reason is the small size of the market in New Zealand. Given a population of just 3.6 million, specialised tools to address codes specific to this country will have problems generating enough income to cover their development costs, let alone return a profit. Another reason is that the research systems, being experimental in nature, consume fairly large amounts of computational resource, more than could be reasonably expected in the typical building design office. A third, and more fundamental, reason is that each of the Kea-based code conformance tools developed to date has been designed for stand-alone usage. This means that designers wanting to check their design for conformance against multiple standards have a considerable overhead in entering the same or similar building information to each such design tool, and this introduces much opportunity for inconsistency of data.

In response to these problems, research directions in New Zealand have concentrated on the problem of integrating multiple design tools, while commercial and pre-commercial development has concentrated on the use of low-cost PC-based platforms and conventional development environments for delivery of applications.

CURRENT RESEARCH AND DEVELOPMENT ACTIVITIES

It has become clear over the last few years that the task of transferring data between design applications working on a common design presents one of the major bottlenecks in computerisation of the building industry. Much effort has been concentrated on the development of standards for the representation of building structures and components (Amor et al, 1993; Augenbroe and Laret, 1989; ATLAS, 1993; ISO/TC184, 1993). These projects have in common the recognition that an integrated model of a building is essential to the creation of a "collaborating" set of design tools. In New Zealand, both BRANZ and the UACS have been working on related aspects of integrated building model construction with the overall aim of constructing a demonstration system integrating at least two of the Kea-based tools, ThermalDesigner and WallBrace, via a common model of residential dwellings. The development of such a system requires a number of tools and techniques including:

- A means of defining the various models involved: the schema of the common building model and the schemas for each of the tools involved.
- A means of developing the common building model schema from the tool schemas: schema integration techniques.
- A means of defining the mappings between the tool schemas and the integrated schema
- A means of creating model instances and moving data between a common instance and the tools using the mapping specifications
- An overall framework for managing the above

In the following we discuss the approaches taken by UACS and by BRANZ in attempting to meet these requirements and the resulting ICAtect framework for design tool integration. Figure 5 shows the structure of the ICAtect system, highlighting the major components of the framework and their connections with the code compliance checkers.

Schema definition

Schema specification in ICAtect is supported by the EXPRESS Programming Environment (EPE) (Amor et al, 1995). EPE permits multiple graphical (EXPRESS-G) and textual (EXPRESS) views of a schema to be created and manipulated as required for various stages of the specification process while maintaining the consistency of the data in all the views. EPE thus permits graphical specification of entity hierarchies in analysis views, graphical specification of entities and attributes in design views, and textual specification of complete entities in late design views. The system also provides navigation methods to find views which reference particular entities and documentation facilities to track modifications and developers' reasons for changes. Figure 6 shows EPE in use, with an EXPRESS-G graphical view and EXPRESS textual view visible.

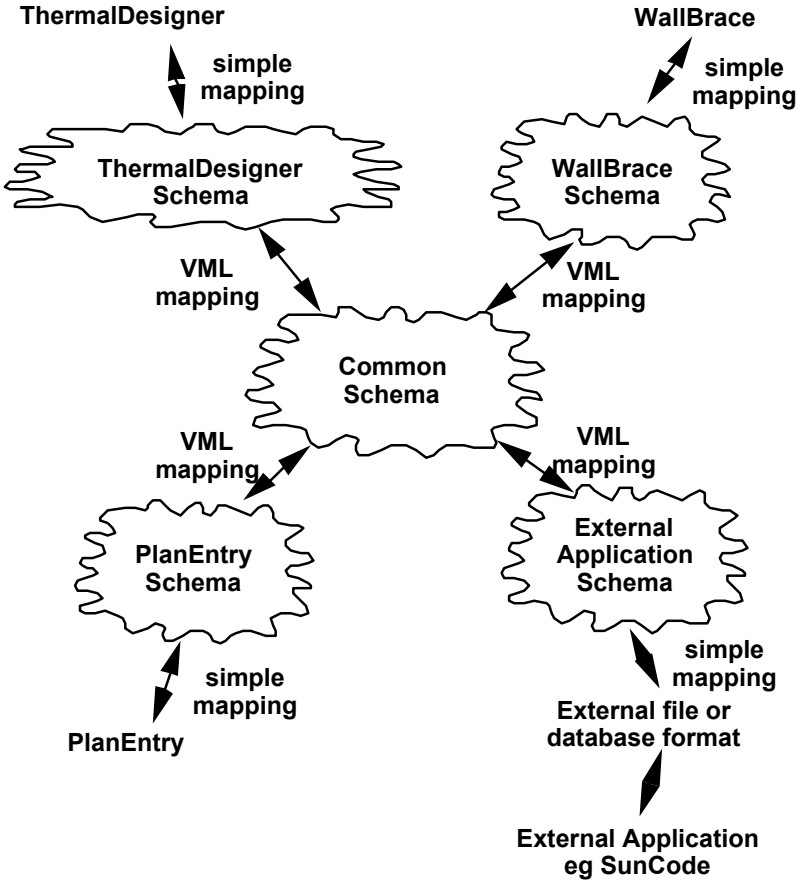

Figure 5: General structure of the ICAtect system

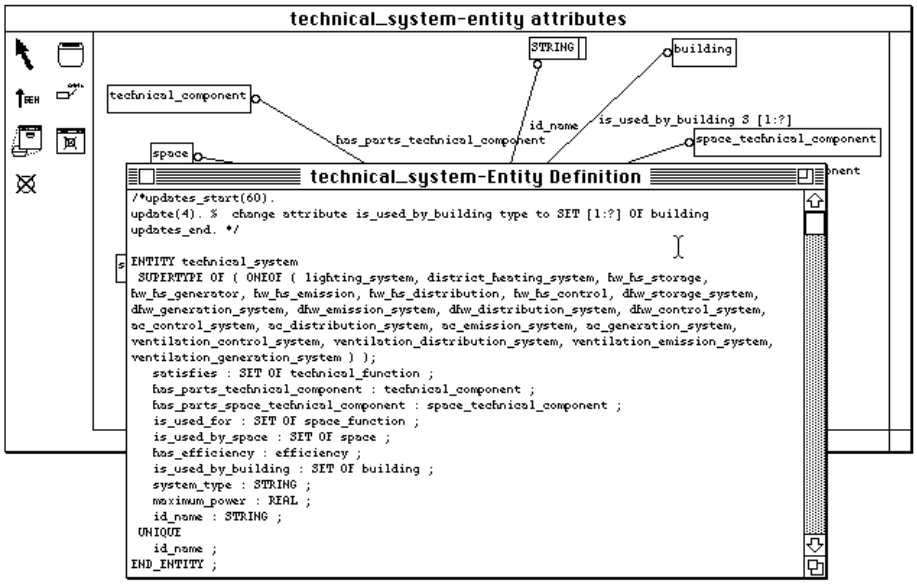

Figure 6: EPE textual and graphical views of an EXPRESS schema

Schema integration and mapping definitions

In both the UACS and BRANZ work, an incremental and evolutionary approach to the construction of a common building model has been taken. In this approach, the common model is extended by the connection of an additional design tool. At Auckland, work has concentrated on schema integration techniques appropriate for such an evolutionary approach (Mugridge and Hosking, 1995), while at BRANZ work has concentrated on appropriate geometric models for a residential building model (Price, 1994).

This schema integration experience, together with earlier experience at Victoria University of Wellington (Amor et al, 1993), has led to development of a mapping definition language VML (View Mapping Language, see Amor, 1994) to both describe mappings between common and tool schema and to assist in extension of the common schema to incorporate changes imposed by the information needs of new tools.

In contrast to other mapping languages being developed, VML takes a declarative approach to the specification of mappings. A VML mapping has three major parts. *Inter class* specifications detail the entities which take part in the described mapping. *Invariant* specifications describe when and if the particular mapping can be applied to a certain set of objects. *Equivalences* specify the relationships between attributes in the entities named in the mapping. The equivalences are mostly specified as mathematical equations or functions which can be applied in either direction (ie only one mapping specification is needed to map data in both directions). For cases where a functional specification of an equivalence is not possible the integrator can define a procedure to handle the mapping.

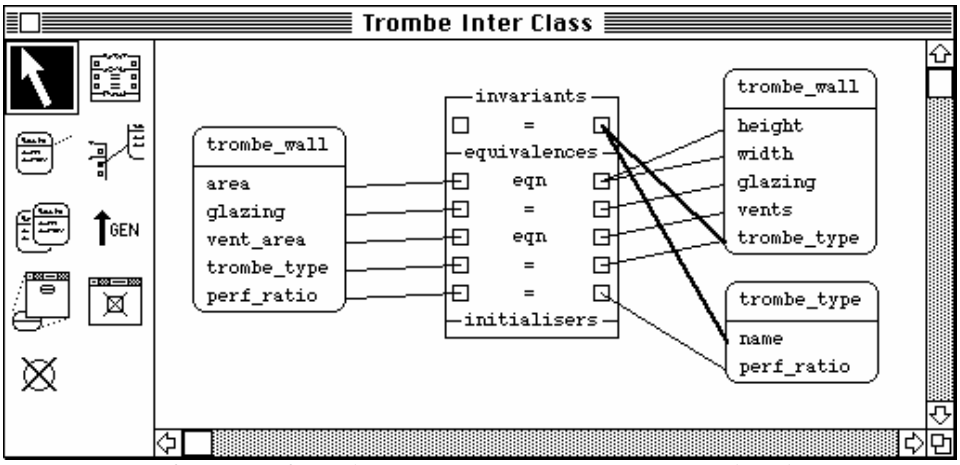

Figure 7: Specification of attribute mappings using VML graphical environment

The VML language is supported by a specification environment in a similar fashion to EPE (see Figure 7). This environment allows the definition of mappings between entities in two schemas in both graphical and textual notations. The environment also supports the modification of the schemas it is mapping between. This allows the addition of new entities, relationships and attributes to the common schema based upon a mapping specification for new types of data in a tool being connected to the integrated system. Integration of a new tool to the common model thus proceeds in the following manner:

- an EXPRESS schema is constructed defining the information needs of the new tool
- VML definitions are used to define the mappings between entities and attributes in the new tool schema and the existing common schema, possibly extending the common schema in the process.

Once the schemas and inter-schema mappings are defined, the final task in tool integration is to define a simple mapping from the EXPRESS-based Tool schema to the actual data format required by the tool. A parser and un-parser kit are used to support specification of these mappings.

Instance construction and manipulation

ICAtect supports instance construction and manipulation via a generic, extendible plan drawing system, PlanEntry, and an object-based model navigation tool, Cerno. Both of these construct and manipulate models implemented in the Snart object-oriented logic language. The EPE tool has facilities for translating EXPRESS schema definitions into Snart implementations which can then be instantiated. Instances of a schema can be loaded into, and saved out of, the EPE system in the form of STEP data transfer format files. These files provide a neutral method of exchange of instance data between the integrated system and other, un-connected, applications.

PlanEntry is constructed using a novel object-oriented constraint based language (Hosking et al, 1994). It allows users to construct an object-oriented model of a building and its components via multiple, editable, 2-D views of the building. Views and the underlying model are kept consistent with one another via constraint propagation

Users can construct any number of building views with PlanEntry. In Fig. 8, four such views are shown, each in its own window. Labelled *view lines* indicate

the plane of orthogonal views and allow rapid navigation between views by clicking on them. They also allow the offset of a view on its view axis to be shifted by dragging the view line. Generic graphics tools are provided for view creation and removal, selection/dragging, space, roof, window, and internal wall creation, and the addition of user specified constraints. The tool palette and underlying building model may be extended to permit graphical definition of other building components.

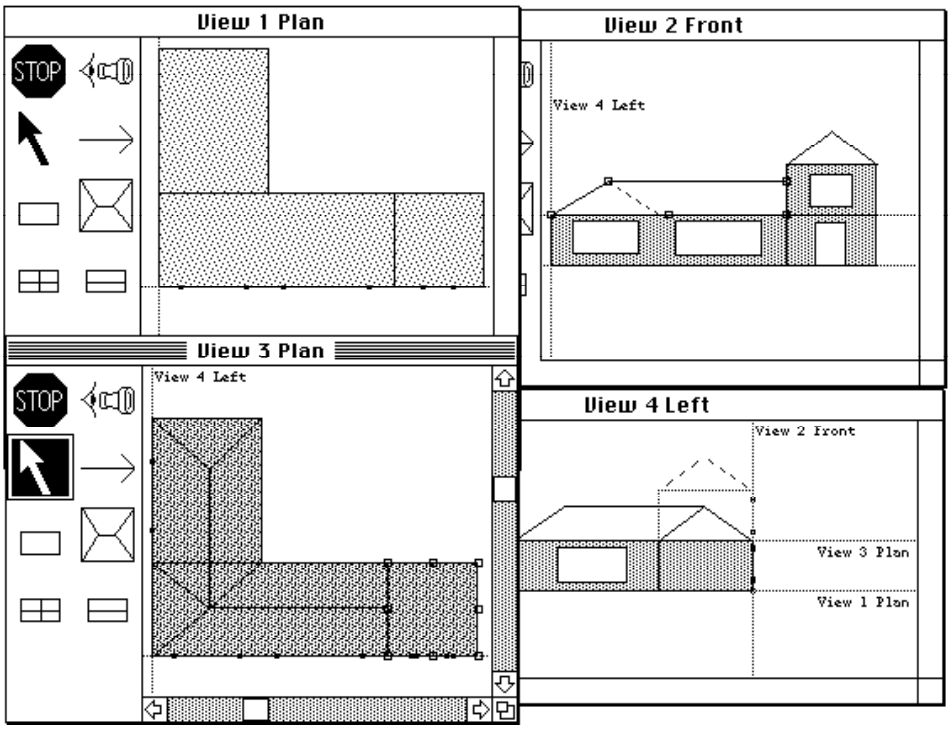

Figure 8: PlanEntry system in use

Some objects are implicitly constrained in how they can be manipulated. For example, ridge lines of roofs are constrained to be parallel to their original axis, so moving one end of a roof perpendicular to that axis causes the other end to shift correspondingly. Users may interactively specify additional constraints between parts, such as constraining two walls to be in the same plane.

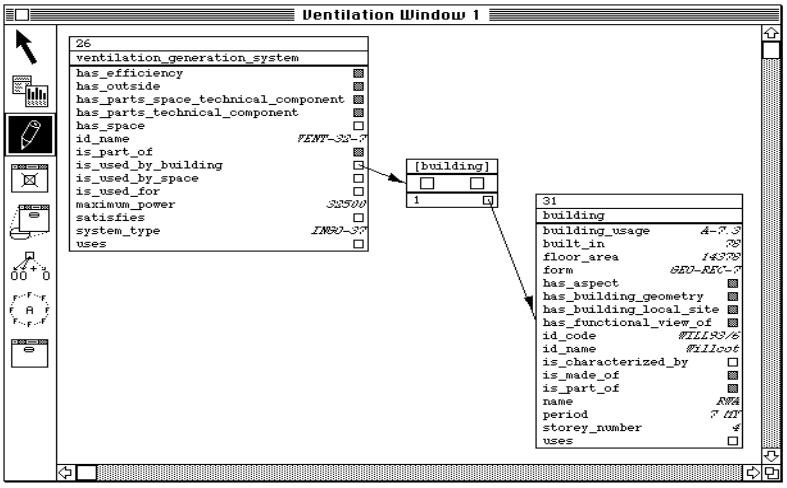

Figure 9: The Cerno object browser

To manipulate the textual attributes of objects the Cerno system provides a graphical model instance browser (Fenwick et al, 1994). Model entities, attributes, and relationships can be visualised and manipulated simply and effectively in a mixture of iconic and textual forms (Figure 9). The amount of information shown, the form of display, and the level of abstraction, are all under the user's control. Cerno updates its visualisations as applications execute, allowing both static and dynamic visualisation of model behaviour.

Mapping between instances

VML has dynamic semantics which permits model instances of a schema to be kept consistent with other schemas via the VML inter-schema definitions. Mapping to and from a design tool and the common model is done in two steps (see Figure 5). VML definitions both control the mapping between an instance of the common building model and an instance of the tool schema (both implemented as collections of Snart objects) and ensure both model instances are kept consistent.

A second mapping process translates from the Snart representation of the tool's data to and from the physical representation of the data actually used by the tool, eg Kea objects for the code conformance tools. The VML mapping is usually a complex one, involving significant transformation in the logical representation. The tool-tool schema mapping is usually more straightforward.

Overall framework

This type of framework provides a very general mechanism for attaching new conformance or simulation tools. The implementor need only specify the schema of the tool to be attached, the mapping between the tool's schema and the common schema and a mapping of the tool's data files to the tool's schema. Therefore, any conformance tool, simulation tool, and user interface that is required by participants in a design project can be linked into a single framework which will manage the consistency of the data throughout the project.

It must be noted, however, that the work described here details a framework with its emphasis on the product of the design process. There is still much work to be performed in introducing notions of process into the framework along with modelling of the users of the system. This is obviously an important step to take as a collaborative design system should have built-in notions of the users of the system, the access privileges they have, and what aspects of the design they have the authority to change. These ideas are of concern to the authors of this paper and are currently being investigated as part of the development of this framework.

The structure of the framework described in this paper has evolved over several years to reach its current state. Many schemas for conformance tools and simulation tools have been developed with parsers and unparsers to transfer data between the tools data files. The tool schemas have been used to construct a common schema which has been revised as new conformance tools have been modelled in the system. With the recent addition of the VML language the mapping between a few tool schemas and the common schema have been detailed. In its current incarnation the ICAtect system contains all the composite parts that have been discussed in this paper tied into a coherent working system. The majority of the implementation entailed incorporating the operational aspects of the VML language into the ICAtect control system, a process which is now complete.

CURRENT COMMERCIAL PRODUCTS

Though electronic information services are currently available to the building industry in New Zealand, there is little on offer that addresses the use of codes and standards. One reason for this is the small size of the market, as mentioned earlier. However, two commercial systems have recently been developed, both of which use a spreadsheet approach to encapsulate the calculations found in a particular standard. The standards covered in these applications are somewhat different to other national standards as they have a very algorithmic specification of compliance criteria, and are therefore prime candidates for computerisation.

The ALF Design Manual (Bassett et al. 1990) is a paper-based design guide developed to assist in the process of checking that a building meets the requirements of the New Zealand thermal insulation standard for residential buildings (NZS 4218P, SANZ 1977). It embodies an empirical approach to the problem that has been developed by the Building Research Association of New Zealand (BRANZ) and is allowed as a verification method for the new performance based New Zealand Building Code (NZBC, see SANZ, 1992). ALF formed the basis of the Kea-based ThermalDesigner system described earlier.

Following on the experience of ThermalDesigner, a spreadsheet based version of the design aid has been developed for BRANZ by the School of Architecture at Victoria University and is now sold alongside the paper version (BRANZ, 1993). This system duplicates much of the style of the original ALF worksheets, both in terms of layout and the data which must be entered by the user. However, the system automatically performs all the required calculations. It also has on-line help for individual fields and when the value for a field is limited it allows selection from a menu. The final output of the ALF system is seen in Figure 10 where the calculated Building Performance Index is displayed along with an indication of whether the design meets the energy efficiency provision of the code.

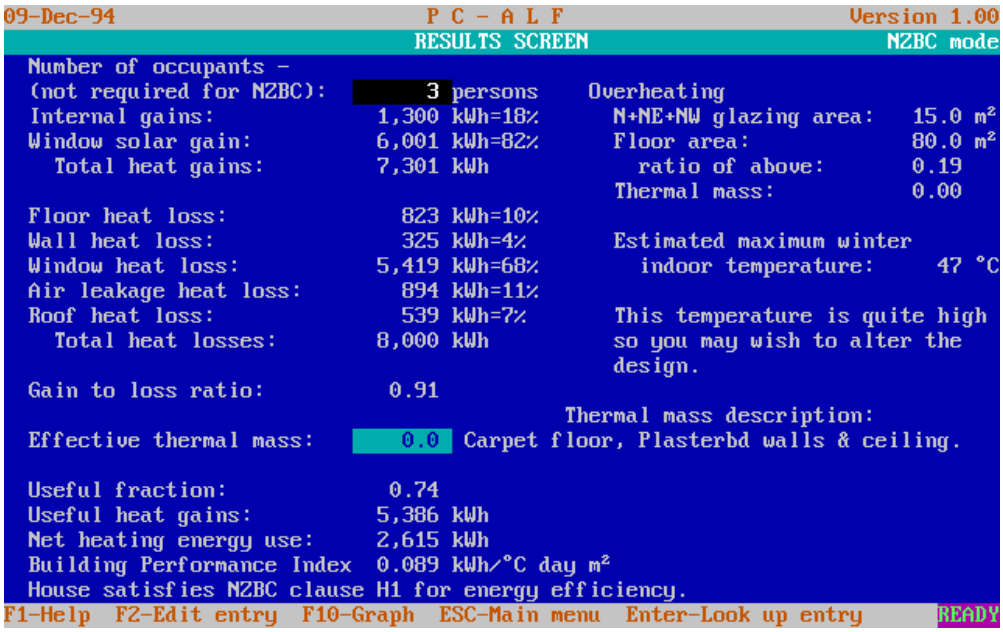

Figure 10: ALF Results screen with code compliance verification

BraceS (SANZ, 1991) allows for the calculation of the bracing requirements for light timber framed buildings in accordance to NZS 3604 (SANZ, 1990), the standard addressed by the WallBrace system mentioned in the earlier research, and is a verification method for Clause B1 of the NZBC. The paper based standard does not have a worksheet like calculation procedure as found in the ALF design manual, so the BraceS system has had to codify the appropriate parts of the standard into a worksheet layout in order to calculate the bracing units required for a building (Figure 11). Thus the user need only enter the most fundamental data on building layout and structure and the program performs all required calculations to determine earthquake and wind loading as well as minimum bracing requirements.

BRANZ have also investigated the development of a code browsing system for the NZBC, implemented as an extension to the Microsoft Windows Help facility. The NZBC comprises 35 approved documents along with verification methods (many of the previous standards can be used as verification methods) and acceptable solutions. The browser allows hypertext like searching and cross-referencing through portions of the building code. As well as browsing the text of the code this system provides precise definitions of key words and diagrams from the code which may be copied from the code into the user's own documents. BRANZ have also investigated the addition of small compliance checkers for portions of the new code as part of their browsing system.

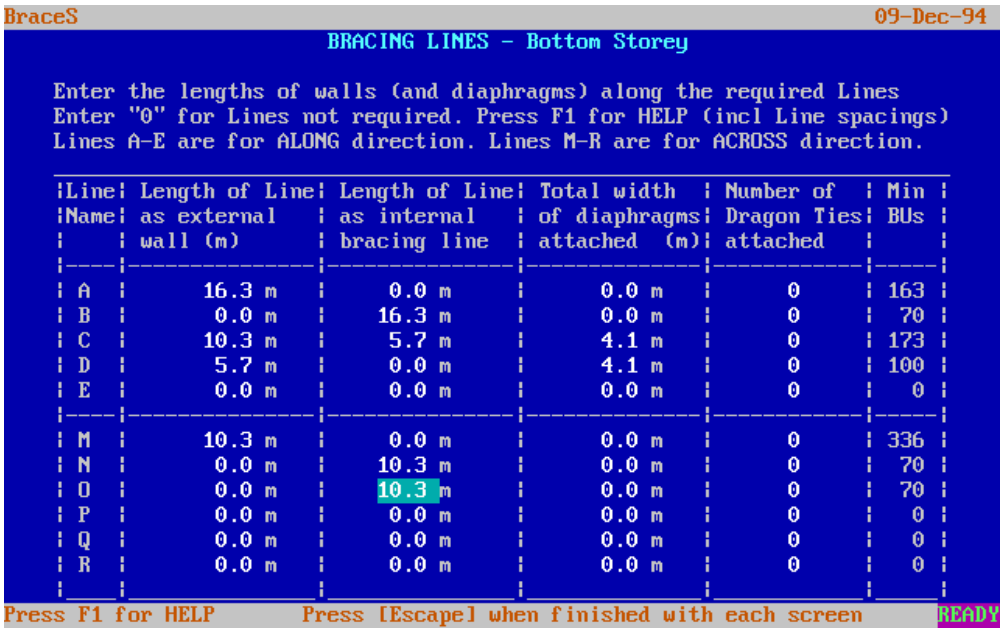

Figure 11: BraceS Bracing Unit calculation phase

FUTURE DEVELOPMENTS

Research and Development

Research at UACS is concentrating on the integration framework for tools in the building domain. The most immediate goal is to complete the current incarnation of the ICAtect system to provide a working platform of integrated conformance and simulation tools. Following on from the completion of this platform will be research into several connected areas which will make the systems more adaptable to individual users needs and the actual projects that they are working on.

The development of support tools for the specification of tool schemas, their connection with the common schema, and possible updates to existing tools is one area of interest. Existing tools generally have poor support for multiple developers working on the same model. Handling concurrency necessitates the addition of versioning control and policies for handling collaboration between the various developers.

Extending the control structures in integrated systems to handle notions of process, and user modelling and control are planned extensions to the ICAtect system. These additions will allow configuration of integrated systems for specific projects, taking into account the actual people who will be involved in a project, their responsibilities and time frames for completion.

Commercial Products

The commercial products planned for the future feature larger institutions in the building industry than those involved in the current computerised standards projects. These players have been working with paper based systems or computerised database systems up till this point. Having gauged the New Zealand market for some years, they have decided that the time is now ripe for further

developments in this area. Reflecting their background in the industry, the three commercial developments that are discussed in this section feature integration of services as part of their overall development strategy.

The Building Industry Authority (BIA), the body responsible for the NZBC, has recently polled the principal users of the NZBC and determined their requirements for using the code. Based upon this survey, the BIA has decided to make an electronic version of the code available to practitioners. To achieve this, the BIA has put out to tender the development of either a fully on-line system, or a mix of on-line and CD-ROM based system that must supply the full 35 approved documents comprising the code as well as verification methods and acceptable solutions. One of the BIA's major concerns in computerising the building code is that the electronic form is easily mixed and linked with other existing forms of electronic data used by building practitioners. Initially the proposed system will be a purely electronic copy of the code as it exists in paper form, though embodied with enhanced searching capabilities. The initial system will not include codification of verification procedures as exist currently for some of the standards, though the BIA hopes to extend the scope of the system to this level at a later date either through funded projects or through development by some of New Zealand's larger institutions and societies.

SPECTEL is one of the major suppliers of electronic information for the New Zealand building industry. Currently they supply information on products and companies as well as information on approved products, existing standards and news from the industry. They have expanded rapidly over the last few years and are planning to develop a more integrated system, heading towards the ability to supply all the building related information a user may require from a single electronic source. To this end they are planning to offer electronic versions of all New Zealand standards as well as the NZBC and other relevant legislation (subject to suitable copyright arrangements). This service will allow users free viewing of the standards, incorporating both text and graphics, but with a fee when the user downloads a copy (comparable to a sale of the standard). Their other major drive in providing a single source of information is the addition of Intelligent Design Systems (IDS) to their system. These are packages, written by suppliers listed in the SPECTEL service, to aid in the correct selection of products for a particular design situation. At the moment, there are 20 IDS packages available for products including fasteners, security systems, paints, gas systems, whiteware, masonry and timber products. For some of the IDS systems it is necessary to check a contemplated design for compliance with portions of a building code before selecting the product to use. SPECTEL expect more IDS systems that provide code compliance checking as part of their selection process to be developed in the next few years.

In order to perform compliance checking from a CAD system, the group at the School of Architecture at Victoria University who developed the computerised version of the ALF Design Manual for BRANZ are investigating its integration with the Archsoft architectural design package (based on AutoCAD). In a manner similar to the Archsoft-Energy product, developed by Pacific Northwest Labs to provide energy analysis of a design, this development will provide an environment where a commercial CAD based design can automatically be checked for compliance with NZS 4218P through the ALF method.

DISCUSSION AND CONCLUSIONS

There has been an evolution over the last ten years of the research and development carried out in New Zealand in the development of computerised compliance checking systems and frameworks for their description and integration. Commercial development of code compliance checkers has begun with a few products suited exclusively to the New Zealand environment. The work to date has highlighted several desirable features of environments supporting computerised code systems, namely:

- A functional representation of code provisions
- The usefulness of object-orientation in code representation.
- The need for flexible user interface tools.
- The need for an integrated framework which manages the consistency of data across connected tools.
- The need for integrated systems to manage more information than purely building related data such as design processes and user models.

Following on from the formulation of the NZBC, commercial interest in making code information available to New Zealand users has increased. However, all developers have realised that some level of integration between applications must be supported. This has lead to current and proposed projects where CAD and compliance checkers are integrated, or where the electronic supply of code text is offered in a form that can be readily integrated with other common applications.

ACKNOWLEDGMENTS

The first author acknowledges the support of the University of Auckland Research Committee and the University of Auckland for their PhD scholarship. The authors also acknowledge the financial support offered by BRANZ and the New Zealand Foundation for Research Science and Technology.

REFERENCES

Amor, R., Augenbroe, G., Hosking, J., Rombouts, W., Grundy, J., 1995, Directions in Modelling Environments, accepted for publication in *Automation in Construction*..

Amor, R., 1994, *A Mapping Language for Views*, Department of Computer Science University of Auckland Internal Report, Auckland.

Amor, R.A., Hosking, J.G., Mugridge, W.B., Hamer, J., Williams, M., 1992, ThermalDesigner: an application of an object-oriented code conformance architecture, *Proc* Joint CIB Workshops on Computers and Information in Construction, International Council for Building Research Studies and Documentation (CIB) Publication 165, Montreal, 1-11.

Amor, R., Hosking, J., Donn, M., 1993, Integrating Design Tools for Total Building Evaluation, *Building and Environment,* **28**(4), 475-482.

ATLAS, 1993, *ATLAS: Architecture, methodology and Tools for computer integrated LArge Scale engineering*, ESPRIT 7280, Delft.

Augenbroe, G., Laret, L., 1989, *COMBINE pilot study report*, CEC-JOULE Report.

Bailey, I., 1994, *EXPRESS-M Reference Manual*, Product Data Representation and Exchange, ISO TC184/SC4/WG5 N51.

Bassett, M.R., Bishop, R.C., van der Werff, I.S., 1990, *ALF Manual, Annual Loss Factor Design Manual, An aid to thermal design of buildings*, Building Research Association of New Zealand, Judgeford, New Zealand.

BRANZ, 1993, *ALF Design Manual: aid to thermal design of buildings*, Building Research Association of New Zealand, Judgeford, New Zealand.

Clark, S.N., 1992, *Transformr: A Prototype STEP Exchange File Migration Tool*, National PDES Testbed Report Series, NISTIR 4944, US Department of Commerce, National Institute of Standards and Technology.

Coad, P., Yourdon, E., 1991, *Object-Oriented Analysis*. Prentice-Hall.

Cornick, S., Leishman, D., Thomas, R., 1991, Integrating building codes into design systems, VTT Symposium 125 Computers and Building Regulations, Technical Research Centre of Finland (VTT), Espoo, 210-236.

de Waard, M, 1992, *Computer Aided Conformance Checking*, Thesis Technische Universiteit Delft, Delft.

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

Fenves, S.J., Wright, R.N., Stahl, F.I., Reed, K.A., 1987, *Introduction to SASE:Standards Analysis, Synthesis, and Expression*, NBSIR 87-3513, U.S. Department of Commerce, National Bureau of Standards.

Fenwick, S., Hosking, J.G., Mugridge, W.B., 1994, *Cerno-II: A program visualisation system*, University of Auckland, Department of Computer Science Report No 87, Auckland.

Hamer, J., Hosking, J.G., Mugridge, W.B., 1992, *Static subclass constraints and dynamic class membership using classifiers*, Auckland Computer Science Report No 62, Department of Computer Science, University of Auckland, Auckland.

Hamer, J., Mugridge, W.B., Hosking, J.G., 1989, Object-oriented representation of codes of practice, *Proc* of the Australasian Conf. on Expert Systems in Eng. Arch. and Construction, Sydney, 209-222.

Hardwick, M., 1994, *Towards Integrated Product Databases Using Views*, Technical Report 94003, Rensselaer Polytechnic Institute, Troy, New York.

Hosking, J.G, Hamer, J., Mugridge, W.B., 1990, Integrating functional and object-oriented programming, *Proc* TOOLS Pacific '90, Sydney, 345-355.

Hosking, J.G., Hamer, J., Mugridge, W.B., 1991, *Kea 1.0 Tutorial Manual,* BRANZ Contract 85-024, Technical Report No18, Department of Computer Science, University of Auckland, Auckland.

Hosking, J.G., Lomas, S., Mugridge, W.B., Cranston, A.J., 1989, The development of an expert system for seismic loading, *Civil Engineering Systems*, **6**(1-2), 27-35.

Hosking, J.G., Mugridge, W.B., Buis, M., 1987, FireCode: a case study in the application of expert system techniques to a design code, *Environment Planning and Design B*, **14**, 267-280.

Hosking, J.G., Mugridge, W.B., Blackmore, S., 1994, Objects and constraints: a constraint based approach to plan drawing, Technology of object-oriented languages and systems TOOLS 15, Prentice Hall, Sydney.

Hosking, J.G., Hamer, J., Mugridge, W.B., Dechapunya, A.H., 1990, From FireCode to ThermalDesign: KBS for the building industry, *New Zealand J Computing*, **2**(1), 23-32.

ISO/TC184, 1993, *Part 1: Overview and fundamental principles in Industrial automation systems and integration - Product data representation and exchange*, Draft International Standard, ISO DIS 10303-1, ISO-IEC, Geneva.

Leler, W., 1988, *Constraint Programming Languages Their Specification and Generation*, Addison-Wesley.

Mugridge, W.B., Hosking, J.G., 1988, The development of an expert system for wall bracing, *Proc* 3rd New Zealand Expert System Conference, Wellington, 10-27.

Mugridge, W.B., Hosking, J.G., 1995, Towards a lazy, evolutionary common building model, *Building and Environment* , **30**(1), 99-114.

Price, C.G., 1994, The development of class libraries for building product model development, Technical Computing, *The ACADS Journal*, **80**, 13-15.

Rosenman, M.A., Gero, J.S., 1985, Design codes as expert systems, *CAD*, **17**(9), 399-409.

SANZ, 1977, *NZS 4218P 1977: Minimum thermal insulation requirements for residential buildings*, Standards Association of New Zealand, Wellington.

SANZ, 1990, *NZS 3604 1990: Code of practice for light timber frame buildings not requiring specific design*, Standards Association of New Zealand, Wellington.

SANZ, 1991, *BraceS the computer program to use with NZS 3604-1990*, Standards Association of New Zealand, Wellington.

SANZ, 1992, *The New Zealand Building Code*, Standards Association of New Zealand, Wellington.

Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T., 1986, The British Nationality Act as a logic program, *Comm. ACM*, **29**, 370-386.

Turk, Z., 1991, Building model standard as a basis for computer integrated design, VTT Symposium 125 Computers and Building Regulations, Technical Research Centre of Finland (VTT), Espoo, 181-194.