

# **Chapter 1**

## **Introduction**

### **1.1 The Problem Domain**

In the fields of architecture and building engineering there is continued dissatisfaction with the state that computerised design has reached. The majority of the claims made for computerisation in these fields have never been achieved. For architects there are few, if any, tools which are useful for initial or sketch design and in many architectural firms the only use of computers during the design process is for detailing the building plans with a draughting tool (CAD). In the engineering domain there are similar problems. Even with a computerised plan of the building it is unlikely that building information can be extracted from this plan for use in other design tools. Many practitioners re-enter building information into their design tools by hand, taking measurements off the printed plans. Despite the existence of a wide range of excellent design tools, such tools are seldom used because of the difficulty and time required to enter building information in order to invoke the tool.

Examining the state of computing in these domains it is clear that the majority of the problems lie with the conceptual models of a building. In a wide range of CAD tools there is no underlying model of a building. That a plan defines a building can only be ascertained by human perception. To the computer a plan is a set of 3D graphical entities. Until computers can mimic the human ability to perceive building structure from a plan, it is impossible to extract useful building information from a traditional CAD system. There is also a problem with the wide range of design tools that model buildings. Each of these design tools has a schema of a building most suited to the type of computation it performs on the building, and in most cases this is a minimal schema to ease

the burden of data entry for the user. As can be imagined, this means that schemas of buildings for tools such as thermal simulation, structural simulation, lighting analysis, fire-code compliance, quantity surveying, etc. have very different structures. As a result, detailing a building for use with two different design tools is like trying to explain exactly the same thing in two totally different languages.

Another important problem is that of consistency and coordination. In a building project there are often several design professionals responsible for different parts of the design. A project manager has the task of coordinating the various designers, and in a manual design practice a range of procedures are used to ensure that the design is kept consistent between all designers. In a computerised design environment it has become no easier to perform this task, and in many cases much more difficult, as the number of plan portions which can be generated with computerised tools is much greater than with manual design.

One obvious solution to the problems outlined above is to find some way to integrate the design tools with a comprehensive conceptual building schema used to transfer building data between the different design tools. There have been several approaches to integrated systems which offer various benefits to the designers working with them. One approach is an integrated design assistant in which it is envisaged that the integrated system would offer a single designer the ability to design utilising any of the various design tools as assistants to check various portions or stages of the design. The integrated assistant would manage the transfer of data to and from the design tools and maintain the consistency of the global model of the designed artifact. Another approach is an integrated design system in which it is envisaged that the system would connect multiple designers working on the same artifact, maintaining the global consistency of the design as it evolves over time. This system would still allow designers to experiment with design solutions independently from other designers (as in the integrated design assistant), but maintain the global model with respect to solutions exported by the various designers. This system would also manage process control to ensure that all design functions were completed during the design and to control the time line of the design process.

The increasing demand for intelligent integrated computerised working environments in architectural and building engineering domains has spawned a multitude of research projects tackling portions of the problem. The main benefits these systems claim to offer their users are:

- Access to all the pertinent information for their portion of the design task
- Access to a range of appropriate design tools which can be utilised in the design process
- Project control management to direct the project development and ensure timely completion
- Automated maintenance of the consistency of the design amongst all designers in a project
- Ability to experiment with design modifications independently before submitting a suitable modification to the global model

- Notification of changes to the global model which affect the design tasks of a particular user
- Management and notification of conflicts between the work of various users

While the benefits noted above exist in current managed projects there are several other flow on benefits possible from an integrated system:

- Final design is likely to be more correct as all designers have been working on the same global model of the artefact
- Final design may be more innovative as designers have the ability to examine and test multiple design solutions to a particular problem through connected design tools
- Possibilities for better performance of the completed artefact as designers have many design tools available to check the properties of the artefact being designed and to evaluate design options quantitatively

There is much prior research in this area of integrated design systems, as will become clear in the next section. The next section will also detail the many other areas of computer science research which are applicable to an integrated design system. To help illustrate the place of the research encompassed in this thesis the author has chosen one of the EU funded projects (COMBINE) as an example of an evolving integrated design system. The framework of the COMBINE system is similar to most other integration projects and is used due to the author's familiarity with the project from a six month period of work in Europe.

## 1.2 Related Research

Whilst this thesis examines an overall framework to allow the development and testing of an integrated design system, the majority of research to date has concentrated on problems associated with the individual components of this framework. These components include developing data models, defining mapping languages, managing model development, etc., and the work in these areas is considered in Chapters 2 and 8 of this thesis. What little work that has looked at overarching development frameworks is considered below.

In the architecture, engineering and construction (A/E/C) domains there has been intense European Union work on the development of integrated design systems since the late 1980s. This has culminated in a range of projects for various subdomains of A/E/C, for example, COMBINE (Augenbroe 1995a; Augenbroe 1995b), ATLAS (1993; Greening and Edwards 1995), COMBI (1995; Scherer 1995), CIMsteel (1995; Watson and Crowley 1995) and more recently ToCEE (Katranuschkov et al. 1996), STAR (Huovila and Seren 1995) and VEGA (1996). These projects started out examining frameworks purely for data integration, but over the years recognised the need for other project aspects, and broadened their scope to process, documents, legality, etc.

Despite their current wide scope very little work has been done in these projects to consider the framework required to ease the development of integrated design systems. Very simple tools to specify data models, or process models, all quite independent of each other are as far as any project has gone. The COMBINE project has developed the greatest number of support tools, and some of these are described in Section 1.3 below and highlighted throughout this thesis as examples of current state of the art.

Research projects undertaken by individual academic institutions have often taken a wider view than the more result-oriented European Union research, with the result that they have developed a more comprehensive framework than many European projects (Papamichael and Selkowitz 1991; Lamb 1987; Subrahmanian 1989; Pena-Mora et al. 1993). Once again the majority of these projects develop small prototype integrated systems with models crafted by hand in text-editing tools, rather than developing their own support tools. The projects which do develop support tools for framework development provide very useful and tightly coupled tools, but for very narrow domains (Poyet et al. 1995; Grundy 1993). In the majority of cases these domains are restricted to the data aspects of the integrated design system, ignoring requirements of support for mapping between data aspects, or incorporating process and project aspects into the framework.

### **1.3 COMBINE Project**

Throughout this thesis I will draw upon several examples and use nomenclature which have originated in the COMBINE project. I will also use COMBINE to help illustrate the place of the various systems that are developed and to highlight their utility in such a project.

COMBINE (COmputer Models for the Building INdustry in Europe) is an EU-funded project which started in 1990 (Augenbroe and Laret 1989). The first phase ended in late 1992 with a seminar and workshop at which the first phase deliverables were demonstrated (Augenbroe 1993). The project, which has now completed its second phase (1992-1995), spans a total effort of 70 man-years spread over 12 partners from 7 European countries.

COMBINE worked towards the practical development of IIBDS's (Intelligent Integrated Building Design Systems) through which energy, services, functional and other performance characteristics in planned buildings can be modelled and integrated. The modelling of information has been one of the greatest concerns in this project, with the first phase of the project concentrating on the integration of data to provide the necessary information for a group of actors. COMBINE's efforts have been concentrated on establishing a data infrastructure and tools for managing the information exchange amongst design actors, i.e., members of a collaborative design team. The project thus represents a good example of international research in product data technology and of the development of an integrated design system.

COMBINE is one of many international projects with a similar aim of integrating the information requirements of a given domain to provide an enhanced working environment to users of that domain. A number of parallel projects in the European ESPRIT-CIME (a full list of acronyms used here can be found in the glossary) program are engaged in similar research and development to that of COMBINE. In time all of these projects may have input into the ISO-STEP standard which has the stated aim of covering the information requirements for all architecture, engineering and construction domains (ISO/TC184 1993). In recognition of the importance of STEP, and the practicalities of interchange of results, most projects have chosen the modelling formalisms specified for the STEP standard as the formalisms for their efforts. These include the EXPRESS language, with EXPRESS-G, NIAM, IDEF1X and IDEF0 diagramming techniques, the STEP neutral file format, for exchange of data, and SDAI, the data access specification. This is the case in COMBINE where all modelling is done with these formalisms (Augenbroe 1994).

COMBINE's first phase was concerned primarily with data integration based upon the concept of a set of actors connected to a central common data repository (Figure 1.1). Its deliverables comprised the first large conceptual integrated data model (IDM) for buildings and a number of interfaced design tools (DT). These results formed the base line technology on which the present second phase builds. The second phase was concerned with developing an operational IIBDS according to functional specifications of particular building projects in practice. In doing this, the number of interacting design tools was expanded to cover existing design applications in the area of costing, HVAC-CAD (Heating, Ventilating and Air Conditioning), architectural CAD, component databases, daylighting and building regulations.

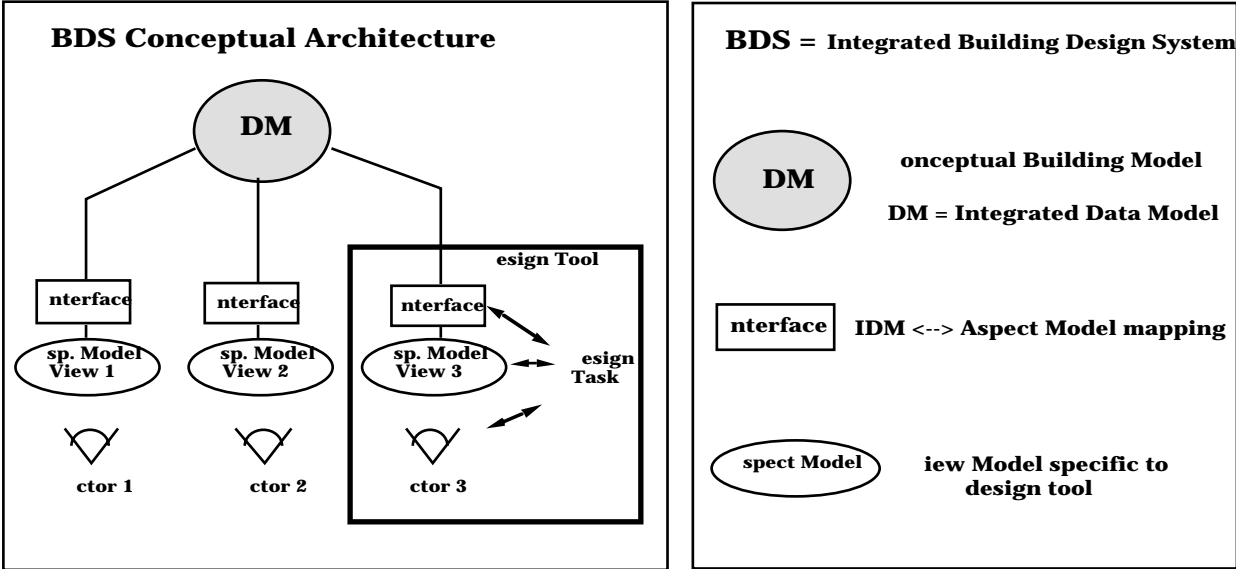


Figure 1.1 COMBINE structure

In the first phase of COMBINE the types of actors considered were limited to those in the fields of energy and HVAC performance in the early design stages of building design. However, even limiting the domain to this small set of actor types, the IDM developed for this project consisted of some 400 entities and 600 relationships between them. The development of this conceptual schema

was a complicated task. The final schema had to incorporate the data requirements of the seven design tools which were used in this first phase, which had very different data requirements to model a building for their simulation needs. The IDM was developed by first modelling the schemas of the various design tools, and then, through schema analysis and use of integration techniques, various portions of the schemas were brought together to help form the final IDM. This was a very time consuming and difficult task. In particular, the developers in the project found it impossible to verify the consistency of the IDM without the aid of computer tools (Dubois 1993).

The second phase of COMBINE involved integration of several more design tools, widening the scope of the IDM. This required a redesign of the IDM to extend the domains supported. Again, the process of development involved comparison of the IDM with the schemas of the various design tools being supported, and incorporation of portions of the design tool schemas into the IDM. The resulting iterative process of schema analysis, integration, and prototyping continued until the various design tool support teams were satisfied that the data requirements necessary for their tools could be met by the IDM.

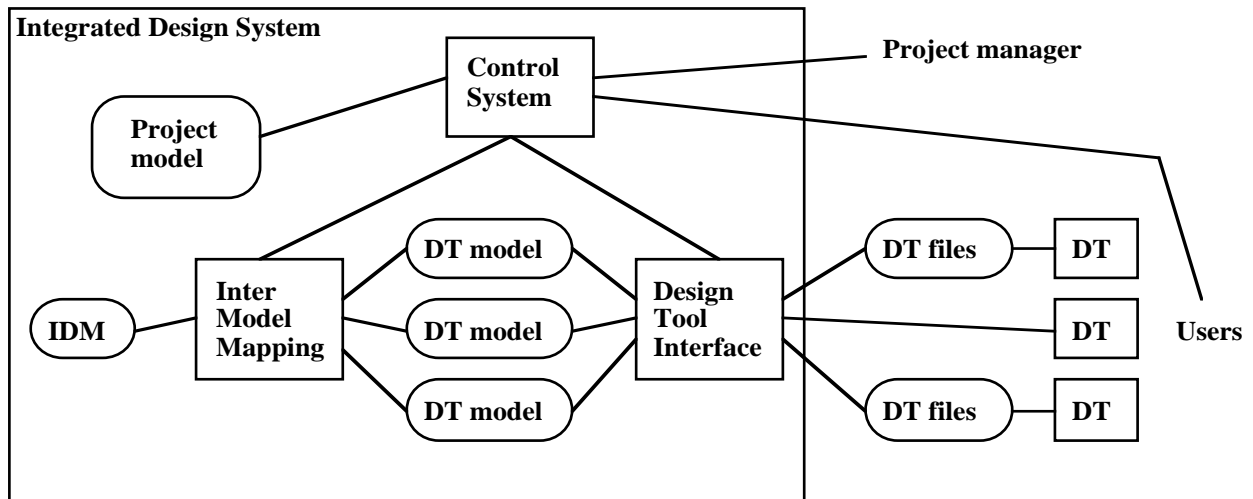
The modelling support presently offered to partners in COMBINE comes through the use of the Configurable Graphical Editor (CGE, see Vogel 1991). CGE can be configured to support a number of diagramming techniques, among which are those mentioned at the start of this section. It also supports a customised version of NIAM called ATLIAM (Vogel 1991) which is capable of exporting an EXPRESS schema of a NIAM diagram and of exporting and importing NIAM diagrams by way of STEP files.

## **1.4 Underlying Framework**

The creation of an integrated design system requires an enormous amount of work constructing specialised modules in a framework of great complexity. The development of a complete framework by one person is obviously unobtainable in a lifetime. However, the general structure of integrated design systems appears fairly stable and it is particular portions of the framework which require research and development to find optimal solutions. In virtually all integrated systems there are a set of common fundamental systems as illustrated in Figure 1.2.

When an integrated design system is in use its schemas and design tools are likely to vary from project to project. This is because a building design project is usually involved with a one-off design and: the group of designers involved; the aspects of the design each designer works on; the design tools required; and the flow of control for the project, are likely to change between each project. This produces a requirement that the integrated system is easily configured for a varying set of designers and design tools. Of more significance is the recognition that to properly cope

with this situation it will be necessary to modify and add schemas to the system. The designer's schemas will have to be changed to reflect the designer's roles in the current project, a new flow of control (project) model will be required for each project, and perhaps new design tool schemas will have to be introduced to cover checking of unusual aspects of the current design (which could in turn affect the scope of the IDM). This leads to the requirement that the final integrated system must contain the modelling tools used in the design of the initial system to enable the system to be configured for individual projects.



**Figure 1.2** Structure of an integrated design system

The integrated design system can be implemented in a variety of ways from a monolithic system with all modules tightly coupled through to a completely modular system with links to relational or object-oriented databases and where many of the modules operate as independent systems utilising a brokering architecture to communicate data between modules. However the system is implemented the same conceptual major subsystems exist in the integrated design system. These are detailed further to give some idea of the scope of the task each must handle.

**IDM:** this is the very general schema of the domain which must cover the information requirements of the designers and design tools which interact with the system. The schema required for buildings is probably the largest and most complex information model developed in the world to date. The ISO 10303 standards committee (STEP) has laboured for over 10 years to define the underlying schemas for graphical, draughting and material representations. Only recently have they moved to create definitions of higher level entities for buildings, ships, etc. Numerous projects have used schemas based around those required for design tools, but expanded to cover requirements of other design tools. Recently several collaborative projects have worked on general schemas of buildings based on the STEP fundamentals for restricted aspects of buildings. Each of these projects has invested in the order of 10 person years of effort in their restricted schemas. The development of the IDM requires the input of domain experts for all the sub-domains being modelled as well as the help of modelling experts to structure the schema. The IDM remains as a fairly static schema, although the introduction of new design tools, construction techniques and building constructs may require modifications to the schema.

The development of an environment to specify the IDM draws upon work in data modelling languages, modelling environments, collaborative work systems and schema integration techniques.

**Project model:** this is a project specific model defining the designers who will be working on a project, the design functions they will need to perform and the design tools that will be utilised during the design process. The project model can also be used to define the flow of control in a project, specifying hand-over points between various designers and sequences of specification that must be followed to help guarantee an optimal design at the termination of the design process. As the design process is not a fully understood process it can not always be completely described at the start of a project. Therefore, it is likely that during the course of the design, new designers may need to be involved in the design or new design tools used for difficult aspects of the design. To enable this flexibility the project model must be modifiable during the project to be able to take into account changes in personnel and new flows of control. The development of an environment to specify the project model draws upon work in process modelling languages, modelling environments and collaborative work systems.

**DT schemas:** each of these schemas defines the structures and data requirements of a design tool. The type of design tool and its capabilities will determine the number of schemas that are required for each tool. In most cases two schemas are used, one to specify the data input to the design tool, the second to specify the data output from the design tool. The creation of these schemas is a task undertaken by modellers who have great familiarity with the particular design tool being modelled. This is because the schema must capture not only the data structures explicitly defined by the tool and the data constraints specified by the design tool (e.g., ranges on attribute values, cardinality constraints on lists, etc.), it must also contain the constraints defined implicitly by the design tool (e.g., that all walls are vertical). A fully specified DT schema allows the system to determine whether there is enough data to create the input file for the DT to execute and to determine whether a semantically correct description of the building can be created from the data available. The DT schema remains static throughout the use of the same version of the DT. The development of an environment to specify the DT schemas draws upon work in data modelling languages, modelling environments and collaborative work systems.

**Control system:** this directs both the inter-model mapping and design tool interface modules to perform their tasks as required by the designers or directed by the project manager. This module simulates the flow of control defined in the project model determining what design functions are able to be performed at any particular time. It also interfaces with designers to let them specify the tasks that they wish to perform next and interfaces with the project manager for decisions upon which designers should be involved in new phases of the project. This system directs the inter-model mapping module to map data between different models, or to ascertain whether it is possible to perform the mapping. It also directs the design tool interface to invoke a design tool with appropriate data and to collect results

upon termination.

**Inter-model mapping:** this module performs the translation of data between the IDM and the DT models. It must determine whether it is possible to create a consistent model from the IDM for any one of the DT models based upon the constraints in that DT schema, and it must ensure that the IDM remains consistent upon update from a design tool's output. It must detect conflicts between various sets of data and must be able to invoke processes to enable negotiation over conflicting data from different design tools and users. Traditionally this module is implemented by hand coding the mappings required between the IDM and each new DT schema. In future integrated design systems it is expected that this module will be realised by modelling the mappings required between the IDM and each new design tool schema. The development of the inter-model mapping module draws upon work in mapping modelling languages and modelling environments.

**Design tool interface:** this module provides the connection between the design tools utilised in the system and their data models. For an off-line DT this module must be able to create the input files required for the DT (from the DT schema in the system), it must then be able to invoke the DT to perform its work and finally retrieve the output from the DT into the DT model in the system. For interactive DT's it must perform the same task but driven by the demands of the design tool. Traditionally this module is implemented by hand-coding the parsing and pretty-printing for each new design tool. In future integrated design systems, and for some classes of design tools, it is expected that this module will be realised by modelling the interaction required by each design tool (for example the data-file structures) and using this model to drive the interaction with the DT. The development of the design tool interface draws upon work in parsing, pretty-printing, interaction and structure modelling languages and modelling environments.

**DTs:** these tools exist outside of the integrated design system. The types of design tools that can be found here include the interface a designer has to the system, off-line and interactive simulation tools, interactive knowledge-based systems, CAD systems, etc.

## 1.5 Research Objectives

Although the basic framework of an integrated design system as described above is fairly universally applied in integrated projects and although many hundreds of man years have been put into these projects there are many aspects of the framework which are poorly understood and not well modelled. To date the majority of the structured effort in integrated projects has been put into the development of the IDM by the domain experts in the project. Systematic modelling of other facets of the integrated system has been bypassed in favour of hand-coded modules which work for particular design tools under particular situations. I believe that it is crucial to model all aspects of the components in an integrated system to guarantee the correct behaviour of the finished system. To enable these models to be developed it is also necessary to have powerful modelling

tools which enable the modellers to describe and manage the enormous models that are being created. The exploration of these objectives forms a major component of the work described in this thesis along with an implemented framework which demonstrates how all the modelled systems can be drawn together into an implemented system. An overview of the major objectives which are examined in this thesis follows:

### **1.5.1 Formalism to specify mappings**

To define the correspondence between two schemas it is necessary to use a formalism which allows the specification of mappings in an easy and intuitive manner. This formalism must closely match the problem domain and have a clear syntax and semantics. In current projects this module has received little attention and only recently has any effort been put into the examination of possible formalisms for this task.

### **1.5.2 Formalism to specify project and flow of control**

As a new project specification is required for every project undertaken with an integrated design system it is essential to have a formalism that will allow the easy description of the participants and their roles in the project. In current projects this module has received little attention and only recently has any effort been put into the examination of possible formalisms for this task. This formalism must offer easy specification of the designers and their roles in the project. It must also be able to specify the flow of control during the project and the hand-over points for the various participants.

### **1.5.3 Comprehensive tool set to support schema specification**

With the enormous schemas which are being specified in these projects more sophisticated modelling environments are required to manage the complexity of the schemas. New environments must be able to support schema specification at many levels of abstraction and with many different formalisms. Multiple views of the schema must be available in any of the formalisms with consistency between the views maintained under modification. There must also be the ability to provide annotation of the schema with documentation. Collaborative modelling must be supported to encompass the range of input from modellers who have input to the schema. Version management for the evolving schema must be supported along with version merging and conflict resolution for the versions of various modellers.

### **1.5.4 Inter-model mapping utilising mapping specifications**

With definitions of the mapping required between two schemas it is necessary to show how they can be used to map whole models back and forth, ensuring the consistency of the models as they are worked upon. The inter-model mapping must be able to create new models from an existing model and it must be able to propagate changes between linked models. When both models have

changed it must be able to arbitrate the merging of changes between the models and when there are conflicts in the modifications being propagated it must be able to manage the resolution of those conflicts between the affected parties.

### **1.5.5 Control system utilising project and flow of control specification**

The control system needs to maintain contact with the project manager and the designers involved with the project. The control system must be able to determine what the allowable tasks are at any time and inform the designers of the options that they have available. When a designer chooses to perform a task the control system determines whether it is possible at the given time, if it is the control system invokes the inter-model mapping to get data into the design tool's model and then invokes the design tool interface to run the design tool and extract results which will be fed back into the IDM.

## **1.6 Description of Example**

Throughout this thesis a single example set will be referenced, as required to illustrate points pertaining to the workings of an integrated system. This is distinct from, and in contrast to, the COMBINE example which illustrates the requirements for the framework. This example set is drawn from a contract completed between the Building Research Association of New Zealand (BRANZ) and the computer science department at the University of Auckland (Hosking et al. 1995). The problem domain of the example is the development of an integrated design environment where several disparate design tools are required to work together to provide enough information for knowledge-based systems. The knowledge-based system demonstrated is ThermalDesigner (Amor et al. 1992) which checks a building using the Annual Loss Factor (ALF) method (Bassett et al. 1990, an empirical estimator of the thermal design code (SANZ 1977)). The building layout and materials information required for this knowledge-based system to operate can be supplied from two stand-alone tools which allow for basic layout design and materials definition. A third program is used to visualise a full 3D model of the building with fly-throughs and shaded images.

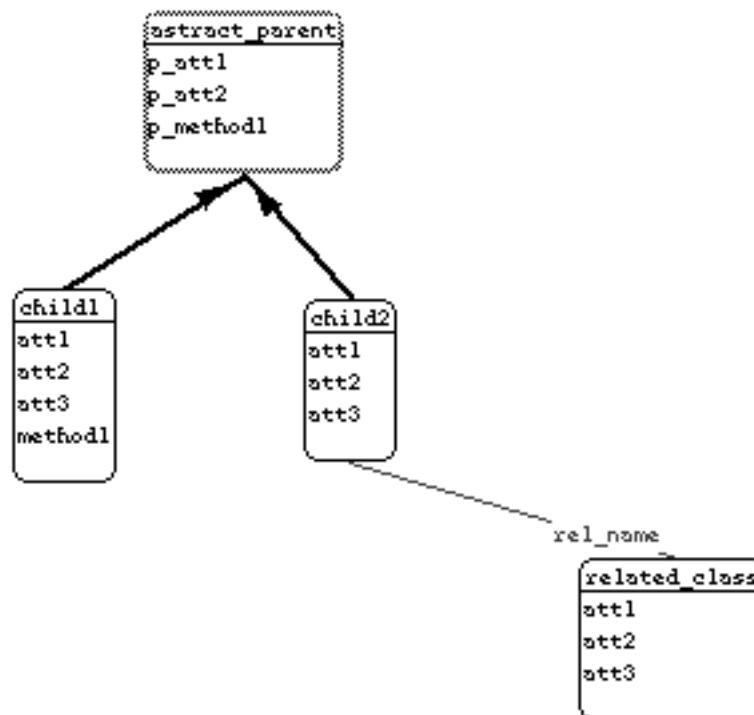
The use of this integrated system in a working design situation would involve several design professionals each charged with different aspects of the design. In this thesis we will envisage a design team consisting of four professionals, an architect, a thermal engineer, a daylighting engineer, and a structural engineer along with a client. The architect would be involved with the initial design of a building after liaison with the client and coordination of the three engineers leading to final design approved by the client. The thermal engineer would select and test various materials for elements of the building to ensure compliance with the thermal code. The structural

engineer would design and approve the structural elements of the design and ensure compliance with the structural code. The daylighting engineer would choose appropriate glazing materials and layout to ensure adequate daylighting whilst checking for potential overheating problems.

The three tools from the example are introduced in the following sections, along with the integrated data model (IDM) through which they will communicate. Following the tool descriptions there is a small analysis of the correspondences between the various design tool schemas and the IDM to give the reader an understanding of the type of information that needs to be mapped between the tools. The full schemas of the tools, the full mappings and the design team interaction specification can be found in Appendix E.

### 1.6.1 The Snart language

All of the development work presented in this thesis has been written in the Snart language (Grundy 1993) and LPA Prolog (LPA 1995). Snart is an object-oriented language, built on top of LPA Prolog, providing notions of abstract and instantiable classes, along with typed attributes and relationships, and class methods. Multiple inheritance is supported, as well as dynamic object classification. The Snart language has a graphical notation in addition to its textual form, and diagrams in this notation are used to illustrate work in this thesis both in this chapter and throughout the rest of the thesis. To allow those unfamiliar with Snart to understand these diagrams, a brief introduction to the Snart graphical notation is provided here using the diagram of Figure 1.3.



**Figure 1.3** An example of the Snart graphical notation

Figure 1.3 shows four classes. A class is represented by a rounded-rectangle. The class name appears at the top of the rounded-rectangle, separated from attribute and method names by a single horizontal line. Figure 1.3 shows an abstract class (a class not able to be instantiated), through the use of a grey edged rounded-rectangle, here named *abstract\_parent*. All other classes can be instantiated and are shown with a solid edged rounded-rectangle. Attribute and method names of a class are shown in the lower portion of the rounded-rectangle. No distinction is made between the representation of attributes and methods in the graphical notation. Inheritance between classes is shown through the use of a thick line with an arrow pointing to the parent class. Figure 1.3 shows *child1* and *child2* both inheriting from *abstract\_parent*. Relationships between classes are shown through a thin line connecting two classes. The line is labelled with the relationship name closest to the class which is being referred to. In Figure 1.3, *child2* has a relationship named *rel\_name* with the class *related\_class*.

### 1.6.2 PlanEntry

PlanEntry (Hosking and Mugridge 1994) is a CAD-like tool to define building layouts. PlanEntry allows multiple 2D views of a base 3D model of a building (see Figure 1.4 for a snapshot of PlanEntry in use). Buildings which can be described in PlanEntry comprise: spaces; internal walls; openings; and roofs. View lines, as shown in Figure 1.4, can cut through any section of the building being described to provide a cross-section of the building. PlanEntry is a constraint-based system, allowing correspondences to be described between spaces, spaces and roofs as well as roof lines (e.g., the connection of the roof sections to form an L shaped roofline in Figure 1.4).

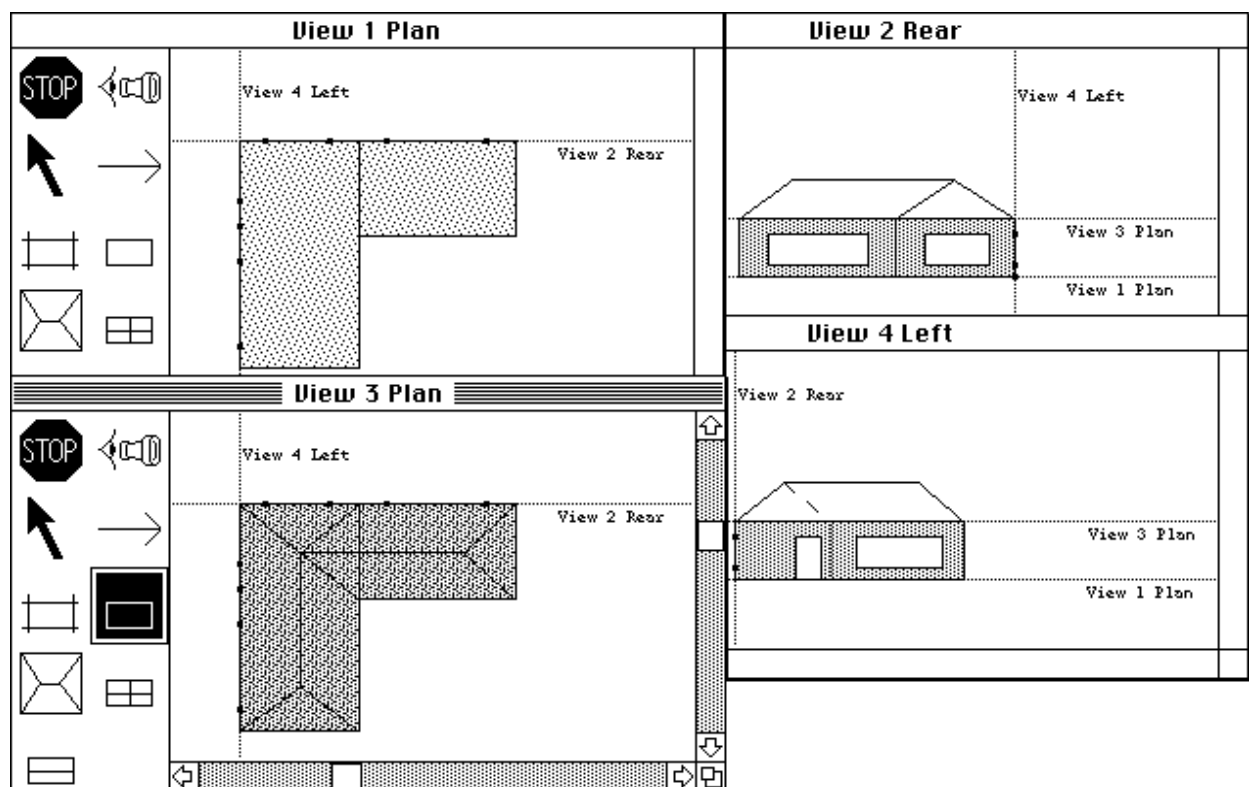
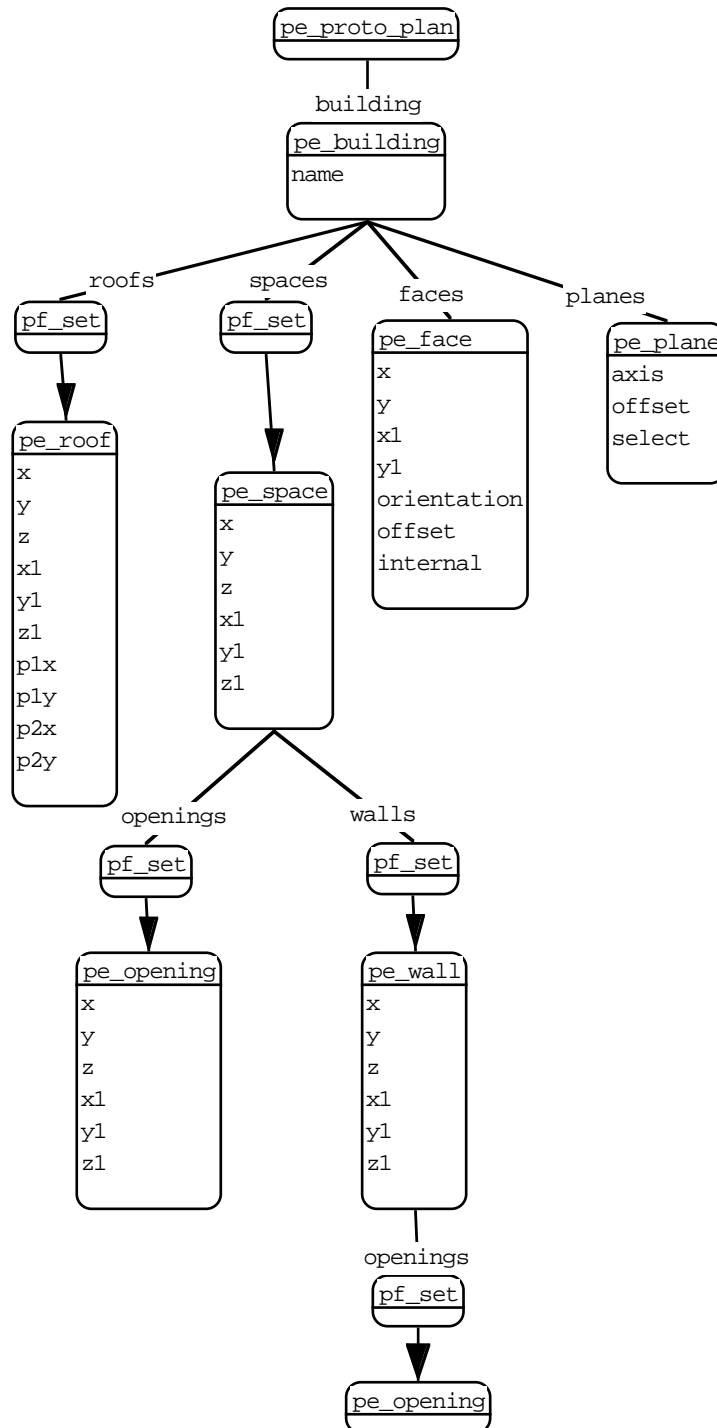


Figure 1.4 A building described in PlanEntry

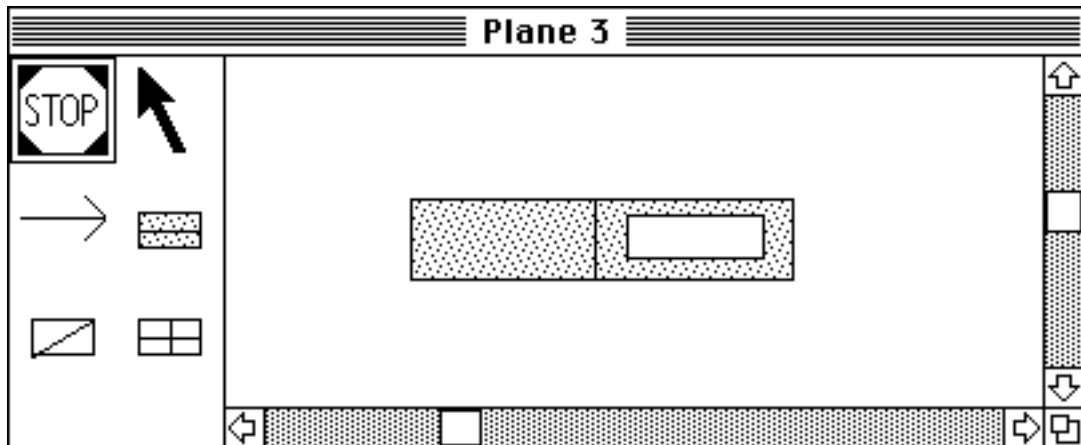
The internal model utilised in PlanEntry is simple. A building is described as a set of spaces and roofs. Spaces can have openings and internal walls. PlanEntry includes a face generating mechanism which takes a building design and identifies all planes along which a wall lies. Using these planes all homogeneous faces for the spaces and roofs of the building are generated. A homogeneous face is a rectangular face which either wholly belongs to one space, or is the interface between two spaces (e.g., an internal wall). The main classes in PlanEntry are shown in Figure 1.5 (note that in this diagram the thin lines with an arrow represent components, e.g., a *pf\_set* containing *pe\_roof* elements, rather than inheritance as in Figure 1.3 with the thick lines).



**Figure 1.5** The PlanEntry schema main classes

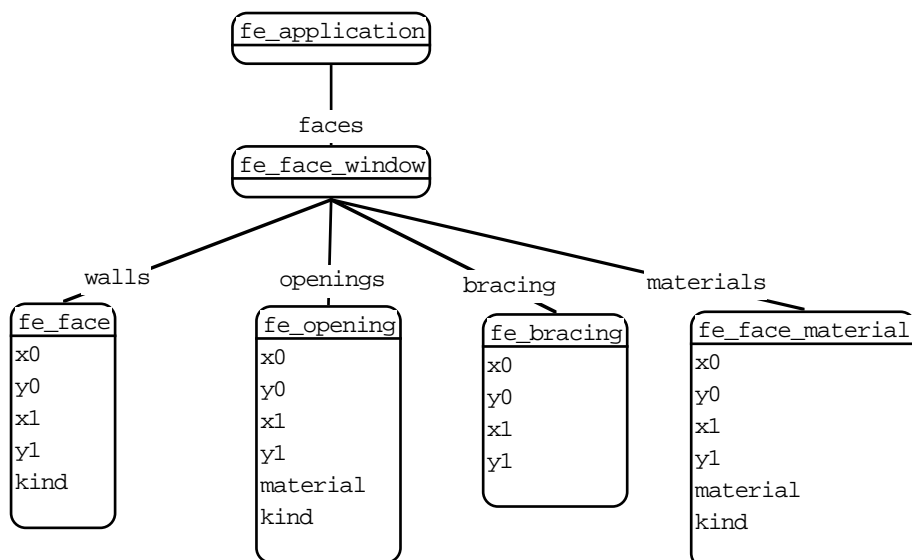
### 1.6.3 FaceEditor

The FaceEditor enables material properties to be assigned to portions of walls and windows which lie along a single plane (see Figure 1.6). A plane in the FaceEditor equates exactly with a 2D plane-line which can be described in PlanEntry. The FaceEditor also allows the specification of bracing materials and bracing properties of walls in a plane. FaceEditor is a constraint-based system, allowing correspondences to be described between sections of material as well as windows and bracing lines.



**Figure 1.6** Wall and window materials being defined in the FaceEditor

The internal model utilised in FaceEditor is very simple. A plane being manipulated in a window consists of wall and window objects which are parallel to the plane (although they need not be on the plane) and rectangular sections of materials and bracing. The main classes in PlanEntry are shown in Figure 1.7.



**Figure 1.7** The FaceEditor schema main classes

### 1.6.4 VISION-3D

VISION-3D (Bourke 1989) enables a 3D representation of a model to be visualised in a wide range of formats. Models may be navigated through static camera placement, or by describing a path for a fly-through of a model. Models may be rendered in formats ranging from wire-frame through to fully shaded. Figure 1.8 shows a wire-frame representation of a model as in Figure 1.4. This tool is used in an off-line manner by importing a data-file containing a geometric description of a building to be rendered.

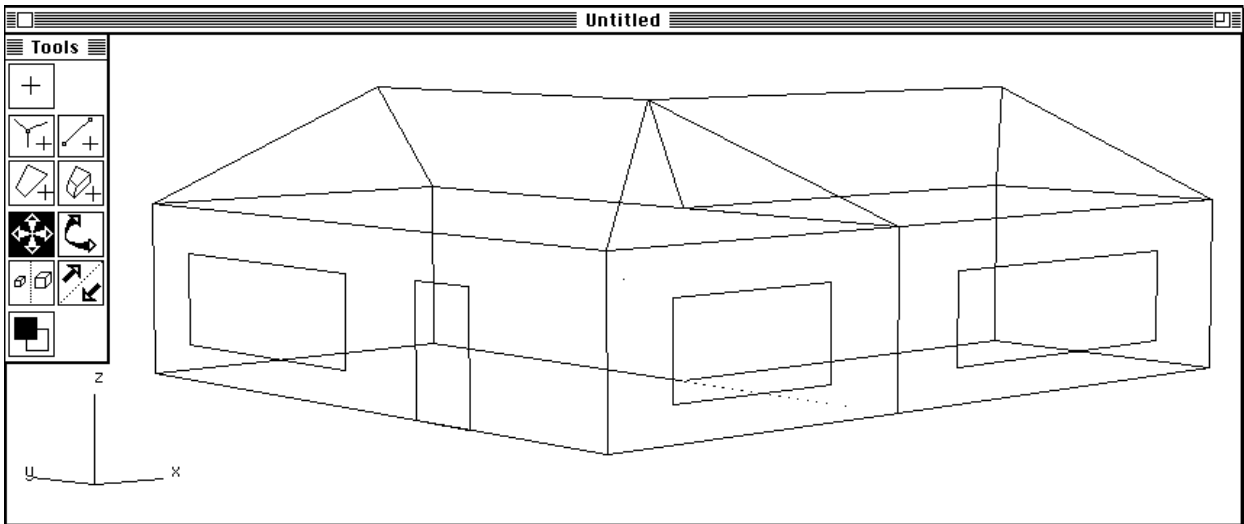


Figure 1.8 A wire-frame representation of Figure 1.3 in VISION-3D

The external model utilised by VISION-3D (see Figure 1.9) has a very simple format consisting only of polygons with some visual properties. Polygons may be grouped together to form more complex objects by specifying the same *object\_id* for each polygon. This feature is used when describing roofs to VISION-3D.

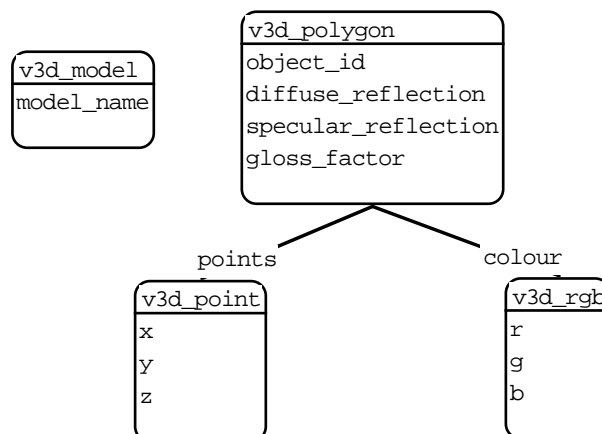


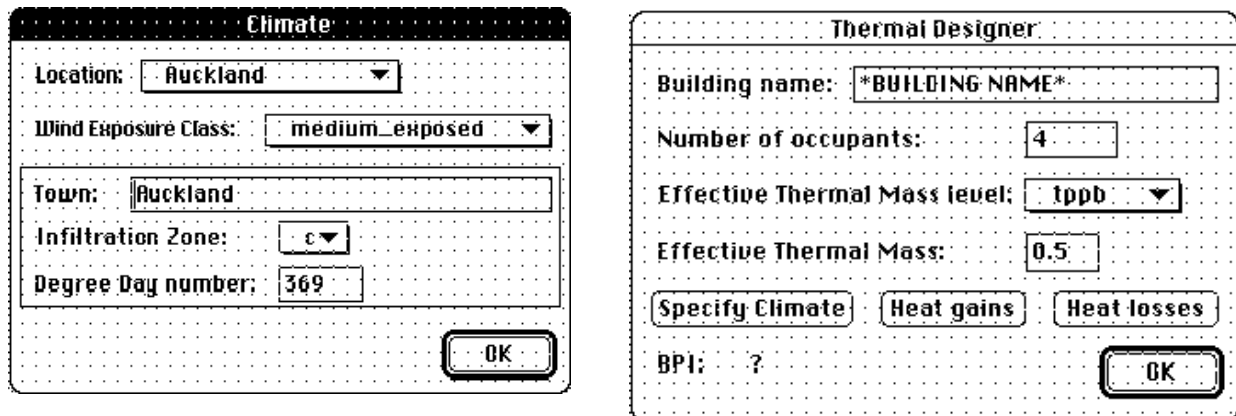
Figure 1.9 The VISION-3D schema classes

### 1.6.5 ThermalDesigner

ThermalDesigner (Amor et al, 1992) helps a designer to check that a building design meets the requirements of the New Zealand thermal insulation standard for residential buildings, NZS4218P (SANZ, 1977). It is based on an approach developed by the Building Research Association of

New Zealand (BRANZ) as a paper design guide (Bassett et al, 1990). The Figure 1.10 shows the forms based interface to the application. The tool has been designed to be used interactively with PlanEntry and FaceEditor providing basic information required to perform the ThermalDesigner analysis.

ThermalDesigner has a very simple model of a building being concerned primarily with the total building area, and the area of walls and windows facing in various directions. To this extent building information from external sources is usually aggregated together to provide the information required by ThermalDesigner.



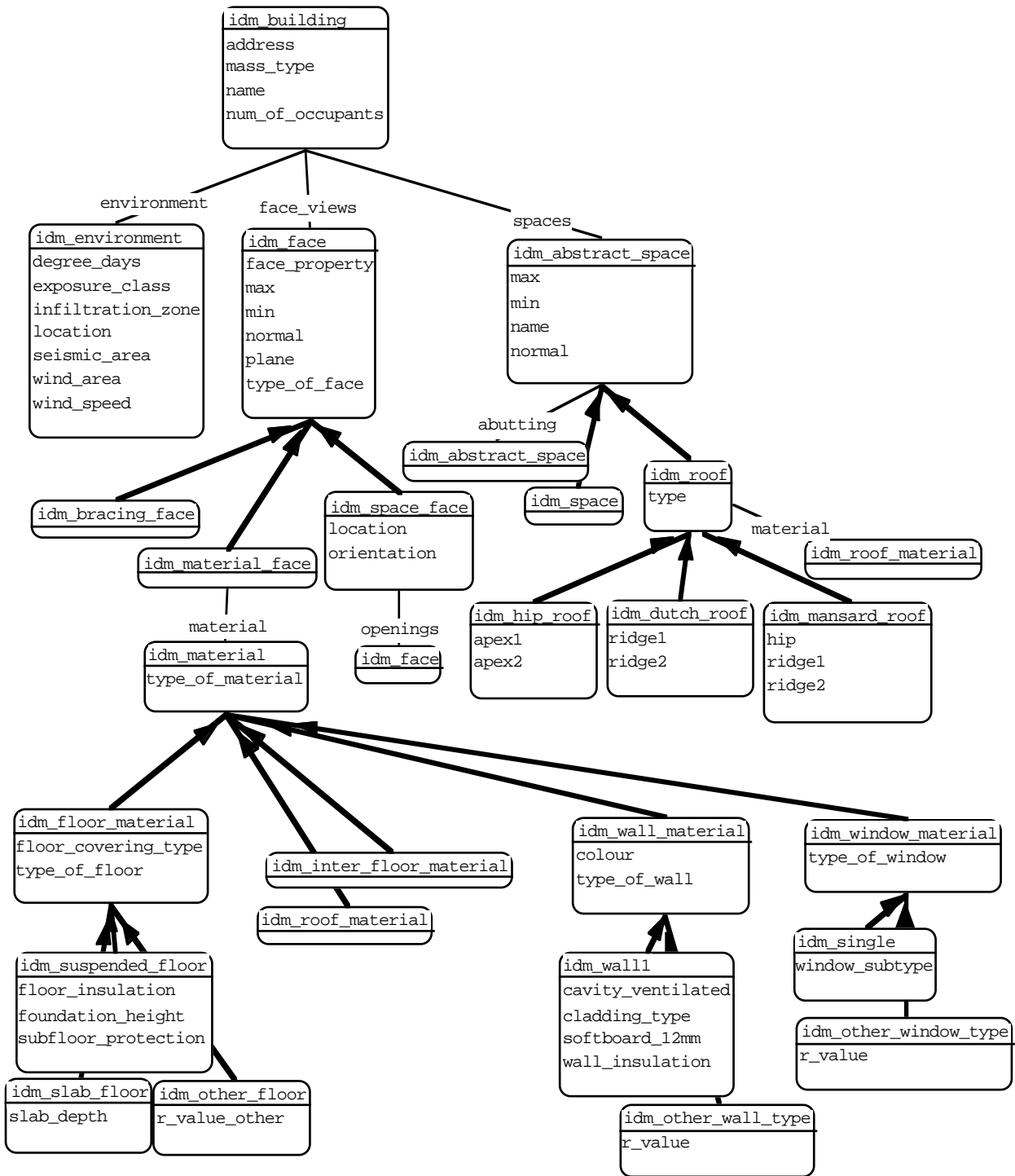
**Figure 1.10** Control windows in ThermalDesigner

### 1.6.6 IDM

The IDM (see Figure 1.11 for a graphical schema) is adapted from a previous integration project (Mugridge and Hosking 1995). The adaptation required the expansion of the old IDM to handle the data requirements of the PlanEntry and FaceEditor packages. The model of a building in the IDM is based on a very generalised and partially redundant notion of a building. A building can be viewed as a set of spaces or as a set of faces. The space view of a building provides for hexahedrons which are placed orthogonally to the major axis. These spaces can be either interior spaces or roof spaces. The main information provided through the space view of the building is the easy determination of space abutment, though this information can be calculated from the face views. The face views break a building into rectangular faces of arbitrary size which have certain properties. In the current IDM the properties which are modelled are geometric, material and bracing properties. An IDM face can transcend space boundaries, and in many cases it is imperative that it does so (e.g., a multi-storey bracing element).

### 1.6.7 Correspondences between schemas

There are many obvious similarities between the schemas of the tools described in this section, along with many differences. The correspondence between each of the design tools and the IDM is further detailed in this section.



**Figure 1.11** The IDM schema classes

PlanEntry’s model of a building maps almost directly into the IDM. Their notions of a building are similar and a PlanEntry space is identical to an IDM space. The roofs that may be created in PlanEntry are hip roofs in the IDM schema. The faces around spaces which are generated in PlanEntry correspond to the geometric faces in the IDM (*idm\_space\_face*) and PlanEntry openings become openings of an *idm\_space\_face* in the IDM.

The FaceEditor's model of a plane also maps almost directly into the IDM. The notion of faces openings, bracing and material in a plane map directly to geometric faces, openings, bracing faces and material faces in the IDM.

VISION-3D has no model of a building, but geometric elements in the IDM such as geometric faces, openings and roofs are easily translatable into the polygons required in VISION-3D.

ThermalDesigner has a very simplified model of a building, being concerned with areas of elements facing certain directions. Therefore, the building structure in the IDM is not found in ThermalDesigner, however, a plane structure is present in ThermalDesigner to represent walls, windows, floors and roofs which maps closely to the IDM.

## **1.7 Outline of Thesis**

Chapter 2 examines the development environment required for an integrated design system. The requirements for such an environment are specified and a possible structure put forward. The requirements of each individual component of this development environment is detailed, along with related research in the area. The individual components are discussed in Chapters 3 to 7.

Chapter 3 looks at what is necessary for a schema modelling and development environment along with the various modelling languages. An implementation of a schema modelling and development environment is presented with demonstrations of how it meets the previously specified requirements.

Chapter 4 introduces the problems of mapping between schemas as defined in environments such as that implemented in Chapter 3. The mapping problem is analysed to provide some measures of what is required and a range of mapping languages are examined to determine their strengths and weaknesses. The chapter summary shows the various language trade-offs and shows a need for a higher-level bidirectional mapping language. Such a language is described in Chapter 5.

Chapter 5 introduces the view mapping language (VML). This language provides a high-level declarative approach to modelling correspondences between schemas. The main constructs of the language are described along with a graphical notation for describing mappings. VML is shown to have a greater expressive power than previous mapping languages.

Chapter 6 details the requirements for a modelling and development environment for mapping languages. The system requirements are very similar to those detailed in Chapter 3 for schema modelling, and a similar approach is followed in demonstrating the utility of such an environment.

Chapter 7 details the requirements for a project specification notation. A developed notation, called CombiNet, is then presented. This notation allows users and their design functions to be modelled, along with flow of control for a project.

Chapter 8 examines the testing and implementation environment required for an integrated design system. The requirements for such an environment are specified and a possible structure put forward. The requirements of each individual component of this development environment is detailed, along with related research in the area. The individual components are detailed in Chapters 9 to 11.

Chapter 9 puts forward the requirements for schema instance management in a design system, looking especially at the requirements imposed by changing schema definitions. A suite of tools developed and used in this thesis are presented to highlight the benefits offered by tools meeting these requirements.

Chapter 10 describes an implementation of an interpreter able to map information between data stores and applications, through specifications in the view mapping language notation described in Chapter 5. Difficulties in implementing the high-level, declarative, and bidirectional, VML notation are identified and addressed throughout the chapter.

Chapter 11 describes an implementation of a control system able to handle a running project through the CombiNet project specification notation described in Chapter 7. This system provides functionality to a project manager as well as to the individual actors in a project by simulating the state of the running project.

Chapter 12 presents the conclusions of the work undertaken, highlighting the advances made in the understanding of the requirements of integrated design system development. A discussion of required future work and research concludes the work.