Building Serv. Eng. Res. Technol. 13(1) 1–11 (1992) Printed in Great Britain

Review Paper

Summary This paper reviews the current state of thermal modelling for buildings, and describes developments in data modelling, object-oriented programming and specific research projects which indicate the likely course of future developments. The limitations of the data structures and processes of current programs are described, in the context of the forces which drive development. Emerging international data exchange standards, which are likely to be a major influence on future engineering software, are examined in the context of building modelling. The applicability of object-oriented programming to building simulation is explained. Finally, a number of research projects are described; some aim to facilitate the production of new simulation programs, while others aim to integrate a set of design tools sharing a common building description.

Building simulation and building representation: Overview of current developments

A J Wright† MSc PhD MCIBSE, **D Bloomfield**‡ BSc MCIBSE and **T J Wiltshire**† BSc PhD MCIBSE
† School of Architecture, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU, UK
‡ Systems Performance Prediction Section, Building Research Establishment, Garston, Watford WD2 7JR, UK

Received 22 April 1991, in final form 5 August 1991

1 Introduction

It would be an enormous and pointless task to carry out a thorough review of all the thermal building programs currently available. Many studies comparing programs have already been carried out[1,2] comparing their performance for the same problem with each other, and with measured data, and reviewing their capabilities.

The purpose of this paper is to take an overview of the current state of thermal modelling, and to look at trends for the future. It is shown how in the short term, the data structures of current programs combined with forces driving program development tend to hinder improvements to building simulation, while in the longer term, the emergence of new tools offers hope for the future.

A major focus of current research in the engineering and construction industries reviewed here is data modelling and the development of an international standard for the exchange of product data (STEP). For buildings, the aim is to have a general schema for describing any building, in order to facilitate the exchange of information between different members of the design team. If this is successful, this would clearly have implications for simulation programs; those conforming to the standard would obviously be favoured by the industry, and integration of design tools would be greatly eased. At present, the language and structure for STEP have been defined but standard sets of entities in specific application areas remain largely undefined. There is currently no agreed data model for a building.

In computing, object oriented programming (OOP) is rapidly becoming established as a powerful and highly productive approach to many areas of programming, as the hardware to support it becomes widely available and inexpensive. Because this technique brings a high degree of modularity to data structures and functions by encapsulating both within software objects which map to real-world objects, it is claimed to be the best available method for producing programs sharing such objects, and also maps closely onto the real-world objects of a data model such as STEP; the language of STEP, Express has several object-oriented features. Four current projects using OOP to produce flexible environments for building simulation, each with a different emphasis, are described here.

2 Structures of current programs

It is necessary to consider the structures of current simulation programs before trying to find a solution to the problems which arise. The relationships between the user and different knowledge sources during the thermal modelling process are shown in Figure 1, from Reference 2, for a generalised plant and control modelling system. (In reality, no single system would offer a wide choice at the Knowledge Base level, and in many cases only one solver would be available.) The situation is very similar when modelling buildings. Clearly, there are a number of different data representations at different stages.

Figure 2 shows the different stages of simulation typical of the current generation of programs. A user defines the problem for the program, using databases both directly and indirectly within the computer. At each timestep, the description of the problem within the computer is then used to form the coefficients for the systems of equations which model the physical processes, and a set of boundary conditions. These equations are solved by a numerical process to give a set of calculated results which are stored as output; some results are also fed back into the system of equations to be used at the next timestep.

The inner box in Figure 2 contains the calculation loop carried out at each time step. The different forms of data representation are now described in more detail for building

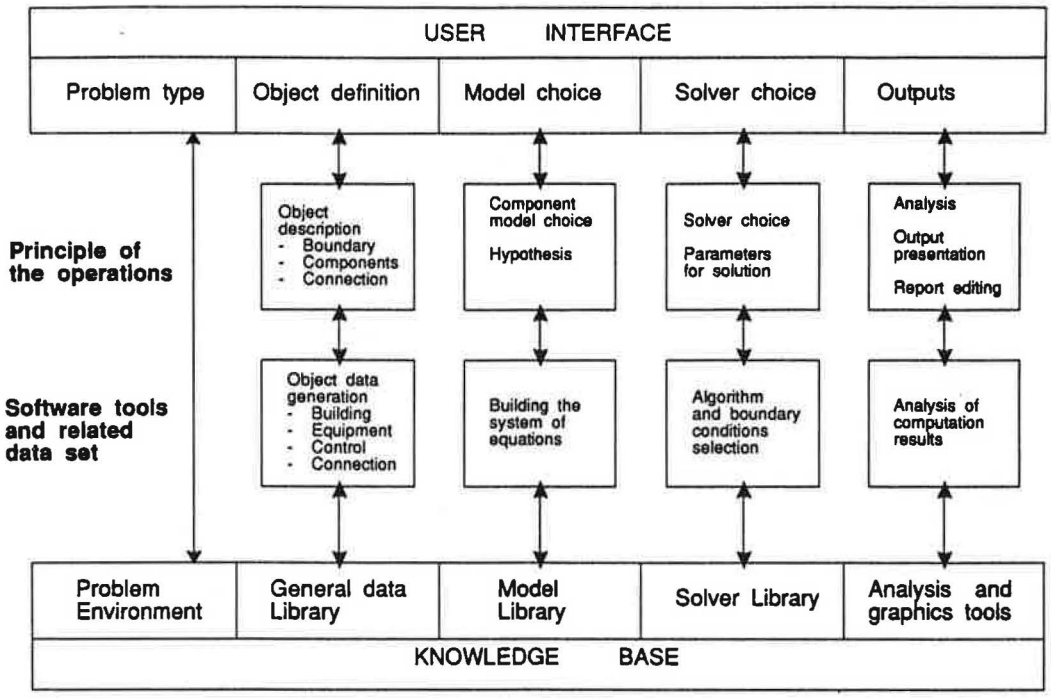

Figure 1 Relationship between use, software and knowledge bases for plant and control modelling

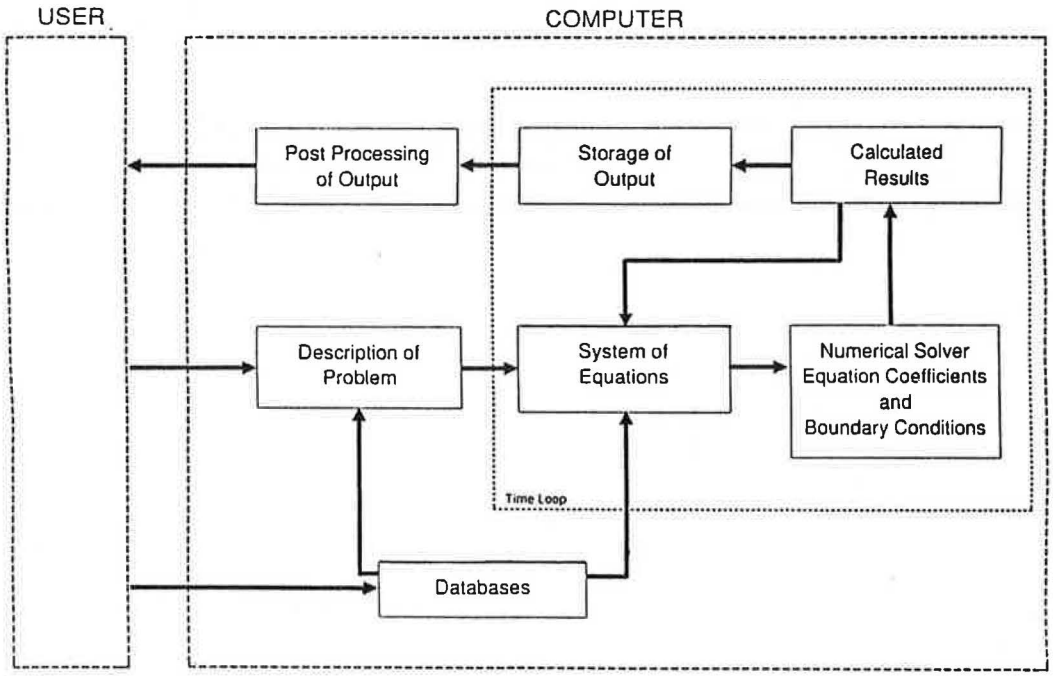

Figure 2 The stages of simulation in current programs

programs; the situation is similar for plant and control programs.

2.1 User level

The user has a concept of the actual problem, and how to represent it in the program. He has data in the form of materials properties, schedules of occupancy, control etc., and geometric information on drawings. Some data can be taken from databases on the computer, if these exist, such as thermophysical properties of materials and time series of weather data.

There are three main ways in which the user's description of the problem is transferred to the program in current models:

- some form of special building description language
- text files, edited using system editors, or customised editors which come with the program
- interactively, using the keyboard, digitisers, mice, third-party CAD systems, etc.

The first method is somewhat archaic and originated in the days of batch job computing and punched cards, but is still used for some USA programs, while the second and third methods are often used in combination; direct editing of text

files is often quicker for experienced users than interactive editing. Building description languages and text file input both map directly onto the data requirements of the program; they may not be most convenient for the user, although text file input offers more flexibility. Whatever system is used, it will involve entering actual data (e.g. building geometry from a file, or with a digitiser), and specifying database references for the computer to access (e.g. building element constructions and weather data). Not only does the form of data entry vary; because requirements of programs vary, the content varies also.

2.2 Problem description level

Whatever the form of data entry, when it is complete there will exist a description of the problem within the program. This can take any form, depending on the language being used. Almost all current programs are written in conventional languages, the vast majority in Fortran; the Pascal language has been little used for thermal modelling, because it is less suited to intensive numerical calculations and interfacing with standard libraries which are usually written in Fortran. This is unfortunate because Pascal offers more facilities for structuring data than Fortran.

At the most elemental level, data are stored as integer and real numbers, strings, and arrays of these. Data may be either private to a subroutine or function, or belong to the main program. There is rarely much structure imposed on the data, as this is difficult in languages like Fortran. Wide use is made of COMMON blocks in Fortran, which are pools of data which can be made available to several subroutines without passing the data through arguments.

This has the advantages of reducing storage requirements, as data are not duplicated in memory, and of simplifying coding because it reduces the need to pass arguments. Both aspects are important for thermal modelling because of the very large data sets involved. However, using COMMON blocks is considered bad programming practice for the purposes of robustness and correctness, as changes to data are not easy to trace through the program, and data items are often difficult to identify.

Programs written in traditional languages tend to be structured around functions, rather than the data upon which they are performed. Data items belonging to the main program, or to COMMON blocks, have a fairly independent existence from the functions; their meaning is usually identified in an *ad hoc* way from the context in which they are found, and from their names, but they are not formally bound to any structure unless private to specific elements of code (in which case they only have transitory existence anyway). This situation is made worse in Fortran because data item names are limited to six characters in length, with no case distinction, severely limiting the semantic information which can be conveyed.

Before the systems of equations can be instantiated with data, some preprocessing of the problem description is carried out. The problem description has considerable redundancy, being in a more general form than is actually required for the calculations. For example, planar surfaces may be represented by co-ordinates, when the calculations require elevation, azimuth, area and inter-planar view factors.

However, the preprocessing varies between programs; for the case of geometric information, some programs do not use co-ordinate geometry, but require elevation, azimuth and area to be given directly by the user. Most preprocessing needs only to be done once, so occurs before simulation starts. Following preprocessing, there exists a problem description which can be used directly in the systems of equations.

The form of the next two stages depends on the type of equations and solution technique used:

— The set of equations to be solved may either be explicit—i.e. each unknown can be calculated directly from known values—or implicit—i.e. a set of simultaneous equations must be solved. (Implicit equations arise when values of state variables at the future time step are used; this avoids numerical instability, hence allowing longer time steps to be used.)

— The solution technique may be direct, or iterative.

2.3 System of equations and numerical solver levels

Conceptually, every program has a set of equations at each time step representing the relationships between boundary conditions, physical characteristics of the system, and the unknown variables (principally heat fluxes and temperatures) to be solved for.

When an explicit solution is used, these equations appear in the form $x_i = f(\ldots)$, where x_i is an unknown, and no other unknowns appear on the right-hand side; thus x_i is evaluated immediately for all i. Because the mathematical model is explicit, the code is easier to understand.

Alternatively, a direct numerical solver can be used to solve implicit equations, which is likely to be faster than iteration. A typical numerical solver requires a set of vectors defining a sparse matrix (i.e. a matrix with many zeros) of equation coefficients, and a vector of boundary conditions. These are derived from the system of equations describing the individual models of behaviour, such as heat transfer processes; the distinction between the mathematical model and the vector and matrix coefficients is rarely made in the code, because it is more succinct to make the model implicit within equations used to derive the numerical solver coefficients directly. Thus the complete set of equations to be solved does not, in general, appear in the code.

Once in vector form, the data structures and semantic information relating to the original description has been lost. These vectors are passed to a numerical solver, which returns a solution vector describing the new thermal state of the building.

Another approach, adopted in the SPANK system[3,4] is initially to try to reduce the number of equations which need to be solved by eliminating unnecessary variables, and then to solve the resultant system numerically. This typically results in solving a smaller, dense matrix (i.e. a matrix without many zeros) instead of a larger, sparse matrix; the reduction in size of the system is partially offset by the greater number of calculations needed to solve a dense matrix system. In this approach, less semantic information is lost because the technique requires that each equation be treated as a separate entity, so that a variety of manipulations are possible.

For a more detailed discussion of the relative merits of different techniques, and their relationship to dynamic modelling of control systems with very short timesteps, see Reference 5.

2.4 Storage and post-processing of output

At this stage, the information of possible interest to the user is retained, and stored in a file. Because of the large volume and diversity of data, some post-processing is essential to make it useful; this usually involves deriving statistical data (totals, maxima, minima being the most important), producing graphical output, and putting data into tabular or report formats. This often takes place interactively. In many cases, output interpreted by the user prompts respecification of the problem, and another simulation is carried out.

2.5 Extensions to capability

For many years, most simulation programs did not model some important thermal phenomena which occur in real buildings; these include moisture effects, airflow within spaces, and two- and three-dimensional heat conduction flows; such effects were usually dealt with by specialist programs lacking most other features.

One of the most frequently requested features in building simulation is airflow modelling. Airflow can be modelled at two main levels; an airflow network of spaces separated by 'resistances' (cracks, openings), or a full computational fluid dynamics (CFD) treatment solving the Navier–Stokes equations of fluid flow. The network approach is used in the ESP system described by Clarke[6] and TARP described by Walton[5], which both operate by iterating between separate solutions of the airflow and the thermal equations.

At present, full CFD modelling is impractical over the length of a typical simulation, but stand-alone CFD programs for building designers are becoming available even on personal computers; for example, the FloVENT program developed by Flowmerics[7] and BSRIA runs on any IBM-compatible PC with an Intel 80386 chip, and includes wall conduction and radiation inputs. Widespread integration with thermal simulation, at acceptable speeds, is only a few years away. However, it is worth noting that for meaningful results, both approaches rely on accurate data on crack widths and external pressure fields which are very difficult to obtain in practice. Dynamic simulation of moisture effects is of less concern to designers, who are usually satisfied with a steady-state analysis to assess condensation risk, but it has been incorporated into a special version of the TARP program[8].

Similarly, edge and corner effects on conduction are thought to be a relatively small part of the total conduction (almost as a tenet of faith on the part of building modellers), which is itself becoming proportionately less important as insulation standards increase. Almost all current programs describe wall, floor and roof elements very crudely, ignoring complications such as bricks, window frames, lintels and floor junctions and treating each element as a series of homogeneous layers.

However, the heat flows which are thus ignored, such as cold-bridging and the effects of poor workmanship, are an important source of problems in buildings and an active area of research, usually using specialist finite-element software at a level of detail orders of magnitude greater than that of a whole-building model. Therefore integration of this type of modelling seems unlikely for many years.

Another very important and closely related area of simulation is plant and control modelling, considered here only in the context of building modelling. There are three basic approaches to modelling plant and control systems:

— Control laws: An ideal plant system is assumed which obeys a selected control law. Most programs using control laws offer several.

— Predefined plant systems: Each plant system is modelled as a complete system. Most programs with predefined systems offer several.

— Component-based systems: A plant system is built up from individual components to any configuration. Most programs of this type only calculate the steady-state solution at each time step; programs calculating system dynamics are usually very slow, sometimes slower than real time.

Plant and control programs are generally heavily biased towards air systems and most have no facility to model, for example, hot water radiator systems.

A fundamental difference between building modelling and plant and control modelling is the timescale of changes; typically of the order of several minutes to hours for buildings, but less than a second to a few minutes for plant and control. To run a combined building, plant and control program, time steps of a few seconds would be much too slow for typical building problems, although hardware developments may make this practical within the next few years.

According to Walton[5] 'It appears impossible for one program to satisfy all needs.' He believes two programs will emerge, one at a building time scale, and the other at a plant and control time scale, taking the output from the first for its boundary conditions. This could either be in parallel, the plant program taking boundary conditions at each time step of the building program, or in series, the plant program being run after the building program. This is in fact the way the UK programs TAS A and TAS B[9] operate.

This separation of the building from the plant and controls leads either to approximations, or to the need for some sort of iterative scheme which, in itself, is likely to require the assumption of quasi-linearity for accurate results to be obtained. There is little hard evidence and no consensus on the adequacy of current solution schemes of this type.

It seems likely that a variety of modes of linking the building and plant modelling will persist; the outcome of research into more flexible simulation programs is likely to result in the ability to produce or configure programs in different ways, according to the application.

2.6 Computing environment

Most of the current generation of building simulation programs were originally developed on mainframe and mini computers, but they are now mainly being used on powerful workstations and personal computers (PCs). The workstation market is dominated by the Unix operating system, while the PC market is mainly for IBM-compatible machines with the Microsoft DOS operating system.

The power of the single-user Unix workstation advocated by MacRandall[10] in 1988 for energy simulation software development has already been surpassed by the current generation of IBM-compatible personal computers costing less than £2000 and running Microsoft DOS and Windows-3, for which a wider variety of tools is available at a much lower price than that available for Unix systems. Meanwhile, Unix machines have dropped in price to become competitive with the PC market and offer even greater power. A number

of the current generation of programs are now available in a PC version.

Since both types of machine have certain advantages and sufficient power and capacity for thermal simulation, both will be used; however, for individual users with a standalone machine, the PC is likely to be favoured because of its lower price, simplicity of operating system and much wider use commercially. In networked environments, the distinction is becoming blurred; windows environments now operate over a number of platforms, and a user may today find himself running a program on a Unix workstation while sitting at his networked PC.

2.7 Differences between programs

At least five reasons for wide disparities in building simulation software can be identified:

— Differences in main application area

— Differing levels of modelling detail

— Modular or non-modular representation

— Time treatment—dynamic or steady-state

— Different forms of equations and method of solution

— Random decisions made during development.

Lack of clear structure causes further problems; although the different levels of data representation described here can be distinguished conceptually, in real programs they are often confused; according to Clarke[11]: 'Complex issues—methods, data, program structure and machine environment—have been hopelessly intermixed. The systems are therefore difficult to maintain and evolve.'

The situation is similar in other areas of simulation; Oren and Zeigler[12] in proposing a methodology for structuring discrete-event simulations into six functional elements, give an example of '. . .how these functional elements appear (albeit in scrambled form) in conventional simulation programs. . .'

3 Forces driving program development

The following selection criteria for commercial energy analysis programs are given by Seth[13] in a survey of the North American market:

(*a*) Ease of use

(*b*) Accuracy

(*c*) Applicability to the particular project

(*d*) Experience with the program

(*e*) Need for consistency and control.

The present situation in modelling makes these difficult to meet:

(i) Most programs are difficult to use, with complicated user interfaces and requiring considerable experience for effective use.

(ii) Program accuracy has to be taken on trust; research has shown that results for the same problem are a function both of the program and the user, while the software structure of current programs makes validation difficult even at the level of individual algorithms.

(iii) Programs are often applied inappropriately to problems they are not designed to solve, while changing to different programs costs money and time.

(iv) Experience with one program is often little help with another; each has unique characteristics which must be learnt.

(v) There is little modelling consistency between programs. Control and management functions in a large simulation exercise are left up to the user, usually at the level of the computer operating system.

3.1 Short-term developments

Several forces act to drive program developments in the short term:

— Research groups improve and generate programs internally for their own needs—few such programs are ever used externally.

— Programs are developed and sold commercially for profit.

— Inertia of previous use and familiarity encourages continued support and improvement of existing programs.

As a result, existing programs tend to continue in existence, often being extended well beyond their original purpose, and failing to exploit advances in computing technology.

3.2 Resource limitations

The number of people involved in building performance analysis is tiny compared with the number involved in more widely applicable fields such as text processing, cost analysis or playing games. At the same time, the profits which can be generated by using building analysis programs are small, and their use is often considered non-essential even for large design projects. Therefore, only a small number of copies are sold, and even large programs cannot be sold at a high price. This contrasts with, for example, satellite communications software for which the market is numerically small, but the use essential and the profits large.

Some figures illustrate the small size of the market for building analysis software; Seth[13] estimated the annual sales of building analysis programs in North America in 1986 to be 1425, while Baxter[14] reported that only about 1000 copies of the most popular suite of programs for building services design in the UK (HEVACOMP) have been sold over the last 10 years.

As a result the resources for program development are limited. A large amount of effort must be put into the modelling and solution part of the program, leaving less effort available for improving other aspects such as the user interface. Consequently, most programs in widespread use have large, unwieldy codes which are many years old; although some have been implemented for workstations and microcomputers, with graphical user interfaces 'bolted on', they remain fundamentally the same underneath.

Building modelling is also widely used in university and government research with the ultimate aim of improving building design. There is considerable dissatisfaction with the current state of modelling; the researcher is left with a choice between several incompatible programs, none of which are entirely trusted or understood, and the alternative of writing software for a particular problem. The user would

Table 1 Long-term influences on development of building simulation

Current feature	Desirable objectives	Emerging tools
Input data allow different structures unique to each program.	Ability to allow different design processes to access common data without translation	International standards for data representation
Unique internal organisation of each program	Rapid creation and modification of programs	New programming techniques, in particular OOP
Numerical solution specific to algorithm	Automatic solution of many different configurations	General numerical solvers for the time domain
Poor user interfaces	High quality, flexible user interfaces	Improved graphical interface toolkits

like to combine the best features of what is available, but cannot afford to write an entirely new program.

However, fossil fuels will become scarcer and more expensive in the coming decades, while the threat of global warming and recent events in the Persian Gulf have given an unexpected impetus to reducing fuel use. The building sector is the largest consumer of energy in the industrialised world; in the UK it accounts for 50% of total energy use. Computer simulation is likely to play an increasing role in improving building energy efficiency, as computers become more widely used in design, and relative efficiency improvements become progressively more difficult to predict.

Building modelling is already being used to assess building performance—the environmental conditions obtaining within buildings are always of importance. This too is likely to play an increasing role for issues such as the Sick Building Syndrome and indoor air quality.

3.3 Long-term developments

The situation is not, however, a stalemate. As shown in Table 1, new tools are emerging which may overcome the problems in the long term. Some of these new tools are discussed in sections 5 and 6.

In October 1989, an International Workshop was held at Chexbres, Switzerland, on computer representations for building information. This produced a diverse and interesting set of papers[15] which form a 'snapshot' of the current thinking in this area, which relates strongly to both emerging data standards and the object-oriented paradigm, discussed earlier. The strongest message to emerge from the workshop was unanimous agreement on the need for better and more general computer representations of building information, in order to accommodate integration of world markets, and the rapidly increasing use of computers by designers for drawing, expert systems, and simulation. At present, there is nothing remotely approaching a standard representation.

A full building product model is an extremely complicated conceptual data model from which information can be extracted in many different ways. Figure 3 from the Finnish RATAS project[16] shows some examples of these. Each viewpoint shows a different aspect of the model, but none describes the model completely.

A building is described by spatial and non-spatial information, and there is agreement that both must be integrated into any building model. But even for purely geometric information where there is no disagreement about what is being represented, there are many ways of representing it;

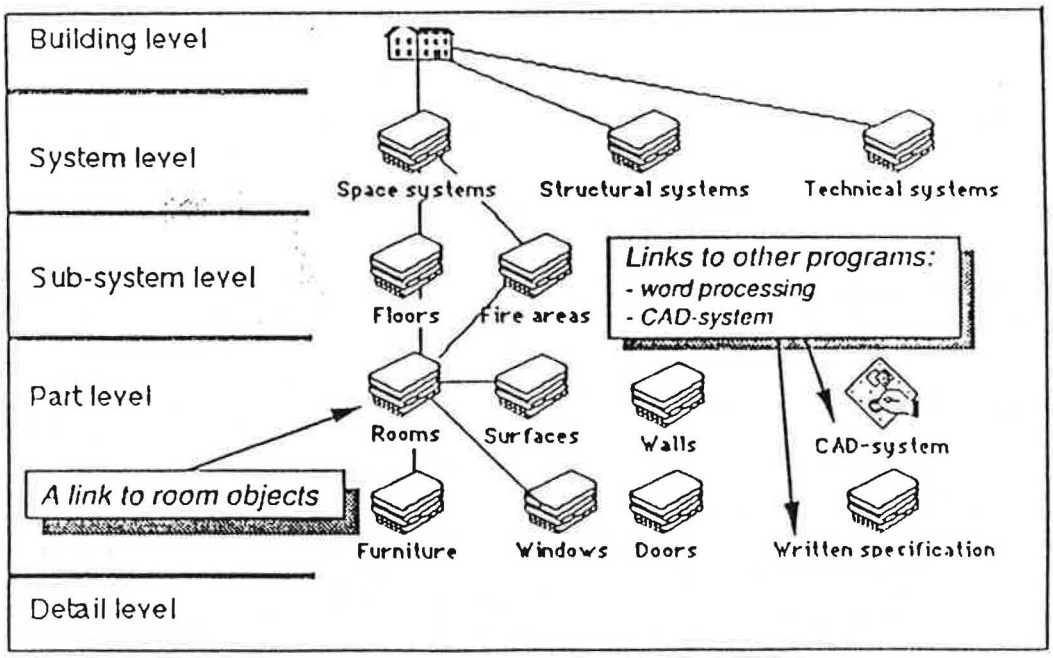

Figure 3 A conceptual overall view of the RATAS building product model implemented in the hypermedia prototype. Several links exist in this window allowing the user to jump to other information in the model.

for example, in CAD for mechanical engineering, there are at least 22 known ways of describing machine parts in three dimensions.

For the non-spatial data, there is much debate about what should be included in the data model, and what left to the applications. Several organisational structures are used by different groups to describe building data, based on different sorts of relationships: inheritance, part-of, slots and fillers, etc., and the examples given of each type all vary widely according to the application. There is agreement on the need for a few basic entities, such as building elements, rooms, and components of building services systems, but not on how they are described or how they relate to each other.

A number of prototype modelling systems were described at the workshop, all, significantly, using very different software implementations which included C, the Oracle relational database, and the Lisp language.

Much work is in progress on the science of data organisation with the emphasis shifting away from relational databases towards object-oriented databases. Unfortunately, the levels of abstraction and the esoteric language used sometimes make this field rather opaque to application developers. This 'bottom-up' work is complementary to the 'top-down' approach of prototyping applications built around experimental databases. These ideas are finding their way into the emerging data standards, described in section 4.

3.4 Conclusions on building representation

The principal findings of the Chexbres Workshop were:

- There is a need for a common meta-level building description language.
- A common building model cannot include all the knowledge about a building, but only a kernel of data common to all applications.
- Future work should combine the building process side with the development of new database models, on a timescale of 5–10 years.
- The most efficient way forward in the short term is to develop prototypes.
- The research should take into account developments of standards and Computer Integrated Construction (part of the ESPRIT programme of the EC).

As it will probably be impossible to agree on a common data schema for building representation, one solution would be an extendible kernel model; this would require an agreed meta-model (a description language) to define extensions to the kernel. The content of a database remains debatable; issues to be resolved include whether the architectural/geometric data should be included, or put in a separate database, and whether only physical information should be held in the database.

Finally, building researchers should not be concerned with low-level database implementation, a problem which should be left to computer scientists; they should instead concentrate on conceptual and higher-level models.

4 Emerging data standards

Standards are of two main types: *de facto* industry standards which result from commercial pressures, and agreed standards resulting from action by international bodies such as Comité Internationale de Normalisation (CEN) and the International Standards Organisation (ISO). Various industry standards have arisen for specific purposes, such as geometric file formats, but the definition of more general standards has been left to public action.

4.1 STEP *and* AEC

A joint project is under way between America and Europe with the aim of standardising methods for the exchange of engineering data, including building and construction data.

This is called STEP (Standard for Exchange of Product Data) in Europe; for historical reasons, in the United States it is called PDES (Product Exchange Standard). It is divided into application areas, with buildings coming under the area of AEC (Architecture, Engineering, Construction). STEP is such a large and complex undertaking that it could only have come about as a public project.

The data in STEP will be described using a language called Express. The basic element of Express is the *entity*. The data associated with an entity is given as a set of attributes, and the behaviour of an entity is defined in terms of rules. Entities are collected to form a common pool. Version 1 of STEP (incomplete) is due to be completed by the end of 1991, with additions to follow.

Application Protocols are defined as a subset of the entity pool relevant to a particular application, together with any special constraints. Each protocol then forms a common

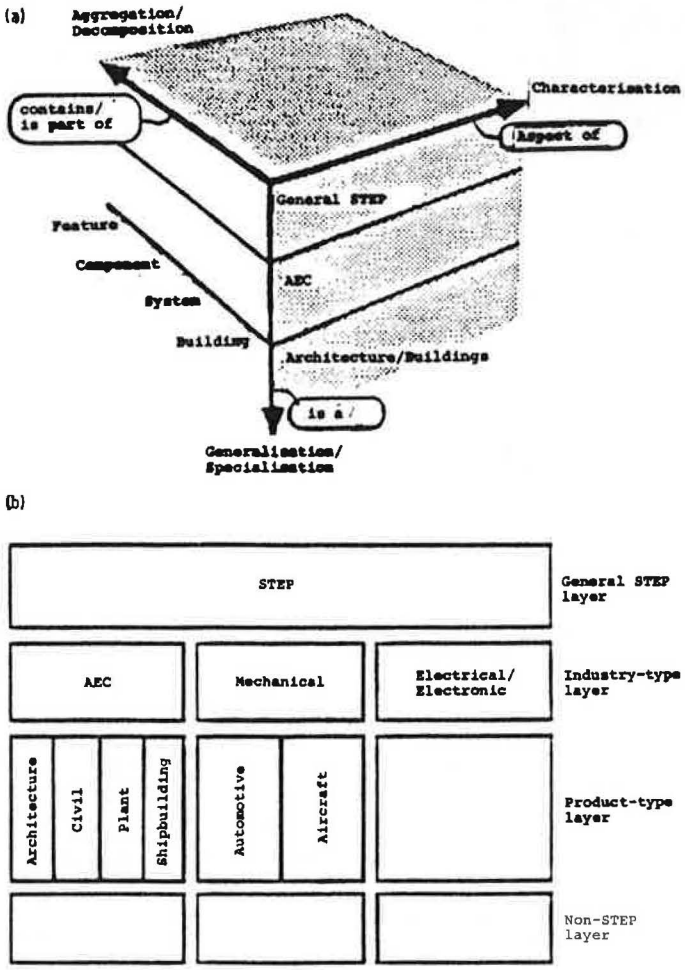

Figure 4 STEP standard: (a) Dimensions; (b) main application areas

means for computer programs to communicate within an application area; for example, a building modelling protocol might be defined for architectural design, environmental analysis and construction costing. The main application areas are shown in Figure 4, from Reference 17.

A high-level, abstract model has been developed for AEC from which models specific to particular types of product may be derived: this is called the General AEC Reference Model (GARM)[18]. More specific to buildings is an AEC building systems model developed by Turner[19,20]. Turner's model describes relationships between the entities which make up a building in detail, but it is not clear how it can be applied immediately to building simulation. However, a prototype product modelling language has been developed based closely on GARM and used to describe a simple building, allowing a variety of calculations including simple thermal calculations and production of bills of quantities.

5 The object-oriented paradigm

In parallel with developments in standards, there is currently intense interest in the use of the object-oriented programming (OOP) paradigm for writing software. Although OOP is not new (the earliest language with widespread use employing OOP principles being SIMULA, developed in the 1960s), it is only recently that inexpensive desk-top machines have been powerful enough for OOP languages; further development has been encouraged by the emergence of C++, a superset of the popular C language which embodies many of the OOP principles and seems set to supplant C in many areas. The increasing use of windows interfaces has also contributed to the use of OOP, because Icons, Pointers, Windows, Scroll Bars and other window features are conceptually 'objects', for which OOP is a natural programming paradigm.

The principles of object-oriented programming are described in detail elsewhere by Meyer[21] and in the computing press, but a brief summary is given here. The basic element of an OOP system is the *object*. Everything, from a single number to the application itself, is an object in a pure OOP language.

An object has a set of instance variables whose values define its state, and a set of functions or methods which can be applied to it. The instance variables themselves may be simple objects (numbers, strings) or complex objects (buildings, polygons). Objects are grouped into *classes:* all objects in a class share the same set of instance variables and methods. A method may create an object, destroy an object, change its state, or return an object to the sender. Methods are invoked by sending messages to objects; the messages form part of methods within other objects or the receiving object itself.

One of the most fundamental principles for producing good software is modularity. When modular programming based on functionality began to be used in languages such as Algol, Pascal and Fortran 77, it was thought by some to offer a solution to the 'software crisis', but this has proved not to be the case.

Meyer has proposed a set of criteria for judging modularity which are not all met by conventional languages, and a set of principles which should be adhered to ensure modularity which cannot all be met simultaneously in conventional languages. Meyer considers that the OOP approach offers the possibility of writing programs which meet his criteria by adhering to the modularity principles, where the software modules are the class definitions. However, simply using an OOP language by no means ensures such modularity, as the definition of classes is left to the programmer.

An important feature of OOP is the use of classes based on physical entities (such as walls, boilers, people, the sun), or non-physical entities with a clear conceptual meaning (such as polygons, matrices, times). Combined with the obvious applicability of data modelling standards (which are themselves essentially object/class based) to building description, the OOP paradigm appears to offer a radical new approcah to simulation software. Some work has already been done on using OOP techniques for building and plant and control modelling, for example in Belgium[22], and in the energy kernel systems described in section 7. Matsson[23] gives an analysis of the OOP approach to the simulation of continuous systems in general.

6 Energy Kernel systems

A number of research groups have been working on producing a more flexible and general environment for building energy modelling, to replace the present generation of monolithic programs; these environments may loosely be called energy kernel systems. The principal groups are in Sweden, the USA, and the UK. Only the UK project is called an energy kernel system. Similar work has been done by the Honeywell Corporation; Benton[24] describes a general solution technique for component-based, state-space representations of control problems, ranging from flight simulation to building systems.

6.1 Sweden: MODSIM

Work is in progress at the Swedish Institute of Applied Mathematics to produce a program for the dynamic modelling and simulation of continuous, component-based systems[25,26]. The aim is to have an efficient numerical solver, and to be able to solve any arbitrary combination of given (input) and calculated (output) variables. This is in contrast to most simulation programs where the algorithms are written for a small number (often just one) of combinations of inputs and outputs. By writing equations in the form of residuals (i.e. one side of the equations is zero), any input-output combination can be solved.

Solution efficiency is achieved in two ways. Firstly, system modularity is exploited by block manipulation of the sparse matrices which represent the equations in residual form, so that different matrix partitions can be partially solved independently. Secondly, the system of equations is reduced by eliminating variables which satisfy trivial 'coupling' equations, to arrive at a minimum number of variables which represent the number of degrees of freedom of the system.

The basic philosophy behind this work is to produce means of solution which can be applied in many fields, before developing full applications. However, the modelling of energy and temperature in buildings and their HVAC systems is being used as a test bed for the MODSIM project. Physical components are represented as objects, where an object contains both the equations and data for that component. Screen manipulation of objects (using icons and windows) is written in Lisp, while the numerical parts are written in Fortran. MODSIM is designed as an efficient solver, and its 'objects' are very specific to this purpose.

6.2 *USA: SPANK*

At the Lawrence Berkeley Laboratory in California, the Simulation Problem ANalysis Kernel (SPANK) is another object-oriented simulation environment for general purposes[3], targeted initially at modelling HVAC systems and similar in many ways to the Swedish approach. The fundamental object in SPANK is the equation. SPANK objects are *equation-based* or *declarative*, in comparison with components in conventional programs such as TRNSYS, which are *assignment-based* or *procedural*[27]. The original SPANK system was for steady-state problems only, but it has now been extended to model dynamic systems, using a separate object to integrate the differential equations[4].

A system is described in terms of components, interface variables and a mathematical model expressed as equations between these variables. As in the Swedish approach, no input/output connotations are implied in the interface variables, but all viable inverses of the equations are given and the appropriate one selected according to the overall solution requirements. This ensures a variable for every formula, and *vice versa*.

The problem is then analysed as a directed graph, with each vertex representing an object, an output variable and an equation (all of these always having a one-to-one correspondence). Graph theory is then used to automatically reduce the graph to leave a minimum number of variables which need to be solved, in a similar way to the Swedish MODSIM technique of eliminating coupling equations, with a typical reduction by a factor of two for an HVAC system. Finally, the system of equations can be solved numerically in a variety of ways, typically by an iterative technique when no direct solution is possible.

Because SPANK was designed to solve algebraic systems of equations, it was initially used only to solve steady-state problems, but it is now being applied to dynamic systems by integrating the differential equations in finite-difference form, so that all the equations are algebraic. The system is written mainly in C. As the concentration is on numerical solution, the object-oriented nature of the system is very specific to that end, and objects would have little meaning in another context. The extent to which SPANK is object-oriented is described more fully by Sowell[3].

6.3 *UK: Energy Kernel System*

The overall objective of the UK Energy Kernel System (EKS) was to provide a better software environment for the development, validation and maintenance of thermal modelling programs for buildings[28,29]. The object-oriented paradigm was chosen as the best means for achieving this. Once complete, the EKS should allow the construction of different thermal programs from a basic toolkit of components which form and relate to buildings, at different levels of physical detail. Some of the results which have emerged so far are now described.

The C^{++} language under the Unix operating system was chosen in combination with an object-oriented database (OODB) called Ontos, which only supports C^{++}. The OODB can store permanent data as objects, such as descriptions of materials, constructions and weather, and temporary data as objects for specific projects which may be used several times, such as building descriptions. The OODB also provides some facilities for manipulating classes to tailor them for new programs, and for program construction itself.

Identifying a set of classes to contain the objects for building simulation was found to be difficult[28]. The process of decomposing a complex system into objects is complicated by the density of connections, both physical and thermal, between entities in a building system, and the complex geometric and topological relationships between entities. No clear methodology yet exists for describing a physical system in OOP form, and it became clear that there is no single or 'right' way of doing this; many arbitrary decisions must be taken on where functions and data belong.

It is clear that different classes are needed to describe the same physical entities for different modelling strategies for two reasons:

(*a*) The level of detail required varies greatly.

(*b*) Different functions may be used to describe the same physical process. To achieve the flexibility to produce a wide variety of programs, dynamic class creation for specific programs is required.

In discrete event simulation, individual parts of the system can respond to events and in turn propagate changes elsewhere. In contrast, the processes in a building system are evolving continuously through time, but our current knowledge about describing these processes requires the state of them calculated at discrete intervals of time. This requires a solver 'engine', just as in existing models. It is intended to include a number of such solvers in the EKS, of different types.

7 Combine

As part of research into improving building design in the European Community, the Combine project[30] is based around the concept of the Integrated Intelligent Building Design System (IIBDS). This has two main ingredients:

— a set of design support tools under complete control of the designer

— a system in which these tools are embedded, providing intelligent assistance to the designer using the tools.

The project is mainly concerned with providing an integrated environment for a number of IIBDSs—that is, with providing interfaces for IIBDSs rather than developing a specific IIBDS. A variety of software tools are being used, but the approach includes:

1 initial concentration on the early stages of design

2 a central integrated data model of the building serving the IIBDSs, in a format based on the STEP standard

3 use of an object-oriented environment for many of the integration tools within an IIBDS, all sharing a common data set derived from the integrated data model.

The main differences between Combine and the energy kernel systems described in section 7 are that the latter are designed to create standalone, much more detailed simulation programs and are aimed at the specialist user, while the former involves simpler tools in an integrated environment, including intelligent design support and aimed at building designers.

8 The AEDOT project

The Advanced Energy Design and Operation Technologies (AEDOT) project is led by Pacific North-west Laboratory in the USA and has as its objective the development of advanced computer-based tools to promote the design and proper operation of energy-efficient commercial buildings[31]. Use will be made of current and emerging new capabilities from the fields of artificial intelligence, advanced database technology, computer graphics, visualisation and system integration. Emphasis is being placed, initially, on the provision of energy design assistance at the early design stage.

An important aspect of this project is the use of computer technology, eventually, to enable an improved transfer of information from the designer to the contractor, commissioner, building owner and operator. It is also possible to envisage feedback on actual building performance to the design team and the improvement of the knowledge base available to them and their successors.

This level of integration, both between different building performance evaluation tools (e.g. for lighting, HVAC design, condensation assessment) and between different stages of the design process will place particular stress on the need for common data structures to be used in describing all the different facets of the building and its associated systems.

9 Conclusions

Current building simulation programs fail to offer adequate facilities for users' needs, and have structures which are too inflexible for continued development and modification. Therefore a new approach is needed.

It is worth noting that the two projects with the broadest scope—COMBINE and AEDOT—both concentrate on the early design stages, and recognise the need for common data structures, while the latter feature is the main content of the RATAS and STEP projects.

In the longer term, data modelling and the international data standard STEP provide a means of representing the core data of a building description in a common format for many applications, while object-oriented programming provides both a way of describing behaviour which maps closely onto a data-oriented representation, and more flexible solution techniques.

It is likely that a data standard will emerge eventually for building descriptions from the STEP programme, but not for several years; as yet, there is no readily available data model for buildings which can be used directly in simulation.

Meanwhile, it is clear that object-oriented programming does not offer a simple solution to data representation for the simulation of something as complex as a building, and many of the techniques for a data-oriented simulation 'toolkit' remain to be resolved. In parallel, simulation systems are being developed which concentrate on generalised solution techniques using object-oriented ideas, but here, the 'objects' do not necessarily correspond to real-world entities. These two approaches may eventually come together in a system where, for the purposes of numerical solution, the 'real-world object' description of the former approach is mapped onto an 'equation-object' description of the latter approach.

All these separate endeavours need to mature considerably before they can be combined into improved simulation environments (a single environment seems unlikely). Projects such as COMBINE and RATAS will provide prototypes of how such environments might work.

Even then, it is certain that a number of solution techniques will remain in use to cater for different types of problem and personal preferences, which may require fundamentally different representations at the physical description level.

References

1 Wiltshire T J and Wright A J *The documentation and evaluation of building simulation models* BEPAC Technical Note **89/2** (November 1989) (Garston: Building Research Establishment)

2 International Energy Agency *Synthesis Report for three years of activities (1983–5): Component Models, Level 1* International Energy Agency Annex **10** (April 1986)

3 Sowell E F, Buhl W F and Nataf J-M Object-Oriented Programming, Equation-Based Submodels, and System Reduction in SPANK *Proc. Building Simulation '89, Vancouver, Canada* (June 23–24 1989)

4 Sowell E F and Buhl W F *Dynamic Extension of the Simulation Problem Analysis Kernel (SPANK)* Internal Paper LBL-26262, Simulation Research Group, Lawrence Berkeley Lab., California (July 1988)

5 Walton G N Considerations for Advanced Building Thermal Simulation Programs *Proc. Building Simulation '89 Conference, Vancouver, B.A., Canada* (June 1989)

6 Clarke J and McLean D *ESP, A Building and Plant Energy Simulation System Version 5.3* (University of Strathclyde, Scotland: ABACUS) (1986)

7 Flowmerics Ltd *FloVENT User's Guide Version 1.2 Beta* (Kingston-upon-Thames, UK) (1990)

8 Winkelman F Advances in Building Energy Simulation in North America *Energy and Buildings* **10**(3) 161–173 (1988)

9 EDSL Ltd *Thermal Analysis Software User Manual* (Milton Keynes, UK) (July 1989)

10 MacRandall D Some trends in computing: The implications for Simulation, Advances in Building Energy Simulation in North America *Energy and Buildings* **10**(3) 249–258 (1988)

11 Clarke J A *A Methodological Approach to Building Environmental Prediction Modelling: Case for Support* (University of Strathclyde: Energy Simulation Research Unit) (1988)

12 Oren T I and Zeigler B P Concepts for advanced simulation methodologies *Simulation* **32**(3) pp 69–82 (1979)

13 Seth D Advances in Building Simulation *Proc. Building Simulation '89, Vancouver, Canada* (June 23–24 1989)

14 Baxter A *The future of building modelling* CIBSE lecture, Newcastle-upon-Tyne (February 1990)

15 *Proc. Polytechnique of Lausanne International Workshop on Building Representation, Chexbres, Switzerland* (October 23–25 1989)

16 Björk B C and Penttilä H A Scenario for the development and implementation of a building product model standard *Advan. Eng. Software* **11**(4) 176–187 (1989)

17 Geilingh W Computer Integrated Construction: a major STEP forward *Computer Integrated Construction* **1**(3) (Thatcham: Wix McLelland) (1990)

18 Geilingh W *General AEC Reference Model* ISO TC 184/SC4/WG1 doc. N329 (Geneva: International Standards Organisation)

19 Turner J A *AEC Building Systems Model* ISO TC 184/SC4/WG1 doc.3.2.2.4 (Geneva: International Standards Organisation) (July 1988)

20 Turner J A The Conceptual Modelling of a Building: Part 1 *Computer Integrated Construction* **1**(1) (Thatcham: Wix McLelland) (1990)

21 Meyer B Object-Oriented Software Construction (London: Prentice-Hall) (1988)

22 Liekens J *Energy Simulations Using An Object-Oriented Approach* Internal Paper, Vrije University, Brussels, Belgium (February 1989)

23 Mattsson S E *Concepts Supporting Re-use of Models* (Dept. Automatic Control, Lund Institute of Technology, Sweden) (1989)

24 Benton R, MacArthur J W, Mahesh J K and Cockroft J P Generalized Modelling and Simulation Software Tools for Building Systems *ASHRAE Trans.* **88**(2) 839–854 (1982)

25 Sahlin P *MODSIM: A Program for Dynamical Modelling and Simulation of Continuous Systems* (Stockholm: Swedish Institute of Applied Mathematics) (1988)

26 Soderlind G, Eriksson L O and Bring A *Numerical Methods for the Simulation of Modular Dynamical Systems* (Stockholm, Sweden: Royal Institute of Technology) (1988)

27 Mattsson S E On Modelling Structuring Concepts *Proc. 4th IFAC Symp. Comp. Aided. Design in Control Systems (CADSC) China* (1988)

28 Charlesworth P, Clarke J A, Hammond G, Irving A, James K A, MacRandall D and Tang D The Energy Kernel System: The way ahead? *Proc. Conf. UK Building Energy Performance Club, University of Kent at Canterbury* (April 1991)

29 Wright A J, Clarke J A, Hammond G, Irving A, James K A, MacRandall D and Tang D The use of object-oriented programming techniques in the UK Energy Kernel System for Building Simulation *Proc. 1990 European Simulation Multiconference, Nuremberg* (San Diego, CA: Society for Computer Simulation) (June 10–13 1990)

30 Augenbroe G Integration of Simulation into Building Design: The Need for a Joint Approach *Proc. 1990 ASME Int. Solar Energy Conference, Miami, Florida* (April 1–4, 1990)

31 Brambley M R *et al. Advanced Energy Design and Operation Technologies Research—Recommendations for a U.S. Dept. Energy Multiyear Program Plan* Report PNL-6225 (Richland, WA: Battelle Pacific Northwest Laboratory) (December 1988)