Integrating the Building Procurement Process Using Knowledge Based Technology

R M DROGEMULLER[1]
J SMITH[2]

ABSTRACT

Computer based methods for facilitating building procurement have been proposed for over twenty years, but progress on such systems has been slow. This paper describes a project built around a three dimensional computer model of the building to be constructed. Knowledge based techniques are used to build up the level of detail required at each stage of development. Data entry requirements are minimised since only the information unique to the project need be entered. Standard information is stored as default values from previous similar projects. The user interface is simple, with a combination of menus to control the flow of information and dialogues to enter textual information. An 'intelligent' CAD interface is used to enter the building geometry.

The system has been developed around the design and construction of detached houses, but the principles demonstrated are relevant across the standard building types. In its current form the user can access the geometric and spatial parameters of the building, derive costing data and perform thermal analyses. There is an option to export scheduling information to an external CPM program. This furnishes the basis for planning the construction activities. The flexibility of the system indicates that knowledge based systems are a viable technology for assisting construction management.

Key Words

knowledge based systems; knowledge based estimating; multi-expert system; geometric reasoning; Prolog

INTRODUCTION

During the procurement of a building many specialists work on the building for varying lengths of time. The increasing pressure on procurement teams to perform their roles under strict cost and time constraints and the increasing complexity of modern buildings increases the likelihood of major misunderstandings and errors during the design and construction of the

1 *Lecturer, Department of Civil & Systems Engineering, James Cook University of North Queensland, Townsville Q4811, Australia.*
2 *Professor of Computing, Department of Mathematics and Computing, University of Central Queensland, Rockhampton Q4701, Australia.*

building.

Nearly twenty years ago, Eastman (Eastman, 1975) proposed a model for the development of information during the design process and indicated how the model extended into the construction and occupancy phases. The information pyramid (Figure 1) indicates how the information developed through the procurement process grows for the various building sub-systems. Currently drawings and written specifications are the main medium of information exchange. Eastman predicted that all project data would be stored on a computer, with the appropriate drawings, specifications and reports being generated from the computer database as required.

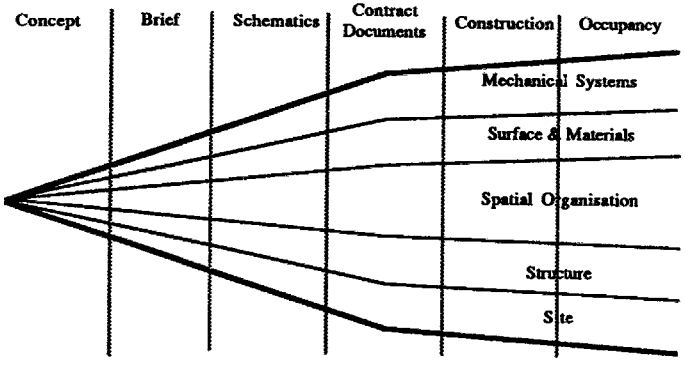

Figure 1. Information Pyramid During Procurement Process (after Eastman, 1975)

Thirty years of research and development has not yet produced a unified computer based system which is capable of providing support for the management of the information generated by the various members of the team. There are many reasons for this, the major one being the sheer complexity of the task. Other reasons are the lack of compatibility between CAD systems and other software and the lack of standardisation in data modelling.

A system called Hester has been developed to examine these issues. It uses the "cooperating experts" model in a similar way to existing building procurement teams. A similar architecture has been used in IBDE (Fenves *et al*, 1990) running under the Unix operating system. However IBDE does not appear to have a CAD like user interface. Hester is currently a working prototype.

Goals of the System

The following criteria were defined to guide the development of Hester:
Minimal Hardware and Software Requirements
Hester is being developed and runs on commonly available personal computers.

Simple User Interface

The interface requirements for knowledge based systems have been given as (Stelnzer & Williams, 1988):
"1. The interface should represent the domain in the user's natural idiom.
2. The interface should provide immediate feedback to the user on the effects of changes to system state by explicitly maintaining and displaying complex relationships and interrelationships.
3. The user must be able to recover easily from trying different alternatives.
4. The user interface must support the user at different granularity's, or levels of abstraction.
5. User interfaces must be implemented in such a way that it is possible to have multiple interfaces to the same knowledge."

Simple Maintenance and Extension

The system has been implemented to use general techniques in the design, estimating and scheduling of building works. It is expected that the system will continue to be developed in a number of ways, such as simplifying the addition of data and heuristics and by expanding functionality. The use of frames and rules as the knowledge representation paradigm and a blackboard architecture have allowed the initial estimating system to be expanded to its current form. Further expansion should cause no major problems.

Minimise Data Entry

The provision of interactive graphics to allow the entry of line drawings solves several problems. It is no longer necessary to produce the drawings using a CAD system and then use a separate estimating package. This avoids duplication of effort and removes mis-measurement as a possible source of errors. Textual information is entered using menus and dialogue boxes where ever possible. The user need only select the desired response from the list of available options.

User Maintenance of Knowledge Base

The users of the system are unlikely to allow outsiders access to their estimating information. Consequently the user must be able to maintain the data.

Modularity

The building team is made up of a number of specialists. The form of the building team varies with different types of contracts and construction methods. This is a useful model for computer software which seeks to emulate the performance of a flexible team.

This system appeared to be well suited to the methods developed in artificial intelligence research. Expert systems have been used extensively in the construction industry for focused tasks, but their use for large systems has been limited to research projects.

Using Hester

When Hester is started (Figure 2) the user sees a screen consisting of three areas. The graphics window is the largest and contains a grid showing the default building module of 1200 mm. The text window at the bottom of the screen is provided for textual messages which should not interrupt the flow of work, such as distances and coordinates. A menu bar appears across the top of the screen to allow the user to issue commands.

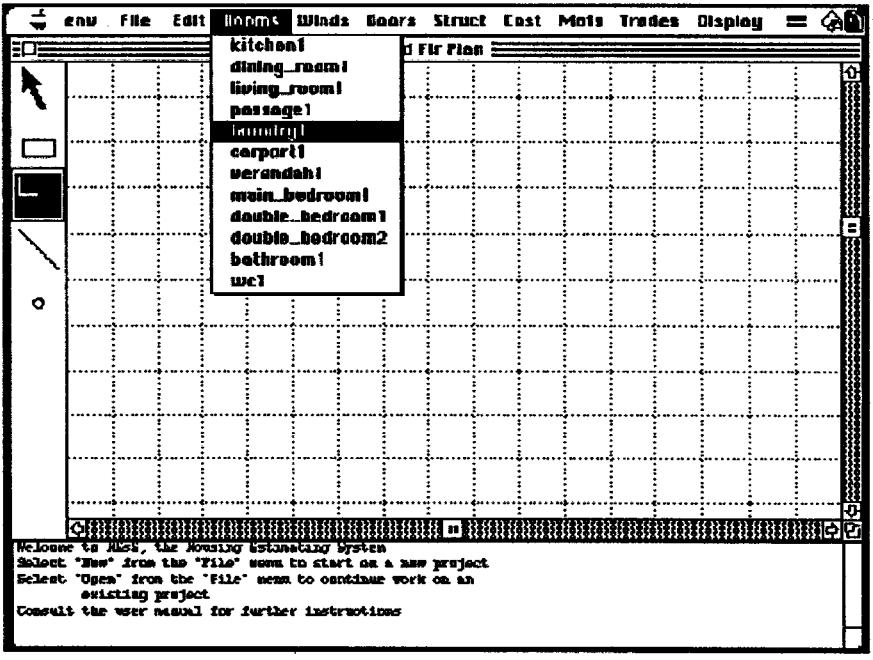

Figure 2. Drawing the Floor Plan

When 'New' is selected from the 'File' menu a series of dialogue boxes appear asking for information about the project. Information about the type of project (Figure 3), the client, the site(s), the building(s) on the site and the rooms within the building (Figure 4) is requested in the order given. Information about the construction and finishes to each building is then requested.

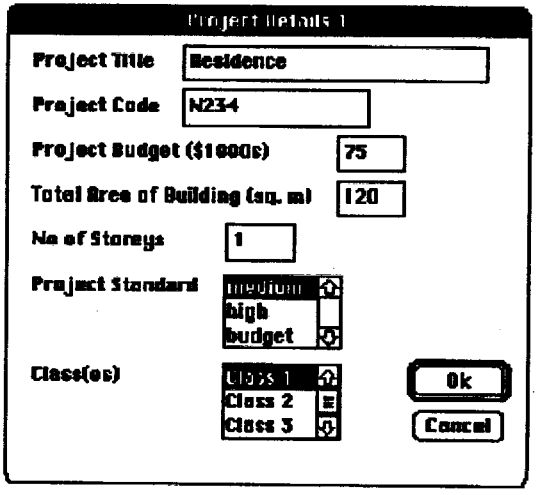

Figure 3. Project Dialogue

Dialogue boxes are used rather than the text window to enter this information since the range of values that Hester 'knows' can be presented with default choices already highlighted. Selection from menus also minimises typing errors.

Once the parameters for the project are entered the user can draw the floor plan of the building in the graphics window. The plan is built up by selecting a room name from the 'Rooms' menu and a graphic tool from the side of the floor plan window (Figure 2). The rectangle or polygon drawn on the screen is accepted as the room and the lines around the perimeter are recognised as walls. Hester automatically distinguishes between internal and external walls and stores the appropriate construction type for each element. As rooms are drawn on the screen their names are removed from the 'Rooms' menu. When the 'Rooms' menu is empty Hester 'knows' the outline plan is complete. Hester can then generate the roof plan using the

requested roof overhangs following one of the standard roof shapes. This also allows Hester to extend the walls up to the underside of the ceiling or soffit (whichever is appropriate). The doors, windows, columns and beams can then be added to the floor plan by the user. The elevations are generated automatically using default values for the window and door head heights and by calculating the height required for walls.

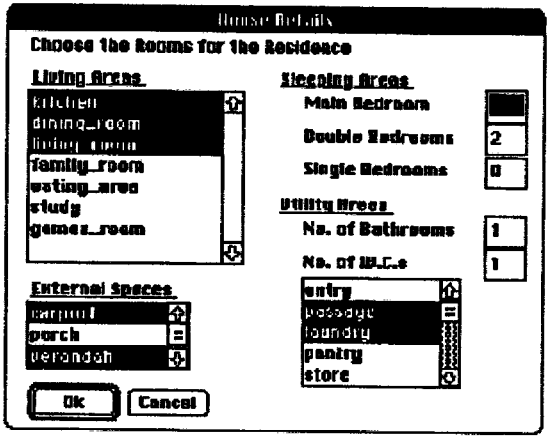

Figure 4. House Dialogue

When the building geometry has been defined various reports can be requested. A total price can be calculated in less than a minute (for a typical detached house) as well as cost breakdowns of the various components. Potential areas of savings can be identified immediately if the cost exceeds the client's budget.

Reports on the thermal performance of the house can also be generated using either steady-state (Szokolay, 1990) or dynamic (Harrington-Lynn, 1974) analyses. Hester will export data to a CPM program so that the construction time can be estimated. The information used is extracted from a variety of sources. The estimating methods and unit rates are those used by a local builder. The time estimates are based on published data (Rawlinsons, 1993).

Graphical Data

The Prolog predicates which store graphical information must allow the data to be represented in a recognisable form in the graphics windows and must also allow for the extraction of areas and lengths for further processing.

In order to maintain acceptable performance the following constraints applied: Minimise computational load by saving intermediate results where possible; Distribute processing between pauses by the user to maintain response times; Keep representation simple but adequate; minimise entry of lines; and allow for non-orthogonal walls.

Hester had to be able to represent and distinguish between internal and external walls, doors, windows, openings, roof structure and miscellaneous structural elements. Linear elements such as columns and beams are represented as a line while planar areas such as walls and roof planes are represented as three dimensional planes bounded by straight lines.

Points are simply stored with an index and the three coordinates and lie in the positive x, y, z sector. The indices of the end points of lines are stored as well as the normal form of the three dimensional equation of the line, the length of the line and the indices of the planes for which the line is a boundary segment. Planes are stored as a three dimensional equation together with lists of the enclosing and enclosed boundary lines and the area of the plane. The equations of the lines and planes are needed for consistency checking and to allow detection of continuous lines and planes. Walls are stored as planes. When a door is inserted in a wall panel the list of perimeter lines is altered. The insertion of a window defines a hole in the plane, which is stored as a separate boundary.

The storage of the plan as lines and planes ignores the actual thickness of wall segments. This is overcome by assuming that the planes represent the centre line of internal walls and that the external walls are offset by the same amount. The thicknesses of walls can then be allowed for by using appropriate formulae. This approach was followed to allow for flexibility in choosing the types of wall construction. If the type of external wall is changed then it is not necessary to redraw the screen. The only change required is the recalculation of areas. This single line representation is also consistent with the desire to keep the user interface simple.

Non-Graphical Data

Hester contains knowledge of the types of rooms in a typical house and building elements, where they are located, the relationships between them and the constraints on the elements. Frames (Minsky, 1985) are used to represent this knowledge. A portion of the building frame is:

```
frame building;
    default class is unknown and
    default no_of_levels is 1 and          % spatial & geometric data
    default gf_rl is 10000 and
    default gl_rl is 9850 and
    default ceiling_ht is 2400 and
```

```
default fl_to_fl_ht is 3000 and
default rooms is empty and
default substructure is unknown and                    % construction data
default structure_type is unknown and
default external_wall_construction is unknown and
default services is {power, water, sewerage} .
```

The house frame is linked to its ancestor, the building frame. Any values for slots which are not explicitly over-ridden are inherited from the building frame:

```
frame house is a building;                    % portion of house frame
    default class is 'Class 1' and            % override default
    default name is 'Residence' and
    default living_areas is {kitchen, dining_room, living_room} and
    default main_bedrooms is 1 and
    ...
    default substructure is slab_on_ground and     % construction data
    default structure_type is load_bearing_masonry and
    default external_wall_construction is conc_block_200mm and
    ...
    default roof_framing is trusses and
    default roof_shape is gable and
    default roof_pitch is 15 and
    default roof_cladding is metal_deck.
```

Frames as implemented in Flex are a powerful method of representing knowledge. An important characteristic of frames is the inheritance of information from parent to child frames. Since the house frame is a descendant of the building frame all of the information stored in the building frame is also part of the house frame, even though it is not explicitly given. The information stored in the slots of the building frame is also available to other descendants of the building frame, making the extension of Hester to other types of buildings much simpler than with traditional programming techniques. The slots with default values of unknown are used as place holders to maintain the inheritance hierarchy across all building types.

Frames can also have procedures attached to them. Launch procedures are used to call dialogue boxes which allow the user to set the appropriate values for the slots when a new instance of a frame is created. The value that is inserted into a slot can be controlled using constraints. This is used to ensure that only legal values are stored. Demons can also be attached to slots and are called immediately after a slot value is updated. When a new instance of a house is created the launch procedure is run and the new values are inserted into the slots of the instance. The values of these slots are checked as the values are inserted. The rooms slot, for example, has a demon attached

which then creates instances of all of the room types stored in the rooms slot. Access to slots can also be restricted used watchdogs, which verify that a request for a value is legitimate. This would be important where information about a project is shared by various firms. Access to information not directly of relevance to this project by other parties could then be controlled.

Frames are used to represent the building elements. For example, the external walls are stored as lists of wall panels which intersect at changes in direction, junctions with internal wall panels and changes of material. If the wall panel is of the standard external wall material then no explicit reference is required. If a panel is of a non-typical material then a slot is created giving this information. When the material type is required, say for estimating purposes, the frame is queried. If there is an explicit type of material stored, then it is returned. Otherwise, the standard value is retrieved from the building frame and returned.

Frames are suitable for storing and maintaining descriptive data, but they are less useful for storing procedural information. An English language-like syntax is the preferred method for achieving this. Flex provides this facility with its rules:

```
rule detail_design_class_6
    if the design_stage is detailed
    and the building_class is 6
    then detail_cost_file is det_cost_6
```

Where information may be modified by non-programmers, Flex has been used due to its English-like syntax. Internal data such as quantities and the thermal properties of materials is stored as Prolog predicates to aid efficiency of execution. In their simplest form these are represented as:

```
quantity(window, 12x12_SGW, 2).
u_value(window, metal_frame_6mm_single_glazed, 6.00).
```

This format is equivalent to the standard relational data base. However, this cannot represent all of the required information. The number of window reveals, for example, depends on the number of windows, so there should be some link between these values:

```
quantity(window_reveal, 12x12_SGW, Quant) :-    % get quantity of reveals
    quantity(window, 12x12_SGW, Quant).         % get quantity of windows
```

The U value of some building elements, such as floors, depends on the geometric properties of the element. A method of calculating the U value once the geometry of the element is known is provided by adding predicates which define relationships between the items of data:

```
u_value(floor, Con_Type, U) :-                    % with the floor element
    element(floor, floor1, _, L, W, Con_Type),    % find length & width
    floor_u_value(Contype, Length, Width, U_Value),  % get the U value
    L > Length,                                   % check for appropriate
    W > Width, !,                                 % values
    U is U_Value .                                % got it

floor_u_value(suspended_timber, 30, 15, 0.39).
floor_u_value(suspended_timber, 15, 15, 0.45).
floor_u_value(suspended_timber, 15, 7.5, 0.61).
```

The use of Prolog has significant advantages for a system that grows to suit the user's needs. Prolog contains database predicates which can ensure that the user interface provides access to all of the data stored in the system. For example, the dialogue that allows the user to select the rooms in the house uses the "findall" predicate to create a list of all of the room types that have been defined. The list of choices automatically changes as the data is modified.

Structure of Hester

Hester is written in PROLOG (Clockson & Mellish, 1984). Flex (LPA, 1989), a knowledge representation language which is implemented in PROLOG is used for large parts of the system.

A modular structure is used to implement the components of Hester (Figure 5). Each module is a stand-alone program that is called by and communicates with the Hester shell and the other modules. The shell is responsible for all interaction with the user. When it needs data from one of the other modules the shell sends a message to the appropriate module, which performs whatever operations are necessary, exchanges messages with the other modules and returns an answer.

Alternative modules can be provided to supply cost data for the various stages of design/documentation. One module could contain unit area rates for feasibility studies, while others could contain elemental rates for sketch designs and unit rates for detailed designs. Communication occurs through Inter Application Communication (IAC) supported by the operating system. The modular structure was chosen to allow incremental development of the system and to allow for alternative sources of data.

An advantage of the modular strategy used in Hester is that all of the modules do not have to reside on the same computer since they are independent programs which communicate through the operating system. Each member of the procurement team can have the appropriate modules on their computer with dial-up access to the central database. The use of multiple computers also means that a degree of parallelism is built in to Hester.

Processing speed is no longer dependant on the speed of one computer since the processing load can be shared amongst several machines.

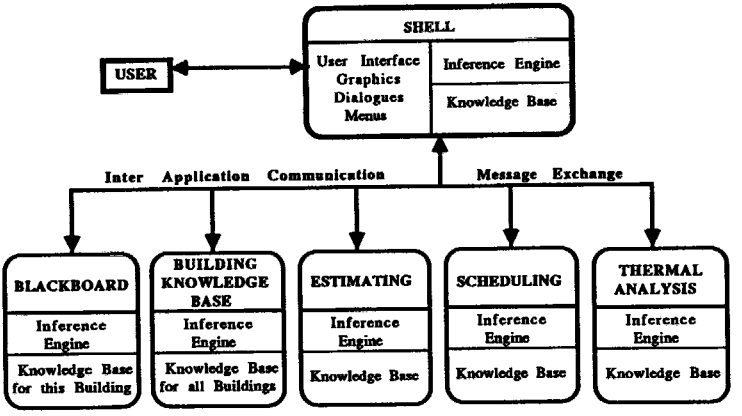

Figure 5. Structure of Hester

Expanding Functionality

The extension of Hester to perform other functions, such as cash flow or the detailed recording of project data throughout the construction period has been catered for by using IAC to communicate between the various programs which make up Hester. New functionality can be added by writing a separate program to perform the function and then adding the required IAC calls to Hester's shell. It would even be possible to replace sections of Hester with other software. For example, AutoCAD could be modified to replace Hester's graphical interface using the AutoCAD Development System (AutoDesk, 1992).

Extension to other Building Types

The intention at the start of this project was to develop a system that would be able to handle the common building types. Housing construction was chosen because it was as complex as the other building types from the computer programming point of view but was not as complex from the building construction perspective.

The use of frames to represent building information has simplified the extension of Hester to other building types. All buildings have common characteristics and this can be exploited by systems such as Hester using inheritance. A frame representing a site can have a slot to store the buildings that are located on that site. It does not matter if the buildings are houses, offices, school buildings or a combination of these. In its current form, Hester

can be easily extended to suit single storey buildings. All that is required is the addition of knowledge about the other types of buildings. The extension to multi-storey buildings will be more difficult since it will be necessary to add knowledge about load-bearing walls and columns and to ensure that load paths in the different types of structure are maintained.

Implications for the Management of the Procurement Process

Hester has implications at several levels for the construction industry. The construction of detached housing represents a significant proportion of Australian building stock. Most of this work is performed by builders with no contribution from the design professions. The resulting designs are often criticised as catering to the lowest common denominator with little consideration given to wider issues such as siting, energy efficiency and the consideration of alternative forms and materials. Hester could be used to provide advice and explanations to the clients as their design is being developed. This can be expanded for other building types to provide advice to architects throughout the design process.

The pressure to decrease overall design/construction time for large projects while keeping cost under control places many strains on the relations between the various members of the building team. Systems such as Hester which allow the various components to be located on different computers connected by a network can alleviate these problems in several ways. If a central computer were provided by the project manager or coordinating consultant the members of the building team could then access the relevant data. This would ensure that conflicts are noticed and attended to at the earliest opportunity.

Distributed software would streamline the competitive tendering process by using computer networks to provide sub-contractor's bids to the contractors tendering for a project. Contractors would submit details of the packages that they want to let out as sub-contracts to a central computer system which also contained the project documentation. Subcontractors could then access this information and submit bids to selected contractors via the central computer with any conditions attached. The exchange of data electronically could solve many of the coordination problems that exist in current methods of pricing using paper documentation. Consistency checking could be performed by the computer software on prices before they were submitted to the central computer and on received prices. This would reduce the chance of errors creeping into tenders due to hasty telephone calls and scribbled notes.

CONCLUSION

Hester has demonstrated that the use of knowledge based techniques and

a blackboard architecture can solve some of the problems inherent in the separation of the design and construction of buildings. The use of a central computer based model of the building to coordinate a range of software packages could reduce problems experienced by the procurement team by ensuring that all of the participants have access to current data on the project. The use of modular software could also have wider implications for the use of computers to exchange data throughout the construction industry. The use of Prolog and Flex rather than the traditional algorithmic languages also has significant advantages for the ease of programming and user extensibility.

References

AutoDesk (1992), *AutoCAD Development System Programmer's Reference*, AutoDesk.

Clockson, W F and Mellish, C S (1984), *Programming in Prolog*, 2nd Ed, Springer Verlag.

Eastman, C (1975), The Scope of Computer-Aided Design. In Eastman, *C. (ed), Spatial Synthesis in Computer-Aided Building Design* . Applied Science Publishers Ltd., pp 1-18.

Fenves, S J, Flemming, U, Hendrickson, C, Maher, M L and Schmidt, G (1990), Integrated Software for Building Design and Construction. *Computer-Aided Design*, Vol 22 No 1, January/February, pp 27-36.

Harrington-Lynn, J (1974), The Admittance Procedure, Variable Ventilation. *Building Services Engineer*, 42, November.

LPA (1989), *Flex Expert System Toolkit*, 3rd Ed. Logic Programming Associates Ltd.

Minsky, M (1985), A Framework for Representing Knowledge. In *Brachman, R.J. & Levesque, H.J. (eds) Readings in Knowledge Representation*, Morgan Kaufmann, pp 245-262.

Rawlinsons (1993), Australian Construction Handbook 1993, 11th Ed, The Rawlinsons Group.

Stelzner, M and Williams, M D (1988), The Evolution of Interface Requirements for Expert Systems. In *Hender J A (ed), Expert Systems : the user interface.* Ablex, pp 285-306.

Szokolay, S V (1990), *Thermal Design of Buildings* , RAIA Education Division.
