Implementation of Simulation Based Design Tools in Practice

J A Clarke
Energy Simulation Research Unit
University of Strathclyde, UK

D F Mac Randal
Informatics Department
Rutherford Appleton Laboratory, UK

Over the past decade developments in the building simulation field have given rise to the prospect of a new generation of design tool which has the potential to allow rigorous hypothesis testing at the design stage. While powerful at their core, simulation based design tools suffer from several user interface limitations. This paper describes an attempt to solve some of these problems by developing an Intelligent Front End authoring environment for building performance appraisal in general. The architecture of the developed system, termed IFe is described and its use explained by demonstrating its application in the context of the ESP-r system for building environmental performance prediction. Pie paper then describes how some elements of the IFe are being used within the CEC's COMBINE project to explore the architecture of an intelligent, integrated building design system.

Introduction

From the viewpoint of contemporary practice in building design, simulation programs are over engineered - both in terms of the functionality they offer and the information requirements to service their theoretical algorithms. This gap, between design practice and simulation tool capabilities, can be overcome either by reducing the order of the applications (and hence their information needs) or by endowing the applications with knowledge based interfaces which are sensitive to the needs of designers and are well mapped to the information flows of the design process. This paper is concerned with the latter prospect.

Firstly, the paper describes the architecture of a knowledge based interface construction environment developed under SERC funding. This environment, known as the IFe (Clarke and Mac Randal 1989), is capable of

- supporting design concurrency (designer to designer and designer(s) to application(s) inter-communication)
- preserving audit trail (who did what, when, why)
- accommodating user interface preferences (style of interaction, feedback and tutoring)
- evolving the product model (incremental problem definition and intelligent defaulting)
- and handling applications (data needs, results interpretation and operational control).

Secondly, the paper describes the use of the IFe environment to prototype an intelligent interface for the ESP-r system (Clarke 1985). This graphical, knowledge based interface is able to accept information against concepts matched to different user types and then, by inference, to complete a full problem specification as required by ESP-r. It is also able to preserve the process chronicle, comprising the user dialogue, the problem evolution and the application returns.

Finally, the paper describes how elements of the IFe - and in particular its 'Blackboard' approach to information sharing and its knowledge based approach to user dialogue support, product model evolution and application control - are being applied within the European Commission's COMBINE project (Augenbroe 1991) to control disparate design tools as they access a central product model under design process rules.

The IFe System

The IFe (Figure 1) is built from cooperating modules organised around a central communications *Blackboard* to facilitate multiple use of information. These modules run asynchronously and can examine the Blackboard for information, posting results back to it.

Blackboard

This lies at the heart of the system and has two major functions. Firstly, it stores the data representing the current state of the problem definition as input by a user or inferred by the Knowledge Handler. Secondly, it acts as a communication centre for the other modules.

Blackboard clients are divided into two classes: those at the user-end (the Dialogue, Knowledge and User Handlers) concerned with extracting from the user the building description and the appraisal definition, and those at the back-end (the Appraisal, Data and Application Handlers) concerned with creating the input data and control instructions to 'drive' the application(s). The dialogues corresponding to these two classes are held within separate communication areas on the Blackboard. All Blackboard clients are autonomous processes running asynchronously, with no selection criteria imposed on incoming messages.

The Blackboard is not just a passive data structure. Clients can ask it to create new, named areas. The various clients can then post requests and information to these areas, the name of the poster and the time of posting being recorded. Clients can either explicitly ask the Blackboard for information by identifying the area and the concept, or can ask the Blackboard to keep them informed of any new information posted to a particular area. Firstly, this avoids clients having to poll the Blackboard. Secondly, it provides a mechanism whereby two, or more clients can create a Blackboard area to serve as a communication channel between them. For example, in the current ESP-r implementation a "user dialogue" area is used to pass information between the Dialogue and Knowledge Handlers in order to validate user inputs, provide essential feedback and feed the inference process.

Dialogue Handler

The Dialogue Handler is responsible for managing and monitoring all user interactions and implementing the level of communication as necessary for a given user and task combination. It acts to coordinate the interaction with users against *User Class* definitions which establish the data and modelling concepts that are acceptable to particular users such as architects, engineers, modelling specialists, students and so on. By possessing a configurable human-computer interface, the Dialogue Handler is able to produce interface instances which are tailored to a user's conceptual outlook, level of experience and stage reached in the design process. Unlike conventional programs where the interface remains static throughout a session, by keeping track of a user's progress, the Dialogue Handler can, in conjunction with the User and Knowledge Handlers, tailor the dialogue to the user's level of expertise and performance history.

The basic function of the Dialogue Handler is to pass the user's inputs to the Blackboard and to pass messages and requests, as posted by other Blackboard clients, back to the user. To perform this function, the Dialogue Handler has several mechanisms at its disposal. The primary mechanism is a generic forms program which can manipulate a set of forms which correspond to a given User Class. Each form entity (a labelled field, a button, a multi-option pop-up, a bit-map, etc.) corresponds to a particular *Concept* within the active User Class: for example window width, number of rooms, project name, simulation time-step and so on. Groupings of related concepts are located on the same form to comprise a *Meta-Concept*: for example building geometry, construction, control system, appraisal definition and so on. Meta-Concepts can contain nested Meta-Concepts, allowing a complete hierarchy to be specified. In this way a set of Meta-Concepts (forms) can be used to define a given User Class in terms of only those related Concepts that are deemed acceptable. Via these forms, the user can ask about Concepts and associate values with them. While one User Class may involve many Concepts (many user inputs), another may involve only a few. By relying on a greater degree of inference in the latter case (the function of the Knowledge Handler), the same application(s) can be offered to both users.

Knowledge Handler

The function of the Knowledge Handler is to manipulate several independent knowledge bases which exist to control the user dialogue, collect and validate user entries, make whatever inferences are appropriate and store on the Blackboard a representation of the building and required appraisals. This requires a mixture of conventional procedural programming, event-driven programming and rule-based inferencing. In essence the Knowledge Handler is an autonomous inference engine based on a Prolog interpreter (Hutchings 1986) with the ability to dynamically load partial knowledge bases each of which are matched to the Meta-Concepts of the active User Class. This means that the user dialogue is at all times under knowledge base control and that User Class definitions, in whole or in part, can be substituted at run-time should a particular definition prove unsuitable. At run-time the Knowledge Handler monitors the user dialogue as stored on the Blackboard in order to

- obtain the user's input statements
- convert these into Prolog predicates
- use these predicates to validate the user inputs
- and, by inference, to further complete the problem/ appraisal definition.

In practice the Knowledge Handler has access to several *Knowledge Bases*, each corresponding to a Meta-Concept; each Knowledge Base only being loaded when the related Meta-Concept (form) is activated. Each Knowledge Bases exists to build on the Blackboard that part of the problem definition to which the Meta-Concept relates. Based on what is already known about the problem, a Knowledge Base can deduce what concept values are sensible (that is intelligent input validation), and how to derive appropriate default values for Concepts required by the target application(s) but not addressed directly by the active User Class (that is intelligent defaulting).

Because User Classes will normally require their own unique dialogue, Knowledge Bases are also responsible for coordinating the user dialogue . This is achieved by including Prolog code to provide feedback, help and guidance to the user. To this end, a Knowledge Base contains knowledge about the Meta-Concept to which it relates and about the capabilities of the Dialogue Handler in terms of its control syntax. In practice Knowledge Bases are constructed from pre-formed predicates, supplied with the IFe, which support dialogue and application control, product model storage, inferencing, feedback and the like.

User Handler

The User Handler has two main tasks. The first is to collect information against which a user's progress might be judged. The second is to modify the interaction in the case of poor user progress.

Like the Knowledge Handler, this module is based on a Prolog interpreter. Its function is to set the appropriate User Class, on the basis of user type and experience information, to ensure that the subsequent session is tailored to the user's skill level and that appropriate guidance, feedback and help is given during the session. At the present time real, dynamic user modelling has not been attempted. Instead, the user is initially classified, from a database or explicit user input, and the corresponding Knowledge Base is established for use by the Knowledge Handler.

During a session, the user's progress is assessed by monitoring the user dialogue as recorded on the Blackboard. From these data the number of errors, changes of mind, backtracks and Knowledge Handler overrides is derived. On the basis of this, or upon explicit user request, the User Handler can change the User Class. From then on, the Knowledge Handler will automatically pick up those Meta-Concepts and Knowledge Bases associated with the new User Class. It should be recognised however that the automatic mapping from one User Class to another is a non-trivial task which has not to date been attempted.

Appraisal Handler

In the current implementation of the IFe, users are required to specify their appraisal objectives in terms that the Appraisal Handler can understand - that is the Appraisal Handler is not knowledge based. These appraisal possibilities are made known to the Appraisal Handler in the form of parameterised Unix^{TM} Shell scripts (Bourne 1982), each representing a particular *Performance Assessment Method* (PAM). It is the Appraisal Handler's task to select the appropriate PAM and, based on the information available on the Blackboard, to assign suitable values for the PAM parameters. These will include the particular application program to be invoked at each stage in the methodology (eg ESP-r for energy appraisal) and the values for the design or performance parameters on the basis of which decisions will be made (eg PMV for comfort if the application is ESP-r). After this task has been completed, the data is posted back to the Blackboard where it is accessed by the Data Handler whose function is to prepare the input data required by the identified programs. The form of a PAM, and its related script are described elsewhere (Clarke and Mac Randal 1989).

Data Handler

This module creates, from the information supplied by the user and that inferred by the Knowledge Handler, the problem description as required by the application(s) identified within the posted PAM. This is achieved via a Blackboard data searching mechanism utilising generalised pattern matching controlled by a *Data Definition Script* (DDS) which defines, again in parameterised manner, the data preparation procedure of the targeted application(s). The parameters of the DDS define the data to be obtained from the Blackboard based on a pattern matching technique. By executing this script, the Data Handler builds the required data-set. Details of a DDS as established for use with the ESP-r system are given elsewhere (Clarke and Mac Randal 1989).

This approach ensures that the data requirements and specific formats of multiple applications can be extracted from a single Blackboard data model, assuming that this data model (as defined by direct user inputs and supplemental inference) is complete. Most importantly, the Knowledge Handler's inference mechanism can be used to map from one application's data entity to the semantically different, but extensionally similar entities of other applications, thus ensuring input equivalence.

Application Handler

This module orchestrates the application(s) against the selected PAM and problem definition. It also receives the application returns for direct display or transmission to the Dialogue Handler via the Blackboard (usually via a results Meta-Concept established for the purpose).

Using the IFe modules it is possible to establish intelligent interfaces of varying complexity which are able to converse with users in the terminology and interaction style deemed appropriate (the

Dialogue Handler); generate problem descriptions against a specific product model deemed subservient to the targeted applications (the Dialogue and Knowledge Handlers); collect, organise and store this product model, and the dialogue and inferences which produced it (the Knowledge Handler and Blackboard); generate the program-specific control inputs (the Appraisal Handler) and the application-specific input data-sets (the Data Handler); invoke and control the application programs (the Application Handler); and monitor the user's progress, giving assistance where necessary (the User Handler). By separating the user dialogue and application control functions it is possible to connect a given interface to several applications or, conversely, multiple interfaces (different User Classes) to the same application(s).

Prototyping an Intelligent Interface for ESP-r

The IFe is a generic, intelligent interface production environment capable of supporting the rapid prototyping of knowledge based, graphical user interfaces. To explore the system's capabilities and test its component parts, the system was used to construct an intelligent interface for use with the ESP-r system for building energy/ environmental simulation. The underlying User Class corresponds to a computer literate engineer.

The starting point was to understand the viewpoint of the User Class and to relate this to the data requirements of ESP-r. For example, while one user may be able (or expect) to input directly much of the data required by ESP-r, another may have difficulty, technically or semantically, with the same data. With the ESP-r interface described here the assumption was that the user is able to understand most of the data but that the interface should assist the input process by providing defaults, inferring values from previous inputs and concealing the underlying data structures. Using the standard, pre-formed predicates offered by the IFe, interface construction proceeded in the following steps:

- Identification of the concepts that the user finds acceptable in specifying the building and the required analysis.
- Grouping of these concepts into a hierarchy of Meta-Concepts to help organise the collection and processing of the user inputs.
- Establishing a Knowledge Base for each Meta-Concept and arranging that each Concept has a Prolog predicate to validate and store the input.
- Designing a form corresponding to each Meta-Concept. (This requires selecting the best field type - text, menu, button, etc. - to help the user input the data.)
- Adding to each Knowledge Base the predicates to handle the switch of focus when the user moves from one Meta-Concept to another. Usually when a Meta-Concept is selected, some initialisation is necessary - for example to remove some previously displayed Meta-Concept or to inform/ remind the user of the information already held/ inferred against this Meta-Concept from a previous session. Similarly, when a Meta-Concept is de-selected this might trigger the calculation of default values for concepts not previously set.
- Adding to the form the necessary fields (usually buttons) to allow the user to switch the focus of discussion to another Meta-Concept (nested form) and back again.
- Installing the Meta-Concepts (set of forms) and matched Knowledge Bases in the IFe library.
- Updating the IFe's User Class directory to make the new User Class available for selection. (Further, the names of any real users of this class can be made known to the User Handler so that, in the absence of a specific selection, the IFe will default to the pre-specified User Class.)

Probably the most important aspect of interface design is establishing the Meta-Concept hierarchy. This should be designed to lead users naturally through the input process while allowing them complete control over the order in which data is input. Figure 2 shows some of the Meta-Concepts as as established for use with ESP-r to represent an Engineer User Class. At the top level, a clear distinction is made between the acquisition of data describing a problem and that which describes the required performance appraisal: each are mutually exclusive Meta-Concepts. On the other hand, the location and function Concepts (within the "bld_spec" Meta-Concept) are independent of the "geometry" Meta-Concept and hence both forms may be displayed simultaneously. (Obviously, this should not be taken to extremes, both to avoid overwhelming the user with requests for data and because of lack of screen space.) It is here that the dynamic nature of the Dialogue Handler and the declarative nature of the Knowledge Handler together permit a human-computer interaction that is truly cooperative, neither the system nor the user dictating the sequence of data collection. Figure 3 shows a typical screen image resulting from an IFe session with this User Class active.

At the data collection level, the mechanics of the interaction required careful consideration. The provision of context-sensitive defaults, as inferred from previously inputs and the offering of multiple ways of supplying data makes for a more natural interface. For example, when inputting the location of the building, the user can:

- select from a set of Knowledge Base specified locations
- type in a location name (and supply its latitude and longitude if the location is unknown to the Knowledge Base.
- or point to a specific place on a map application from which the place name, and its latitude and longitude can be deduced.

The important point is that using the Dialogue Handler and a single Prolog predicate, the inclusion of the map program was almost as simple as providing the boxes into which the user could type the latitude/ longitude. Based on the location, however supplied, it is then a straightforward task to infer values for standard and obscure Concepts such as climate collection, site elevation, atmospheric turbidity, fuel type, control schedule (if the building type is known) and so on. These data could then be offered as Concept suggestions for, as yet, unaddressed Concepts.

The idea of providing multiple ways of supplying data can be taken further than the simple "name or numbers" options for specifying the location. When inputting building geometry, for example, ESP-r demands that exact dimension be specified. However, in many cases, and especially at the early design stage, such detailed information may not be available. A completely different way of describing a building from the traditional "what its made of" method is, by analogy, "its like this ... but with these differences ...". To this end the developed User Class includes the classification and presentation of a set of past designs from which selections can be made and changes applied until an acceptable surrogate design is obtained. This browse and edit mechanism, though simple in concept (and implementation), when combined with the IFe's inferencing mechanism allows simulation programs such as ESP-r to be employed at a much earlier design stage than is possible with traditional interfaces no matter how user friendly. To this browser mechanism add alternative approaches to geometry definition based on a CAD modellers and the like and the user now has a range of basic but different approaches for the definition of building geometry each adapted to a different information context and user preference. Figure 4 shows the geometry definition options as established within the ESP-r interface: form filling (the "form_fill" Meta-Concept), a CAD modeller ("draw"), foreign file import ("cad_file") or on the basis of previous design browsing ("bld_browse"). In the first case - form filling - the user is required to associate values with the specific data entities which define a zone's topography and topology. To do this, a separate form ("zone_geom") is displayed for each zone. This form offers several alternatives, from simple box dimensioning to full vertex/ edge specification. Once checked, the data is held on the Blackboard and also asserted as facts in the Knowledge Base. In the second option, some external CAD modeller (external to the IFe but invoked by it) is used. The button associated with this option posts a tuple to the Blackboard requesting that the CAD modeller be run, the results to be placed in a specified external file. This output file is processed as a foreign file, the third option, which requires filtering before being imported to the IFe (where it is held in the same manner as if it had been input using the first option). The fourth option starts the browse program. This scans a standard directory containing a number of designs previously defined by, or otherwise made known to, the IFe. Once selected, the data for that design is loaded onto the Blackboard and asserted within the related Knowledge Base. The detailed geometry forms of the first option can then be used to modify the geometry as appropriate. In this way the browse facility allows past designs to be offered to an ESP-r user so that entire models can be easily selected and modified.

One further action is now required to complete the ESP-r interface: the incorporation of a representative set of Performance Assessment Methods (PAM) within the Appraisal Handler together with the corresponding Data Definition Scripts (DDS) within the Data Handler. When the user selects the analysis to be carried out, the Knowledge Base activates the appropriate PAM and DDS and provides the necessary parameterisation. As well as invoking a specific PAM/ DDS, the predicates associated with the "analysis" Meta-Concept can run pre-defined groups of assessments to automatically generate standard reports.

COMBINE

The COMBINE project (Augenbroe 1991) is a European concerted action to develop an integrated data model which can support the needs of the range of design tools as typically employed within the building design process. Essentially the project entails the following tasks:

- The identification of the data requirements of several design tool prototypes (for CAD, for lighting design, for thermal design, for HVAC design, for layout planning, etc.), the conflation of these data into an integrated data model (IDM) and their representation in the EXPRESS language (Spiby P 1991) to define, unambiguously, the semantics associated with each data entity.
- The representation of the IDM to emerge from the first task within an object oriented paradigm, the encapsulation of this data model within an object oriented database and the provision of a set of interface tools, resulting in a

general purpose data exchange system (DES).

- The building and field testing of a prototype, integrated building design system (IBDS) using the IDM and DES in conjunction with the selected design tool prototypes.
- And, finally, the experimental addition of design semantics, inter process control and transaction management to research the feasibility of producing an intelligent IBDS, or IIBDS.

It is within the context of the last two tasks that elements of the IFe will be used to explore an approach to the construction of an IBDS which surpasses the limitations of mere data exchange by supporting application interaction handling. This issue will be researched by experimenting with alternative approaches to design tool interaction handling and scheduling - that is transaction management superimposed on the data exchange facilities provided by the IFe's Blackboard. This experimentation will, in turn, be carried out against a background of the different possible models of design (generate and test, goal directed search, constraint propagation, heuristic search, autonomous agents and the like) to ensure that the IIBDS has the necessary functionality to enable at least existing design tools to be used within these design model contexts. (Note that the intention is not to develop an IIBDS which adheres to any one of these possible models of design merely to ensure that it could.)

Specifically, the IIBDS will be constructed against a research programme which explores the following issues:

- Application interaction management including process chronicle storage and manipulation (in support of future concurrency needs and audit trail requirements).
- The imposition of inter-application control and/ or scheduling using i) the IFe's knowledge-based approach (ie a Prolog inference engine configured as an additional supervisory Blackboard client and/ or ii) a programmable Blackboard facility for implementing/ enforcing pre-specified application processing sequences.
- The incorporation of an DES schema searching mechanism (ie remove the necessity for the design tools to know the DES schema under which the data is actually held). This could be based on i) a simple Blackboard design tool control mechanism (design tools communicate directly with the DES on instruction from the Blackboard) and/ or ii) full transaction processing with the DES (the design tools get their data from the Blackboard which alone communicates with the DES).
- A Blackboard extended syntax to enable a two-way actor dialogue (the intention here is only to handle the semantics of design tool interaction, not the internal processes of the design tools).

In relation to the two Blackboard/ DES interaction possibilities: the former is the easier given the existence of a separate DES and the fact that the design tools are already conceptually arranged around the DES. The latter approach - in which the Blackboard controls the DES directly - is more powerful in that it is extensible (any new design tool can be accommodated by adding it to the Blackboard), flexible (design tools can be isolated from storage schema changes) and, if achieved, would be the first step towards semantic information exchange as opposed to simple data exchange.

Thus, within COMBINE, the IFe is being used as an IIBDS prototype builder to conjecture, (relatively) rapidly prototype and test alternative approaches.

Conclusions and Future Work

An intelligent front end environment has been developed which enables a human-sensitive approach to building performance appraisal in the context of the multiplicity of applications now available. Given the complexity of the subject, it will require a further R&D effort to evolve robust products which can be used routinely to create intelligent interfaces for the many applications and user types.

The IFe in particular could be refined by improving the efficiency and flexibility of its knowledge handling, user class definition and data manipulation functions. In the first two cases this could be achieved by the development of software tools to assist in knowledge base creation and user class definition. This would reduce the level of computing science expertise required and so allow a greater number of application specialists to work with the IFe directly. In the last case a simple schema definition language could be defined which is capable of handling the requirements of the IFe. The Blackboard would then be enhanced by the addition of a mechanism to search the schema, so that clients can request specific data without specifying where in the schema it is stored. This would greatly increase the efficiency with which new or existing clients could be serviced, and by reducing the need to know about the data structuring used by other clients, will further improve the IFe's modularity. This, in turn, would ensure that major new components, such as new appraisal methods or user classes, could be added more easily in future. In the longer term, the development of a plan recognition function for the Appraisal Handler and a user modelling capability for the User

Handler would enable the IFe to address the wider, less technically focussed goals characteristic of many designers.

By giving the profession access to the power of contemporary and future software systems from a single building description achieved from only the information the user is able to supply, the possibility of truly integrated, multi-criteria design appraisal is enabled. This, in turn, will allow designers to make the necessary trade-offs in the search for an optimum solution and so arrive at more robust designs.

References

Augenbroe G (1991) 'Integrated Building Performance Evaluation in the Early Design Stages' *Proc. First Int. Symp. on Building Systems Automation-Integration* Madison.

Bourne S R (1982) *The UNIX system* Addison-Wesley

Clarke J A (1985) *Energy Simulation in Building Design* Adam Hilger Ltd Bristol and Boston.

Clarke J A and Mac Randal D F (1989) 'The Application of Intelligent Knowledge Based Systems in Building Design', *Final Grant Report*, Science and Engineering Research Council, Swindon, England.

Hutchings A M J (ed) (1986) 'Prolog User Manual' *Report AIAI/PSGml/86* AI Applications Institute, University of Edinburgh, UK.

Spiby P (Ed) (1991) 'EXPRESS Language Reference Manual' *ISO TC184/SC4/WG5 Document N14.*

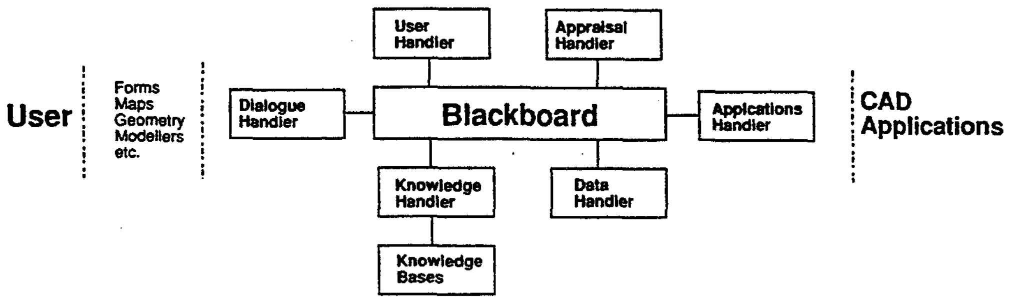
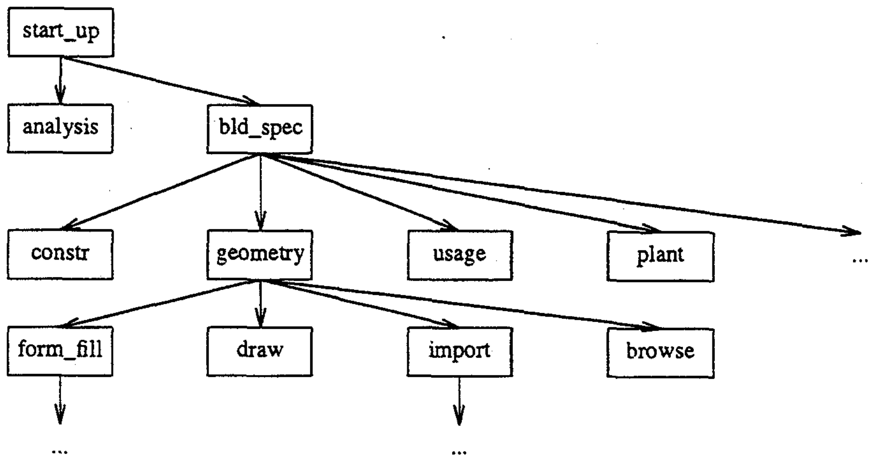


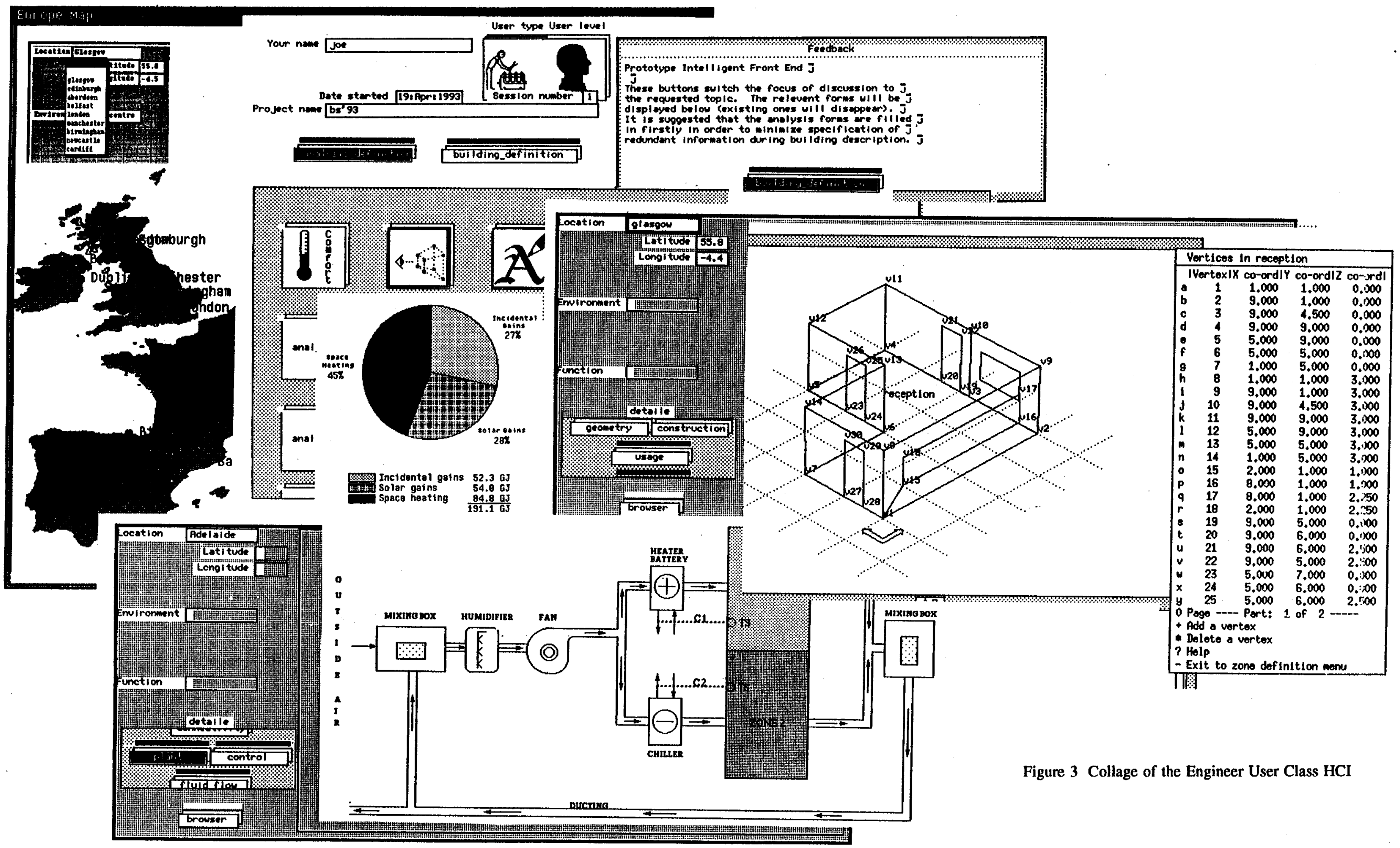
Figure 1 Architecture of the IFe



Key:

Meta-Concept	Purpose
start-up	User Class setting and project identification.
analysis	Selection of required PAMs.
bld_spec	High level building description information (e.g. location and function).
bld_browse	A building design browse facility.
geometry	Building geometry definition.
form_fill	Zone-by-zone geometry definition by form filling.
cad_file	Building geometry definition by importing a foreign CAD file.
draw	Building geometry definition by a CAD modeller.
construction	Constructional attribution of defined geometry.
usage	Zone operation definition.
plant	Definition of plant components and networks.

Figure 2 Meta-Concepts for the ESP-r Engineer User Class



Your name joe

Date started 19:Apr:1993

Project name bs'93

Session number 1

Feedback

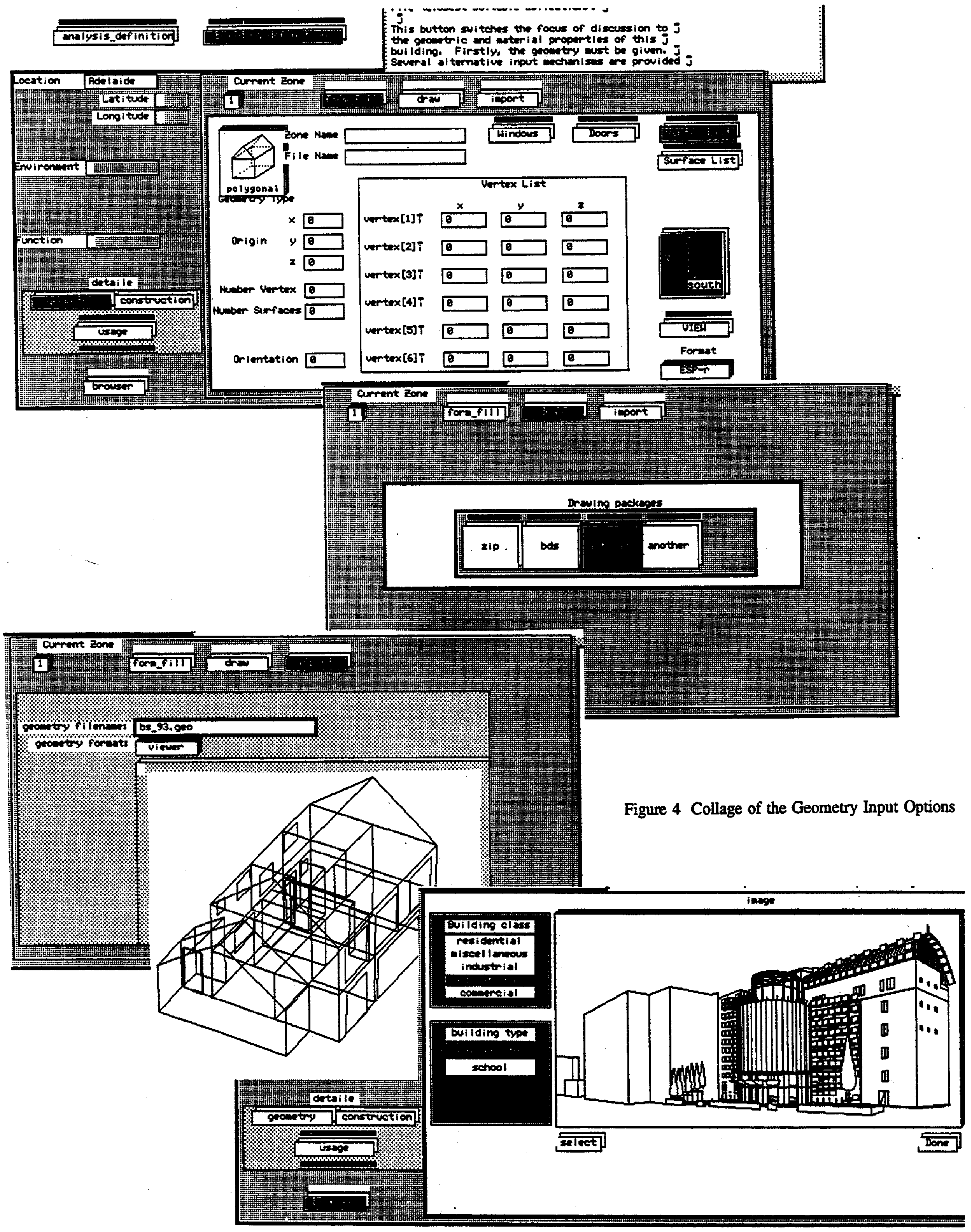
Prototype Intelligent Front End
These buttons switch the focus of discussion to the requested topic. The relevent forms will be displayed below (existing ones will disappear). It is suggested that the analysis forms are filled in firstly in order to minimize specification of redundant information during building description.

Location glasgow
Latitude 55.8
Longitude -4.4

▨	Incidental gains	52.3 GJ
▦	Solar gains	54.0 GJ
■	Space heating	84.8 GJ
		191.1 GJ

```
Vertices in reception
 |Vertex|X co-ord|Y co-ord|Z co-ord|
a     1   1.000   1.000   0.000
b     2   9.000   1.000   0.000
c     3   9.000   4.500   0.000
d     4   9.000   9.000   0.000
e     5   5.000   9.000   0.000
f     6   5.000   5.000   0.000
g     7   1.000   5.000   0.000
h     8   1.000   1.000   3.000
i     9   9.000   1.000   3.000
j    10   9.000   4.500   3.000
k    11   9.000   9.000   3.000
l    12   5.000   9.000   3.000
m    13   5.000   5.000   3.000
n    14   1.000   5.000   3.000
o    15   2.000   1.000   1.000
p    16   8.000   1.000   1.000
q    17   8.000   1.000   2.250
r    18   2.000   1.000   2.250
s    19   9.000   5.000   0.000
t    20   9.000   6.000   0.000
u    21   9.000   6.000   2.500
v    22   9.000   5.000   2.500
w    23   5.000   7.000   0.000
x    24   5.000   6.000   0.000
y    25   5.000   6.000   2.500
0 Page ---- Part:  1 of  2 -----
+ Add a vertex
* Delete a vertex
? Help
- Exit to zone definition menu
```

Figure 3 Collage of the Engineer User Class HCI



This button switches the focus of discussion to
the geometric and material properties of this
building. Firstly, the geometry must be given.
Several alternative input mechanisms are provided

Location: Adelaide

geometry filename: bs_93.geo

geometry format: viewer

Format: ESP-r

Figure 4 Collage of the Geometry Input Options