The Development of an Intelligent, Integrated Building Design System Within the European COMBINE Project

J A Clarke*, J W Hand*, D F Mac Randal#, Strachan

ABSTRACT

There are two main issues to be resolved in order that design tools can be used in cooperative mode, each communicating with the other. Firstly, there is a need to put in place a consistent product model of a building and its systems from which disparate design tools can obtain their inputs and return their outputs. Secondly, there is the requirement to manage the transactions between users and design tools. Tbese issues were addressed within the European COMBINE project. This paper is concerned with the latter issue. It describes the basis and operation of an intelligent, integrated building design system, or IEBDS, which is able to coordinate designer-to-designer, designer-to-application and application-to-application transactions, against rules which describe the purpose of a given design session. The HBDS is able to address 'shallow' control, where the design tools are sequenced, and 'deep' control, where knowledge is introduced in relation to design purpose so that design tool use is constrained within a given design session.

INTRODUCTION

To bring real benefit, building performance modelling must be integrated within the design process. Traditionally, as summarised in Figure 1, the use of design tools has followed a *tool-box* approach in which the designer is expected to recognise a particular task, locate a suitable program, run it and translate its outputs to appropriate changes to the design hypothesis.

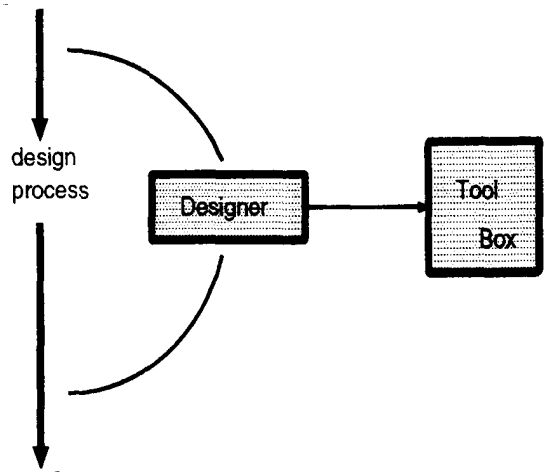

Figure 1: Tool-box approach to design. (After MacCallum 1993)

Clearly this is an inadequate approach in that the tools are decoupled from the design process and require the designer to be knowledgeable about each tool's capabilities, control syntax and semantics.

An alternative approach is summarised in Figure 2. Here the computer resource is (somehow) integrated within the design process. In such a *Computer-Supported Design Environment* (CSDE), the designer evolves the design hypothesis in such a way that the tools are able to automatically access the data describing

* ESRU, Energy Systems Division, Faculty of Engineering, University of Strathclyde, Glasgow G1 1XJ. email: esru@strath.ac.uk, phone: +44 141 552 4400 X3986, fax: +44 141 552 8513.

Informatics Department, Rutherford Appleton Laboratory, Chilton, Oxon. 0X11 0QX. email: damian@inf.rl.ac.uk, phone: +44 1235 445403, fax: +44 123541 44589.

the design and give feedback on all aspects of performance and cost in terms meaningful to the designer.

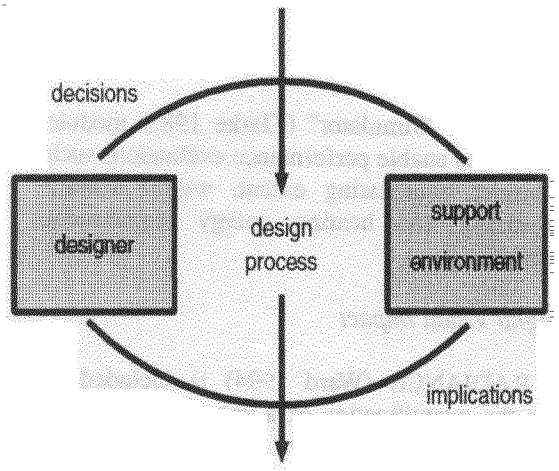

Figure 2: CSDE approaches to design. (After MacCallum 1993)

The attainment of such a CSDE is a non-trivial task involving the development of *integrated product models* and *intelligent interfaces*.

In the former case, the complexity stems from the temporal dimension of the design process, i.e. the evolution of the describing data against an uncertain information context and the different professional viewpoints and vocabularies. Within the European Commission's COMBINE project (Augenbroe 1992), the objective was to evolve an integrated data model (IDM) which could satisfy the needs of a representative set of design tools (for energy analysis, CAD, lighting, regulations compliance, layout planning and the like). The IDM (or strictly speaking the Data Exchange System, or DES, which is an implementation of the conceptual IDM) is then able to receive from, store and deliver data to these design tools in a manner which ensures that these data are accompanied by their related semantics: in the COMBINE project the EXPRESS language (Spiby 1991) is used to achieve this end. A key issue is the structure of the data model to ensure efficient exchange and allow future extension as additional or more powerful design tools are added. The decision to base this data model on the object oriented paradigm and to contain it within an object oriented database was seen as the way to achieve these goals.

The development of an intelligent interface, the subject of this paper, is non-trivial because of the complexity of the transactions which require to be managed in terms of:

- supporting design concurrency (designer to designer and designer(s) to application(s) inter-communication)
- preserving audit trail (who did what, when and why)
- supporting a constructive user dialogue (style of interaction, feedback and tutoring)
- evolving the product model (incremental problem definition and intelligent defaulting)
- and handling application semantics (application to product model and application to application).

One COMBINE task explored the form of an intelligent, integrated building design system, the IIBDS, which could handle these issues. This was done via a rapid prototyping approach by which different scenarios for design tool transaction management and data exchange were explored. The prototyping environment was the IFe system (Clarke and Mac Randal 1993), the modules of which are:

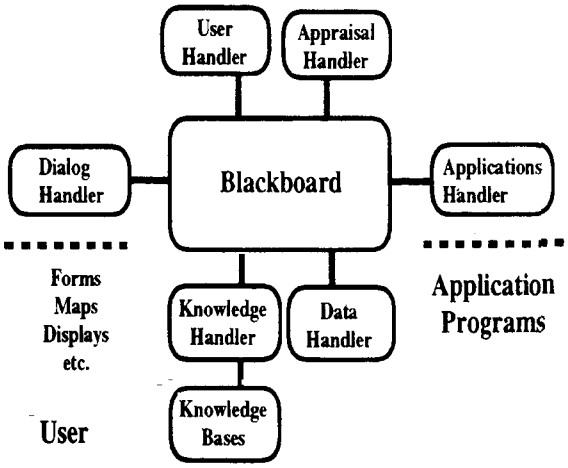

Figure 3: The IFe System.

- A Blackboard to serve as a communication centre for its various clients. By this means concurrency can be supported and traceability achieved through the collection, organisation and storage of the session chronicle.
- An Application Handler to control the various design tools, pass them their data and receive their returns.
- A Knowledge Handler to control design tool access to the product model and the

communication with the designer (verification of entries, supplemental inferencing and feedback/guidance).

- A Dialogue Handler to converse with the user by means of acceptable concepts which relate to the different user types and levels of expertise.
- A User Handler to track the user's progress and ensure the system responds in an appropriate manner.
- An Appraisal Handler to hold the design tool control syntax against standard performance assessment methodologies.
- A Data Handler to extract an application's data from the Blackboard and organise these data in the required format.

The IIBDS is therefore based on a Blackboard/Knowledge Handler architecture to effect purpose-specific design tool control. It incorporates several real design tools (DTs), which can be configured to support real design sessions.

IIBDS DESIGN TOOLS

The aim of the IIBDS prototype was to provide a mix of design tool functions (DTFs) by which a number of realistic design sessions could be accommodated, each to a realistic level of complexity. While some DTFs require substantial interactions with the user (e.g. a CAD program), others may be purely computational (e.g. a regulations compliance program) so that the user need not be made aware of their existence. The DTs known to the IIBDS at the present time are as follows.

Architectural CAD

Two Architectural CAD DTs are supported: AutoCAD Release 12 (Autodesk 1989) running in native mode with a constrained set of commands and drawing entities consistent with the DES and MicroStation (Conforti 1994) configured as an on-line interface to the DES.

Geometrical Attribution

The attribution of the problem geometry generated by AutoCAD is accomplished via the "Project Manager" module of ESP-r (Hand 1994), hereinafter called ATTRIBUTE. Attribution is in terms of construction, occupancy and control.

U-Value Compliance

The regulations compliance of a design is assessed by BRC (Rode 1993) which relates to several European national building regulations.

Thermal Energy

ESP-r's "Simulator" (Clarke 1985) module is included to enable performance evaluations such as summer overheating extent, winter heating plant sizing and heating energy requirement estimation.

Daylight/ Visual Impact

RADIANCE (Ward 1994) is included to enable the quantification of a zone's illumination levels and the production of visual impact information for the overall building. It accepts problem descriptions as generated by the ATTRIBUTE DT.

PROCESS MODELLING

In dealing with the design process at the level at which COMBINE operates, the IIBDS must support the flow of data/information between work-steps (or DTFs) and event-handling in terms of starting and stopping the design tools. The mechanism adopted to handle these issues within the IIBDS is as follows.

The required process model is captured in the form of a Petri-Net (Javor 1993) and then this is transformed into a file of Prolog facts. This gives the basis of the process as a formal description. This file (hereinafter termed the PNF for Petri-Net File) is then dynamically read into the IIBDS' Application Knowledge Handler (AKH) where it is used by a "process support inference engine" to animate the process. By modifying this process knowledge base (as held within the AKH), it is then possible to control the rigidity of the system, its handling of parallelism, etc.

In the current IIBDS, three process models are available (each with potentially many instances) corresponding to:

Case 1: where DT invocation is not sequenced nor functionally constrained so that the designer is able to invoke the DTs in any order and activate their internal functions as required. That is the PNF is used only for DT access control.

Case 2: where the DTs are sequenced but not functionally constrained so that DT selection is prescribed while function invocation is not. That is the PNF controls DT ordering but DT use is opportunistic; concurrency is allowed.

Case 3: where the DTs are both sequenced and functionally constrained so that the system, not the designer, controls the order of DT selection and the invocation of the DTF. (But note that it is the DTFs that are being automated, not the design evolution. The designer remains in control of the process and whether the outcome of a DTF is acceptable or otherwise.) In this way the PNF enforces rigid DT use but no concurrency is allowed.

The process model corresponding to Case 1 therefore relates to the 'shallow' control issue, by which DT transactions are managed, while the model corresponding to Case 3 relates also to the 'deep' control issue, by which knowledge is introduced in relation to design purpose so that the use of the DTFs is constrained within a specific design session.

The PNF can be changed in mid-process, should it become necessary to adapt the rigidity of the design process. The external Petri-Net description and the dynamic loading makes it easy to change the process being enacted. Note however that at the present time no tools are available for process model design or to check that any new process model is consistent with the current state.

Each node in the Petri-Net corresponds to a a design function and triggers a knowledge predicate which "knows" what should be done at this point in the process, i.e. it handles the internals of the design function. This is where the problematic issue of concurrency is handled. The knowledge base has access to the IIBDS' Blackboard (i.e. the design process state) and to the DES (i.e. the state of the problem description). This knowledge base will either be established to react only to the Petri-Net (when in prescriptive mode of operation), or to react to the design process state (when in reactive mode of operation). At the present time these two state are mutually exclusive since they are controlled by preventing the knowledge base from examining the Blackboard's Journal area in the former case. After deciding to carry out the design function, the knowledge base ensures that a) the data required for the task is available, b) starts the appropriate design tool at the appropriate point and then c) hands control to the user. It then monitors what the design tool is doing and finally ensures that the results of the tool are captured and propagated. While the DES is responsible for the handling of the data, the AKH is responsible for driving the process (i.e triggering state changes in the Petri-Net), propagating information to other knowledge bases and keeping track of the design status and history.

The process and design tool knowledge bases are event driven and operate asynchronously. This enables concurrency. Event driven controllers can handle any amount of concurrency, subject only to their ability to "understand" what the other controllers are "saying". In practice, unconstrained concurrency is of little value as it is inherently unstable and unpredictable. The design tool knowledge base is therefore made subordinate to the process knowledge base, which activates/de-activates the former as appropriate. By activating more than one at a time, Petri-Net handling can effectively move from single token passing to coloured token based. Furthermore, design tool knowledge bases can be forced to listen only to the Petri-Net, giving a slavish compliance to the specified process, or encouraged to react to other tools giving a more dynamic, context sensitive system.

IIBDS EXTERNAL VIEW

Figure 4 shows the arrangement of the IIBDS DTs corresponding to a Case 2 process model (DT invocation sequenced but use unconstrained). On entering the design session the user is required to use AutoCAD to create a new problem geometry (while complying with a set of entity and topological constraints). On exiting AutoCAD, ATTRIBUTE is used to complete the site, composition and operational characteristics of the problem. On completing attribution the user is presented with a choice of compliance checking or thermal/lighting performance appraisal. In the case of compliance checking the conclusions provided will influence a user's choice to modify the problems geometry (via AutoCAD), its composition (via ATTRIBUTE) or invoke either an energy/comfort assessment via ESP-r or a lighting/visual evaluation via RADIANCE. Finally, the user can either revisit AutoCAD or ATTRIBUTE or exit the design session. It is emphasised that although the IIBDS supports a cooperative dialogue between the user and the above DTs, this design session, though sequenced, possesses no knowledge about design purpose.

Figure 5 summarises the functionality of AutoCAD as deployed within the IIBDS. Because this tool is not "on-line" with the DES, its use must be constrained so that, for example, an isolated line cannot be defined which would have no meaning within ATTRIBUTE.

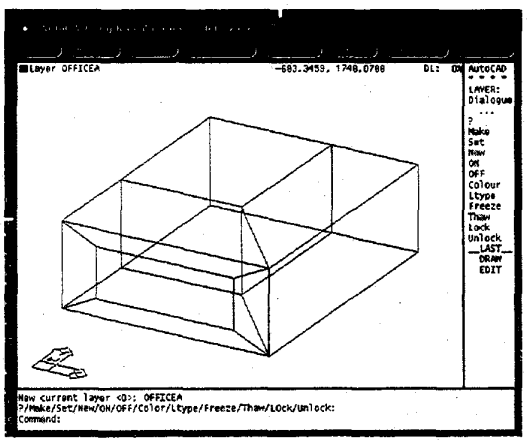

Figure 6: Initial AutoCAD session geometry.

The result of an initial AutoCAD session might result in a problem representation such as that shown in Figure 6. This contains a simple cubic space bounded on two sides by an 'L' shaped space which includes a window. This geometry is passed to the DES before the ATTRIBUTE DT is invoked. Upon entry to ATTRIBUTE, the geometry as described within AutoCAD is recovered from the DES and the following functionality is activated.

- Description of the site in terms of location and climate.
- Checking of each of the geometric entities for significant errors. If any are found they can be corrected and reported back to the DES.
- Contiguity checking for all zones.
- Constructional attribution of the geometric entities. A typical session is shown in Figure 7.
- Operational attribution by zone.
- Control attribution by zone.
- Zone quantification in terms of areas, volumes, U-values, etc.
- Zone view factor estimation.

After the problem is attributed, the current problem state is returned to the DES. Depending

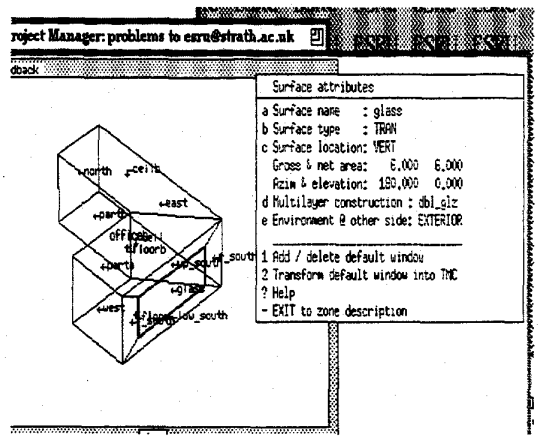

Figure 7: Constructional attribution of a surface.

on the complexity of the problem and working preference, the Figure 4 Petri-Net allows the designer to revisit this DT to add further attribution as design information becomes available.

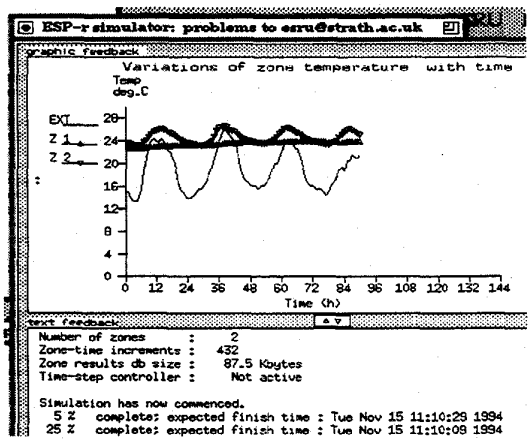

Figure 8: An ESP-r simulation in progress.

Figure 8 was captured during the operation of the thermal evaluation DT and indicates a slight overheating problem.

Finally, Figure 9 shows a typical image as generated by the visualisation DT.

IIBDS INTERNAL VIEW

Figure 10 shows the internal structure of the IIBDS. There are two knowledge handlers, corresponding to user and application control, communicating through the Blackboard areas as

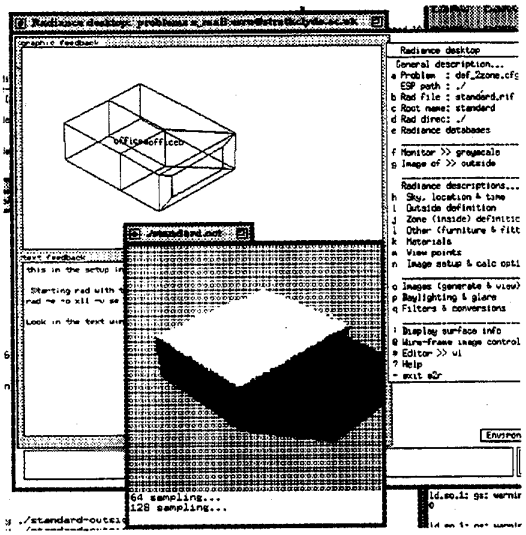

Figure 9: Visual assessment via RADIANCE.

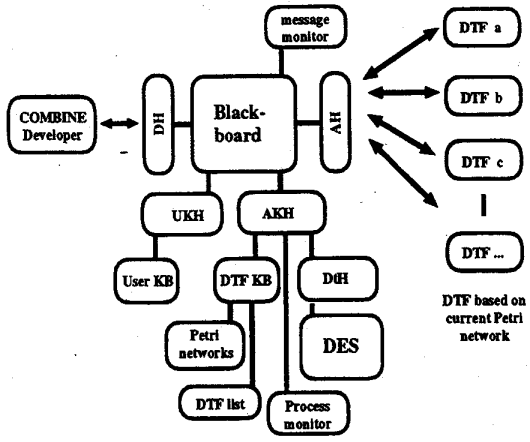

Figure 10: Structure of the IIBDS.

indicated. The message passing between the user and application domains has been isolated within a "transaction" area of the Blackboard. The aim of introducing knowledge in relating to design purpose is further supported by the addition of a "journal" area on the Blackboard. This is a repository for the aggregate log of transactions within the system and is used to feed the Prolog predicates of the design session knowledge base. In particular, the nature of the DTs presented to the user, and how they are sequenced and constrained, is supported by the addition of design process knowledge to the AKH. This has been achieved by arranging for the AKH to load the Petri-Net representations as implied by the user's choice of design session.

Between the AKH and the Data Handler/DES resides the Process Monitor which presents the current position of the token in the Petri-Net and the passing of STEP files to and from the DES.

Also shown in Figure 01 is the Transaction Monitor (TM) which observes the transactions between the knowledge handlers, the DTFs and DES. While the TM is an aide to IIBDS development, it can also be used to observe and analyse an active design session.

In order to explain the working of the IIBDS, a series of snap-shots follow which record a user's progress with the active design session corresponding to Case 2 as outlined previously. In the snap-shots the arrows show the potential flows of information: a single arrow indicates a notification while double arrows indicate sending and listening. The "user_dialog" area of the Blackboard is reserved for user interaction transactions, while the "application_dialog" area is reserved for transactions related to the DTFs. The "journal" area receives messages from the various knowledge handlers and organises these for subsequent analysis and process control.

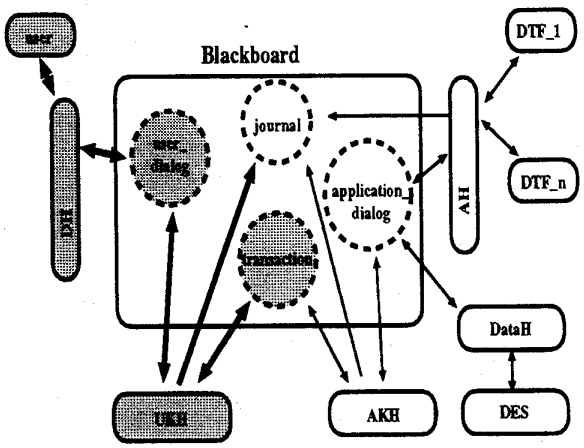

Figure 11: State of the IIBDS after user action.

Figure 11 defines the state of the system after the user has selected a DTF and the actions triggered:

1) a message passes to the Dialogue Handler (DH) indicating the requested interaction;

2) the DH passes the message to the "user_dialog" area;

3) and the User Knowledge Handler (UKH) tells the Blackboard to "start DT".

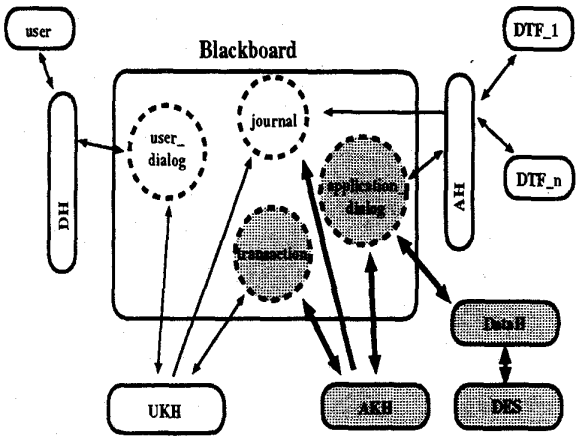

Figure 12: User knowledge handler requesting a DTF.

Figure 12 is the state of the Blackboard after the UKH has issued a message "application_dialog start DT" to the transaction area. The Application Knowledge Handler (AKH) finds the actual application and posts the message "new_application DTx" to the "application_dialog" area. The Data Handler (DtH) then queries the DES - "get_data_for DTx" - and the DES returns the appropriate data for DTx as "data_for DTx file".

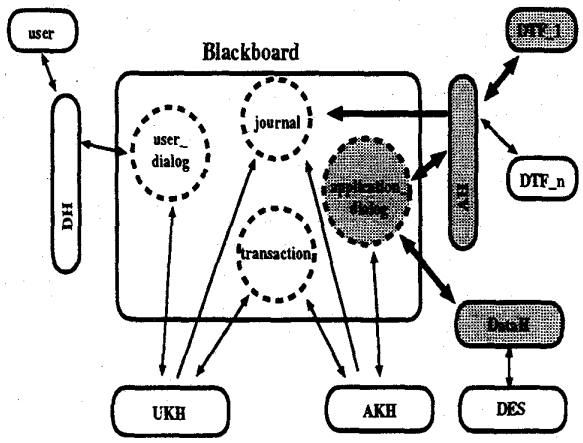

Figure 13: DES returns STEP file and DTF starts.

In Figure 13, after the DES issues "data_for DTx file":

1) The DtH posts "new_application application parameters" to the application dialog area.

2) The Application Handler (AH) starts the application and establishes a pipe to receive the performance return(s).

3) When the application is complete the AH records this and sends "closed DTx revised_data_file" to the application_dialog area. The AKH posts "closed DTx" to the transaction area, which is receives by the UKH for transmission (not shown) back to the user interface.

INTRODUCING DESIGN PURPOSE

Consider now a session which incorporates knowledge in relation to design purpose (Case 3). With reference to the Petri-Net in Figure 14, the initial portion of the design session might proceed as in the previous unconstrained session in terms of geometric specifications and attribution. For those users who enter the summer overheating design session with an existing problem there is a direct path to the overheating assessment.

Simulation environments, such as ESP-r, have traditionally provided facilities to enable direct access to their internal DTFs. In the current example some of these ESP-r DTFs are accessed, with the user involved as the arbitrator of acceptability or otherwise of the performance returns. Here, the process model involves the determination of the climate patterns which would constitute an acceptable test of summer overheating risk. While such a decision is implicit in most simulation based studies, here it has been made explicit. Next, the focus is shifted to a simulation of the current problem and then determination of what constitutes the worst spaces in terms of overheating. The search rules operate on the basis of the highest resultant temperature in an occupied space as determined from a series of inquiries of the database of simulation results. It is quite possible to have alternative rules governing this DTF.

Assuming that overheating has been detected, two presentations are made to the user. Firstly, a frequency binning of temperatures and a graph of temperatures in the worst zone. This sets the context which violated the 'rules' of the assessment. The process model then calls for the presentation of information on the likely causal factors. For example, if high internal gains were the cause of the overheating then only this information would be provided. The user can then either exit the design session or return to the architectural CAD or to the ATTRIBUTE DTF. A typical design session is shown in Figure 15.

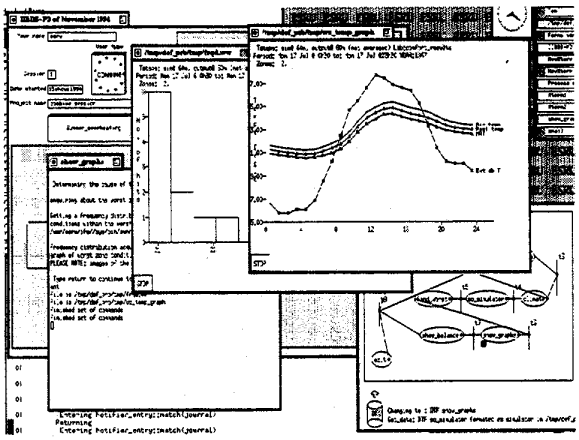

Figure 15: Summer overheating design session.

CONCLUSIONS

Technology has now reached a stage where it is possible to bring together product modelling and intelligent interfaces. The COMBINE project has undertaken developments in these areas, resulting in an Integrated Data Model (IDM) and Intelligent Integrated Building Design System (IIBDS). The former is able to service the building description requirements of several disparate design tools; the latter is able to control the operation of these same tools against rules which define the purpose of their use. Taken together, these developments are helping to evolve the prospects for a Computer-Supported Design Environment by which the analytical power of the computer can better complement the creative power of the designer.

REFERENCES

Augenbroe G (1992) 'Integrated Building Performance Evaluation in the Early Design Stages' *Building and Environment* **V27 N2** pp149-61.

Autodesk Ltd (1989) 'AutoCAD Release 10 Reference Manual' *Autodesk Ltd.* Exeter

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

Conforti F (1994) Inside MicroStation *OnWorld Press* Santa Fe.

Hand J W (1994) 'Enabling Project Management Within Simulation Programmes' *ESRU Pub T94/14* .

Javor A 'Petri Nets in Simulation' *EUROSIM - Simulation News Europe* November 1993 pp6-7.

MacCallum K (1993) Private communication.

Rode C (1993) 'Specification of the Generic Tool: BRC, the Building Regulations Compliance Checker' *Danish Building Research Institute* Horlsholm Denmark.

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

Ward G (1993) 'The RADIANCE 2.3 Imaging System' *Univ. of California* Berkeley.

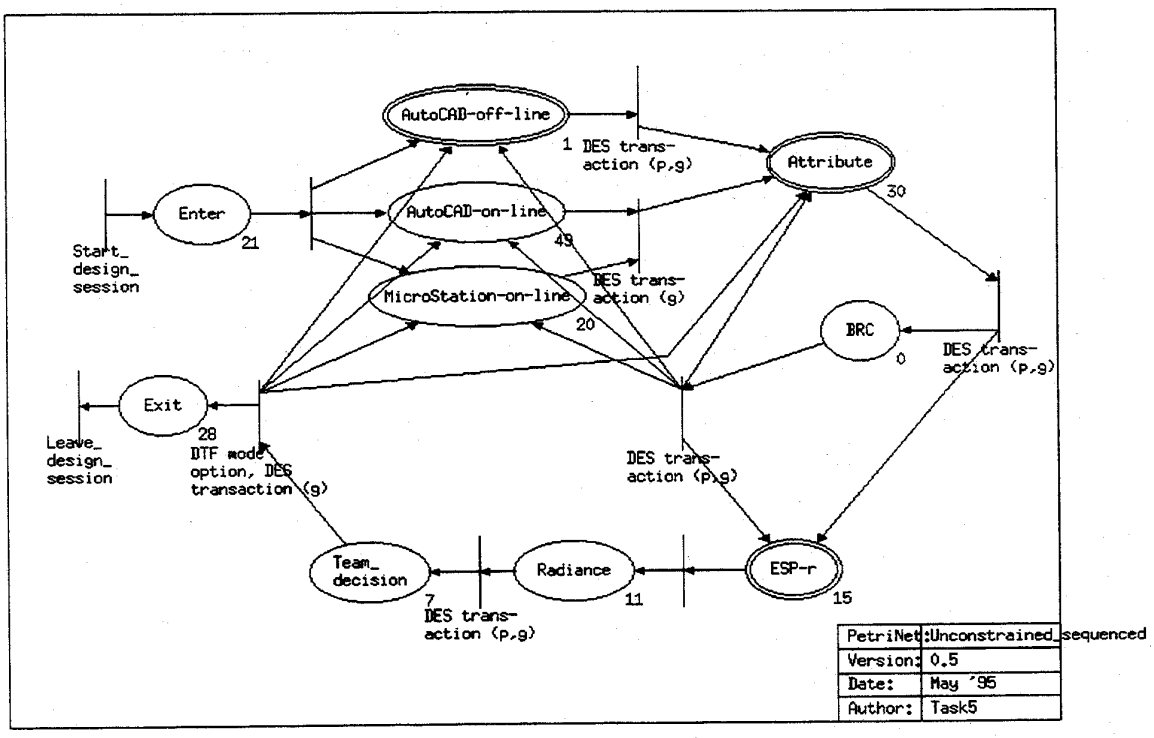

Figure 4: Case 2 process model Petri-Net.

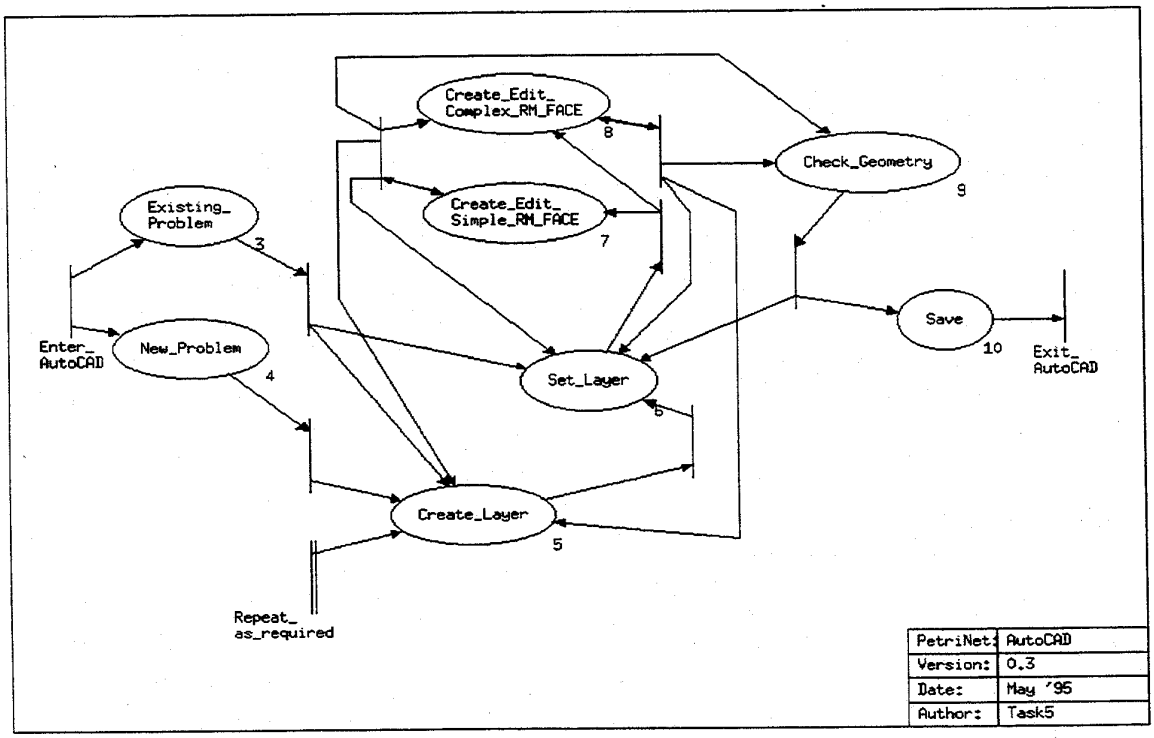

Figure 5: Arch_CAD Petri-Net.

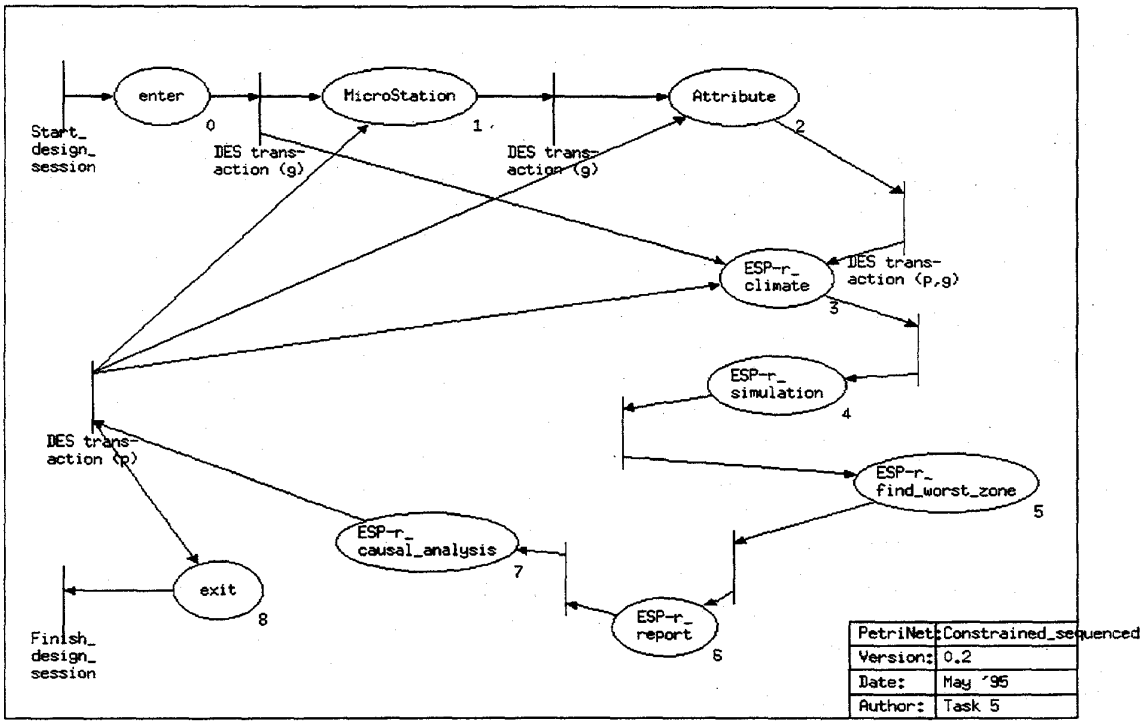

Figure 14: Summer overheating Petri-Net.