Infrared Professor - Design Phase

G.Z. Brown
J. Kline
T. Sekiguchi

Energy Studies in Buildings Laboratory
Department of Architecture
University of Oregon
Eugene, OR, 97403
USA

This paper describes diagnostic and advising modules that are being added to existing energy analysis software. The diagnostic module helps users understand what's causing their building to have certain energy use characteristics by juxtaposing performance data with climate and building use data. The advisor is a rule- based expert system which tells the user what to do to improve the energy performance of their building design.

Keywords: advisor, architectural design, buildings, energy, expert system

1 Existing energy design software

The diagnostic and advising modules described in this paper are connected to existing energy analysis software (Brown, 1989) which is intended to help architects think simultaneously about architectural design and energy design early in the design process when key energy decisions concerning building form, organization and materials are made. The inputs and outputs are completely graphic, and building components are described in architectural terms such as wall constructions rather than thermal analysis terms such as "u" value. Therefore the program fits with an architect's ways of thinking and drawing. The program emphasizes reductions of heating and cooling loads using passive strategies such as daylighting, solar heating, night ventilation of mass, cross and stack ventilation, and shading. The user may select any four days from a particular climate base representing, for example, four seasons or the same day with different occupancy patterns.

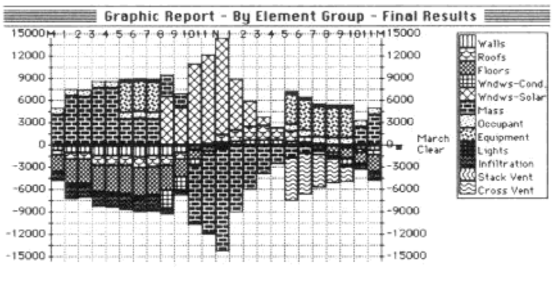


Figure 1: Graphic Report: Heat gain and heat loss for 24 hours (BTU's)

The program displays its thermal analysis in several ways. Figure 1 shows the output for one day graphed by building element. The horizontal axis is time of day with noon in the middle. The vertical axis is in Btu's with heat gain shown above the origin and heat loss below. The heat gain minus the heat loss represents the buildings heating or cooling load for any hour. In order to improve the performance of the building the user would modify the architectural design to balance heat gain and loss for each hour and to reduce the height of the bars, especially those, such as electric lighting that represent manufactured energy.

2 The problem

Feedback from users concerning the existing software indicated that even though the output was graphic and much easier to grasp than the equivalent information in tabular form, non energy experts had difficulty understanding what caused gains and losses to occur and what to do to the building design to correct the problem. Understanding what to do was particularly difficult when interactions between the user's modifications caused synergistic or counter intuitive effects. To address this problem we decided to add two modules to the existing software - a diagnostician and an advisor. The diagnostic part of the system helps the user understand why something is happening. For example, by looking at graphic data showing that roof heat gain increased from 10 am to 11 am juxtaposed with graphic data on sun position, the user can conclude that the increase in roof heat gain was due to the fact that the sun`srays were more perpendicular to the roof plane at 11 am than they were at 10 am. The advisor, on the other hand, tells the user what might be done to solve a problem. For example, for the problem of heat gain through the roof at 11 am, the advisor might suggest, "use a lighter roof material, increase the r value or reduce the roof area to reduce heat gain."

3 How the diagnostician works

Originally we had intended to use an expert system for the diagnostician as well as the advisor. Based on the effort required to develop the expert system for the advisor module and the expected benefits to the user, we have elected to take an approach that requires the user to analyze data and reach their own diagnosis of what is wrong with their building. We were more comfortable using this approach with the diagnostician than the advisor because diagnostics require primarily analysis capabilities on the part of the user (something students are capable of because much of their education is about analysis techniques) as opposed to the advisor which requires more synthetic skills. Synthetic skills are required because there is not a one-to-one relationship between the problem and the solution. For example, if the problem is building over-heating as a result of solar gain through the windows, the one-to-one solution is to reduce the solar gain by shading. However, there are several other solutions like adding mass to the building to store the heat for flushing at night, using ventilation to cool the building or raising the thermostat setting to reduce the temperature difference between inside and outside (the temperature difference is the driving force for heat movement through the building envelope). Furthermore some solutions, like adding mass, may solve other problems like winter heating, as well. In general, we find student to be much less capable of synthetic thinking. The major causes or drivers of building energy use are climate, building design, and how the building is used. The major drivers in climate are temperature, radiation, and wind. In building design they are the r value of the envelope, the transparency of the envelope to solar radiation and wind and the building's thermal massiveness and in building use the number of people present over the day and when the lights and other equipment are turned on and their efficiency.

Because there are a number of variables and many of them are dynamic (climate, some building elements like windows, and occupancy, lighting and equipment) it is difficult for users to keep track of all the variables and especially to keep track of their changes over time.

The diagnostician has only been partially designed and not coded. Therefore descriptions of the diagnostic module should be regarded as preliminary. We have developed two approaches to helping the user diagnose the buildings problems. The first augments the graphic report shown in figure 1 by adding climate and building use information to the graphic report. This information is organized in an hourly format as are the heat gains and losses, making it possible for the user to look down a column for any given hour and determine the magnitude of gain and loss by element and to simultaneously see climate and building use variables.

The section of the diagnostics labeled occupancy displays variables associated with occupant patterns. On top of the section is shown the temperature difference, or delta t, for each hour. As seen in figure 2, the hour 7 am to 8 am has a temperature difference of -17 degrees, while from 5 pm to 6 pm there is a positive temperature difference of 4 degrees. Hours when there is no temperature difference are those in which the outside temperature is within the minimum and maximum thermostat settings.

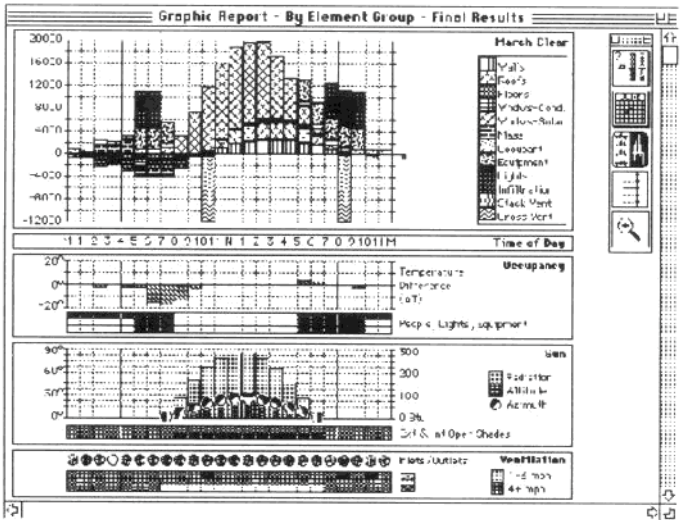


Figure 2, diagnostic report

Below the temperature difference graph are shown the hours when people, lights, and equipment are present or turned on. For instance, from the illustration we see that people are in the building from 5 pm until 8 am.

The next portion of the diagnostics, labeled sun, shows solar radiation and sun angles on the upper graph and schedule information about shading devices below. Radiation is indicated by a bar graph and is read using the vertical scale to the right. Overlaid on this is a circle indicating the sun's altitude and azimuth. The altitude is given by the vertical position of the circle using the left hand graph scale while the light vs. Shaded interior of the circle shows what building surface azimuths would receive sunlight.

Exterior and interior operable shade schedules are indicated below this. Hours filled with a gray pattern show that a shade type is scheduled on to be used if needed while the black filled hours are those when shades are actually in use. In figure 2 it is seen that while both exterior and interior shades are scheduled on only interior shades are used from 8amto7pm.

The final piece of the diagnostics is ventilation. The circles on the upper portion show that for the wind direction for the hour which window orientations will be considered inlets and which outlets for cross ventilation. The software understands windows oriented within 45 degrees of the wind direction to be inlets and all other to be outlets, therefore the circle is divided into one-quarter for inlets and the remaining three-quarters for outlets. Circles without a subdivision indicate hours when there is no wind. elative wind speed is also shown by the shading of the outlet portion of the circle. Since beyond a certain wind speed there is usually excessive cross vent capacity we show only 2 ranges of speeds. For example, the hour of 1 am to 2 am has a relatively high wind speed of at least 4 mph and cross vent inlets would be windows facing southwest through northwest while all other windows would be considered outlets.

Below the wind direction and wind speed circles are the schedule information associated with windows and ventilation. These are cross and stack vent and night vent of mass. Similar to the schedule information for shades, the illustration shows cross and stack vent turned on for use if needed at all hours while night vent of mass is scheduled on if needed only from 7 pm to 7 am. Only cross vent is actually being used and then only for 3 hours of the day.

While the first part of the diagnostician helps the user relate variables to performance, it is one step removed from the building in that it represents the elements of the building as patterned sections of a bar graph. In addition, while the graph represents 24 hours, its static quality doesn't fully capture the dynamic quality of the building heat loss and gain. In order to better communicate the dynamic quality of heat flow and more closely tie performance to building elements. We will animate the building drawing itself.

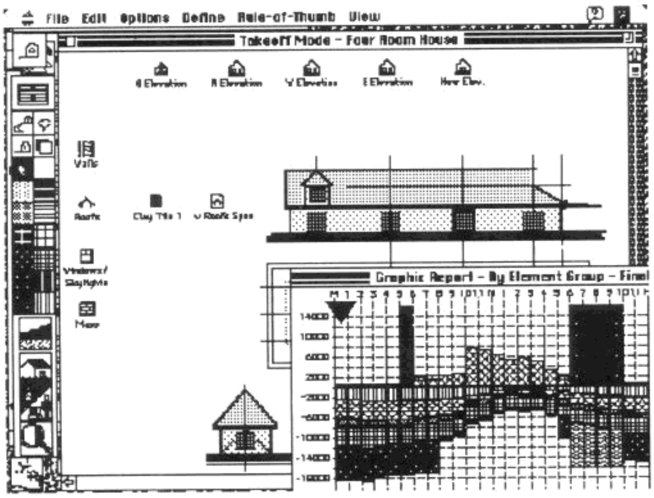

File
Edit
Options
Define
Rule-of-Thumb
View
Takeoff Mode – Four Room House
Graphic Report – By Element Group – Final

Figure 3: building elements

Figure 3 shows the building elements highlighted with dot patterns (color in the software) to represent their rate of heat gain or loss for a single hour. We expect this visualization technique to be very powerful because it allows visually oriented architects to see the invisible--the flow of heat in their buildings. Our experience working with infrared scanning cameras in existing buildings reinforces the importance of visualization. When contractors are shown infrared images of their buildings, they are amazed to see thermal leaks at studs or light switches. They frequently comment that seeing thermal leaks convinces them how important these elements are to energy performance.

As seen in figure 4, we use a black pattern with diagonal white stripe (red in software) to signify heat gain and black (blue in software) to signify heat loss, with the degree of saturation of the color (in the software) corresponding to the rate of gain or loss. White signifies thermal neutrality.

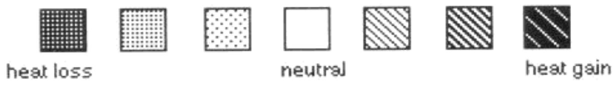


Figure 4: heat gain/loss spectrum

The user will be able to control the speed of the animation and the hours or times of year shown. Figure 5 shows the building at the same hour for four different seasons. Notice that the windows not only gain the most heat in summer but also lose the most in winter. This clue alerts the designer to investigate the windows as a thermally weak area.

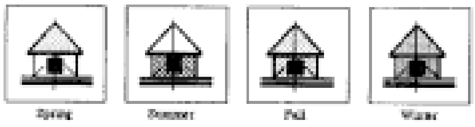


Figure 5: same hour, different days

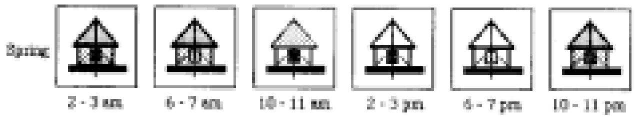


Figure 6: same day, different hours

Figure 6 shows the building over the course of a typical spring day, cool at night and warm during the day. Figures 5 and 6 show a series of elevations all visible at the same time. In actuality, the elevations would appear one at a time in one location with one image dissolving into the next to stress the changing conditions that result from the passage of time.

4 How the advisor works

The advisor, as currently operating, has four major components: problem identification, resource check, conflict resolution, and advice generator.

The problem identification component reviews each hour of all days selected and classifies them as having one of four load types: type h (heating required), all hours exhibiting negative net flow (gain - loss <0); type cc (cooling required, cool outside), all hours exhibiting a positive net flow (gain - loss > 0) and the outside temperature is below the thermostat maximum set point; type ch (cooling required, hot outside), all hours exhibiting positive net flow and the outside temperature is above the maximum thermostat set point; and type (.),when the net flow is equal to zero (gain - loss = 0).

Solutions are then associated with each hour type and the cause of the net gain or loss. For example, if the hour is type h (gain - loss <0) and the causes of the heat loss are windows, walls, and roofs, the solution might be to increase the r value of the walls, roof and windows, or add night insulation to the windows. In addition to generating specific solutions that addresses the "cause" of the problem (heat loss through windows, walls and roof), the advisor also generates general design strategies such as, increasing the amount of south facing glazing and thermal mass or reducing the thermostat set point. Using this technique, each hour is associated with several pieces of advice.

Before these pieces of advice are presented to the user, however, the advisor regroups them according to the building element problem they pertain to and orders these problems from greater to lesser magnitude. For example, if heat loss through windows were a greater problem than heat loss through the walls, then the advice about windows would come before advice about walls. All advice about that element causing the problem is listed. For example, even though heat loss through the window is the greatest problem, the advisor would also list suggestions about shading (for overheated hours) under the windows category. The advice is grouped in this manner to make it easy for the user to understand and make all changes necessary to an element such as a window at a single point in time.

Once this list of advice has been generated and ordered it is reviewed from several perspectives. The first is resource availability. As an example, the advisor checks to see if solar energy is available before giving the advice to "add south glazing and thermal mass." This kind of checking applies to wind speed and direction, illumination, and temperature, as well as to solar radiation. If the resource is not available, the advice is removed from the list. The specifications the user has input are also checked. For example, before giving advice to lower the thermostat's heating set point, the advisor checks to see what the user has already specified. If the thermostat set point is already at the allowable lower limit (based on recommended comfort standards), the advice is removed from the list. Similar checks are made for lighting, occupancy, equipment, envelope, etc. The advisor also checks to see if building elements have already been specified before giving advice recommending that a certain element be added.

Once the list of advice has been pruned by discarding advice when resources aren't available, when specifications are at the limits of standards, and when elements have already been specified, the advisor searches the list for conflicting advice. For instance, the list might include the advice to 'make the roof dark to increase solar gain to reduce heat loss" (occurs in type h under heated hours) and to "make the roof light to decrease solar gain and reduce heat gain" (occurs in type cc and ch over heated hours). Or the advice might be to "decrease the window area to reduce heat loss" and to 'increase the window area to increase daylighting. ' These conflicts are resolved by calculating which advice will offer the greatest improvement in performance. That advice is retained on the list, and the advice with the least impact is removed from the list.

4.7 *Further development of the advisor*

Although not implemented at the time this paper was written, we will also develop code that reorders the advice list based on the potential of the solution rather than size of the problem. Currently, for example, if heat loss through the roof were greater than heat loss through the walls, we would give advice about the roof first and walls second. However if the roof were already well insulated and the walls were not, we would get the greatest energy return on investment by insulating the walls before the roof. Therefore walls would move to before roofs on the advice list.

5 Software design

Our approach to implementing the advisor has been incremental. That is, first we implemented the problem identification, then the resource check, then conflict checks, and so on as opposed to trying to design the whole advisor and implementing it all at once. This approach has proven to be extremely beneficial because we can always give the user advice. Granted the advice is sometimes not possible to implement because of resource limits or other factors; however, but it is not wrong, just not as smart as one might like. Secondly, in working incrementally on designing the advisor, we can avoid the difficult cognitive task of working out in advance the interaction of several variables over time.

Most of the existing energy analysis program is written in the c programming language in the mpw programming environment. The diagnostic and advisor sections of the program are written in C++.

Figure 7 is an overview of the advisor section of the program which consists of eight objects. There are three objects in the top layer: the evaluation day object, problem list object, and solution list object. These three objects in turn possess other objects.

An evaluation day object is created for each The four evaluation days in the program. The evaluation day object takes information from the building database created during the calculations portion of the program. It extracts the essential information that is needed for writing to the advice window. This is the raw data that the advisor will need. In addition to extracting information, it keeps track of a new piece of information called type. Type is the concept that determines a major portion of the advisor code design. There are four types and they refer to what the net flow (loss or gain) is for an hour and if a building element contributes to the net flow problem. The types are h for heating required, ch for cooling required with to > ti max, cc for cooling required with to < ti max, and (.) for zero net flow. If an element contributes to the net flow problem for a particular hour then it is one of the first three types, otherwise it is type (.).

An element is significant only if the total heat flow for that hour is positive and the element has a heat gain value or if the total heat flow for that hour is negative and the element has a heat loss value (see figure 8). For example, if the hour is a type h hour (heating required) and the element is electric lights, the type is (') because electric lights do not contribute to the net flow problem (loss) for the hour. Figure 8 represents the format of the resulting element list which is part of the evaluation day object. From this information a problem list object is created.

The problem objects consist of elements of the building such as window conduction, heat gain from occupants, etc. That have type information of type (h), (ch), or (cc). In creating the problem list, the format of the objects is changed. Now the objects are sorted by the cause of the problem and type. Figure 9 represents the format of a problem list.

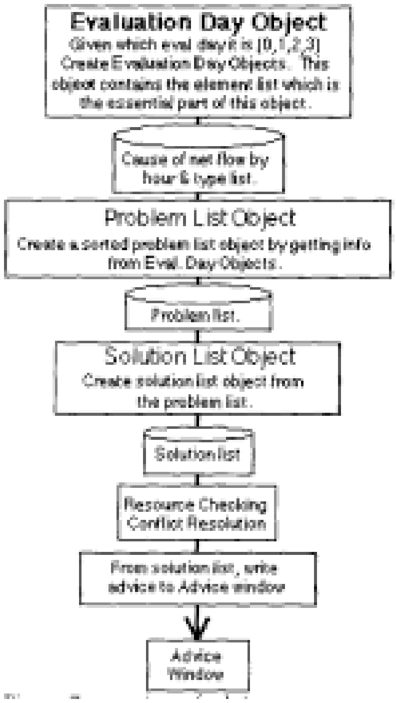
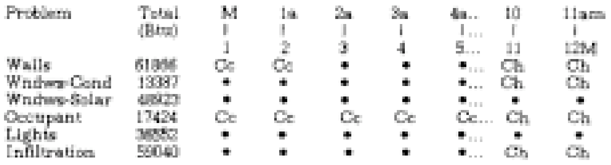

Evaluation Day Object
Given which eval day it is [0,1,2,3] Create Evaluation Day Objects. This object contains the element list which is the essential part of this object.
Cause of net flow by hour & type list.
Problem List Object
Create a sorted problem list object by getting info from Eval. Day Objects.
Problem list.
Solution List Object
Create solution list object from the problem list.
Solution list
Resource Checking Conflict Resolution
From solution list, write advice to Advice window
Advice Window

Figure 7: overview of advisor

After the problems are identified, each problem can then be associated with one or more solutions or pieces of advice. This creates a solution list. This list may be rather long, but it is eventually thinned as a result of checking for duplication, checking the availability of resources, and resolving conflicts. The culled list is used to write the advice window.



Problem	Total (Btu)	M | 1	1a | 2	2a | 3	3a | 4	4a.. |... 5...	10 | 11	11am | 12M
Walls	61866	Cc	Cc	•	•	•...	Ch	Ch
Wndws-Cond	13387	•	•	•	•	•...	Ch	Ch
Wndws-Solar	48823	•	•	•	•	•...	•	•
Occupant	17424	Cc	Cc	Cc	Cc	Cc...	Ch	Ch
Lights	38552	•	•	•	•	•...	•	•
Infiltration	59040	•	•	•	•	•...	Ch	Ch

Figure 8: element list: june clear - day 1

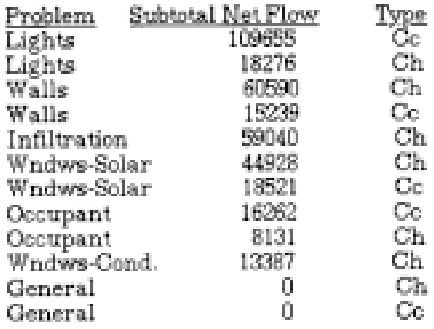
5.1 *Description of the objects in the advisor code*

The following section describes the evaluation day object, the problem list object, and the solution list object. Within the evaluation day object is an object called the hour information object and an object called the element list object, within which is the element object.

An evaluation day object:

• knows: which day it is (given 0,1,2,3), and its hour type information objects. The evaluation day object contains the hour information object, and the element list object (described below).

• can: fill in its own information, destroy itself, tell which day it is, tell what type it is for any hour, get the beginning of the element list, set the type information for each hour, and create the element list objects.



Problem	Subtotal Net Flow	Type
Lights	109655	Cc
Lights	18276	Ch
Walls	60590	Ch
Walls	15239	Cc
Infiltration	59040	Ch
Wndws-Solar	44928	Ch
Wndws-Solar	18521	Cc
Occupant	16262	Cc
Occupant	8131	Ch
Wndws-Cond.	13387	Ch
General	0	Ch
General	0	Cc

Figure 9: problem list

The hour information object knows: what type of hour it is for each of the 24 hours, namely h, ch, cc or (.) can: initialize itself, tell what type it is for any given hour, and set what type it is for any given hour.

An element list object:

• knows: its element objects (described below).

• can: create element objects, destroy each element object. Given which element number, it can retrieve an element object, and given an element id, it marks the element record as being scratched or removed.

An element object:

• knows: its significant net flow, which element it is, if it is scratched off the list or not, and for each type, if it is significant for each hour.

• can: set its initial values (net flow (nfl=0, which element it is from what is given, scratch the flag off, schedule by hour type (figure 10), destroy itself, tell what its total net flow value is, tell what its cause is, tell what its scratch flag is, given a type, tell which hours are significant for all 24 hours, and set its scratch flag to true.

A problem list object:

• knows: the number of items in the list, the head of the linked list of problems item, and the last item in the linked list.

• can: initialize what it knows, delete all items in its linked list, add items to the end of the linked list, and retrieve the head of the linked list of problems.

A problem item object:

• knows: its cause, which type it is, its schedule of significant hours for a particular evaluation day; the values for each contain the 24 hour significance flag, its subtotal net flow value by problem and type, and the next item in the list.

• can: initialize itself, retrieve the next item, given a problem item, set it to the next item in the list, tell what its cause is, tell what its type is, tell what its significant hours are, tell what its total net flow is, set its total net flow value.

A solution list object:

• knows: the number of items in the list, the head of the linked list of solution items (described below), and the last item in the linked list.

• can: initialize itself, delete all items in its linked list, given A problem item object, add an item to the end of its linked list, given a solution item object, add an item to the end of its linked list, retrieve the head of the linked list of problems, remove an item from the list, and insert an already created solution item into its linked list.

A solution item objects:

• knows: its cause, type, solution number; category, how its advice is given (1=by hour; 2=by day; 3=n/a), schedule of significant hour for a particular evaluation day; the values for each contains the 24 hour significance flag, information on why the solution is given, the next item in the list, the previous item in the list, its scratch flag telling if this solution is still in the list.

• can: initialize itself, retrieve the previous item, set its previous item, tell what its cause is, tell what its type is, tell what its solution number is, set its solution number, tell what its category is, get information on how advice is given, set information on how advice is given, tell what its significant hours are, set what its significant hours are, tell why the solution is being given, set why the solution is being given, get its scratch flag, set its scratch flag (0 means it is still on the list; non-zero gives a reason for scratching code).

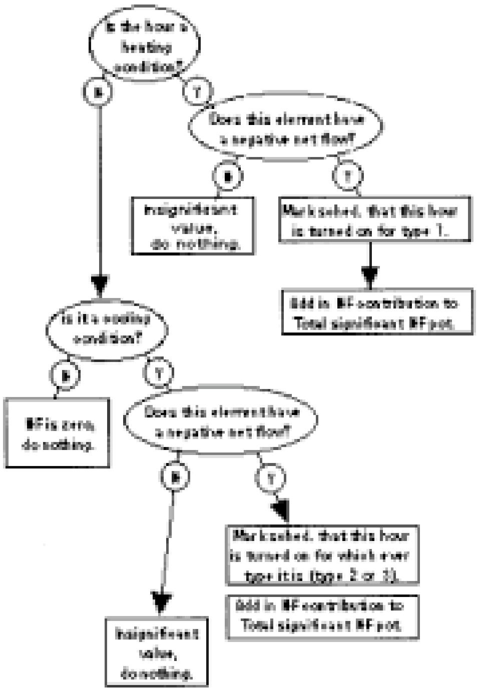
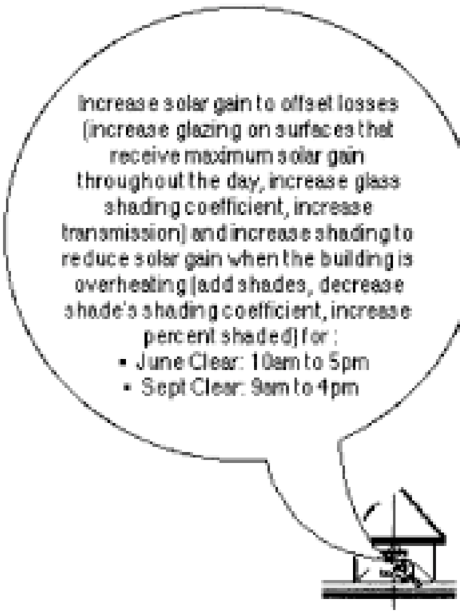

Is the hour a heating condition?
N
Y
Does this element have a negative net flow?
N
Y
Insignificant value, do nothing.
Mark solved, that this hour is turned on for type 1.
Add in NF contribution to Total significant NF pot.
Is it a cooling condition?
N
Y
NF is zero, do nothing.
Does this element have a negative net flow?
N
Y
Mark solved, that this hour is turned on for which ever type it is (type 2 or 3).
Add in NF contribution to Total significant NF pot.
Insignificant value, do nothing.

Figure 10: schedule by hour type flow diagram

6 Advisor user interface

For debugging purposes the output of the advisor is currently textual. We will retain the textual presentation of advice for printing in a report, although the user probably will not be able to see it on screen. The user will be able to get a report which describes the building performance as well as all inputs and a list of advice about what to do to improve the building's performance.

The primary means of accessing the advisor will probably be graphic and auditory. For example in figure 11 we show the user placing the cursor on the window and receiving spoken advice on what to do to that building element to improve performance. Because the user will be able to get individual advice about each element of the building it is important that there be some way for the user top keep track of what he or she wants to change in the design. To accomplish this we will implement a "to-do" list. This is a text window to which the user may transfer information about building elements by typing or clicking on the building elements themselves. In the context of student exercises the 'to-do" list can be frozen at any point in time so that it can be turned in with exercises that ask questions like "what is causing this performance?" Or, "how should the design be changed to improve performance?"



Increase solar gain to offset losses [increase glazing on surfaces that receive maximum solar gain throughout the day, increase glass shading coefficient, increase transmission] and increase shading to reduce solar gain when the building is overheating [add shades, decrease shade's shading coefficient, increase percent shaded] for :
- June Clear: 10am to 5pm
- Sept Clear: 9am to 4pm

Figure 11: advisor interface

7 Evaluation

The advisor section of the program has been tested in a seminar at the university of oregon. Students were introduced to the existing energy analysis software and completed four exercises. The advisor was then added and the students completed four more exercises. Evidence from student course evaluations and faculty evaluations of student exercises suggests there is a positive correlation between using the advisor and better understanding of energy phenomena and better exercise grades.

The diagnostician and the advisor combined with the existing energy analysis program are scheduled to be tested in a large lecture class in the fall, 1995 . One half the class will use the software to do their exercises and one half the class will do the same exercises by hand. The exercises will be similar to those used in the past so they will be biased towards hand calculation methods and traditional techniques. The midterm and final examinations will be evaluated to determine if there is a difference in comprehension between the two groups of students. After these tests are completed the infrared professor will be redesigned in response to student suggestions.

8 References

(1) Brown, G.Z., Sekiguchi, T., 'Energy Scheming 1.0,' Proceedings of the International Solar Energy Society World Congress, Kobe, Japan, 1989.

9 Acknowledgments

This project has been supported by the fund for improvement of post secondary education, the US Department of Energy, and the University of Oregon. The authors would also like to acknowledge the importance of their colleagues at the energy studies in buildings laboratory, who have reviewed the software as it was being designed and offered valuable suggestions. We would also like to thank the users of Energy Scheming who have given us valuable feedback, especially Mark Dekay, Lance Lavine And Joel Loveland.