LBL-31212
UC-350

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

ENERGY & ENVIRONMENT DIVISION

Presented at the First International Symposium on
Building Systems Automation-Integration, Madison, WI,
June 2–7, 1991, and to be published in the Proceedings

A Computer-Based Building Design Support Environment

K. Papamichael and S.E. Selkowitz

June 1991

ENERGY & ENVIRONMENT DIVISION

REFERENCE COPY
Does Not
Circulate
Bldg. 50 Library.
LBL-31212
Copy 1

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

Neither the United States Department of Energy (DOE) nor any agency thereof, nor The Regents of the University of California (The Regents), nor the California Institute for Energy Efficiency (CIEE), nor any of CIEE's sponsors or supporters (including California electric and gas utilities), nor any of these organizations' employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by DOE or any agency thereof, or The Regents, or CIEE, or any of CIEE's sponsors or supporters. The views and opinions of authors expressed herein do not necessarily state or reflect those of DOE or of any agency thereof, of The Regents, of CIEE, or any of CIEE's sponsors or supporters, and the names of any such organizations or their employees shall not be used for advertising or product endorsement purposes.

Lawrence Berkeley Laboratory is an equal opportunity employer.

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBL-31212
DA-301
UC-350

Presented at the First International Symposium on Building Systems Automation-Integration, June 2-7, 1991, at the University of Wisconsin, Madison, WI, and to be published in the Proceedings.

A Computer-Based Building Design Support Environment

Konstantinos Papamichael and Stephen E. Selkowitz
Energy and Environment Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

June 1991

The research reported here was funded, in part, by the California Institute for Energy Efficiency (CIEE), a research unit of the University of California. Publication of research results does not imply CIEE endorsement of or agreement with these findings, nor that of any CIEE sponsor. This work was also supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Technologies, Building Systems and Materials Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

recycled paper

A Computer-Based Building Design Support Environment

- Design process and expertise
- Integration of design tools
- Interactive multimedia environment

Konstantinos M. Papamichael and Stephen E. Selkowitz

Windows and Lighting Program
Center for Building Science
Applied Science Division
Lawrence Berkeley Laboratory
University of California
Berkeley, CA

1.0 Introduction.

Continuously decreasing cost has brought computers into most architectural and engineering offices, most commonly for activities such as drafting, accounting and word processing. Computers are used less often to predict the performance of design solutions. However, most performance simulation software packages are simplified versions of main-frame analytical tools, originally developed for research. Such software packages are focusing on specific design issues according to the research needs that originated them. In addition, the data input requirements are complicated and incompatible with each other, and the output data are usually specialized and difficult to interpret. It is yet to be seen how the increasing memory and processing speed of computers, the two main advantages that computers have over the human brain, can be used to assist designers throughout the design process, allowing them to organize design projects electronically.

The number of building design criteria exceeds by far what the average human can handle simultaneously. The same is true with the number of the technological options for building systems that emerge as a response to human needs, and whose performance cannot be known a-priori since it depends on the context of their application. The design process is becoming increasingly demanding and complicated. The development of appropriate design tools to assist designers in managing this data-explosion situation is essential for the successful outcome of design projects.

A major effort is now under way to utilize the power of computers to assist designers directly in the design process, possibly at the early stages of design, where most of the important design decisions are made. Since performance assessment through simulation is not possible during the early design stages, heuristics are implemented, in the form of the so-called Expert Systems, which emerged from the field of Artificial Intelligence [Rich 1983].

Example applications within the building design domain include diagnosis of problems with various types of building equipment [Haberl et al. 1989; Ruberg and Cornick 1988] and selection of various building components and systems [Degelman and Kim 1988; Tuluca et al. 1989]. These initial attempts have led to the identification of problems in knowledge acquisition and representation, as well as integration with existing software [Hall and Deringer 1989].

The reason for such problems is the lack of an appropriate, comprehensive design theory, which would serve as the foundation for the development of building-related data and knowledge representation schemes. Unless such a theory is developed, the various attempts to utilize computers to assist designers will follow different modeling methods resulting in a variety of incomplete and incompatible with each other knowledge-based systems. Moreover, such systems may prove inappropriate, forcing designers to premature judgements and/or misleading them due to hidden subjective preferences.

We describe the design and initial implementation of a computer-based Building Design Support Environment (BDSE) whose structure and operation are derived from a detailed theoretical analysis of the design process, into the iterative and interactive activities that contribute towards the formulation of design criteria, the generation of potential solutions, and their evaluation. ***The identified design activities are characterized with respect to the nature of knowledge requirements and the degree to which they can be specified and delegated to computers. The results are considered as criteria to determine the level of automation and the interaction between designers and computers, to model the delegateable and non-delegateable activities, respectively.*** We believe this approach, when fully implemented, has a good chance of providing building designers with a powerful environment to enhance building design.

2.0 Design Theory.

Design is an activity aimed at producing a plan, which is expected to lead to a situation with specific intended properties, but without undesired side- or after-effects [Rittel 1972]. Design, then, presupposes a discrepancy between a situation *as is* and a situation *as it ought to be*, and requires at least three distinct activities:

1) the formulation of the specifications of the ought-to-be situation,
2) the generation of plans to lead from the as-is situation to the ought-to-be one, and
3) the checking for undesired side- and after-effects.

However, the specifications of the ought-to-be situation are formulated as a set of *performance characteristics*, while a plan is developed as a set of *descriptive characteristics* of a *will-be* situation, whose performance characteristics have to be determined and checked against the performance characteristics of the ought-to-be situation. Moreover, consideration of identified undesired side- or

after-effects is the equivalent of *updating* the performance specifications of the ought-to-be situation. When the will-be performance characteristics do not match the ought-to-be ones, then the will-be descriptive characteristics are *modified*, or the ought-to-be performance characteristics are *degraded*, or the designer gives up. Even if a solution without side- or after-effects has been found, the ought-to-be situation may be *improved* in search of a *better* solution.

Based on the above considerations, there are seven main design activities, two initial ones and five that are performed iteratively according to four main decisions (Figure 1). These seven activities can be grouped under three tasks, based on their contribution towards the specification of the ought-to-be performance, the will-be description, and the will-be performance:

A. Activities towards the formulation of the ought-to-be performance:

1. Specify the initial ought-to-be performance,
2. Update the ought-to-be performance,
3. Improve the ought-to-be performance, and
4. Degrade the ought-to-be performance.

B. Activities towards the development of the will-be description:

5. Specify the initial will-be description, and
6. Modify the will-be description.

C. Activities towards the determination of the will-be performance:

7. Determine the will-be performance.

The four decisions that control the sequence of these activities are:

1) Decide if the will-be performance matches the ought-to-be one,
2) Decide if there are undesired side- or after-effects,
3) Decide if there is time / hope for solution, and
4) Decide if there is time / hope for improvement.

The seven activities are further analyzed as they contribute to the development of the three sets of specifications.

2.1 Formulation of the ought-to-be performance.

The performance of a design solution is described through the use of *performance variables*, that is variables based on the values of which designers judge the appropriateness of design solutions. The ought-to-be performance is specified as a set of *performance criteria*, that is conditions on the values of performance variables. Performance variables are determined initially based on consideration of the design program objectives, and then during the design process through consideration of undesired side- and after-effects of potential design solutions.

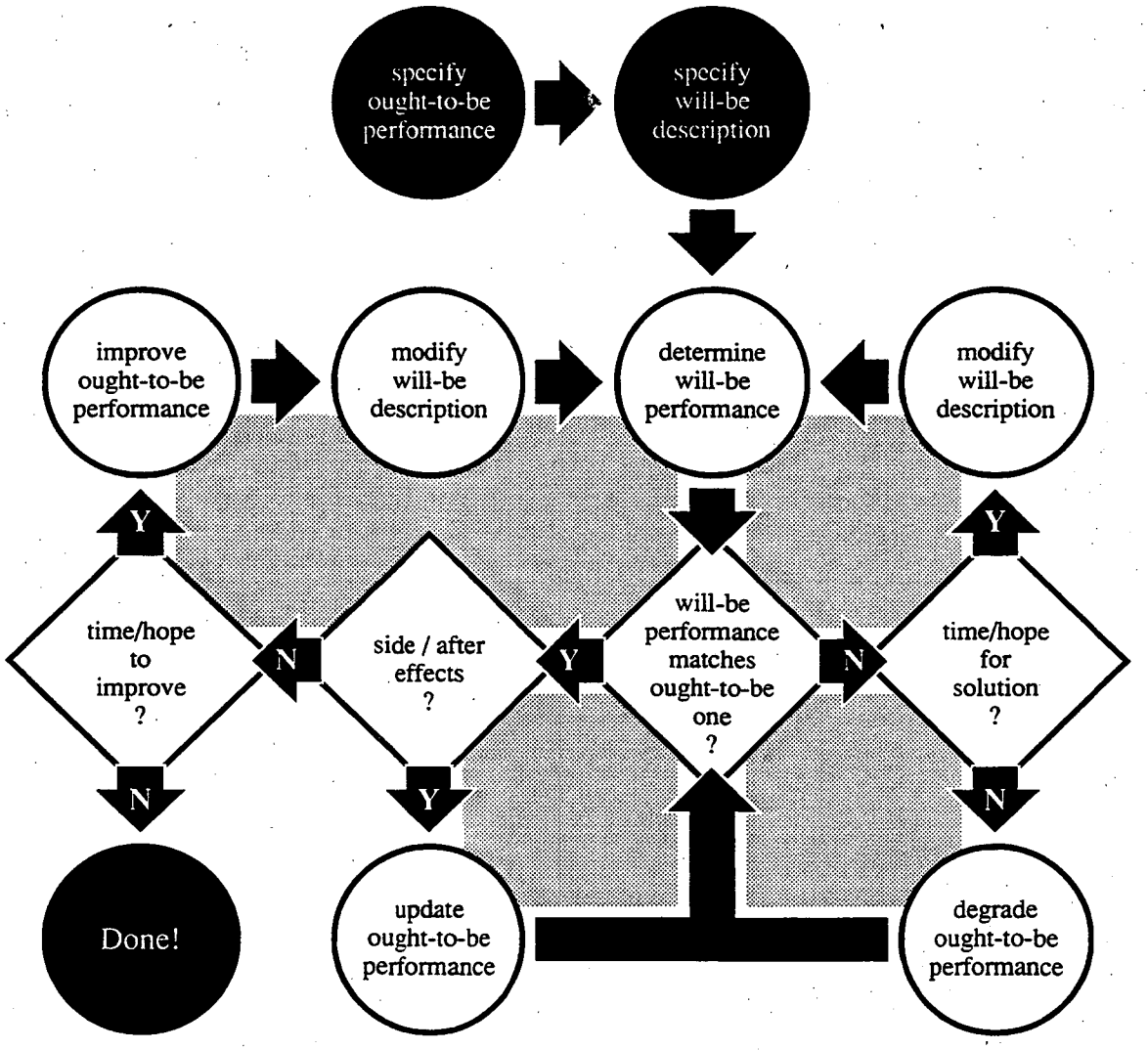


Figure 1. Design process flow chart.

Performance criteria may or may not be specified for the initial performance variables. ***Design is a compromise between what is desirable and what is possible. The design process is the equivalent of exploring what is possible under the specific design context and adjusting performance criteria accordingly, since what is desirable is not always possible*** (for example, zero cost). The performance criteria are formulated throughout the design process. The ought-to-be performance is either *updated, improved,* or *degraded.* It is updated through the identification of side- and after-effects to include new performance variables along with the associated performance criterion. It is improved when a solution has been found but there is still enough time / hope for a better solution It is degraded when the will-be performance does not comply with the ought-to-be one and there is no time / hope for

a solution. ***This updating, improving, and degrading of the ought-to-be performance continues throughout the design process. The final version of the ought-to-be performance is that of the final will-be one.***

Performance variables and associated criteria are derived through deliberation on the overall performance of design solutions, that is the identification of a set of performance variables whose consideration is equivalent to the consideration of the overall performance. Each resulting performance variable may in turn be considered through an equivalent set of performance variables, ultimately resulting in a hierarchical, tree-like structure.

Performance variables can operate either on *nominal* or *continuous* scales. They may take any values, such as numeric quantities, images, sounds, video segments, smells, as well as sets of such values. Performance criteria that are formulated as conditions on performance variables that operate on continuous scales (for example, cost) are called herein *quantitative*, and can be specified with respect to both, performance *acceptability* as well as performance *improvement* and *degradation*. Performance criteria that are formulated as conditions on performance variables that operate on nominal scales (for example, esthetics) are called herein *qualitative*, and can be specified only with respect to performance acceptability. Quantitative criteria are specified as *acceptable value ranges*, while qualitative criteria may be specified as *acceptable value sets*, in which case they may be treated as *pseudo-quantitative criteria*. ***However, qualitative criteria are usually not specified at all, in which case the delegation of judgement is impossible.***

The judgement on a qualitative criterion is actually a deliberated judgement, in which the branching criteria are not independent, that is the appropriateness of the values of the branching performance variables depends on the values of some, or all, branching performance variables. Considering the performance variable "esthetics," an image can be seen as a luminance distribution, that is values of a set of luminance variables. Since the values of these luminance variables are considered for the evaluation of the performance with respect to esthetics, they are actually branching performance variables operating on continuous scales, thus appropriate for the development of quantitative criteria. However, the criterion for the appropriateness of the value of a luminance variable depends on the values of the rest of the luminance variables. As a result, if the branching criteria of a deliberated judgement are not independent, then the deliberated judgement is not delegateable since it is the equivalent of a qualitative criterion, that is a non-specifiable criterion. However, the branching criteria of a deliberated judgement can never be independent. They are always linked through their *relative importance*.

A deliberated design criterion represents the ought-to situation, which, may or may not be possible. If it is not possible, that is, if a solution that satisfies all branching criteria cannot be found, then the branching criteria have to be modified. If it is possible, there may be more than one solution that satisfies all branching criteria. Selecting one is the equivalent of further specifying the ought-to-be situation,

that is, modifying it. Both cases, then, result in a need for modifications of the branching criteria. Here is where the relative importance of performance criteria is considered to select which ones to modify and how. In fact, since the relative importance of performance criteria is required for their modifications, its determination is part of the formulation of the performance criteria, rather than an independent activity. ***As a result, all deliberated criteria (multi-criterion judgements) are qualitative, that is non-specifiable. The ought-to-be performance is, then, only partially specifiable, which means that judgement is only partially delegateable.***

2.2 Development of the will-be description.

The values of performance variables depend on the values of *control* variables, that is, variables used to describe the will-be situation. The names of control variables depend on design domain. In architectural design they are variables that describe the building and its context. In automotive design they are variables that describe the automobile and its context. These variables may be either *design*, or *context* variables. Design variables are those whose values are directly controlled by the designer, such as the dimensions of a room, or the color of a wall. Context variables are those that the designer *decides* s/he does not want to control, such as the height of people, or the cost of utilities [Rittel 1973]. ***Design can be seen as the direct control of the values of design variables to indirectly control the values of performance variables for a given set of values for context variables.***

Modifications of the will-be description are made when one or more performance criteria are not met, based on the relationship of design and performance variables for the specific set of values of context variables. When a performance criterion is not met, each design variable that affects it represents an option for modifying the will-be description. However, modifying the value of a design variable does not guarantee that the required performance will be met. Determination of the value of the specific performance variable is required. Moreover, a design variable usually affects the value of more than one performance variables. Trying to improve performance with respect to one performance variable may, then, result in degrading performance with respect to other performance variables. ***The trade-offs among performance variables due to such inter-dependencies make performance criteria difficult to meet, often requiring non-delegateable adjustment.***

The will-be situation is usually referred to in terms of *objects*, which have *attributes* and may be *children* or *parents* of other objects, and can be seen as a hierarchical, tree-like structure. The overall parent object, such as building, automobile, is usually what defines a specific design domain. Usually, the specifications of the will-be situation are initiated by assigning values to the attributes of the overall parent object and its children and then, progressively proceed down the hierarchy towards attributes of the terminal objects. Since the will-be description is subjected to the available technologies within the design domain, not all attributes may be directly controlled

by designers. Rather, in many cases, designers are limited to assigning values to objects, rather than their attributes (for example, glazing type), in which case the values of the attributes of the object are predefined (for example, glazing transmittance, U-value, shading coefficient, etc.). In such cases the attributes are *pseudo-design variables*. When the values of design variables are "fixed" during the design process (for example orientation, number of floors), they become *pseudo-context variables*. This conversion of design variables into pseudo-context ones is often perceived or interpreted as the equivalent of *design phases*. In the beginning of the design process all variables that affect the performance variables considered may be realized as design variables. Some of them are set as context variables based on the design program and the rest are gradually converted into pseudo-context variables. At the end of the design process all design variables are considered as pseudo-context ones, since no modifications are desired.

The development of the will-be description is then specifiable and, thus, delegateable, only for quantitative criteria. The related, domain-specific knowledge consists of the values for context variables, the available alternative values for design variables, and the relations among context, design and performance variables.

2.3 Determination of the will-be performance.

Since design is aimed at producing a description of the will-be situation, rather than actually creating it, the only means to determine the will-be performance is by *simulating* it. Simulation of performance is achieved in various ways: through sketches, drawings, scale models, hand calculations, nomograms, charts, or computer-assisted calculations.

Since the will-be description is specified gradually from the attributes of the overall parent object towards the attributes of the terminal objects, designers make decisions before the will-be description is explicitly specified. Simulation procedures to determine the values of performance variables are then used at various degrees of detail. They range from simple, inexpensive ones for quick estimates of the order of magnitude (for example, sketches, crude scale models, simplified calculations), to complicated, expensive ones that are highly detailed and accurate (for example, working drawings, detailed scale models, sophisticated calculations). Moreover, different performance variables may be considered during the design process. Once performance appears *promising*, some, or all of the design variables that have been addressed may be treated as pseudo-context ones for the specification of the values of the rest of the variables down the hierarchy of the will-be description.

Determination, then, of the will-be performance is specifiable within a specific design domain, thus delegateable. In fact this is what most of the design education covers, since this is what designers do with most of the time allocated to a design project. In many cases, however, this determination of the will-be performance is almost concurrent with the determination of the will-be

description and appears to be the latter, as is the case with sketching, drafting and drawing.

3.0 Design Modeling.

A proper model of the design process should be independent of design domain and provide means to incorporate all of the specifiable knowledge associated with any specific design domain. All design activities contribute to the specification of the ought-to-be performance, the will-be description, and the will-be performance. ***A model of the design process should then provide means to represent these specifications, automate the delegateable activities, and interface with designers for the consideration of the non-delegateable activities.***

It is important that a model of the design process does not become a model of "the" or "a" designer. A model of "the designer" assumes that designers operate in a specific way (for example they know the relative importance of design criteria a-priori and can even specify it using weighting factors) which is usually compatible with algorithms that offer opportunities not only for design automation, but optimization as well. A model of "a designer" assumes that one designer's ought-to-be performance is the ought-to-be performance for all designers, thus automating decisions based on "hidden" criteria (from handbooks, standards or "experts"), which, however, may not match the designer's preferences for a specific project, or in general. ***Design tools that are based on such models may prove ineffective, forcing designers to premature judgements and/or misleading them due to hidden subjective preferences.***

Design can be realized as decision-making through argumentation, that is the process of *raising* (asking) and *resolving* (answering) *issues* (questions) through consideration of alternative *positions* (answers) that are *supported* (advantages) and/or *negated* (disadvantages) by *arguments*. This design theory has been applied towards the development of Issue-Based Information Systems (IBIS), used to help record an argumentative process and organize it so that it is transparent and retraceable. Various approaches have been taken, expanding IBIS to incorporate references, notes, etc., allowing the specification of issue relations and identifying problems with respect to the operation and growth of IBIS [Dehlinger and Protzen 1972; Kunz and Rittel 1970].

Designers assign values to design variables by monitoring the resulting values of performance variables and translating them into advantages and disadvantages of the will-be descriptions. Each value assignment can be seen as the resolution of an issue, where alternative positions (will-be descriptions) are evaluated considering the arguments that support (advantages) and/or negate (disadvantages) them. Any value assignment may become an issue, for which *conditional positions* may be taken, that is variable positions that depend on the value (resolution) of one or more variables (predecessor issues). The conditions of a conditional position may be considered as supporting or negating arguments, depending on whether or not they were met, respectively.

Design can be realized as argumentation towards the assignment of values to variables through delegateable and non-delegateable activities. All design activities are further analyzed into sub-activities which are characterized with respect to specifiability and delegateability. Moreover, all input and output data are identified, as they relate to the three sets of specifications, that is the ought-to-be performance, the will-be description, and the will-be performance.

3.1 Formulation of the ought-to-be performance.

The initial specification of the ought-to-be performance consists of two sub-activities:

1) specifying the performance variables, and,
2) specifying the performance criteria.

The other three activities of formulating the ought-to-be performance either add new performance variables along with the associated performance criteria (updating), or modify the already specified performance criteria by constraining (improving), or relaxing (degrading) them (Figure 1).

The ought-to-be performance is specified through deliberation that starts with the consideration of the "overall performance" and ends with the determination of the terminal performance variables and their associated specifiable performance criteria. Based on the theoretical analysis, performance variables can be structured hierarchically from the "overall performance" variable to the terminal performance variables. In fact, since the aggregation of deliberated judgements is non-specifiable, only the terminal performance variables are required, which can be structured as a list. A hierarchical structure, however, reflects the involved deliberation, allows for global enabling and disabling of terminal performance variables, and supports possible argumentation involved in deciding what the performance variables ought to be.

The deliberation for the determination of the names of the performance variables can be seen as the assignment of *string values* to *set variables* for each deliberated performance variable. The deliberation for the determination of the specifiable performance criteria for the terminal performance variables can be seen as the assignment of *float values* to *minima* and *maxima acceptable variables*, or *any values* to *acceptable value set variables*, for each quantitative and pseudo-quantitative performance criterion, respectively.

Each deliberation step is an issue to be resolved, since there may be more than one positions to be considered. When a position is conditional, it is automatically supported or negated by the values of the context and/or design variables, according to whether or not they meet the specified conditions. Additional argumentation is optional, as for the case of non-conditional positions.

In the beginning of the design process only few performance variables are associated with performance criteria, that is, those that describe the intended specific properties. The rest of the performance variables are associated with performance criteria at the time of their identification, that is, through checking for undesired side- and after-effects.

Although the activity of formulating design criteria cannot be delegated, it is specifiable and can be partially recorded as positions to the issues of assigning minima and maxima acceptable values for performance variables. Different performance criteria for the same performance variable would result either in conditional positions, when taken by the same designer, or in conflict of opinions, when taken by different designers, that is issues that need to be resolved among the participants of the design process. ***In this way, a computer may be used to record and then duplicate, that is, learn, the specifiable subjective preferences of designers.***

Usually, common practice within a design domain dictates the consideration of certain performance criteria. ***Considering the above "learning" scheme, the proper alternative of "a designer" models is to record subjective preferences, such as those found in handbooks, codes, standards, and opinions of experts, as positions taken against the related issues.*** In this way, designers may consider them towards the resolution of the related issues, rather than having them imposed on them, possibly without even being aware of it.

3.2 Development of the will-be description.

The initial specification of the ought-to-be performance consists of three sub-activities:

1) specifying the context variables,
2) specifying the values of the context variables, and
3) specifying the values of the performance variables.

Context and design variables are identified through the specification of procedures used to determine the values of performance variables. However, additional context variables may be defined to support the consideration of conditional positions.

The will-be description is specified through the names of design and context variables, which are either objects or objects' attributes. An object-oriented data structure can then be used to represent the will-be description in terms of objects with attributes, which may be parents or children of other objects. The parent-children relations support the specification of constraints related to the values of control variables. Databases may include available value options. Finally, pseudo-context variables may be considered as design ones, thus supporting the exploration of "new" solutions which, however, may, or may not, be possible.

The modification of the will-be description is the result of unsatisfactory performance with respect to one or more of the performance variables considered. When performance with respect to a performance variable needs to be improved, the value of the particular performance variable should either increase, or decrease. Each performance variable is associated with a list of design and context variables that affect it. Increasing or decreasing the value of a performance variable can then be translated into increasing and/or decreasing the value of one or more design variables that affect it, assuming that they operate in continuous or pseudo-ordinal scales. The available value options of such design variables may then be considered as positions which are automatically supported by the performance variables that are affected positively and negated by the performance variables that may be affected negatively.

Excluding *creativity*, or considering it as the assignment of "new" values to "old" variables, and assuming that the determination of the will-be performance is clearly specified through performance variable functions, it is possible to automate the development of the will-be description. This is possible only with respect to quantitative performance criteria. It becomes the problem of selecting the proper combination of values for design variables that will result in values of performance variables within the specified acceptable value ranges. ***This automation is useful only with respect to determining if one or more solutions are possible, that is, if one or more combinations of values for design variables satisfy the quantitative performance criteria under the specified context. Optimization is impossible, since the relative importance of performance criteria is non-specifiable.***

Even if all of the above assumptions are satisfied, the only way to conclude that a solution is impossible is through an exhaustive enumeration of the combinations of values for design variables. However, the solution space may be impractically large, since each design variable considered increases the number of possible solutions by multiplying it by the number of its value options. Moreover, there are usually terminal qualitative criteria that require direct evaluation by the designer. ***Automatic search for solutions is, then, theoretically possible for quantitative criteria, however of questionable practical usefulness.***

3.3 Determination of the will-be performance.

Assuming that all performance variables are specifiable functions of control variables, the values of which have already been determined through the specification of the will-be description, the determination of the will-be performance appears to be a fully delegateable activity. However, there may be more than one function to determine the value of one or more performance variables, raising the issue(s) of which one(s) to be used.

A performance simulation model, that is, a performance variable function, is characterized by *modeling capabilities*, *time requirements*, and *accuracy*, all of which depend on the model's (function's) inherent assumptions. Time requirements and accuracy are usually proportional,

while modeling capabilities refer to different values of design and context objects and objects' attributes that the simulation model has been designed for.

The simulation models that are used in the initial stages of the design process are usually fast and inaccurate, since they are usually meant to provide an estimate, that is, order of magnitude, for the values of performance variables. As explained, a design solution is a compromise between what is desirable and what is possible, both of which are determined throughout the design process. These "order of magnitude" simulations are meant to initiate the process of understanding what is possible under a specific design context.

The determination of the will-be performance is, then, delegateable through the specification of appropriate simulation algorithms, that is, functions of design and context variables.

4.0 A Building-Design Support Environment (BDSE).

The modeling approaches described in the previous section are applied to the building design domain towards the development of a computer-based Building Design Support Environment (BDSE) [Papamichael and Selkowitz 1990; Selkowitz et al. 1986]. ***All components of the BDSE operate on common data structures for the representation of the ought-to-be performance, the will-be description, and the will-be performance, with respect to building design.***

4.1 Structure.

The BDSE is currently structured around six software modules (Figure 2):

1) an Issue-Based Information System (IBIS),
2) a Rule-Based Information System (RBIS),
3) a Performance Simulation Algorithms Library (PSAL),
4) a Design Projects Archival Database (DPAD),
5) an Objects Value Options Database (OVOD), and
6) an electronic Handbook Information System (EHIS).

A general IBIS is implemented to record and organize positions taken for the formulation of the ought-to-be performance, that is, the names of performance variables and the associated acceptable value ranges. A specific IBIS is created for each design project, to record and organize the development of the will-be description, using the values of performance variables as supporting and/or negating arguments. Conditional positions for both types of IBIS (for example, the minimum and maximum acceptable values for work-plane illuminance may depend on task) are implemented using the RBIS, which is also used to specify the relations among performance, design, and context variables, which may require accessing the PSAL. The RBIS is also used to specify constraints for the values of the control variables, that is conditions among the values of objects and/or objects' attributes (for example, the

window width should be less than or equal to the width of its parent wall).

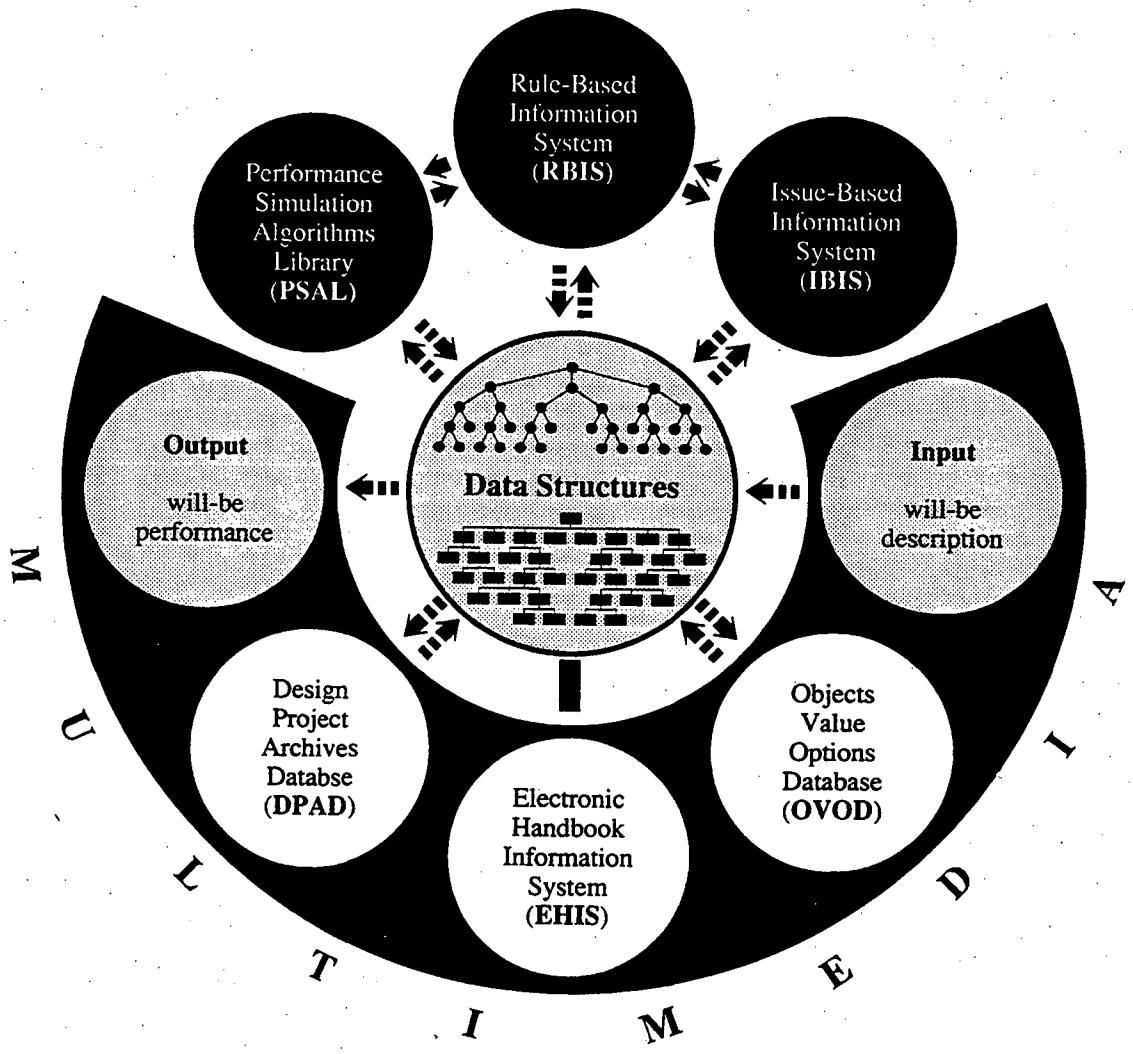


Figure 2. The structure of the BDSE.

The project-specific IBIS is also used to record the design process with respect to "*who* decided *what*, *when*, and *why*." The DPAD is an archival record of all project-specific IBIS created through the use of the BDSE, as well as non-BDSE design projects, as case studies. The OVOD contains data for the various objects and their attributes, that is design or context variable value options, such as product catalogs and climatic data. The DPAD and OVOD are implemented under a multimedia environment and are complimented by the EHIS. All three modules are linked through the names of the performance, design and context variables considered within the specific design domain.

4.2 Operation.

The operation of the BDSE relies entirely on the user, that is the designer. At any point, the designer can access and edit any of the data related to the various modules of the BDSE, whether s/he is working on a specific design project, or not. Through the general IBIS, the DPAD, the OVOD, and the EHIS, the BDSE can be used as an educational tool.

When working on a specific project, the user can consider any of the performance variables that have already been declared to the BDSE, or introduce new ones. New performance variables require the specification of the function(s) to be used for the determination of their value. The introduction of a new function may introduce new control variables, that is, new attributes and/or objects for the objects that have already been declared to the BDSE. Note that the introduction of a new performance variable is the equivalent of expanding the BDSE's design domain. Performance variables may be considered or ignored at any time during the design process, as long as at least one is being considered.

Once a performance variable is being considered, the associated quantitative performance criteria may be specified as acceptable value ranges. However, this is not required, since it is used only for "design automation," as explained later in this section. The specification of the performance variables to be considered and their associated performance criteria are supported by the general IBIS which includes various positions based on handbooks, standards and codes. The user can take her/his own position and argue for and/or against any of the already recorded positions.

Control variables are introduced automatically based on the performance variables considered. All control variables are considered by default as design variables. The designer may declare the context ones for which s/he has to assign values, which, however, must comply with the available OVOD entries. The assignment of values to design variables may be manual or automatic.

In the manual mode the designer assigns values directly, in the same way that s/he assigned values to the context variables. S/he can then ask for the determination of the values of the performance variables. After the initial specification of the will-be description, the BDSE can operate in two different ways: 1) the designer may specify which performance s/he wants to improve and ask the BDSE to indicate possibilities, that is which design variables to manipulate and how, and 2) the designer may propose new values for design variables and ask the BDSE to indicate the resulting effects on the values of the performance variables considered.

In the automatic mode, the BDSE considers the available value options and selects a combination that satisfies the specified quantitative performance criteria. If a solution cannot be found, the performance conflicts are presented to the designer, who may degrade the ought-to-be performance, that is enlarge the acceptable value range of one or more performance variables, and ask again for a solution. If a solution is found, it is presented to the designer, who may improve the

ought-to-be performance, that is narrow the acceptable value range of one or more performance variables, and ask again for a solution.

The degrading and improving of the ought-to-be performance does not need to be explicit, since there is ignorance on what is possible. Rather, the designer indicates the performance that *may* be degraded, or the performance that *must* be improved, without specifying new limits for the associated acceptable value ranges.

Finally, the designer has access to the DPAD, OVOD, and EHIS at any point. Access to these modules may be specific, that is, based on specific values or conditions of performance and control variables, or general, for browsing purposes.

4.3 Growth.

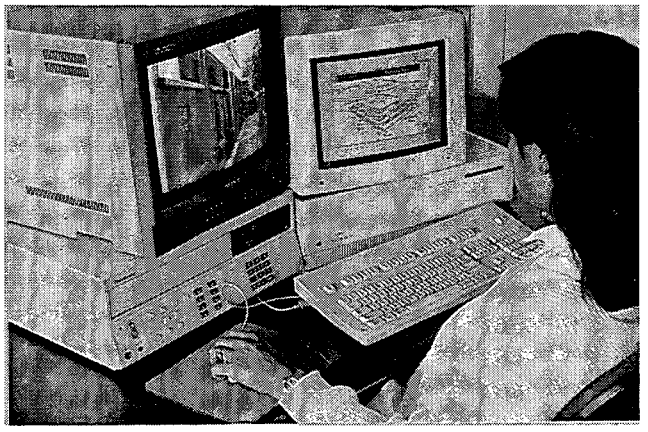
There are two ways that the BDSE grows with respect to its capabilities and knowledge. One way is through further development for the consideration of additional performance variables, which, is expected to be integrated into the BDSE's operational capabilities. The other way is through the use of the BDSE.

The general IBIS can be extended at any point by entering new positions for existing issues and new arguments for existing positions. New issues are raised only through the introduction of new performance variables along with new associated control variables. Moreover, the designer's actions towards improving performance are translated into positions for the minima and maxima acceptable values of the associated performance variables. When different positions are taken by the same user, they are treated as conditional positions and the user is prompted to specify them using the values of "known" control variables, or "new" context variables. In this way, the system learns the subjective preferences of its users. Differences in opinions among design participants can be thus identified by the BDSE, in which case it asks the users to resolve their conflict by agreeing on a single position for the specific design project.

The OVOD can be extended at any point by entering new values for the existing objects along with the values of their attributes. Finally, the HIS can be extended as well, however through development efforts rather than automatically through the operation of the BDSE. However, the existing entries of the EHIS may be linked to the DPAD and OVOD entries, as well as to the general IBIS for reference purposes.

4.4 A BDSE prototype.

Simplified versions of all BDSE components are integrated into a BDSE prototype, which is used to demonstrate its operation, and explore development options (Figure 3). The BDSE prototype addresses the design of fenestration and lighting systems for office spaces, considering comfort (luminous and thermal), energy (fuel requirements, electricity requirements and peak electricity demand), and cost (first, operating and life-cycle) (Figure 4).



XBC908-6697

Figure 3. The BDSE prototype workstation.

The prototype's EHIS, covers daylighting and electric lighting explanatory information. The EHIS and DPAD are both implemented in a multimedia environment making use of sounds, images, animation and video (Figures 5 and 6). [Shuman et al. 1988]. The performance simulation procedures of the BDSE prototype are based on simplified algorithms that were derived from detailed parametric analyses using an hour-by-hour simulation of a building's annual operation [Sullivan et al. 1988]. However, more powerful algorithms are also being developed for inclusion in future versions of the BDSE [Selkowitz et al. 1982; Ward 1990]. Moreover, various CAD software packages are explored as alternatives to graphical input of the will-be description.

5.0 Acknowledgements.

This work is supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Technologies, of the U.S. Department of Energy, under Contract No. DE-AC03-76SF00098, and the California Institute for Energy Efficiency.

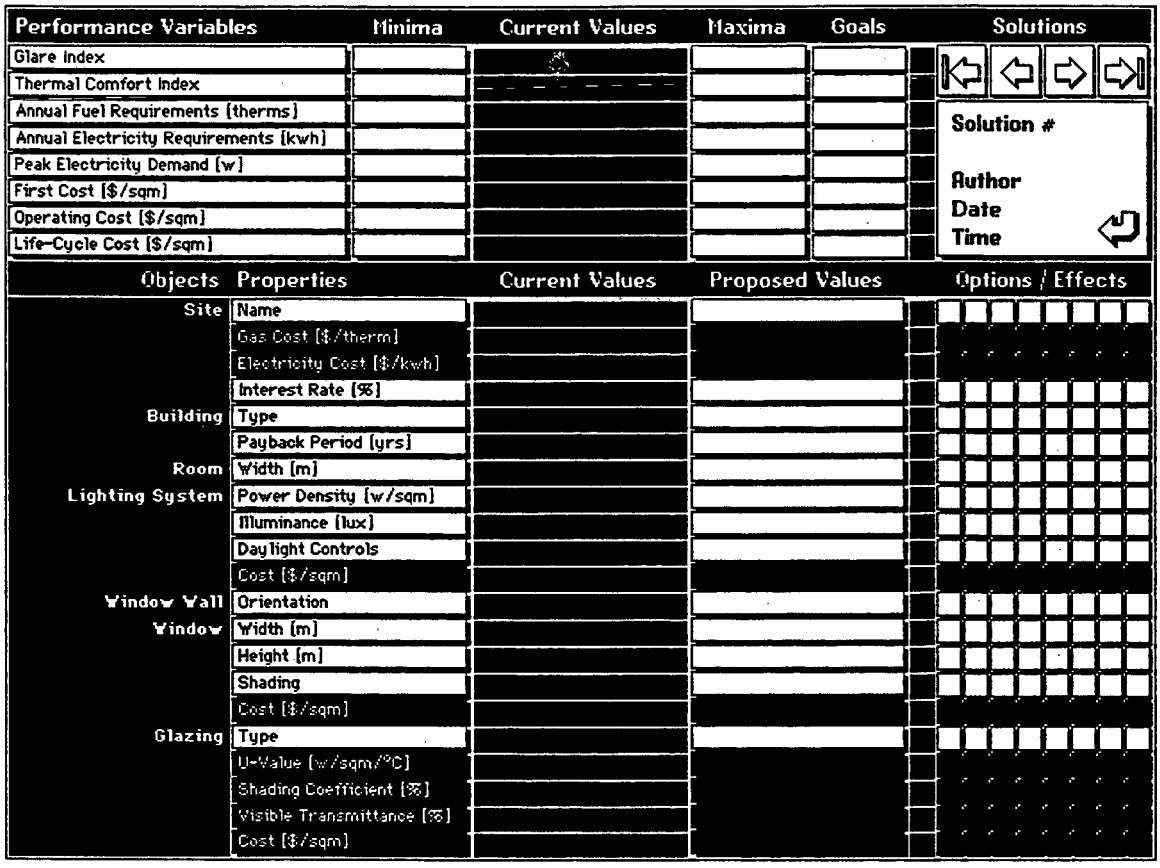


Figure 4. The main screen of the BDSE prototype.

6.0 References.

Degelman, L.O. and Kim, Y.S. 1988. "A knowledge-based system for energy efficient building design." Proceedings of the Third National Conference in Microcomputer Applications in Energy, November 1-3, 1988, Tucson, Arizona, pp. 83-89.

Dehlinger, H. and Protzen, J.P. 1972. "Some considerations of the design of Issue Based Information Systems (IBIS)." *DMG-DRS Journal: Design Research Methods*, Vol. 6, No. 2 (April-June),

Haberl, J.S., Norford, L.K. and Sparado, J.V. 1989. "Intelligent systems for diagnosing operational problems in HVAC systems." *ASHRAE Journal*, Vol. 31, No. 6 (June), pp. 20-30.

Hall, J.D. and Deringer, J.J. 1989. "Computer software invades the HVAC market: Overview of knowledge-based systems research and development activities in the HVAC industry." *AHRAE Journal*, Vol. 31, No. 7 (July), pp. 32-44.

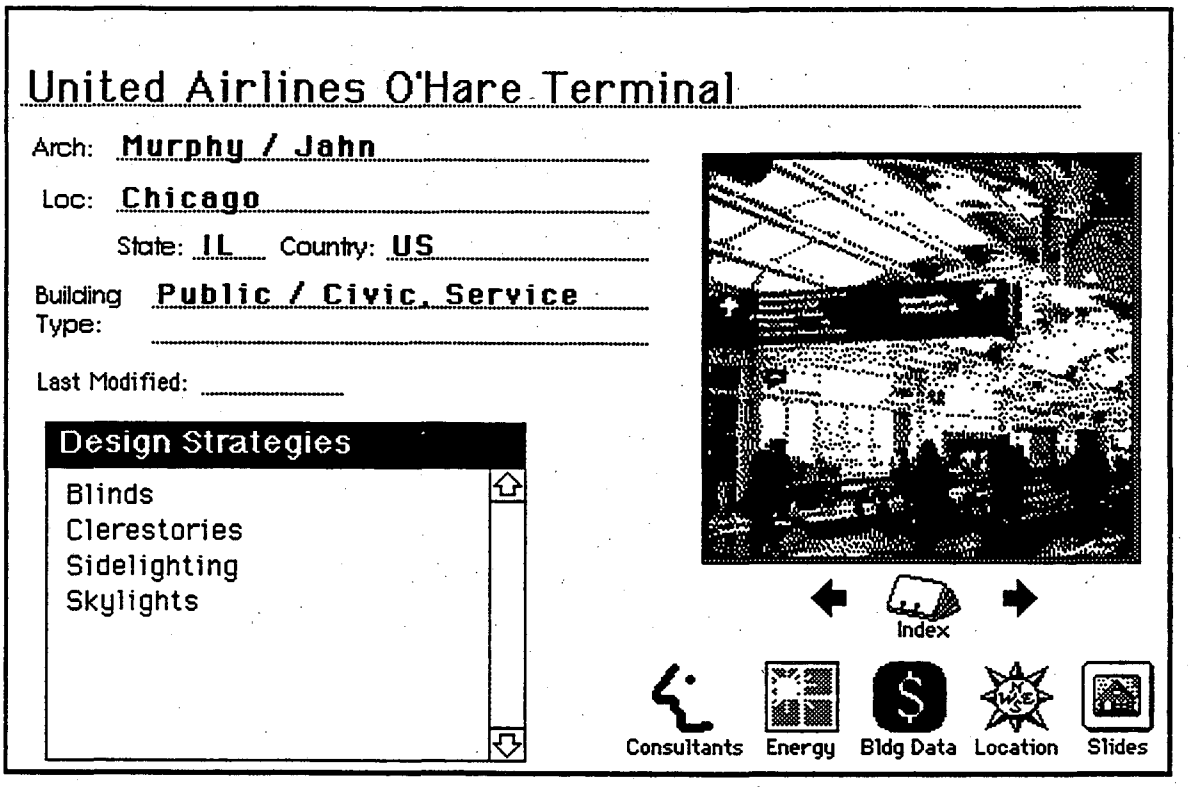


Figure 5. A sample DPAD case study screen.

Kunz, W. and Rittel, H.W.J. 1970. "Issues as elements of information systems." Institute of Urban and Regional Development, CED-UCB, Berkeley, California, Working Paper 131.

Papamichael, K.M. and Selkowitz, S.E. 1990. "Modeling the building design process and expertise." *ASHRAE Transactions*, Vol. 96, No. 2.

Rich, E. 1983. *Artificial Intelligence*. New York, NY: MacGraw-Hill.

Rittel, H.W.J. 1972. "On the planning crisis: Systems analysis of the 'first and second generation'." *BERDIFTSOKONOMEN*, No. 8 pp. 390-396.

Rittel, H.W.J. 1973. "Some principles for the design of an educational system for design." *DMG-DRS Journal: Design Research Methods*, Vol. 7, No. 2.

Ruberg, K. and Cornick, S.M. 1988. "Diagnosing window problems: Building an expert system." Proceedings of the Fourth Conference on Building Science and Technology, February 18-19, 1988, Toronto, Ontario, pp. 101-120.

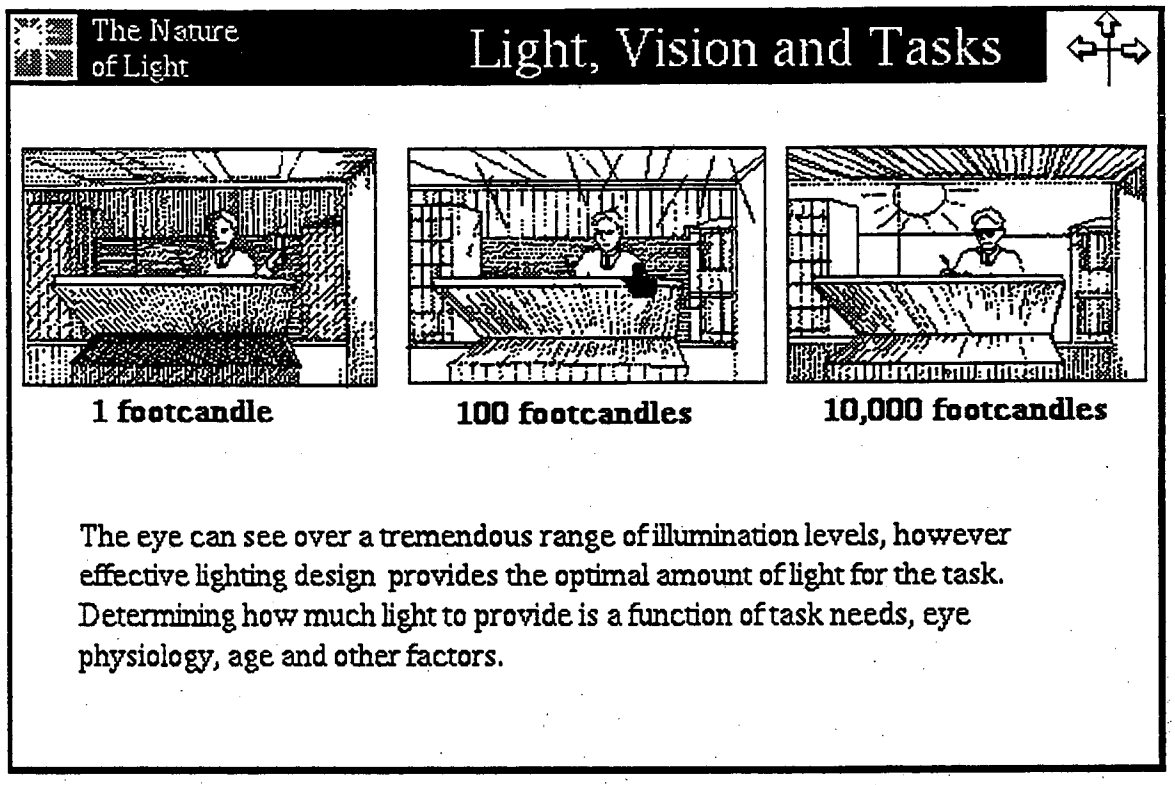


Figure 6. A sample EHIS screen.

Selkowitz, S., Kim, J.J., Navvab, M. and Winkelman, F. 1982. "The DOE-2 and SUPERLITE Daylighting Programs." Proceedings of the 7th National Passive Solar Conference.

Selkowitz, S.E., Papamichael, K.M. and Wilde, G.M. 1986. "A concept for an advanced computer-based building envelope design tool." Proceedings I of the 1986 International Daylighting Conference, November 4-7, 1986, Long Beach, California, pp. 496-501.

Shuman, J., Sullivan, R., Selkowitz, S., Wilde, M. and Kroelinger, M. 1988. "A daylight design tool using Hypercard on the Macintosh." Proceedings of the Third National Microcomputer Conference, November 1-3, 1988, Tucson, Arizona.

Sullivan, R., Arasteh, D., Papamichael, K.M., Kim, J.J., Johnson, R. and Selkowitz, S.E. 1988. "An indices approach for evaluating the performance of fenestration systems in non-residential buildings." *ASHRAE Transactions*, Vol. 94, No. 2.

Tuluca, A.N., Turner, G.E. and Krarti, M. 1989. "Expert system for selection of standardized research greenhouses." *ASHRAE Transactions*, Vol. 95, No. 2.

Ward, G.J. 1990. "Visualization." *Lighting Design + Application*, Vol. 20, No. 6 (June 1990), pp. 4-20.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
TECHNICAL INFORMATION DEPARTMENT
BERKELEY, CALIFORNIA 94720

BH360
LBL Libraries