PLAN FOR THE DEVELOPMENT OF THE NEXT-GENERATION BUILDING ENERGY ANALYSIS COMPUTER SOFTWARE

James J. Hirsch

Applied Science Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

ABSTRACT

Recent years have seen a number of building energy modelling systems reach the concluding phase of their development. Although these current systems offer sophisticated modelling capabilities, a number of deficiencies can be identified which will restrict any future adaptation to satisfy the needs of an increasingly demanding user community. Most of these systems were designed during the mid-1970's and were tailored in an inflexible manner for a now-outdated computing environment. Additionally, these systems were developed independently by groups around the world (mainly the U.S. and Europe') using different types of algorithms and solution techniques. Thus there is no way for these researchers to directly compare or exchange existing approaches; it is also difficult for them to collaborate on improving the existing systems. These problems are all solvable and it is against this background that a new idea has begun to emerge simultaneously from a number of research groups: to develop collaboratively a plan of work which will lead to a clear specification and subsequent development of the next-generation of building energy modelling systems.

1. INTRODUCTION

This document presents a plan for the development of the new whole-building energy modelling computer software. The program will be a significant advance over existing programs and will be designed to meet the simulation needs of the building research and design community of the 1990's. The work will be carried out over the next five years by Lawrence Berkeley Laboratory (with funding from the U. S. Department of Energy) and public and private sector organizations in the U. S. and other countries.

The new program is intended primarily for researchers in building energy science and engineering. However, since this program will be easy to use (so that it is attractive to researchers) it will necessarily have a wider audience of design architects and engineers requiring a detailed simulation program to produce a successful building design.

In the following sections we describe our initial ideas on the need for a new program, the possible structure and capabilities of the program, management of the project, and a schedule for planning, development, and implementation. This paper is intended to be a starting point from which a formal working plan will evolve after extensive review and input by the project collaborators.

2a. Background A simulation program is the only practical way of assessing -- under a wide range of climates, building types and operating conditions -- the performance and cost-effectiveness of the rapidly emerging new technologies, products, and techniques aimed at energy efficiency. Reliable assessment requires a simulation program that can easily and accurately model not only individual components, but also the interaction of these components with the rest of the building and its HVAC system.

There are two broad classes of simulation programs that are relevant to building simulation. Whole-building energy analysis programs such as DOE-2 and BLAST can model with reasonable accuracy a wide range of building types at an intermediate level of detail. Other whole-building programs, such as the proprietary Honeywell GEMS or public domain U.K.-sponsored ESP, have a more detailed simulation capability but have small sets of component and system models. Detailed simulation programs -- such as two- and three-dimensional finite element heat conduction and convection programs, and sophisticated physical models of primary and secondary equipment -- can model with much greater accuracy certain limited situations or specific building components.

The whole-building programs were designed for and are widely employed by commercial, non-research users. These programs meet most, but not all, of the needs of this group. The detailed simulations were designed for use by researchers; however, most of these programs cannot simulate a whole building. Moreover, there is no easy way for a researcher to integrate his detailed models into the whole-building programs to obtain a "bottom line" number, e.g., the effect on whole building energy usage of a single component or system. Thus, researchers developing new or detailed algorithms for an individual building component are often required to develop their own simulation for the entire building or make crude modifications to an existing program.

There are now several good whole-building and detailed programs in the public domain. Each has one or more of the capabilities needed by the design or research community. However, none has enough capabilities to allow it to become a "standard" in all usage areas. Although these current systems offer sophisticated modelling capabilities, they all have deficiencies which restrict future adaptation. For example, most of the systems have an inflexible software framework forced on them by the batch oriented, non-interactive computing environment within which they were developed. There is also an obvious lack of common standards in relation to system I/O, thermal network representations, component linking protocols, numerical solution techniques, operational interfaces, results display, and so on. Lastly, and most importantly, there is no consensus statement on the long term objectives of model formulation and the mechanism by which the user community can make more effective use of simulation programs. One result of this situation is that the user is forced to spend time, effort, and money to become familiar with several different programs or he must rely on a single program which may inadequately simulate important aspects of his problem.

It is unlikely that a single model or technique will become a standard throughout the building science research field (either in the U. S. or the rest of the world). However, it is possible to design a framework to ensure that future developments are facilitated, the effectiveness of narrow investigations are enhanced, collaborative developments (including algorithm exchange and model validation) are encouraged, and the as yet unforeseen needs of users several years ıce can be accommodated.

The need for a new program comes from both the research and commercial user communities. Commercial users want to model innovative buildings that cannot be handled by the existing whole-building programs. For instance, an A&E firm might want to model an apartment building with an innovative system in which a rooftop greenhouse is used to preheat supply air for the rest of the building; none of the available whole-building programs can accurately handle this kind of calculation. Researchers, on the other hand, need a straight-forward means to integrate their detailed simulations into a whole-building model. For example, a one-room detailed convection program has been developed at LBL but there is no simple way to feed the results of this detailed program into the whole-building program (BLAST) that was being used. As a result, the connection was made by hand -- a separate building energy simulation had to be done for each hour of interest. The new program would give the architectural/engineering community the single tool needed to model innovative designs and at the same time would provide researchers with a framework specially designed to allow easy integration of detailed simulation models. The development of this new program will also provide the common framework to facilitate future model development by independent groups as well as to enhance the capabilities to exchange and compare methods and results.

2b. Limitations of Existing Computer Programs Although the whole-building programs like DOE-2 have been successful at simulating conventional buildings, they all have limitations which make them difficult or impossible to modify in order to handle future research and design needs. In DOE-2 and similarly structured programs the most important of these limitations are:

The envelope simulation (LOADS), the secondary HVAC system ulation (SYSTEMS), and the primary equipment simulation (PLANT) are done separately and sequentially, with no feedback possible from SYSTEMS to LOADS or from PLANT to SYSTEMS. For example, the heat transfer through the envelope is calculated separately from the HVAC simulation which determines the air temperature in a room, making it impossible to accurately model zone-temperature control of envelope components like movable sun-control shading devices. The network programs (like GEMS and ESP) do not have this limitation in principle, but do not allow fast simulation or simple solution techniques when the problem is simple.

(2) The calculation time step is fixed at one hour, corresponding to the time interval at which weather data has historically been recorded. However, many physical processes, such as the cycling on and off of heating and cooling equipment, have characteristic times of minutes (or even seconds) over which large variations in thermal behavior can occur. Some problems require a short timestep to ensure accurate results, while many problems are handled well by a long timestep. The ability to integrate these two types of models is important.

(3) Some calculations have intentionally been simplified to reduce computer time -- for example, conduction is assumed to occur in only one-dimension, radiation exchange factors are approximated by simple area ratios, and storage of moisture in building elements is ignored. Other calculations have been drastically simplified due to lack of understanding of the physical process involved; for example, temperature stratification and convection of the air within a zone are ignored.

(4) The program's internal structure is very rigid. If a researcher wants to add a new component simulation, he must modify the FORTRAN code, recompile, debug, and test the program. This is a time-uming process that requires expertise in the internal workings of the program. For example, to add a simulation of phase change materials to DOE-2 would require several months of effort and could only be done by someone familiar with most aspects of the program's data structure, input language processor, and simulation algorithms.

(5) The user has almost no choice of solution techniques; he must accept the algorithms used in the program even though their accuracy, speed, or level of detail may be much better or worse than is required for the problem at hand.

2c. General Capabilities of the New Program The objective of the development of a new program is to integrate the best capabilities and attributes of existing programs, both detailed and general, as well as to allow variability in the level of detail of analysis and the completeness of the user's description of the problem. An example of the physical problem which the program is intended to solve is given schematically in Fig. 1, which illustrates a room being supplied with conditioned air by a secondary HVAC system (which coils are, in turn, supplied with hot and cold water by a primary system). Figure 2 shows the conditioned room in more detail, where the different forms of heat transfer that can take place in the room are schematically illustrated.

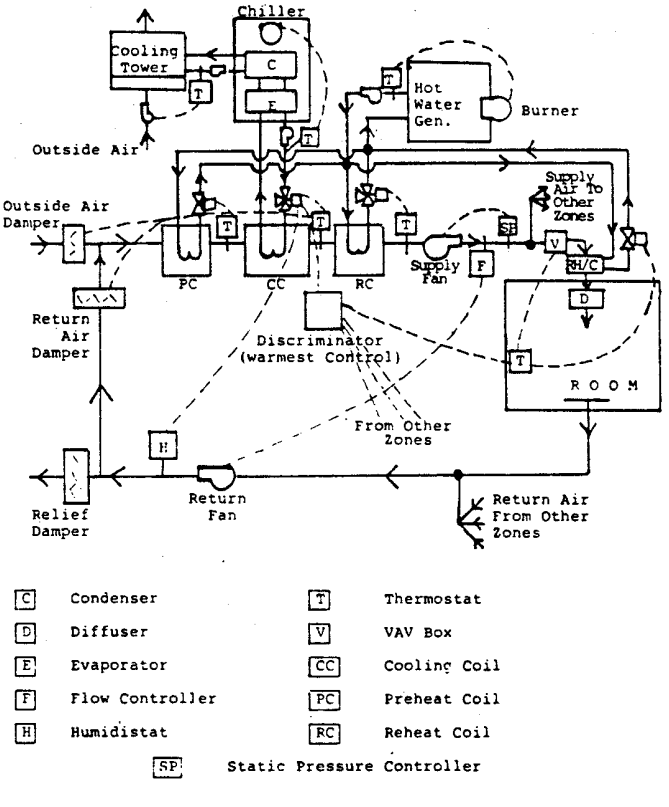


Figure 1. Typical system configuration showing room, airside (secondary) system, and plant (primary) system.

To model a coupled room/secondary system/primary system of this kind, the program must be able to do a simultaneous solution of the heat and mass flows for each component, taking into account the driving forces (outside temperature, solar radiation, etc.), building envelope characteristics, equipment characteristics, and control mechanisms. The solution approach will be to model a building as a network of discrete components, where the thermodynamic behavior of each component is represented by differential and/or algebraic equations (Sowell 84, Miller 82, Benton 82, Irving 83, Clark 85). The solution of these equations determines the outputs of each component in terms of time-varying input temperatures and flows, fixed parameters, and control variables.

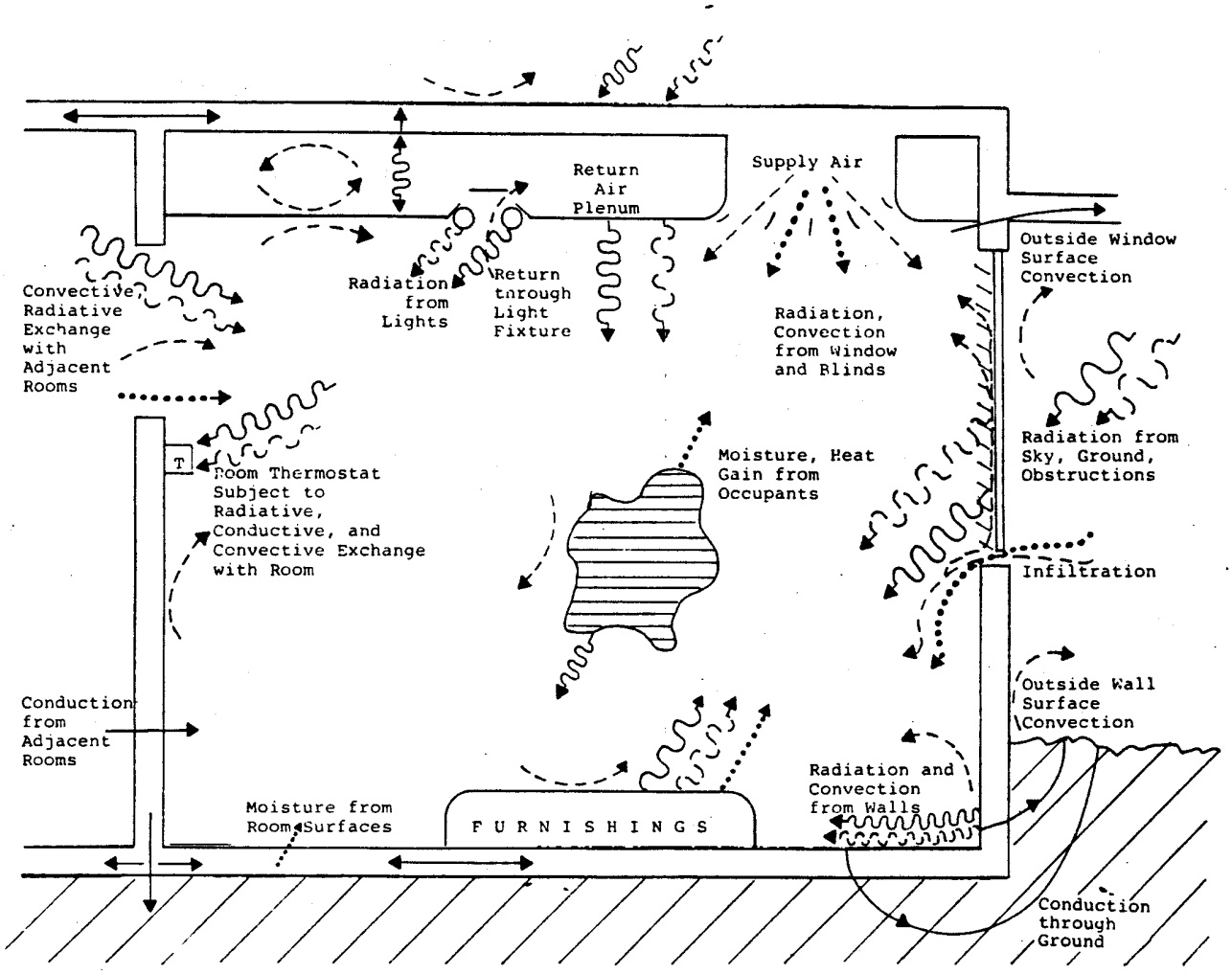


Fig. 2. Different forms of heat and moisture transfer that can take place in a room.

The program input will allow the user to specify the relationships between discrete components, i.e., how the outputs from one component are connected to the inputs of one or more other components. The physical/mathematical model which simulates the thermal performance of a particular component can be chosen from a library of algorithms, or, alternatively, the user will be able to enter his own component model through the input.

With this kind of component-based input and solution structure, it will be possible for an analyst to answer questions with many different levels of detail. These will range from fairly broad questions having to do with overall building energy consumption on a month-by-month or annual basis, to specific questions about the dynamic, small timestep behavior of specific components and control processes.

The general capabilities of the new program will include the following:

(1) The envelope and HVAC systems calculation will be fully integrated, with interactive loops allowing both feed back and feed forward. A simultaneous calculation will be performed of room-level heat flows, air and moisture transport and conditioning by the secondary HVAC system, and hot/cold water transport, conditioning and storage by the primary system. This will allow simulating many processes which are difficult or impossible to handle in programs like DOE-2 because they involve strong coupling between the envelope and the secondary system or between the secondary system and primary system. Examples of such processes include: (a) sun control with window shades based on room air temperature; (b) effect on conductive loss due to supply air blowing across windows, a common technique to maintain higher glass temperatures for thermal comfort during the heating season; (c) use of an atrium or Trombe wall to preheat ventilation air; (d) passing supply or return air through the gap in double-pane windows to pick up or discharge heat; (e) effect on room conditions of plant which is undersized, i.e. which cannot meet coil loads under peak demand or pulldown situations; (f) convective film conductances which depend on surface-to-air temperature differences; (g) passing cold air through hollow core concrete floors to precool a building at night.

(2) The timestep will be variable; it can be chosen by the user or determined dynamically by the program depending on the process being simulated. This includes the ability to allow different timesteps for individual components in the simulation.

The small timestep option will allow simulation of processes which have time constants smaller than an hour, or will be used for solution techniques, e.g. some types of finite-difference schemes, which require a small timestep for accuracy or stability. Examples of effects which can have characteristic times on the order of seconds or minutes include (a) process controls; (b) cycling on and off of HVAC equip-

ment; (c) air temperatures, flows, humidity ratios in the room and throughout the secondary system at startup and shutdown; (d) convective heat transfer between rooms with a large initial temperature difference, where this temperature difference can change rapidly when convection begins; (e) room air temperature change due to sudden increase in load from solar gain, natural ventilation, process equipment, etc.; (f) response of airsolar collectors to insolation fluctuations artly cloudy conditions.

(3) The user will be able to add, via the input language, algorithms to simulate innovative components which cannot be modelled by the basic program.

(4) The program will have built-in algorithms and solution techniques ranging from fast, simplified methods to slower, more detailed solutions where a high degree of accuracy is important. A capability to allow "mixing" of the level of detail for different components in a single simulation will be available.

The user can thus tailor his choice of algorithms and solution techniques to the problem at hand. For conceptual design studies, where the envelope or its systems are not yet well-defined, fast-running algorithms requiring minimal input would be chosen. For HVAC equipment studies, where the emphasis is on equipment performance rather than optimizing the envelope, the user could choose the simpler envelope calculations and concentrate his input effort and execution time on detailed equipment algorithms and calculation techniques.

As the design develops, the analyst may decide that more information is required for a particular component or process in which case he could select higher level algorithms. For example, for initial runs the analyst might not be concerned with the internal distribution of sunlight in a room, so the fast and and approximate shading coefficient approach might be selected for modelling the effect of window shades on solar gain. In later runs to calculate the cost-benefit of a dimming system for the electric lights, he could easily switch to using a library of optical transmission functions to model various shading devices, and he could invoke the algorithm which performs an interior visible radiation flux balance to determine the daylight illuminance distribution.

etailed thermal modelling of the envelope will be possible, including: multi-dimensional heat flow; convection and stratification of air; precise calculation of interior and exterior IR, solar, and visible radiation exchange; storage, release, and convection of moisture and contaminants; and calculation of combined illuminance distribution from daylight and electric lighting.

(6) Many building energy analysis programs have fixed connections between the building components. A dual duct system in DOE-2, for instance, has a central fan, a heating and a cooling coil, two ducts leading to the zones, and so forth.

The models for the fan and coils are connected in a fixed way. there is no way to add a fan to the configuration to produce a *dual* duct system, even though models for each of the components are in the code.

Similarly, the connection between the system models and the building envelope is pre-defined in DOE-2. A Trombe wall model exists, and various HVAC systems are modeled, but there is no way to connect the two so that outside air is preheated for the HVAC system by the Trombe wall.

The advantage of a component based program, then, is that the user can build up models of systems that are not pre-defined in the program. Such programs do exist; TRNSYS, for example, is a component based program primarily used for active solar systems. Its ability to model building envelopes is limited, but it does allow detailed modeling of solar collectors and various primary equipment.

(7) The user will be able to simulate the complex control schemes which are becoming possible with microprocessor-based energy management control (EMC) systems. The program will be able to model EMC functions such as time programming of HVAC equipment, electrical load cycling and shedding, electrical load peak demand reduction, chiller optimization, boiler plant optimization, and optimum fan start.

(8) The new program should have an enhanced library capability. Existing building energy analysis programs have libraries, but their capabilities are rather limited. For instance, only certain pre-defined items (materials, schedules, etc.) can be stored in the library, and adding, deleting, and listing items is usually clumsy. The new program should allow any item or component that can be defined by the input to be stored in the library. In addition, the library should allow combinations of components to be stored. For instance, the user should be able to store and retrieve a space with all its associated walls, windows, and internal gains and schedules; or a system with its coils, fans, sensors, and controllers.

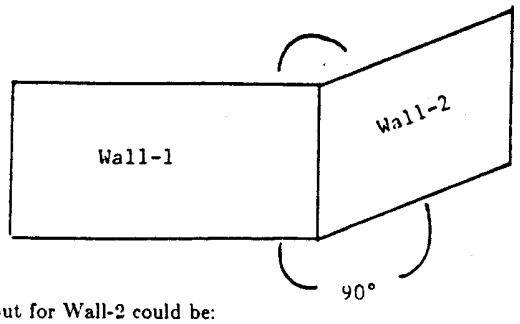
(9) It is important to reduce the difficulties encountered by the non-expert user of a building energy analysis program. At present, a user needs a great deal of background information about how a building envelope and its associated HVAC and primary equipment work in the real world, how the program does its simulation, and how to input the details of the building and equipment being modelled in order to successfully use an existing building energy analysis program. The typical user may be an expert in only one portion of the building energy analysis field -- an architect knowledgable about the envelope, or a mechanical engineer conversant with HVAC equipment, for instance. The new program should contain a database of pre-defined spaces, floors, buildings systems, and central plants that can be accessed by a user who is interested in only part of the building energy analysis problem. For the envelope, for instance, the database might contain one of the favorite floor plans used by building modelers: a core zone surrounded by four exterior zones. The shape and much of the geometry of the floor is thus pre-defined. The user could alter the details -- amount of glass, dimensions, type of construction, etc., with very little labor. An entire building could then be readily assembled out of such building blocks.

Similarly, the database should contain favorite pre-configured HVAC systems -- central VAV, for example. Again, the user could supply the details -- coil capacities, fan size, etc. We can imagine the user typing SHOW DEFAULT-VAV and obtaining the text of the input for the system. Typing DISPLAY DEFAULT-VAV could yield a diagram of the system on his terminal screen.

(10) The methods used for inputting the geometry data in most of the present building energy analysis programs are tedious and error prone. In DOE-2, for example, the X,Y,Z coordinates and the azimuth and tilt must be entered for each wall. Although an attempt is made to alleviate the problem by introducing local coordinate systems, the lack of graphical feedback or connectivity checking by the program, combined with the complexity and tedium of the input, frequently result in erroneous input.

In the interest of preserving portability, it is not our intention to work on digitizing geometric input directly from drawings or to develop 3-D graphical verification output. Instead, we would like to simplify the input and provide some simple, graphical feedback as to what the user's input looks like.

The geometric input could be relational -- the position and orientation of a wall are specified relative to a wall already input.



The input for Wall-2 could be:
location - right edge of Wall-1
orientation - 90°

This reduces the input and makes sure all the surfaces are connected properly. entire spaces could be located in the same way -- Space-2 top of Space-1 for instance. The user could then alter the dimension of the copied space.

In addition a prompt, simple graphical feedback on where the walls (and windows and shades) are is needed. We might show a plan view of the walls -- ceiling and floors might be cross-hatched or shaded. Elevation views of different sides of the building could also be provided.

(11) A built-in postprocessor will allow the user to configure and print reports summarizing the results of multiple (parametric) runs. Capabilities will include (a) tabulation of variables over user-selected time periods; (b) statistical analysis, such as calculation of averages, standard deviations, maxima and minima; (c) numerical manipulation, i.e. calculation of sums, differences, and user-specified mathematical functions of one or more variables; and (d) accumulation and display of frequency distributions (histograms and scatter plots).

(12) At the user's option, the weather processor will produce compressed weather files for superfast analysis. Various techniques are currently available which will take a year of weather data (8760 hours) and reduce it to a few representative days per month or to short sequences of heating, cooling, and swing-season days. Using compressed files of this kind can reduce running time for an annual simulation by an order of magnitude. This reduces computer costs and facilitates multi-variable optimization of building parameters which can require hundreds or thousands of runs.

In addition to the simulation capabilities of a program, other important issues are the availability of the program (where it can be acquired, for how much money, for what computers), the support of the program (how can usage be learned, who is called if problems are found), and the extent of documentation on both input and algorithms.

3. NEW PROGRAM PROJECT AREAS

The development of the new program will require a long-term effort in five major project areas:

(1) the planning and coordination among the multiple participants in the development and overall management of the technical projects;

(2) research into mathematical models and computer algorithms which will form the basis of the library components that can be simulated;

(3) research into solution techniques to solve combined non-linear partial differential and algebraic equations;

(4) user interfaces, including algorithm, language, report and building descriptions; and

(5) the overall structure of the program and how its parts fit together and communicate.

3a. Planning and Coordination Many organizations (and groups within organizations) will be involved in this project over its lifetime. It is very important that the strengths and potential contributions of individuals working in this field be known so that the best possible resources are made available to the project. No one group has all the necessary background and experience to carry out the project without significant investment in exploring the experiences of other workers in the field. The plan is to identify individuals and groups in the U. S. and other countries that have knowledge and experience in the various technical areas required for this project (short timestep finite difference solutions, lumped parameter solutions, convection calculations, controls simulations, etc.) and to bring them together to work on the project. In this manner, we can start with the best parts of existing solutions and experience and build up from there, rather than redo what has already been accomplished. To date, discussions have taken place with many researchers, including Kusuda and Walton (NBS), BLAST program developers, Huang (Chinese Academy of Science), Chounet and his group (CNRS/AFME, France), Sornay (CSTB/AFME, France), Clarke (United Kingdom), LeBrun (Belgium), Isfalt (Royal Technical University, Sweden) as well as several U. S. private sector researchers including Miller (Johnson Controls), Nall (Heery), and Rodgers (Skidmore-Owings-Merrill).

Non-U.S. participants will not only contribute significantly in general areas like solution techniques and program structure, but will also be responsible for development of specific algorithms to simulate systems which are particular to individual countries. Such systems include, for example, hollow core concrete flooring used for thermal storage in Sweden, hot water radiators (which are common in Europe), and the floor-slab radiant heating system used in France.

Meetings have begun to be held to identify the level of interest, types of involvement, level of potential personnel commitment and timetable of involvement for the various participants. The subprojects must be planned in a controlled and uniform manner to ensure that they fit together to form a workable solution. However, the projects must also be highly separable so as to allow a significant degree of independent work by participants. This will require considerable central planning and coordination. We have had experience of this kind during the LBL/ANL/LANL development of DOE-2 and found it very possible to centralize the control and planning of the program design while allowing sufficient flexibility to allow individuals the freedom to provide the best solution within the available time and budget.

3b. Research into Mathematical Models and Computer Algorithms The program will require algorithms for the simulation of processes (such as convection), components (such as cooling coils), heat and mass transfer between components, and control of components. We will distinguish three types of mathematical algorithms for describing physical processes, components or systems. Type (1) comprises algorithms which are (a) curve fits to measured data, (b) curve fits to detailed first-principle calculations, or (c) time-series transfer functions based on first-principle calculations. An example of (1a) is the performance curves for HVAC equipment used in DOE-2, which are based on manufacturer's data. An example of (1b) is the polynomials used in DOE-2 to calculate solar transmittance and absorptance of windows. These curves were obtained by solving Fresnel's equations for different incidence angles and fitting the result to a third-order polynomial. The DOE-2 response factors and weighting factors are examples of (1c).

Type (1) algorithms are fast running since they basically involve evaluation of an algebraic function or series. They usually require less input than their Type (2) or (3) counterparts. However, they may in some cases be less accurate because some physical parameters are

ignored or assumed constant. For example, the DOE-2 weighting factors which relate cooling loads to instantaneous heat gains are determined from a detailed thermal balance calculation using constant convective air film coefficients, whereas in actuality these coefficients depend on the surface-to-air temperature difference. On the other hand, there are cases where only a Type (1a) algorithm, i.e. an empirical correlation, is available because the physical component or process is complex or poorly understood that a first-principles calculation is impractical.

In a Type (2) algorithm the fundamental physical equations which describe a process or component are solved each timestep. Generally, some simplifying assumptions are made to reduce calculation time or input requirements. An example of a Type (2) algorithm is the use of Fresnel's equations to determine the transmission and reflection of solar radiation for the actual distribution each timestep of incident direct and diffuse solar radiation. Some assumptions usually made in this calculation are that the incident radiation is unpolarized, the glass is non-dispersive, and the window consists of perfectly flat and parallel panes of glass.

Type (3) corresponds to "state-of-the-art" algorithms which model physical processes in very great detail with a minimum of simplifying assumptions. An example of an algorithm in this class is a two- or three-dimensional solution of the Navier-Stokes equations to calculate the temperature and velocity field for in-room air convection. Type (3) algorithms are expected to be very slow running and require a high level of input detail. This will limit their use to specialized studies or to generation of coefficients that can be used in Type (2) algorithms. Additional fundamental research will be needed to develop Type (3) algorithms in some areas (research which we wish to facilitate but do not plan to undertake as part of this project).

Present whole-building programs rely heavily on models of Type (1), with some use of models of Type (2). The new program will include Type (1) and Type (2) models for each element of the building energy analysis problem. Some elements will have Type (3) models. The higher level models can be used to check the range of applicability and the accuracy of lower level models, as well as used for simulation of situations that cannot be handled by the simpler models. The input would first identify and select the relevant models at each node and then merge them into the overall program structure. The structure would be such that models of Type (1), (2), or (3) could be employed, depending on the user's needs, and new models of all three types could be added without expert knowledge of the program itself.

It is expected that the new program will primarily employ models that already exist or are presently under development. For instance, work is in progress at SERI and NBS in the U. S. and at CSTB/Poitier in France on inter- and intra-zone convective algorithms that could eventually be adapted for use in the new program. Detailed HVAC models are also being developed at NBS (SES, for example, Hill 85, Clark 85); these could form the basis for the HVAC models of Type (2) in the new program. In some cases, adequate algorithms already exist, for instance, in the area of two or three dimensional heat flow. An extensive search will be conducted in the first phases of the project to identify, test, and select appropriate existing models and algorithms. At the same time, individuals from the ongoing development projects will participate by including their results in the new program.

3c. Solution Techniques There already exist programs that can solve parts of the building energy analysis problem. Any of the widely available mathematical libraries (e.g. NAG) contain FORTRAN subroutines that can solve systems of algebraic equations, or systems of first order ordinary differential equations. There also exist programs such as SPICE and ASTEC that can solve complicated electrical networks. Some general simulation programs have been written and applied to building analysis. The proprietary GEMS program at Honeywell allows the simultaneous solution of algebraic equations and an analog electrical network representation of a building. Non-linear processes can be handled by a lumped parameter representation in GEMS. The new program will have at a minimum the capabilities of a program like GEMS.

In general, the new program will have to solve a combined system of algebraic and non-linear differential equations. The algebraic equations may include discontinuities or constraints. We need to establish whether a general technique exists for solving such a combined system or, if not, what the most reasonable compromise is -- e.g., linearization or separation of the problem into smaller pieces.

The first step will be to find what solution techniques are available, implement them, and test their range of validity. The next step will be to combine the various techniques to solve the more general building analysis problem in as much detail as possible. It is expected that the results will be useful to a broader group than just building energy researchers.

To facilitate the numerical solution of the various problems to be formulated in the project, well documented and kept-up general mathematical software, such as IMSL, NAG, and the Sandia Mathematical Library will be available. After careful implementation and comparison of the existing numerical methods in the special context of energy efficient buildings, it may be decided that certain mathematical subroutines be included in Energy-1. The numerical methods of these subroutines will be carefully studied and similar subroutines which are not proprietary can be used. For this purpose, public-domain mathematical packages (other than Sandia's) will be selected and obtained; e.g. LINPACK for systems of algebraic equations, EISPACK for eigenvalues and eigenvectors, some versions of GEAR for ordinary differential equations, etc. Some subroutines may be extracted, modified, and incorporated accordingly.

For example, suppose that a system of algebraic-differential is formulated and that, as for circuit analysis, it can be put into form

$$\frac{dy}{dt} = f(y,t) \tag{1}$$

where y is the vector of unknowns and f is a nonlinear operator. If the system is simple, the trapezoidal rule algorithms can be used

$$Y_{n+1} - Y_n = \frac{h_n}{2}\left[f(Y_n,t_n) + f(Y_{n+1},t_{n+1}) \right] \tag{2}$$

where $h_n = t_{n+1} - t_n$; Y_{n+1} may be iterated as

$$Y_{n+1}^{(k+1)} - Y_n = \frac{h_n}{2}\left[f(Y_n,t_n) + f(Y_{n+1}^{(k)},t_{n+1}) \right] \tag{3}$$

with a time step limit. (2) may also be considered as a general system of nonlinear equations,

$$G(Y_{n+1})=0 \quad or \quad G(Y)=0 \tag{4}$$

for simplicity, and solved with the standard Newton-Raphson method

$$G(Y^{(k+1)}) \approx G(Y^{(k)}) + J(Y^{(k)})\,(Y^{(k+1)} - Y^{(k)}) = 0 \tag{5}$$

where J denotes the Jacobian matrix $\frac{\partial G}{\partial Y}$; for each iteration (5) can be solved with a subroutine for a system of linear algebraic equations. (2), or (4), can also be solved directly with a subroutine for a system of nonlinear equations. On a Runge-Kutta subroutine from any ordinary differential equation package can be used for (1), if the system is non-stiff. If it is stiff (i.e. it involves very rapidly changing and very slowly changing components), then some Gear subroutine should be used.

There are softwares for partial differential equations (Sweet 79); however, their use has been limited because the numerical solution of partial differential equations is even more problem-dependent than algebraic and ordinary differential equations, and research on success and efficiency of a method in this area is especially important. Take efficiency, for example: in general, for explicit finite difference

methods, the timestep must be smaller than that corresponding to the fastest possible process at any point, but calculation for each point is independent of the other points. Whereas, for implicit finite difference methods, the timestep need only be small enough to resolve the time evolution of the process of interest. However, for each time level a system of linear algebraic equations usually results for linear problems and a system of non-linear equations for non-linear problems. Which is the more economical depends on the particular problem under consideration.

3d. User Interfaces One of the design goals of present day Building Energy Analysis Programs was to reduce the labor of user input and create a more friendly user interface. To this end, fixed format input was discarded in favor of free format input languages. Nonetheless, the same amount of data at the same level of detail still has to be input, and the input phase of existing Building Energy Analysis Programs remains a major barrier to the wider use of building energy analysis.

A major problem in designing the I/O for the new program is that greater detail in the simulation usually requires greater detail in the input supplied by the user. Similarly, a more general and flexible program means that the user has more options to choose from. Rather than welcoming the free-claim to choose, the user may feel overwhelmed by it. The new program will not be successful, even as a research tool, unless these problems can be mitigated.

The output side of the I/O interface also merits attention, though not to the extent of the input side. A research oriented simulation program requires that the user be able to choose the output variables, the output format, and the output timestep. That is, the user must be able to design, through the input, his own output reports, graphs, and histograms. Creating this capability is not inherently difficult and should be one of the more straightforward tasks in this project.

The participants in the new program project, at least at this time, are more interested in the simulation side of the project than in the "user friendly" input task. New developments in user friendliness will come from the commercial computer industry, not from government-sponsored research. We hope to choose judiciously from already-existing techniques to create a user interface that will ease the burden of using a powerful, flexible simulation program.

The input processor can be regarded as one of the two main elements of the new program -- the simulation itself being the other. thus it corresponds to the input languages of existing Building Energy Analysis Programs such as BLAST and DOE-2.

Our strategy, to combine ease of use with flexibility and generality, is to create a user input processor that will accept input at three levels of detail. The highest (least detailed) level might be called the conceptual level. This is the level the user would employ when a building or system is in the early design stage -- when the problem the user is trying to solve is not yet well defined and is subject to major alterations. This type of input (Level 1) will be interactive. The user will be able to display on his terminal various pre-assembled systems of building components. For instance, the library will contain building floor plans with all the geometry and schedules in place, or preconnected HVAC systems, which the user will simply access and alter. This level will be used to describe the bulk of many simulation problems. The more detailed levels will be used to describe only those areas in which the user is most interested.

The second level of input (Level 2) will correspond to the detail provided by the input languages of existing Building Energy Analysis Programs. Unlike these languages, however, the new program input will be interactive. If we imagine the input problem to be defining "components" (windows, walls, coils, etc.) by assigning values to "component parameters" (location of window, coil capacity, etc.), the program will have menus for each component and will prompt the user for input for each component parameter, or display the already entered value or the default.

Simple graphical feedback at this Level and at Level 1 is very important. As mentioned in the general capabilities section, a great deal can be accomplished with rather simple graphical feedback. Simple histograms of the operating schedules on the user's terminal would be one example. Plan views of the various spaces and floors, component diagrams of HVAC systems showing already entered values are other examples of prompt, simple feedback that will eliminate much input error.

Level 2 will be used to input the bulk of the geometry for the simulation of existing or fully designed buildings. The method of specifying the geometry will be relational, rather than absolute. That is, the position of a wall or other geometrical element will be specified relative to another wall that it is attached to. This will allow the program to ensure connectivity, simplify the input, and reduce user geometry errors.

The lowest, most detailed and general level of input (Level 3) will be a non-interactive input language. The choice of the type of language – context free or context dependent – has not yet been made. This level of input will have several uses. first of all, the two higher levels will be translated into Level 3 input rather than being fed directly to the database. Thus, Level 3 can be used for detailed, minor changes to models created at a higher level.

Secondly, Level 3 can be employed by experienced users who grow impatient with the interactive style of the higher levels. Finally, Level 3 will be used to alter and add to the capabilities of the simulation itself.

At Level 3, the user will be able to define new "components" and their associated component parameters. These new components and parameters will be automatically added to appropriate tables so that the Level 3 input language will be able in the future to recognize the new input and transform it correctly onto the database. This is similar to adding new commands and keywords to the DOE-2 program by changing the keyword table. In DOE-2, this is accomplished by running a stand-alone, fixed input format program. Also, assigning the keywords to the correct position in the database is not exactly automatic. In the new program, this task will be accomplished via the Level 3 input language.

Also at Level 3, the user will be able to define new simulation algorithms for new or existing components. This will require including in Level 3 an interpreter with FORTRAN-like or APL-like syntax. The syntax will be as similar as possible to the language used in writing the simulation program, so that the user, when satisfied with the algorithm, can move it permanently into a solver in the simulation program. An alternative would be to include a call generator in Level 3, in addition to or instead of the interpreter. The user would describe his algorithm in some high level language and the code generator would translate this into FORTRAN (for example), which could be sent to the compiler. Simply having the interpreter would mean longer execution time than using compiled code.

The Level 3 input will also allow the user to alter, to a certain extent, the structure of the language itself. At the simplest level, this could involve changing the default tree; that is, changing the manner in which defaults are assigned to component parameters. At the highest level, it could involve changing the language (via the syntax table) so that it would no longer resemble the original. Since the two higher input levels feed into Level 3, this would be inadvisable, and it is likely that the power of the language to alter itself will be severely restricted.

Finally, the user at Level 3 will basically be able to design his own simulation program. He will be able to tell the controller in what order to call the solvers, what level of solver to use, and what convergence criteria, time steps, etc., to use for each solver or group of solvers. Of course, the controller will be "intelligent" (we hope!) and will be able to make good choices for solver calling sequence, convergence criteria, and time step without external aid. But these choices will always be capable of being overridden by the user.

The output processor will be a separate program element that will operate on a separate database – the output database. The user will be able to specify (via the Level 2 and 3 input processors) what variables he wishes to save, and at what time intervals. This information will be given to the controller, and a special "solver", the output solver, will be used to access the appropriate data from the database at the desired intervals, and move it to the output file.

The output processor will be interactive, similar to the Level 2 input processor. The user will be given the option of using certain predefined summary reports, or creating his own by summing, averaging, scaling, combining, the variables on the output database and displaying the results as formatted reports or as graphs and histograms. Like the Level 1 and 2 input processors, this task is a separate well-defined work module that can be performed by an independent group.

3e. Program Structure To allow the capabilities and flexibility described above will require an internal structure completely different from existing programs. We envision four main components in this structure:

(1) a database system which contains user input information, detailed parameter description for simulation algorithms, intermediate and computed results, and other categories of information;

(2) algorithm routines that are the heart of the actual simulation;

(3) a computation control structure; and

(4) a user interface module.

In existing programs, items 1, 2, and 3 are merged together or "hard-wired". This hinders substitution of different, more exact algorithms. Since the algorithms, solution techniques and parameter data are separate within the new structure, the user will be able to independently change each of these items. Additionally, each of these items can be described through the user interface or language module. Through the input the user will be able to control not only the values and constants necessary for the simulation, but also which algorithms are used and where they are performed in the simulation. The control structure will determine which algorithms to use and how the algorithms are sequenced, and even allow multiple algorithms to be applied to the same computation so that results can be compared.

All communication between algorithms will be through the database. When a new algorithm is added, the control structure will extract required information from the database system for that algorithm, pass control to the algorithm, then place the results back into the database structure. Output subprograms will use the database system to extract results to be reported, summarized, or analyzed.

This kind of modularity allows the input and output phases of the program to reside on different hardware if desirable. The input and output modules could be running on a desk-top workstation (micro computer) and simply request or supply information to the database on the main computer where the simulation or number-crunching parts of the programs reside.

From the programming perspective there are several issues that need to be examined. The resulting program must be easily portable to different computers. This usually requires it to be written in FORTRAN or some other widely available computer language. However, using a language like FORTRAN makes the programming of complex systems very difficult. For this reason, it is likely that one or more pre-processors will be developed that allow the code to be developed in a "higher level" language (with capabilities -- for scientific computation -- of less portable languages like ADA, C, and PASCAL) and automatically translated for the target machine compiler (such as FORTRAN). This higher level language must allow more general data structure, expandable or dynamic memory allocation, recursive routine (algorithm) calls, and sophisticated looping and logic structures. As languages like C and ADA become generally available on scientific computing hardware, the pre-processor could produce code for these types of compilers instead of FORTRAN.

For the user interface, allowing the input of new algorithms requires an interpreter which will be used by the simulation to follow the steps of the algorithm described by the user. It is also envisioned that a code generator will be implemented which will translate the user's input into new FORTRAN code so that a permanent, fast version of the new algorithm can be added to the program's algorithm library with minimal effort.

There has been a substantial growth in the use of computer-aided-design (CAD) systems in the building design process, and many companies using these systems also use energy analysis programs. Some have attempted to integrate the energy programs into their CAD systems, with varying levels of success. Although we feel that the private sector has the required resources and capabilities to develop the future CAD systems for the design community, the new building energy analysis program must contain the necessary interfaces to allow integration into these CAD systems without major modifications. This will require interaction between these two groups of program developers.

4. PROGRAM SCHEDULE

There are two main stages for this long-term effort: the development stage and the maintenance stage. The development stage consists of three main phases: (1) conceptual design, (2) prototype implementation and testing, and (3) final implementation, testing, and release. Development is expected to take approximately five years; maintenance follows development and continues for the useful life of the program. The effort level during development is expected to increase during the first two years, peak during the third and fourth years, and decrease during the fifth year. Maintenance effort is expected to be level during the first few years after the program release then decrease.

Although development will have three phases, as outlined above, each of the five project areas of this stage will proceed through these phases on different schedules with different effort levels. Much of the conceptual design is expected to be finished during the first year. This phase will produce a conceptual design document which describes all of the major components of the program, including:

- the gross structure and information flow between the program's main subsystems;
- the fine structure of communication between the main subsystems;
- the fine structure of information storage and flow within the main subsystems;
- a detailed description of the various research efforts that must be undertaken to answer any unresolved questions, including who will perform the work and the schedule for completion;
- a more detailed plan and schedule for the next phases of the project; and
- an agreement among the participating research groups and organizations outlining the expected contributions, responsibilities, commitments, and deliverable schedules for each.

The second development phase will proceed for roughly the following two years and will result in a *working prototype* of all major components of the program. The prototype will be used by project participants in testing and further refinement of the program subsystems. This prototype will not contain *all* the capabilities of the final program, but will act as a testbed for the work remaining to be done. It will also allow all the aspects of the conceptual design to be tested to ensure that a workable solution has been produced. In this way, all necessary revisions can be made before the final version of the program is produced.

The third development phase will take approximately two additional years and will result in the first version of the code and its documentation for public release and wide distribution.

The planning and coordination activities will be active over the entire development stage. During the first phase, these activities will concentrate on identifying potential contributors and bringing the participants together to produce an acceptable conceptual design. This includes insuring that users and developers interact in the conceptual design phase so that the program meets the needs of the various user types. During the second phase the efforts will concentrate on coordinating the research tasks being performed by each participant to facilitate solving problems as they arise, to ensure adequate information flow among the groups (especially among those whose work is highly independent of the others), and to track the progress towards milestones for each sub-task area. During the final phase, the coordination activities include the same activities as the second phase with the additional responsibility of insuring that the documentation and testing groups receive adequate information and assistance from development groups.

The second two project areas (research into models and algorithms and research into solution techniques) will proceed along similar strategies although they will have different timing and effort levels. Both areas will begin with a search for existing work that can be used in the new program. These potential contributions will then be evaluated and tested to decide if they are useful as is or with modifications. At this point, it will be known what new research is needed in these two areas.

The solution technique research is expected to be concentrated during the second and third years of the project. During the first year, various potential techniques will be identified, and testing on their limit of applicability will begin. In the following two years, final approaches for solution techniques will be developed and implemented; during the last two years they will be fine tuned.

In the model/algorithm project area, again, the first year will mostly be devoted to identifying existing work or work in progress which will contribute to the effort. Identified contributions will then be acquired and tested. In the next four years, work will proceed to produce models and algorithms where necessary. This work is expected to peak in the fourth and fifth years of the project and will continue after release of the first public version of the program. In fact, one of the objectives of the new program is to facilitate this area of research.

The user interface project area will concentrate first on the design and implementation of computer software rather on the production of useful documentation. During the first year it is expected that much, if not all, of the language processors, database systems, and report generators will be fully designed and implementation begun. The bulk of the implementation effort will proceed over the second and third years. During the last two years, refinements and improvements will be made, but the main effort will switch to the production of good documentation for release with the public version of the code.

The overall structure design project area will proceed at a high level of effort during the first two years of development. This will primarily contribute to the conceptual design document at the end of the first year. During the second year, as development and testing of models, algorithms, and solution techniques proceed, it is expected that some of the initial design will require modification or refinement; also, new areas that need work will be identified. During the final three years of development, activity in the structure design area will be at a low level.

The second stage of the project is maintenance. Once an initial public release is made and the program is in wide distribution, problems identified by users in the computer code and its documentation will be fixed. During this stage, work will also be carried out on new models and algorithms to augment the capabilities of the program. These efforts are expected to be highest during the first two years of this stage. At that time, very mature code and documentation will exist and maintenance efforts can be substantially reduced.

Obviously, the above described schedule is highly dependent on yearly funding levels for the organizations involved and decisions on technical issues made during various phases of the program development. This initial discussion is intended to provide a framework within which more detailed discussions can proceed; thus management and technical decisions can be made.

REFERENCES

Benton, R., MacArthur, J. W., Mahesh, J. K., and Cockroft, J. P., "Generalized Modeling and Simulation Tools for Building Systems", ASHRAE Trans. *88*, Pt.II (1982)

Building Energy Simulation Group (BESG), *DOE-2 Engineers Manual*, LBL-11353, 1982, Lawrence Berkeley Laboratory.

Clark, D. R., Hurley, C. W., and Hill, C. R., *Dynamics Models for HVAC System Components*, ASHRAE Transactions *91*, pt. 1 (1985).

Clark, Daniel F., *HVACSIM$^+$ Building Systems and Equipment Simulation Program Reference Manual*, NBSIR 84-2996, January 1985.

Clarke, J. A., *Energy Simulation in Building Design*, Adam Hilger Ltd, P.O. Box 230, Accord, MA 02018, 1985.

DOE-2 Engineer's Manual, 1979.

Hill, C. R. *Simulation of a Multizone Air Handler*, ASHRAE Transactions *91*, pt. 1 (1985).

Irving, S. J., and Quick, J. P., *Simultaneous Solution of Room Response and Plant Performance*, Proc. of the Int. Conf. on System Simulation in Buildings, Liege, Belgium, 1983.

Miller, D., "A Simulation To Study HVAC Process Dynamics", ASHRAE Trans. *88*, Pt.II (1982)

Sowell, E., Taghave, K., Levy, M., and Low, D. W., *Generation of Building Energy System Models*, ASHRAE Trans. *90*, Pt.I (1984)

Strang, W. G. and Fix, G. J., *An Analysis of the Finite-Element Method*, Englewood Cliffs, New Jersey, Prentice-Hall, 1973.

Sweet, R. A., *A Survey of the State of Software for Partial Differential Equations*, STAN-CS-79-704, 1979.

Winkelmann, F. C., *Matrix Formulation of Solar Heat Gain Through Multi-Layer Fenestration Systems*, Building Energy Simulation Group internal report, LBL, 1984.