Building Simulation

Seventh International IBPSA Conference
Rio de Janeiro, Brazil
August 13-15, 2001

HVAC SYSTEM DESIGN AUTOMATION: ISSUES, METHODS, AND ULTIMATE LIMITS

Charles S. Barnaby, Mikhail Shnitman, and William A. Wright
Wrightsoft Corporation
Lexington, MA 02420, USA

ABSTRACT

We have developed software applications that design HVAC systems (that is, select, locate, and size components and their interconnections) given building descriptions, user preferences, and built-in rules. When this capability is combined with other key application characteristics (graphic interface, interactive recalculation, full integration, self-contained reference data, strict retention of user modifications, and automatic parts selection), the result is a powerful assistant that greatly increases user productivity and design quality.

Our application structure is a framework for iterative improvements in design and analysis algorithms and the addition of optimization techniques. We see this type of software ultimately replacing most design functions now performed by people.

INTRODUCTION

One of our company's long-term goals is to automate the HVAC system design process. Over the last several years, we have developed highly integrated multi-function applications that can design building HVAC systems, as opposed to simply analyze them. The software uses an interactive graphic diagramming tool as its primary interface, and we have automated the takeoff of building plans to allow heating and cooling load calculations. In addition, we apply user preferences, standard engineering methods, and common practice to lay out and configure HVAC systems. The software also produces parts lists and cost estimates.

The ultimate goal of our work is automatic production of complete system designs that do not require user correction or addition. Reaching this goal requires an iterative development process, and we are only part way through the iterations. Our work to date has led us to a number of observations about the nature and capabilities of interactive design automation software.

The paradigm underlying our applications is that of the helpful assistant. Reference information is readily at hand, tedious calculations are performed automatically and quickly, initial HVAC system design suggestions are offered, and required project documentation is produced (all with no back-talk).

Our company is currently shipping two products that incorporate design automation features. Right-Suite Residential calculates peak heating and cooling loads, positions and sizes ducts, designs radiant heating panels, sizes ground-coupled (geothermal) heat exchangers, performs operating cost comparisons, and produces a system parts list. An alternative version offers the same functions but uses Canadian analysis procedures. Target users are contractors, who often install systems without participation of a design engineer.

The second product, Right-Suite Commercial, provides similar functions for commercial projects (not including radiant heating or ground source design) and is of interest to engineers as well as contractors.

We believe our users have three main reasons for using our applications: increased productivity, better design quality, and reduced liability. Fast, interactive, and flexible software allows correct application of design procedures and rapid exploration of alternatives, yielding quality designs in minimal time. This has obvious commercial advantages. In addition, users want designs that conform to sanctioned procedures that limit liability exposure in the event of operational problems.

Our applications are PC-based, operating on all 32-bit Microsoft platforms (Windows 9x/ME/NT/2000) and are implemented in C++ using Microsoft Foundation Classes (MFC).

IDEAL DESIGN AUTOMATION BEHAVIOR

The behavior of our applications is based on an ideal model of how a designer might work with a software-based assistant. We imagine the following project process –

- The user initiates the project, either from “blank” or from a template of a similar building, and enters basic information, such as project location and client identity. The application supplies reference weather and fuel cost data as needed.
- If not already specified in the template project, the user indicates HVAC system design preferences, for example system type, duct type and shape, design methods, maximum water temperature, and so forth.
- Using a graphic interface, the user describes the building by manipulating objects that represent building components and assigning construction characteristics to them. Defaults are provided to aid data entry speed and accuracy. The display allows immediate data checking. CAD files or other pre-existing data sources, if available, can be exploited to avoid duplicate data entry. Building heating and cooling loads are calculated in response to the building description. Optional displays and reports allow detailed checking of load calculations; the user is not obligated to wallow in thermal minutia.
- The application designs a system, based on the building description, preferences, and built-in rules. A suggested design should always be produced, so the user has something to react to, rather than a blank screen.
- As the user alters and refines the suggested design, the application checks all data and if valid, recalculates all dependent results (e.g. heating and cooling loads).
- The design responds to changes in the building description and/or preferences. Items such as additional registers may have to be added in response to building changes; such changes are made as consistently as possible with the existing design.
- To document the project, the application makes available reports and drawings that can be used without further modification. Graphic information can be inserted directly into CAD drawings, as appropriate.

Figures 1-3 show a simplified example of the central portion of the ideal behavior, as implemented in one of our applications.

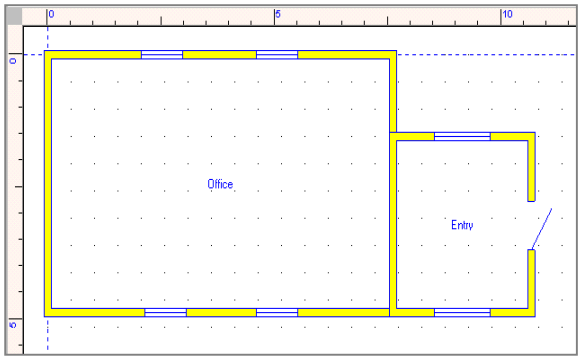

Figure 1. Sketch plan as entered by user (duct layout hidden)

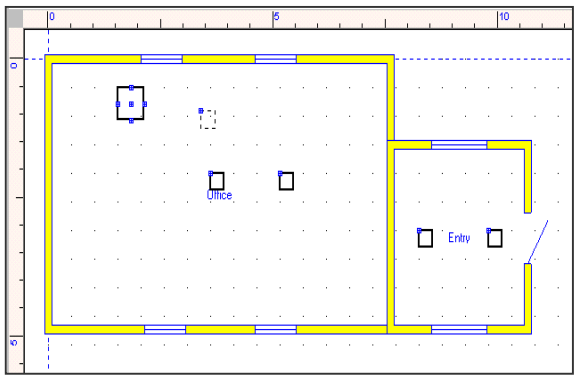

Figure 2. Equipment and register placement as suggested by application. The number of registers is determined by the space load and maximum air flow per register (user preference). Initial placement is along the central axis of the space.

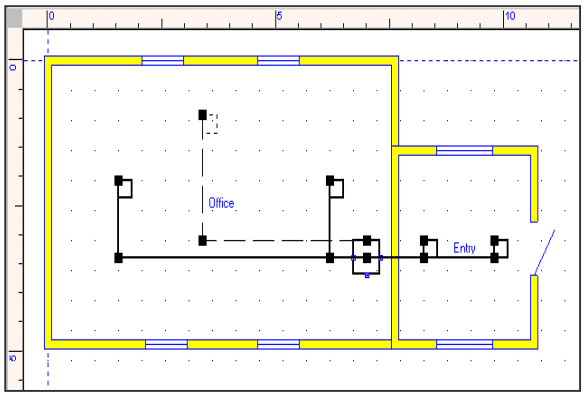

Figure 3. User drags registers and equipment to desired locations, then selects duct layout type. Application creates and sizes required duct runs.

CHARACTERISICS OF HVAC DESIGN AUTOMATION SOFTWARE

Summarizing the prior section, it is our view that HVAC design automation software should take a building description as input and produce as output all documents and information required to build,

sell, and/or procure the HVAC system for that building.

Fully achieving this goal is extremely difficult and remains years away. However, we contend that *imperfect is still very powerful.*

The relationship to the user is the key to successful implementation of HVAC design automation software given current computing limitations. We have found that the following characteristics provide an environment in which users can work effectively:

- *Interactive.* Results from input changes should immediately propagate. Within practical limits, there should be no "calc" command or operating modes.
- *Integrated.* A fully integrated implementation quickly and invisibly handles data transmission between analysis steps. The user should not be distracted by delays or administrative steps such as data exchange.
- *Everything editable.* Automated features notwithstanding, the software should retain all user changes to the HVAC design, allowing the user to retain ultimate control. The user should also be able to easily perform "what if" experiments, to explore and verify design choices.
- *Graphic interface.* The software should display representations of the building and HVAC system. It should allow direct editing of equipment layout.
- *Self-contained.* The application should contain required reference data. The user should never have to refer to additional sources of information or perform side calculations.
- *Retained preferences.* User preferences should guide the design process so all common system variants can be generated. Preferences should be retained so similar designs can be generated without repetitive input.
- *Specialized, process-oriented output.* Given the productivity orientation of design applications, output should exactly fit the user's process. Printed reports should be usable without any "minor reformatting" and graphic information should be directly inserted into CAD files. User-customizable reports are helpful in this regard.

In short, the software should be a highly integrated productivity assistant that supplies design suggestions and records all user changes. This might be thought of as a word processor that writes the first draft, tracks and checks modifications, and produces camera-ready output.

ANALYSIS METHODS

The analysis methods behind our applications are standard, sanctioned procedures that are in wide use for both hand calculations and in various computerized implementations.

In our residential applications –

- Heating and cooling loads are calculated according to Manual-J (ACCA, 1986) or, for our Canadian variant, F280 (CAN/CSA-F280-M90, 1990 and HRAI, 1996).
- Duct sizing is done according to Manual D (ACCA, 1995) or, in the Canadian variant, HRAI methods (HRAI, 1998).
- Radiant design relies on ASHRAE procedures (ASHRAE, 1996) and advice regarding common practice from a group of radiant heating manufacturers.
- Ground-coupled heat exchanger sizing follows International Ground Source Heat Pump Association and ASHRAE publications (Oklahoma State University, 1988; Bose, 1985), guided by consultation with several ground source heat pump manufacturers.
- Operating cost calculations use bin methods (Kelly and Parken, 1978 and Domanski and Kelly, 1980) using additional data and procedures obtained from unitary equipment manufacturers.

In our commercial application –

- Heating and cooling loads are calculated using the CLTD method (ASHRAE, 1989 and ASHRAE, 1977).
- Duct sizing is done using equal friction or static regain procedures derived from ASHRAE (ASHRAE, 1997), SMACNA (SMACNA, 1990), and Manual Q (ACCA, 1990).

None of the methods we currently use is particularly new or sophisticated per se. Instead, they are standard, common practice methods. Involvement of industry groups during development ensures that we apply the methods in practical ways. Our applications produce results and designs that are readily acceptable to current practitioners, both because they are familiar and because they are based on sanctioned methods that provide liability protection in the event of operational problems.

DESIGN ALGORITHMS

An important innovation in our software is implementation of design algorithms for duct and radiant systems. User preference settings control the

behavior of the algorithms; preferences are usually preset and saved, so they need not be modified in each project. A fundamental implementation issue is identifying the appropriate set of preferences. Some are obvious (duct material, for example). Others were found through discussion with industry members ("you could do it this way, you could do it that way").

The following briefly describes the design sequences we use for duct and radiant systems.

Duct design starts by determining the number and placement of registers. The number is derived from the room load and the preference of maximum air flow per register. Placement is initially along the central axis of each room. Next, an air handler is placed centrally in each zone. The duct system layout is established according to a preference – radial or one of several plenum schemes. Then all ducts are sized using the preferred calculation method, required air flows, duct material characteristics, and preferences concerning pressure drop. Fittings are selected based on duct sizes and user preferences.

When the user alters the position of a register or air handler, the duct layout is regenerated and recalculated, resizing runs as required. A "user defined" layout preference allows arbitrary arrangements to be drawn and sized. An "as built" mode is also provided, so systems with fixed dimensions can be analyzed. This is useful for commissioning and troubleshooting.

Hydronic radiant heating systems rely on serpentine runs of tubing embedded in floors or ceilings. Warm fluid (generally water) is pumped through the runs. The radiant system design algorithm determines tubing run lengths, fluid temperatures and flow rates, and grouping of runs onto manifolds operating at the same temperature. Panel area is determined by user input via the graphic interface. Tubing spacing is set by user preference. Run length is derived from preferences of preferred and maximum run length.

Panel output depends in a non-linear fashion on surface temperature. Surface temperature depends on tubing layout, fluid temperature, and panel construction. The software searches for the fluid temperature that results in panel output equaling room load. Many combinations of entering fluid temperature and flow rate will produce the same average fluid temperature (and thus the same output). All runs attached to the same manifold have the same entering temperature; the software groups the runs onto manifolds according to feasible entering temperatures and then determines flow rates, pressure drops, and balance value settings.

PARTS LISTS

Our applications include modules that prepare parts lists for the HVAC system. Parts lists are an example of specialized output that has proven extremely popular, since counting fittings is tedious and error prone. In addition, parts list data can be aggregated to produce other common reports, such as proposals.

The core problem associated with assembling parts lists is manufacturer dependency. Various manufacturers offer different assemblies that provide the same function. In many cases, the same manufacturer offers several alternatives. Each assembly consists of one or more parts, each of which must be identified by part number on the final list. Furthermore, many parts are available only in certain package sizes.

We have addressed this problem by distinguishing generic parts and actual parts. A generic part number designates an assembly; generic part numbers are related to one or more actual part numbers with a manufacturer-specific table. Some rules dealing with alternative assemblies are currently implemented in code. We plan to generalize these capabilities so all situations can be handled by external rules.

Special cases must be handled as well. An example is allocation of tubing lengths required by radiant panels to a set of rolls available in various fixed lengths. Doing this with minimal waste is the "cutting stock problem" and is NP-hard. We devised an algorithm that always finds a satisfactory if not optimal arrangement and avoids long run times.

IMPLEMENTATION ISSUES

Our applications evolved from prior stand-alone programs that performed load calculations, duct sizing, operating cost comparison, and so forth. We explored several implementation strategies in an effort to integrate these modules. Early experiments established that automation schemes such as OLE introduced excessive overhead and essentially eliminated the interactive nature of the applications.

The advent of the 32 bit Windows programming environment provided sufficient address space to simply combine multiple modules into a single application. Data structures have been harmonized as required. Because all modules are statically linked together, any required data exchange, in any direction, is extremely efficient.

To achieve the desired interactive behavior, we use the Document/View model provided by MFC. We avoid modal screens and use conventional Windows message-based programming techniques. Any

number of views can be simultaneously active and there is no rigid navigation structure – the focus moves among views with simple mouse clicks. When data is changed on any view, results are recalculated as required and all views are updated. We have found that on current Pentium-class PCs, the overhead associated with display updating is minimal.

Calculation time, on the other hand, is the rate-limiting step for interactive responsiveness. In the commercial load calculation, we use a graph-based dependency analysis that omits unnecessary calculation steps in response to numerical user input. However, we have found that as the interactivity becomes more sophisticated, the number of entry points to the calculations increases and it is difficult to reliably identify what data has changed. For example, the graphical user interface allows selection and dragging of multiple non-identical objects, possibly between spaces.

Simultaneous automatic design combined with retention of all user changes requires special attention. The automated design process is confounded by many non-linearities. In general, any change to a building, even an "insignificant" one, can result in a substantially altered HVAC system.

For example, consider the duct design process described above. A small change in room load can require the addition or elimination of a register. A room load change might result from a modification in overall building volume (due to distribution of infiltration loads among rooms), so a small change can alter duct layout essentially anywhere. The application must redesign at each calculation step, but reuse pre-existing registers and duct runs if they are still required – thus retaining user changes.

OPTIMIZATION

Integrated HVAC design software offers an obvious framework for optimization. Given the project data maintained in our applications, it is in principle possible to minimize construction costs, operating costs, greenhouse gas emissions, or some life-cycle combination of all three. There are many exciting possibilities that we intend to pursue.

In our development efforts to date, we have concentrated on achieving the essential level of function and responsiveness. For the reasons discussed elsewhere, we have relied on pre-existing, sanctioned design methods. We have emphasized the efficient and correct application of those procedures, allowing the user to quickly produce high quality designs. Now that these first steps have been successfully completed, optimization has become a real possibility. Coincidentally, commonly available desktop systems now have the power necessary to perform some computationally intensive optimization procedures on an interactive basis. Extremely ambitious calculations from today's point of view, such as multiple full-year simulations, will be practical in a very few years.

ULTIMATE LIMITS

Our experiences with implementing HVAC design software yield some observations about the factors that limit attainment of the ultimate goal of automatic, optimized, and complete designs.

The most obvious limitation is computing power. Sophisticated analysis and optimization procedures will require significant additional power. Perhaps two orders of magnitude will be needed to perform global optimization in a practical (that is, interactive) fashion. However, PC system power has increased more than two orders of magnitude since introduction and it is expected that this rate of improvement will continue. Thus, we anticipate that in a few years, standard desktop systems will have sufficient power to handle very large HVAC design problems using advanced analysis algorithms and optimization.

Given available computing power, it seems only a matter of time that software will progress to the point where full, integrated 3-D models will exist for essentially all building construction projects. Interoperability standards will facilitate exchange of information among applications. HVAC design software will have available sufficient project detail to avoid errors such as intersection of duct runs and structural members. Communications technology and product description standards will provide ready access to detailed component information, allowing sophisticated software-based equipment selection. User correction and refinement will become less and less necessary. All-in-all, we do not find it difficult to imagine an HVAC design robot that can lay out, size, and specify high quality systems starting from a building design and user preferences.

But what about elimination of user preferences? Is it unreasonable to expect that an automated designer should operate starting from only the building design? This is where we see a possible ultimate limit. It may prove impractical, perhaps impossible, to reduce the pertinent human preferences to cost functions that would allow software to reliably make acceptable design decisions – some fair number of human preferences are probably irrational by most objective standards. We also speculate that there is not a sharp optimum design for many problems. Several (or many) alternatives would provide good

performance within the uncertainty of available data. It may not be possible to build software that can make final choices that people do not challenge.

There are two other areas where we see possible limits to design software capabilities: imagination and innovation. In the area of imagination, will software be able to recognize design opportunities and identify novel solutions, without human hints? The answer to this is certainly yes, assuming the software is working from a list of possible solutions that includes the novel ones. Given this brute-force approach, software will at least look like it has imagination. However, we question whether software will be able to find creative solutions as efficiently as people do.

Innovation is another matter. When a new component becomes available, human designers can immediately see how to exploit it in systems. Similarly, people recognize missing capabilities, leading them to refine or invent components. It is a big step from a robotic designer that cleverly combines and sizes known components to software that announces "I really could use a 3 inlet mixing box". The latter capability clearly requires a level of artificial intelligence that will probably not arrive in the near future.

In a different realm, automated design software may be limited by social and legal constraints. Design software with the capabilities we envision could replace most of the functions of the HVAC engineer. To date, legal responsibility for designs remains with licensed professionals – it will probably be some time before software is allowed to stamp drawings. If people are obligated to take responsibility for designs produced by automated systems, perhaps the capabilities of those systems will be deliberately limited in order to facilitate human verification.

CONCLUSIONS

Our experiences developing and marketing HVAC design automation software has shown that interactive, graphically-based, integrated applications can provide significant productivity and design quality improvements. Our applications rely on integrated implementations of standard, sanctioned design methods and common practice. The key innovation is reducing design procedures to algorithms, allowing automatic generation of system designs.

Now that integration is complete and operating, we can undertake inclusion of additional design procedures, more advanced analysis methods, and optimization techniques.

Given the rapid increase in available computer power, software capability, interoperability standards, and communication networks, we see no major obstacles to development of automated HVAC design software capable of replacing most of the functions of current design engineers. We expect development of software that offers true design innovations to be more difficult. Progress in that area will be linked to advances in artificial intelligence research.

REFERENCES

ACCA, 1986. *Load Calculation for Residential Winter and Summer Air Conditioning -- Manual J 7th Edition.* Air Conditioning Contractors of America, Washington, DC

ACCA, 1990. *Commercial Low Pressure, Low Velocity Duct System Design – Manual Q.* Air Conditioning Contractors of America, Washington, DC

ACCA, 1995. *Residential Duct Systems – Manual D.* Air Conditioning Contractors of America, Washington, DC

ASHRAE, 1977. *Cooling and Heating Load Calculation Manual.* GRP-158, William Rudoy, Project Director. ASHRAE, Atlanta, GA.

ASHRAE, 1989. *Handbook of Fundamentals.* American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., Atlanta, GA.

ASHRAE, 1996. *HVAC Systems and Equipment.* American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., Atlanta, GA.

ASHRAE, 1997. *Handbook of Fundamentals.* American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., Atlanta, GA.

Bose, J. E., Parker, J. D., and McQuiston, F. C., 1985. *Design/Data Manual for Closed-Loop Ground Coupled Heat Pump Systems.* American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc., Atlanta, GA.

CAN/CSA-F280-M90, 1990. *Determining the Required Capacity of Residential Space Heating and Cooling Appliances.* Canadian Standards Association, Rexdale (Toronto), ON, Canada.

Domanski, P. and Kelly, G. E., 1980. Estimating the Heating Seasonal Operating Cost of Residential Hybrid Heat Pump Systems, Including Units Retrofitted to Oil, Gas, and Electric Furnaces. NBSIR 80-2090. National Institute of Standards and Technology, Washington, DC.

HRAI, 1996. *Residential Heat Loss and Gain Calculations, Student Reference Guide*. Heating, Refrigerating and Air-Conditioning Institute of Canada. Mississauga, ON, Canada.

HRAI, 1998. *Residential Air System Design Guide*. Heating, Refrigerating and Air-Conditioning Institute of Canada. Mississauga, ON, Canada.

Kelly, G. E. and Parken, W. H., 1978. Method of Testing, Rating, and Estimating the Seasonal Performance of Central Air-Conditioners and Heat Pumps Operating in the Cooling Mode. NBSIR 78-1271. National Institute of Standards and Technology, Washington, DC.

Oklahoma State University, 1988. *Closed-Loop/Ground Source Heat Pump Systems, Installation Guide*. Stillwater: International Ground Source Heat Pump Association.

SMACNA, 1990. *HVAC Systems Duct Design*. Sheet Metal And Air Conditioning Contractors' National Association, Inc.