To the Graduate Council :

I am submitting herewith a thesis written by Bharati E. Jog entitled "*CAAD*: A Computer-Aided Architectural Design System." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Maria Zemankova

Maria Zemankova, Major professor

We have read this thesis
and recommend its acceptance

Joan R. S. Blair

J. R. B. Corbett

Accepted for the council :

Lawrence Winkel

Vice Provost and
Dean of the Graduate School

CAAD: A COMPUTER-AIDED ARCHITECTURAL DESIGN SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Bharati E. Jog

March 1988

To my father, Dr. E. S. Jog.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Maria Zemankova for her guidance in this thesis. Her encouragement and assistance throughout my work has been invaluable. Without her help and support, this thesis would not be nearly as complete. I would like to thank Dr. Jean Blair and Dr. Robin Cockett for being on my committee. They provided ideas, support and criticism in a highly effective manner.

I would also like to thank Professor Carl Bovil of the Department of Architecture for making the "Energy Graphics" manual available for this thesis. He answered all my questions regarding energy analysis of buildings.

My special thanks to Subhendu Ghatak and Sandeep Kumar for going through this thesis thoroughly. They provided valuable suggestions and comments on my work. Thanks to Bill Rosener for his help in proofreading my report.

Finally, I would like to thank my parents and my sister, Dr. Vidya Jog for their support and encouragement.

ABSTRACT

Computer-Aided *Drafting* in architecture is the process of creating drawings with the help of a computer. This process does not involve any decision making which relates to designing a building. However, Computer-Aided *Design* (CAD) in architecture includes creating a building design using a Computer-Aided *Drafting* system, using the information about the building from the drawing files, evaluating the building design and giving directives for improving the design.

Computer-Aided *Drafting* systems are being used in the architectural profession and in academic environments. However, there are no Computer-Aided *Design* (CAD) systems available to aid architects in the design process. In this thesis, *Computer-Aided Architectural Design (CAAD)*, an integrated CAD system for architects is developed. This system will be helpful in the design process and should not affect the architect's originality. A prototype model of this system has been developed on an IBM PC AT running the DOS operating system.

Contents

1 Introduction **1**

1.1 Architectural Design 2

1.2 Literature Review 4

1.3 Purpose of The Project 5

2 The System **7**

2.1 *CAAD*: Overall Structure 7

2.1.1 Computer-Aided *Drafting* Tool 10

2.1.2 Evaluators and Analyzers 14

2.2 Implementation Details 15

2.2.1 System Requirements 15

2.2.2 System Development 19

2.3 Summary 26

3 Energy Expert **28**

3.1 Terminology 29

3.2 Methodology 30
3.3 Data Collection 32
3.3.1 Site Information 32
3.3.2 Building Design Information 36
3.4 Calculation Details 37
3.4.1 Internal Heat Gain 39
3.4.2 Solar Heat Gain 40
3.4.3 Envelope Heat Gain and Loss 41
3.4.4 Ventilation Heat Gain and Loss 43
3.4.5 Allowable Heat Gains From Envelope & Ventilation 44
3.4.6 Total Heat Gain and Loss 45
3.4.7 Cost Estimations 45
3.5 Expert Advice 47
3.5.1 Rule-Based Expert Systems 47
3.5.2 Advice in *Energy Expert* 49
3.6 Graphical Output 57
3.7 Summary 59

4 CIC **61**

4.1 CALPAS3: Description 62
4.1.1 Reporting Features 63

4.1.2 Basics of CALPAS3 64

4.2 Implementation Details 66

4.2.1 The Site Database 68

4.2.2 User Interface and Warning Messages 69

4.2.3 Extraction of Information from AutoCAD Drawings 74

4.2.4 Use of Extracted Information to Create new Information . 76

4.3 Summary 78

5 Conclusions **79**

5.1 Summary of *CAAD* 79

5.2 Future Directions 81

Bibliography **84**

Appendices **90**

Appendix A 91

Appendix B 104

Vita **107**

List of Figures

1.1 The Architectural Design Process 2

2.1 Schematic Design Stage 9

2.2 Final Design Stage 11

2.3 Command *bed* 12

2.4 Command *bed* at Level 1 13

2.5 Concept of an Interface 18

2.6 *CAAD*: The Prototype System 20

2.7 Entities for Plan Drawings 23

2.8 Entities for Elevation Drawings 24

2.9 Extraction of Information at Schematic Design Stage 25

2.10 Fire-Safety Expert 27

3.1 Graphical Output: The Three Windows 58

3.2 Graphical Output 60

4.1 An Interface between CALPAS3 and AutoCAD 67

4.2 Polygon *ABCDEFA* 77

Chapter 1

Introduction

Although architects have been reluctant to change their manual methods of drafting and designing, they have begun to use computers in architectural projects. It is now not uncommon to see architects using Computer-Aided *Drafting* systems even in small firms. However, there is no Computer-Aided *Design* system for architects on the market. The reason that no such systems exist is that architects have had a fear that such systems might hamper their creativity.

Computer-Aided *Drafting* is different from Computer-Aided *Design*. Computer-Aided *Drafting* is a process of creating drawings with the help of a computer. During the *drafting* process decisions about the drawing itself, such as thickness of lines, scale, format and drawing layout are made. However, the process does not involve any decision making which relates to designing the building. Computer-Aided *Design* (CAD) in architecture, involves creating a building design using a Computer-Aided Architectural *Drafting* system as well as extracting information about the building design, using this extracted informaion to evaluate the building design, and issuing directives (advice) for modifying the design for better performance.

The goal of this thesis is to develop an integrated Computer-Aided *Design* system and implement it at a prototype level. It is hoped that the prototype model can later be developed into a marketable system.

Section 1.1 describes the architectural design process. The literature review is given in Section 1.2. The purpose of the project is presented in Section 1.3.

1.1 Architectural Design

Typically, an architectural design project is divided into three major phases: pre-design, design and construction-supervision (Figure 1.1). A brief description of each of these phases is presented in the following paragraphs.

Figure 1.1: The Architectural Design Process

1. **Pre-design Phase:** In this phase a feasibility study and preliminary research studies are carried out. Requirements for the building design, such as space allocation, budget and tentative deadlines, are finalized. A site analysis which includes the analysis of soil structure, slopes, cuts and fills, views, drainage lines and accessibility is made. Based on these studies a schematic site plan is prepared. A bubble diagram or a circulation[1] diagram of building spaces is generated.

[1] In this context, circulation refers to the movement of occupants.

2. **Design Phase:** Based on the studies done in the pre-design phase, a building design is developed in the design phase. The design phase is further subdivided into schematic design and final design stages.

   The schematic design is iterative. At this stage, a sketch design is created and an approximate evaluation of the sketched building for its energy performance, circulation, cost, aesthetics, and structural soundness is carried out. Based on this evaluation, the sketch design is modified or an altogether different alternative sketch design is created. During this process several sketch alternatives of the building design may be created. After considering the different trade-offs, the most suitable design alternative is selected as the basic conceptual design.

   At the final design stage, the basic conceptual design is developed into a detailed design. Design decisions about structural details, and building materials are made. At this stage a detailed and accurate evaluation of the design for fire-safety requirements, energy consumption and structural soundness is carried out. If the results are unsatisfactory, the design is modified accordingly. The process continues until the evaluation yields satisfactory results. Once a detailed design is developed: working drawings are made; specifications are prepared accordingly; a detailed cost estimation is done; and the contractors and subcontractors are appointed.

3. **Construction-Supervision Phase:** This phase includes the preparation of a construction-schedule. While preparing this schedule, factors such as the expected date of completion of the project, finance, etc. are taken into consideration. On the whole, the architect manages the project and monitors the contractors' operations at this stage.

In this thesis, an integrated Computer-Aided *Design* system is created for the design phase of the architectural design process.

1.2 Literature Review

An integrated CAD system in architecture which aids throughout the architectural design phase has not been developed yet. However, there have been some efforts at a few universities and vendors to develop computer systems that aid the architects in part of the design process. This section lists some of these programs.

At the University of Michigan ARCH:SKETCH, a sketching and drafting program, has been developed in which a database of building information can be maintained [23]. The program runs on the Amdahl 5860. In the same university an interactive program, ARCH:FIRESAFETY has been developed which uses the databases stored by ARCH:SKETCH. The program is designed to check the fire-safety requirements of a building at the initial design stage of the project [24]. ARCH:FIRESAFETY works only on Apollo work stations. The program made it possible to carry out many of the routine calculations required by fire and life safety codes at a much faster rate.

At the University of California at Los Angeles Solar5, a program in Fortran, has been developed for the U.S. Department of Energy [20]. The program was developed to help architects understand the implications of alternative building forms at the schematic design stage. Using the user given information about a building design, Solar5 analyzes the design for its energy performance. The program can also create a basic geometry of an energy efficient building based on the building type, weather station, area in square feet and the number of stories of the building. The user needs to provide this data. This computer generated

conceptual building form is rather crude and is a long way from being inspiring architecture [20]. The program runs on the IBM PC and compatibles.

There are some sophisticated programs available on the market in the area of energy analysis and structural design . An example of such a program is CALPAS3, a program developed by the Berkeley Solar Group for detailed energy analysis [3]. The program runs on IBM PC and compatibles. CALPAS3 and most of these programs require the user to go through a lengthy and complicated data input stage.

All these existing system are important as they transfer the tedious task of calculations to a computer. However, all these systems aid only in different parts of an architectural design process, are not compatible, and require the user to go through a lengthy data input session for each separate system. The need of the hour is to develop a single integrated system which will be able to aid the architect in the entire design phase.

1.3 Purpose of The Project

The main idea of the project is to design an integrated Computer-Aided *Design* system for architects and to create a prototype model to examine the feasibility of the new system. This system will be utilized in the design phase of an architectural project. Pre-design and construction-supervision phases are not considered. The proposed system should be useful at both stages (schematic and final) of the design phase. Such a CAD system in architecture needs to be affordable even for small architectural firms. This aspect is considered while implementing the system. The modules which evaluate the building performance at the schematic design stage are rule-based expert systems. These modules analyze the building performance.

Based on that performance they also give directives for the improvement in the building design. This shows that the principles of expert systems can be applied in the architectural knowledge domain. The system should be highly graphical, as architects usually understand graphical results better than numerical results.

CAAD (Computer-Aided Architectural *Design*), a prototype model is developed in this project. Because of the time constraint all the modules of the system have not been implemented. For some of the modules that are not developed, commercially available software is used. At a later stage the remaining modules can be developed with the current programs acting as guidelines.

In the next chapter we present the details of the system. Chapters 3 and 4 provide detailed descriptions of the main modules of the system. Chapter 5 presents the conclusions and provides directions for further work.

Chapter 2

The System

The architectural design process has been described in Chapter 1. The process involves three phases: pre-design, design, and construction-supervision. While developing this project, it is assumed that pre-design has been completed, and that all the requirements of spaces, budget and site have been finalized.

CAAD, the prototype design system has been developed for the design phase. Using the developed *CAAD* system, working drawings of the design can then be made. The third phase, construction-supervision is not considered in this system.

This chapter describes the overall system design and some of the implementation details. In the next section, the overall structure is presented. Section 2.2 describes the implementation details. Section 2.3 provides summary of this chapter.

2.1 *CAAD*: Overall Structure

As described in Section 1.1, the design phase is divided into two parts: schematic design and final design. *CAAD*, is designed to aid in both of these stages.

At the schematic design stage, a conceptual sketch design of a building is created by using a Computer-Aided Architectural *Drafting* tool. The information about the sketch is extracted and used in modules called *evaluators*. The *evaluators* carry out quick and approximate evaluations of the sketch design for its energy performance, fire-safety requirements, structural design and circulation. Throughout the reminder of this thesis, we will refer to these *evaluators* as *Energy Expert, Fire-Safety Expert, Structural Expert* and *Circulation Expert* respectively. Based on the evaluation, the *evaluators* give advice about possible modifictions in the design that would result in better performance. If the building performance is unsatisfactory, the design sketch can be modified or an altogether new sketch design can be created. On the other hand, if the performace is satisfactory, the sketch design is passed on to the final design stage. The process for the schematic design is illustrated in Figure 2.1.

At the final design stage, the selected schematic design is developed into a detailed design using the Computer-Aided Architectural *Drafting* tool. The overall design approach in this stage is similar to that in the schematic design stage. But, unlike schematic design stage, this stage involves designing a building in much more detail and hence is a more time consuming process. At this stage, all the building elements (e.g. walls, windows, doors) are defined in detail with regard to their sizes and types. The design is then accuratetely evaluated for its energy consumption, fire-safety, structural and circulation performance. Here onwards, we will refer to the modules which carry out accurate and detailed building evaluation at the final design stage as *analyzers*. If the results of the evaluation are unsatisfactory, the design is modified. If the results are satisfactory then working drawings are made

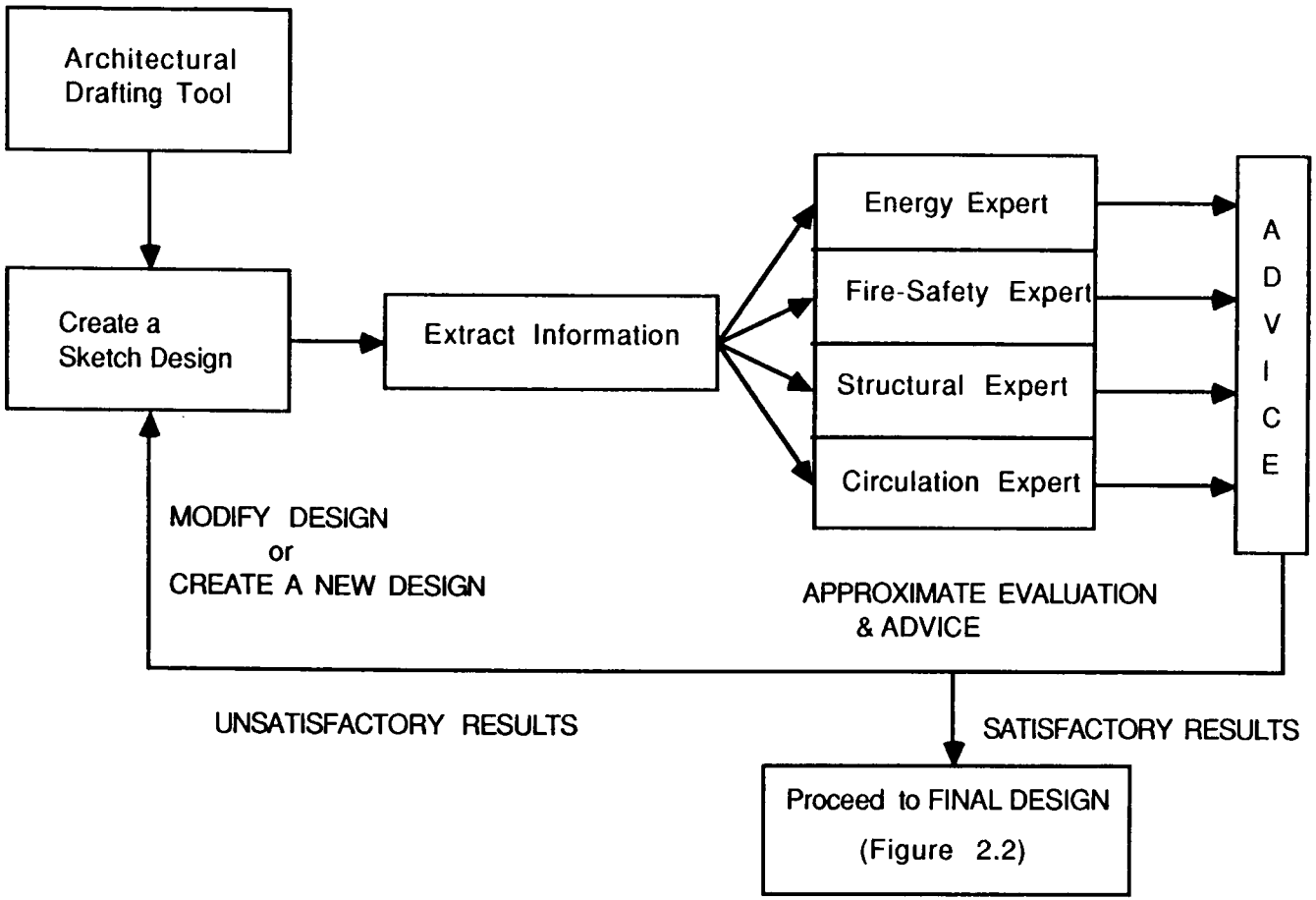

Figure 2.1: Schematic Design Stage

using the selected Computer-Aided Architectural *Drafting* tool. The process is shown in Figure 2.2.

Now we will present the ideal structure of the important modules of the integrated Computer-Aided Architectural *Design* system. Section 2.1.1 explains the structure of the Computer-Aided *Drafting* tool. The structure of the *evaluators* and *analyzers* is presented in Section 2.1.2.

2.1.1 Computer-Aided *Drafting* Tool

Sophisticated drafting programs such as AutoCAD [1], MicroCAD [19], are available on the market. These packages have been developed mainly for engineering drafting. Thus, the basic drawing entities are points, lines, circles, and arcs. These packages provide graphic aids to copy and transform objects. Typical transformations include scaling an object, rotating an object along an axis or a point and translating an object at a distance. To use a drafting package to draw an architectural design, it is more appropriate to have architectural specific entities such as walls, windows, doors, staircases, and furniture. Using these entities, it is easier to create architectural drawings than to use meaningless entities such as lines and points. This, inturn, makes it easier to extract information about these architectural objects.

Now we will present the data structure of each of the entities for *CAAD*. Each of these entities can be a tree structure. Let us take an example of the entity *bed*. A tree structure for *bed* is as shown in Figure 2.3. At the first level of the tree, the size and type of the bed are not taken into consideration. For example, when the approximate width, length and base points are given by the user at level

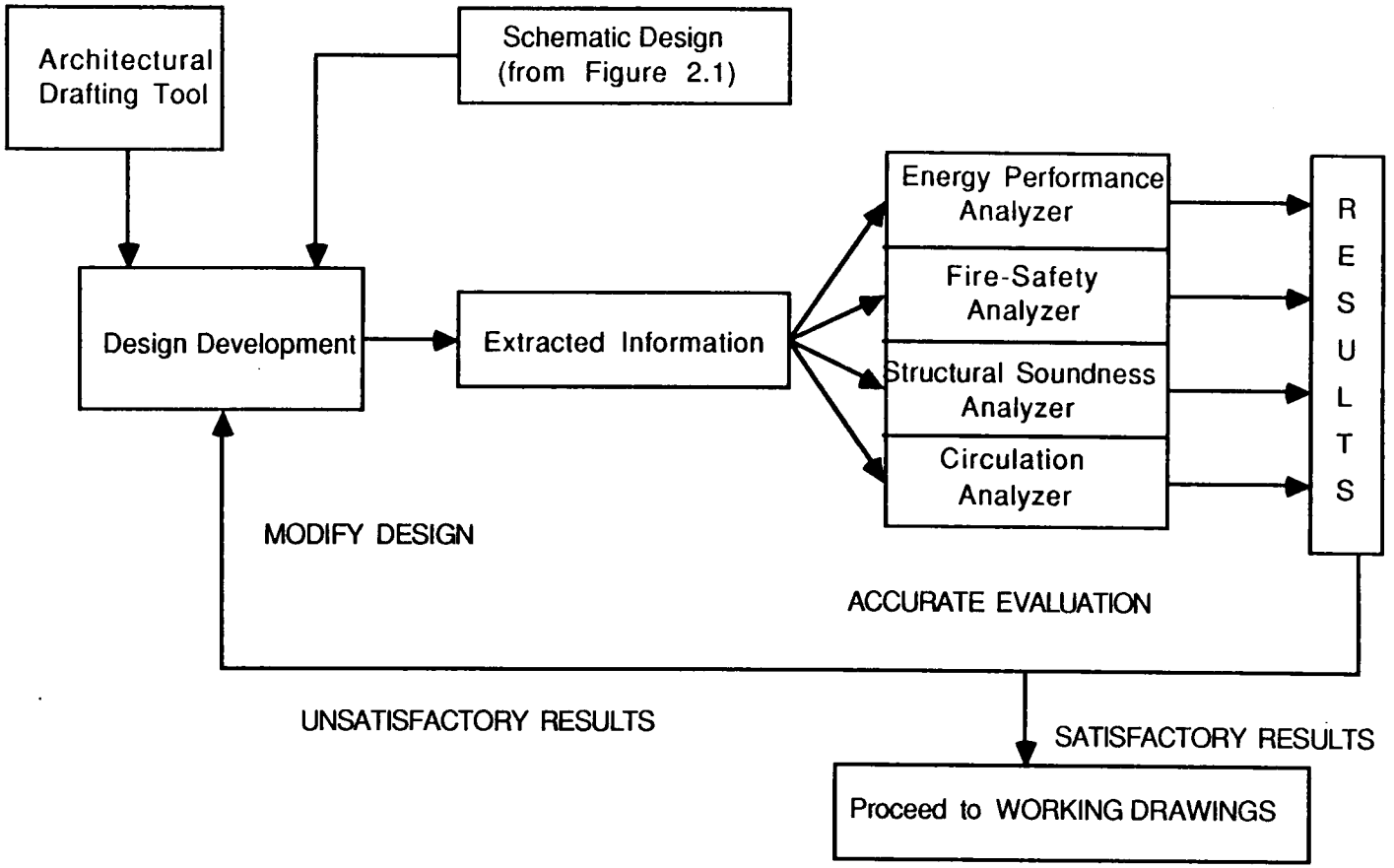

Figure 2.2: Final Design Stage

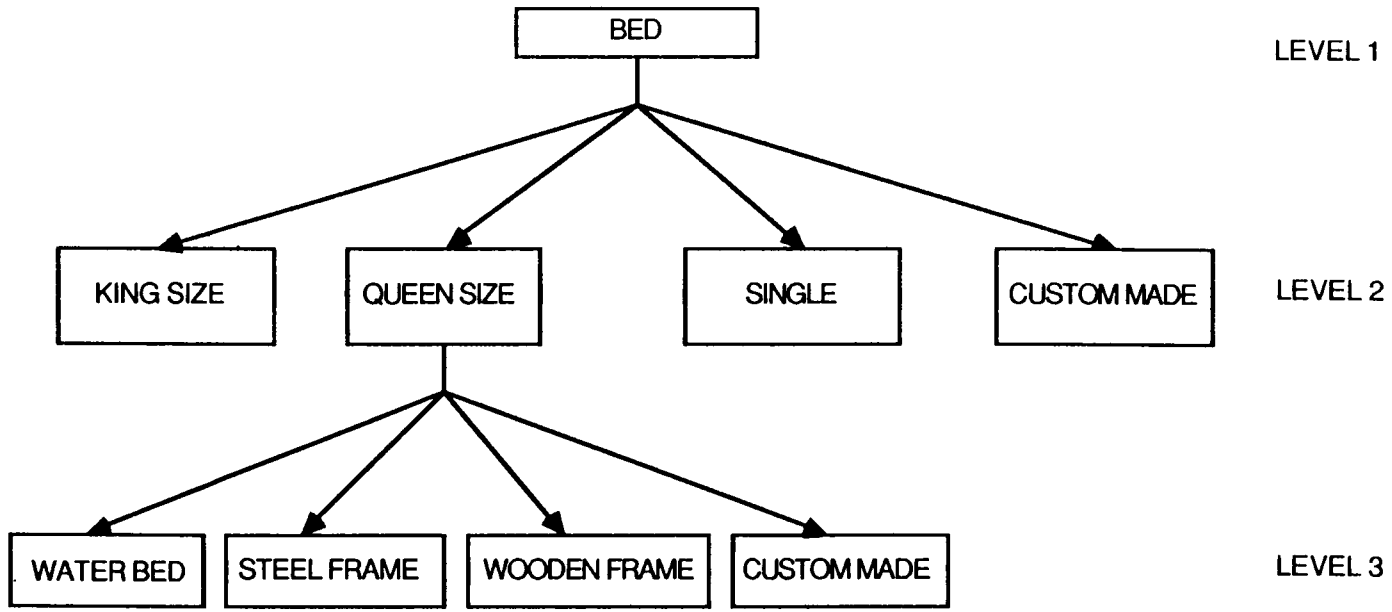

Figure 2.3: Command *bed*

1, the bed can be drawn as in Figure 2.4. This level can be used for creating quick sketches at the schematic design stage. Levels 2 and 3 can be used at the final design stage to create a detailed design. At these levels the entity is defined in greater detail. For example, at level 2 *bed* is defined as *queen size, king size, single* or *custom made*. When a bed is drawn using the corresponding commands at level 2, the exact sizes and proportions are drawn. The third level of the *bed* structure defines its type.

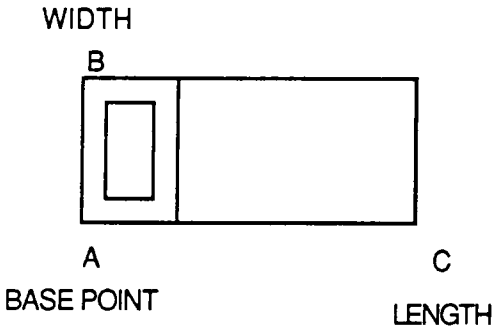

Figure 2.4: Command *bed* at Level 1

Ideally, there should be a method for attaching information to an entity. If the entity *bed* is used to draw a queen size bed, there should be a way of extracting information such as size of the bed, and the location of the bed on the plan. Here onwards we will refer to these architectural entities as *drawing database*. Creating such commands for all architectural objects with all the possible standard sizes and types is a large project in itself. The author does not know of such a system available on the market.

While implementing the Compute-Aided *Drafting* tool in this thesis, routines to draw few architectural entities at the first level of the tree are developed. A routine to extract information about the entities is also developed in the prototype.

2.1.2 Evaluators and Analyzers

It should be noted that some criteria for evaluating designs are not measurable and thus not computable. An example of such a criterion is aesthetics. Such criteria can be subjectively evaluated by examining a design using 3D modeling. Typical examples of measurable criteria (and thus computable criteria) in a design are its structural feasibility, circulation efficiency, energy and fire-safety performance. All these evaluations are based on a series of rules and building bylaws. While developing *evaluators* and *analyzers* in *CAAD*, only these measurable evaluation criteria are considered.

As stated earlier, at the schematic design stage, the *evaluators* carry out approximate and quick evaluation of the sketch design. Rule-based expert systems can be developed as *evaluators* at the schematic design stage. Ideally, these *evaluators* should get the information about the design from the *drawing database* and from the user. Additional information about the site can be fetched from a database of sites. Such a database needs to be developed in *CAAD*. These expert systems should quickly evaluate the design for meeting the selected criteria and advise the user of possible modifications in his design to improve its performance. Since rule-based expert systems have been proven to be efficient in providing analyses based on a set of rules, this strategy has been adopted for the *evaluators*.

At the final design stage, the evaluation carried out by the *analyzers* is detailed and accurate, and hence the process is lengthy. Sophisticated programs are available for a detailed structural and energy analysis. At the final design stage, instead of building large expert systems as the *analyzers*, these programs can be made operational on the selected machine and interfaces can be developed between the programs and *CAAD*. A detailed fire-safety check program can be

developed separately and interfaced with the integrated system.

2.2 Implementation Details

Before implementing *CAAD*, certain decisions about the system requirements had to be made. These decisions include the type of computer, the language tools, the drafting tools, and the evaluation tools to be used for this implementation. The next section gives a brief description of the decisions made to meet these requirements. The prototype developed in this thesis is discussed in Section 2.2.2.

2.2.1 System Requirements

Computer System: During the last decade researchers in universities, have tried to develop prototype programs for architects. These programs were developed either on main frames such as a VAX 8600 or workstations such as the Apollo. Most of the architectural firms and schools cannot afford these expensive systems. As a result, these programs either die or are used only at the particular university where they were developed. To overcome this problem and to make the new system more usable, a popular and inexpensive system was used so that even small architectural firms and schools might be able to afford the system. Considering these requirements, an IBM PC AT with an enhanced color display adapter and 640 K memory, running the DOS operating system was chosen to implement *CAAD*. The machine has been popular in architectural firms and schools.

Language Tools: In this prototype, different modules of *CAAD* were developed separately and then they were integrated by developing interfaces. Appropriate languages were chosen for the development of each separate module. A procedural language was selected for developing program modules which contain

many calculations and for the interfaces. A backtracking language was selected for the rule-based expert system modules.

The available *procedural language* on the IBM PC AT at the University of Tennessee is Turbo Pascal. Turbo Pascal supports Turtle graphics. There is also a graphics toolbox for Turbo Pascal. However, the toolbox was not available while developing the project. At a later stage, the toolbox can be used to enhance the user interface. Turbo Pascal supports color display and windows which make the system more user friendly and attractive for a naive user.

To develop the expert *evaluators* two available *backtracking languages*, Arity Prolog and Turbo Prolog, were considered. Turbo Prolog supports Turtle graphics, windows and color display. Hence the chosen tool was Turbo Prolog.

Computing the energy or structural performance of a building requires many calculations. If a procedural language is used for computing the energy and structural performance, and a backtracking language such as Prolog is used for the advisory task performance, it will cut down on execution time. Therefore, it will be useful to couple Turbo Prolog with Turbo Pascal to develop *evaluators*. However, currently Turbo Pascal programs cannot be linked with Turbo Prolog programs. Borland plans to release Turbo Pascal 4.0 in late 1987 which will allow such linking. At the next stage of the development of *CAAD*, Turbo Pascal can be used for calculating energy and structural performances of a building in the *evaluators*.

Drafting Tool: One of the major considerations in developing *CAAD* was that this integrated system should be affordable even for small firms. To achieve this goal, some of the already existing popular programs have been used in *CAAD*. The popularity of the drafting tool with practising architects and in schools was

a major consideration in the selection of a drafting tool. As a result, the drafting package should not be an additional cost to the architects and schools as they probably have copy of the package already. The other important criteria affecting the selection was the requirement that the package needs to support the development of new commands to draw architectural entities. Also information about the drawings should be extractable.

Of the available packages, AutoCAD [1], and MicroCAD [19], the former was chosen. AutoCAD is a Computer-Aided *Drafting* program developed by AutoDesk Inc. It is written in C and runs on an IBM PC (and PC compatibles) under MS-DOS or PC-DOS. AutoCAD is becoming an extremely popular tool. Surveys published at A/E/C System conference '84 at Los Angeles, show that more than 60% of architectural firms which use computers for drafting, were using AutoCAD. It has over 60,000 licensed users and has become a standard drafting package in the industry and educational institutions.

AutoCAD supports AutoLISP. AutoLISP follows most closely the rules and syntax of "Common Lisp". Using AutoLISP, the programmer can develop his or her own functions. These functions can be included in customized menus, tablet overlay menus or accessed from the keyboard. This ability makes AutoCAD an ideal drafting tool to develop *architectural entities* and to embed them in the system.

Information about the AutoCAD drawing entities can be stored in the standard *drawing interchange format (DXF)* in a file and hence can be easily extracted. For more details about the drawing interchange format refer to Section 4.2.3.

Interfaces: In the prototype system different modules have been interfaced by developing different programs. These programs can be called as *interfaces*. All the interfaces in the prototype are developed in Turbo Pascal. In the reminder of this section we will present the general concept of an interface.

Two or more independent programs/systems can be interfaced to produce an integrated system. An interface between two programs takes the output of one program and converts it into the input for the other. This process may not be simple. The programmer may need to handle problems such as compatibility and completeness. Often an auxiliary input is necessary. The auxiliary input can be obtained interactively from the user. The user interface needs to be friendly and simple. Another way to obtain the auxiliary input is to read user-created input files. Figure 2.5 illustrates the concept of an interface.

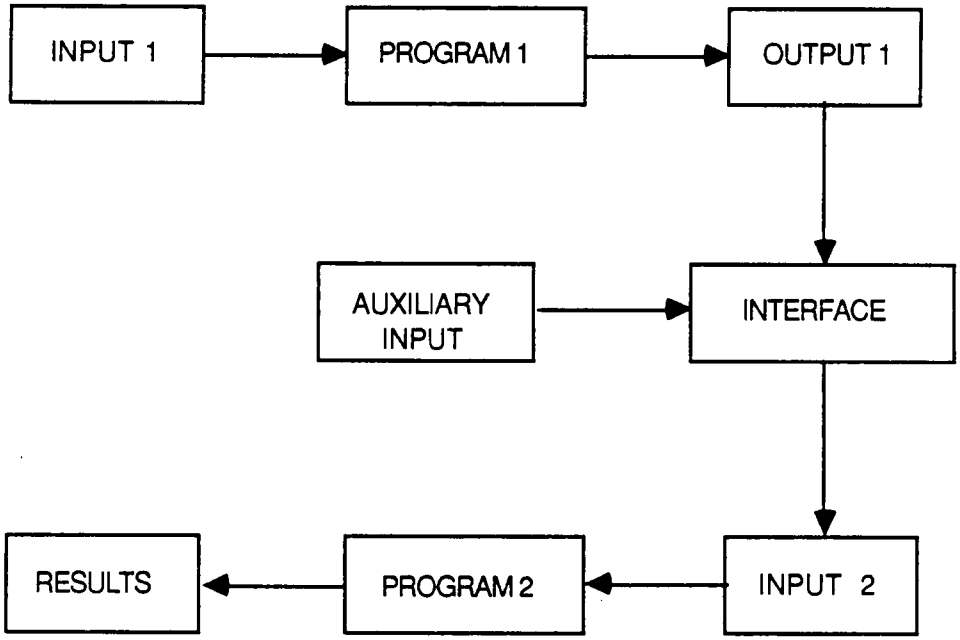

Figure 2.5: Concept of an Interface

Evaluating Tools: *Energy Expert* is implemented in this prototype as an example of *evaluators* at the schematic design stage. *Energy Expert* is based on the method described in *Energy Graphics*. *Energy Graphics* is a manual program developed for a quick and approximate energy analysis of a building at the schematic design stage.

As an example of *analyzers* at the final design stage, CALPAS3 was selected as an energy analysis program. CALPAS3 carries out accurate and detailed analysis of a building. The program takes a long execution time and requires the user to enter detailed information about the building in a specific format in a file. In this prototype, in order to aid the user in entering the data an interface, *CIC* is developed between the system and CALPAS3. Hence, the usage of CALPAS3, is transparent to the user.

2.2.2 System Development

The goal of the prototype system was to show that an integrated Computer-Aided Architectural *Design* system can be developed and to show that the principles of the rule-based expert systems can be implemented in evaluating a building design. In this thesis, all of the modules of the system have not been developed. Only the modules that were necessary to achive the goal have been developed. However, based on the structure of the developed modules, the remaining modules can easily be developed at a later stage. The modules of the developed prototype system are shown in Figure 2.6. These modules are interfaced to create a single integrated system. In this section, we will discuss the different implemented modules of *CAAD*.

The main implemented modules of this prototype system are *Energy Expert*

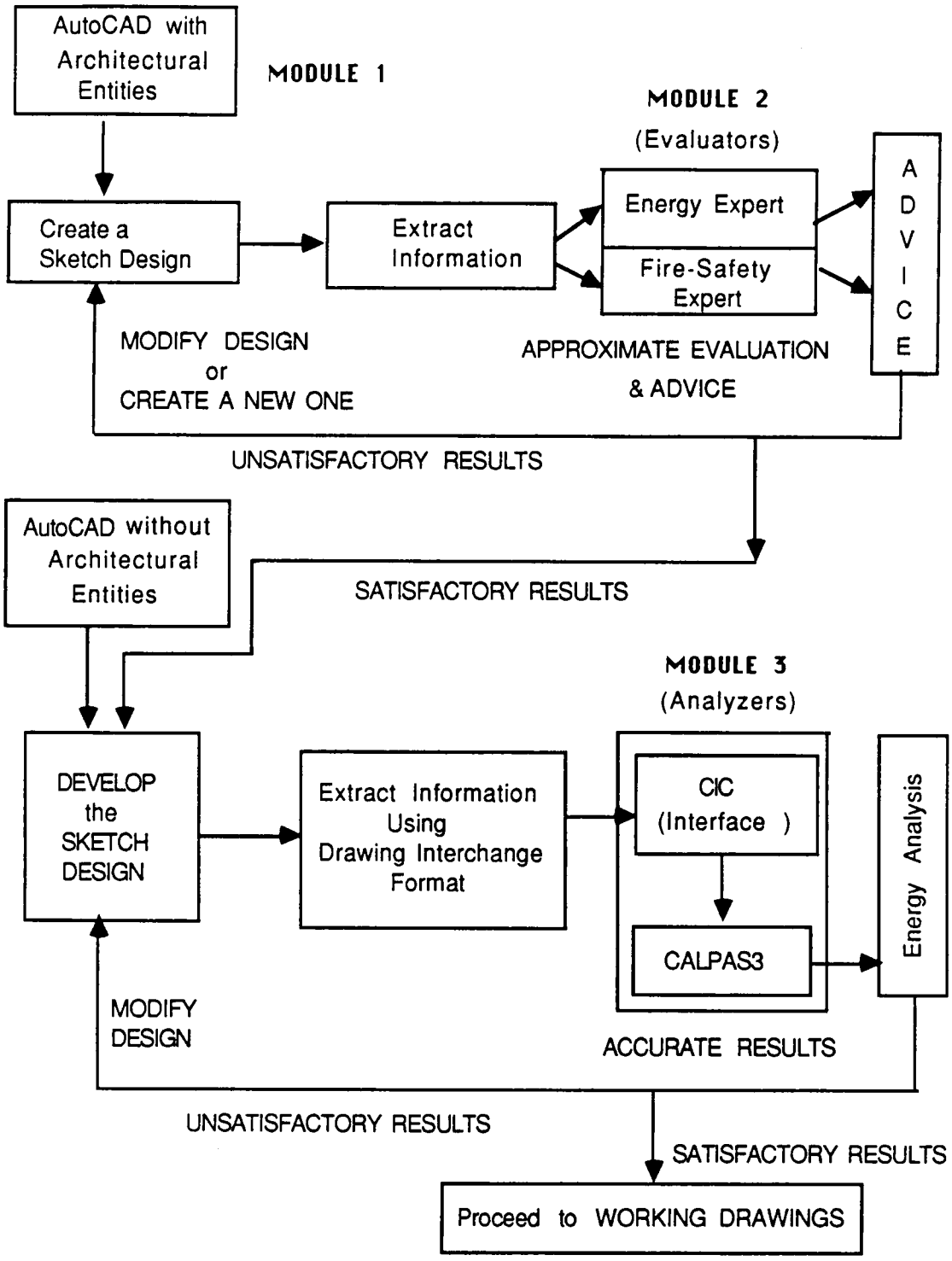

Figure 2.6: *CAAD*: The Prototype System

as an *evaluator* at the schematic design stage and the program *CIC* (Create Input file for CALPAS3) which interfaces CALPAS3 in the prototype. *CIC* together with CALPAS3 can be considered as an *analyzer* at the final design stage.

Energy Expert: As stated earlier, this program is based on the quick and approximate energy analysis method described in Energy Graphics [10]. *Energy Expert* collects some data from the AutoCAD drawing files and some data interactively from the user. Additional data about the site is collected from the database called *site.dat*. The output of *Energy Expert* is a graph of the energy performance of the building and two pie charts of the costs of heating and cooling during each of the four seasons. The output also consists of advice giving directions to modify the schematic design to make it more energy efficient. The details of the module are given in the Chapter 3.

CIC: CALPAS3 is a program, accepted by architects, to perform detailed energy analysis. The program requires the user to create an input data file of CALPAS3 commands and arguments in the CALPAS3 standard syntax. The CALPAS3 commands and arguments (Appendix A) are difficult to remember. As a result, the program is not used as frequently as it could be. To overcome this problem, a program is developed in *CAAD* to create an input file for CALPAS3. The program *CIC*, collects some information from the AutoCAD drawing files and some information interactively from the user. Additional information is collected from the database called *site1.dat*. A detailed description of *CIC* is given in Chapter 4.

The modules that were straightforward to implement are now explained:

AutoCAD with Architectural Entities: In this module architectural entities are developed so that AutoCAD along with these new entities can be used as an architectural drafting tool. These entities are created in AutoLISP as AutoCAD supports AutoLISP. In *CAAD*, AutoLISP routines have been written to draw basic architectural objects such as doors, windows, walls, and single flight and double flight staircases. Ideally, the user should be able to draw any architectural entity in plan[1] as well as in elevation[2]. In the prototype, the first three entities can be drawn in plan as well as in elevation (Figure 2.7 and 2.8). The staircases can be drawn only in plan. As stated earlier in Section 2.1.1, each of these entities is a tree structure. The tree structure has three levels. The entity tree structures are implemented only at the first level to be used at the schematic design stage (Figure 2.3). Appendix B illustrates the program module to draw a double flight staircase in plan. At the final design stage, AutoCAD without these architectural entities is used as a drafting tool.

Extraction of Information at Schematic Design Stage: The command called *information* has been developed which obtains the necessary information from the user about the drawing on the screen and stores it in a file called *data.dat*. The routine developing this command is written in AutoLISP. A program called *convert.inf* written in Turbo Pascal, converts the information stored in *data.dat* into a Turbo Prolog readable form and writes it to a file called *expert.dat* (Figure 2.9). The *evaluators* obtain data about the drawings from *expert.dat*.

[1] A *plan* is a view of an object as seen by looking down on the object from above.
[2] An *elevation* of an object is one of its side views.

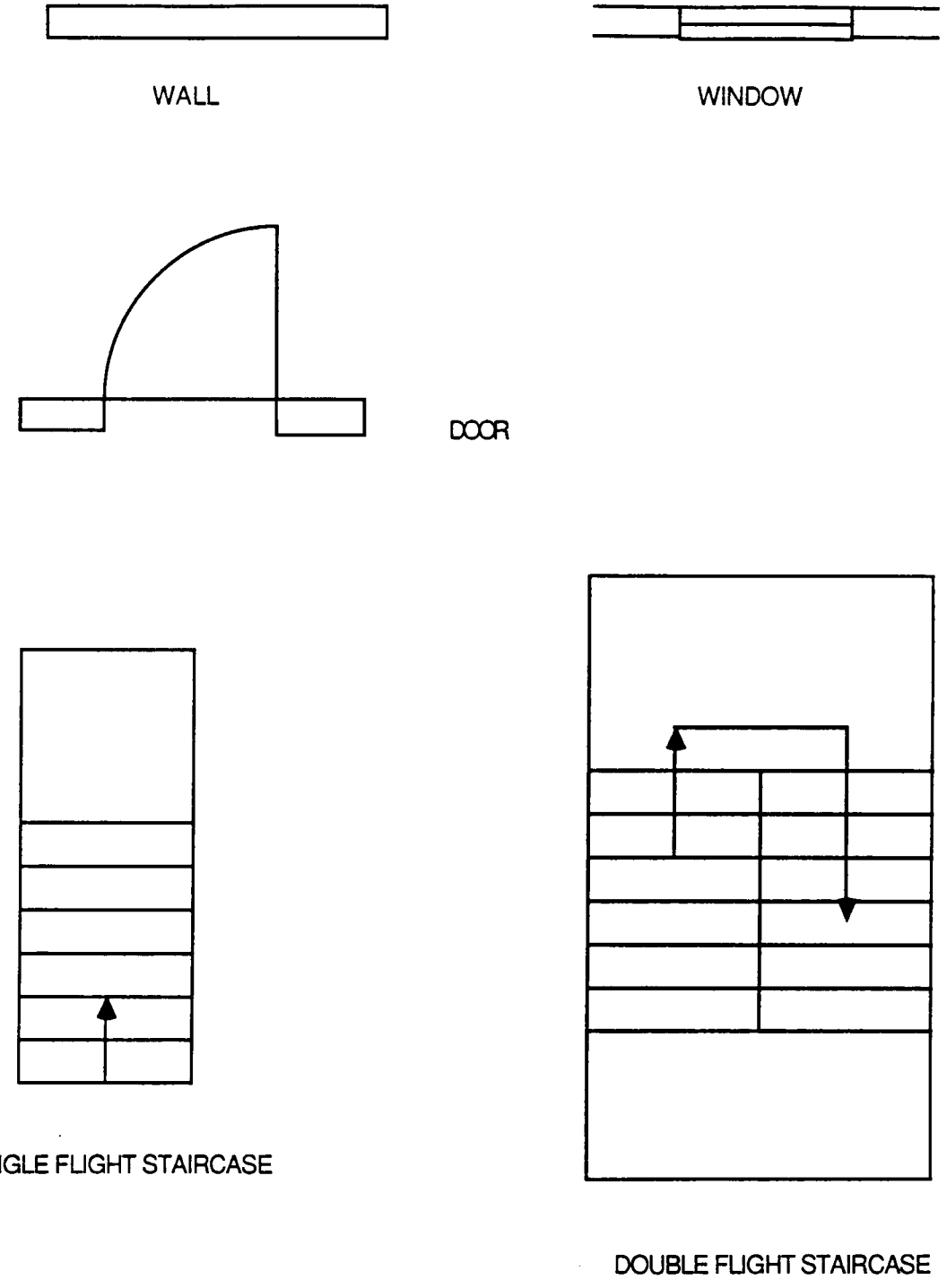

Figure 2.7: Entities for Plan Drawings

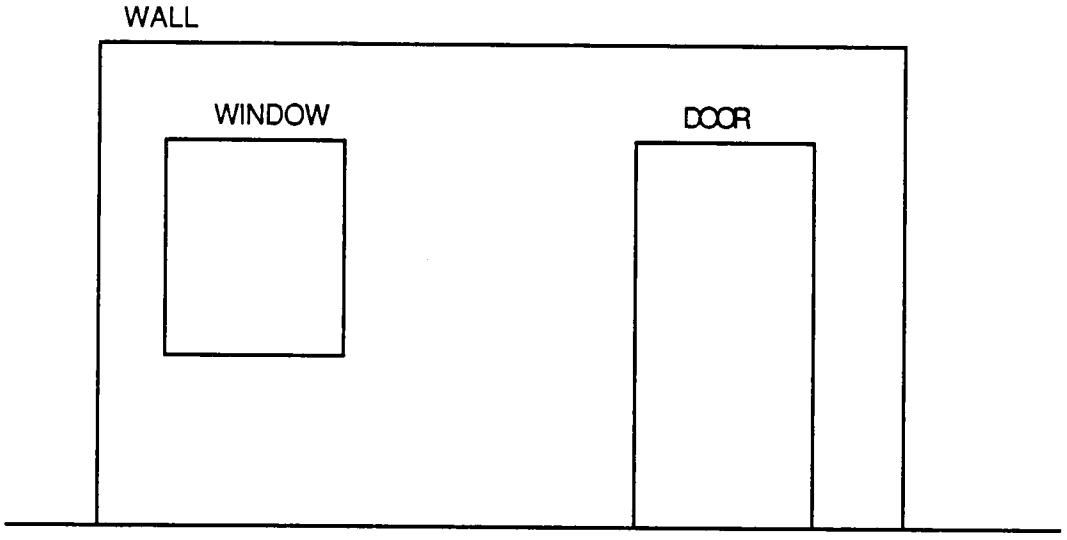

Figure 2.8: Entities for Elevation Drawings

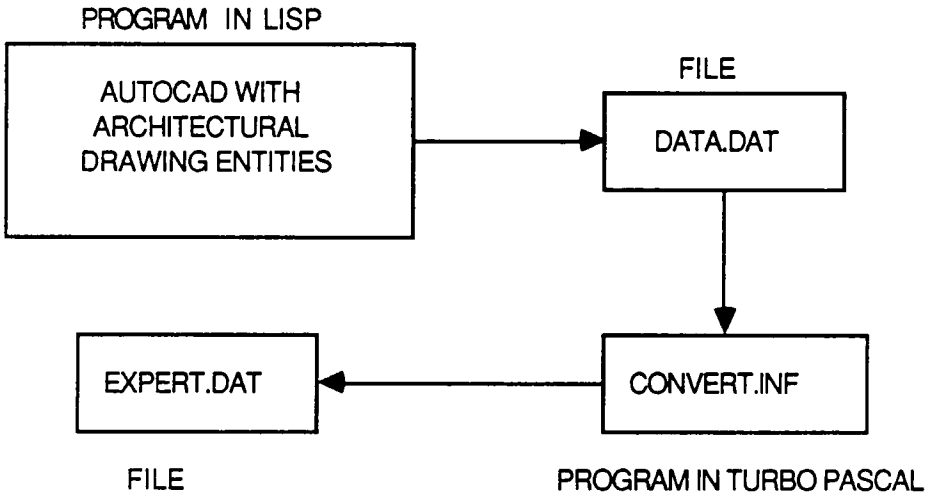

Figure 2.9: Extraction of Information at Schematic Design Stage

Site Databases: As mentioned earlier, information necessary for the *evaluators* about the site of a building is obtained from a site database. The database called *site.dat* contains information about 70 locations in the U.S. which is needed in *CAAD*. The information is in the form of predicates. The predicate is as follows:

places (city,state,latitude,longitude,time_zone,winter_highest_temp,

winter_lowest_temp,...)

The information can be read by the e*evaluators* at the schematic design stage as they have been developed in Turbo Prolog.

Fire-Safety Expert: A fire-safety expert system has been developed as an example of the *evaluators* at the schematic design stage. As stated earlier, we call this module the *Fire-Safety Expert* (Figure 2.10). Five basic rules have been developed to check the fire-safety of a building at the schematic design stage. The rules are based on NFPA 101 Life Safety Code [22]. The rules check the building for its floor heights, number of exits and staircase requirements. This module collects the building data from the file *expert.dat*. The remaining data is obtained interactively from the user. The output is a list of suggestions for modifications to the building, in textual form.

2.3 Summary

CAAD, a Computer-Aided Architectural *Design* system has been designed for application to the design phase of an architectural project. A prototype model of the system has been developed as part of this thesis. This chapter touched upon the general design of the prototype model. The chapter also described the simple modules of *CAAD*. In Chapters 3 and 4, the main modules are described in detail.

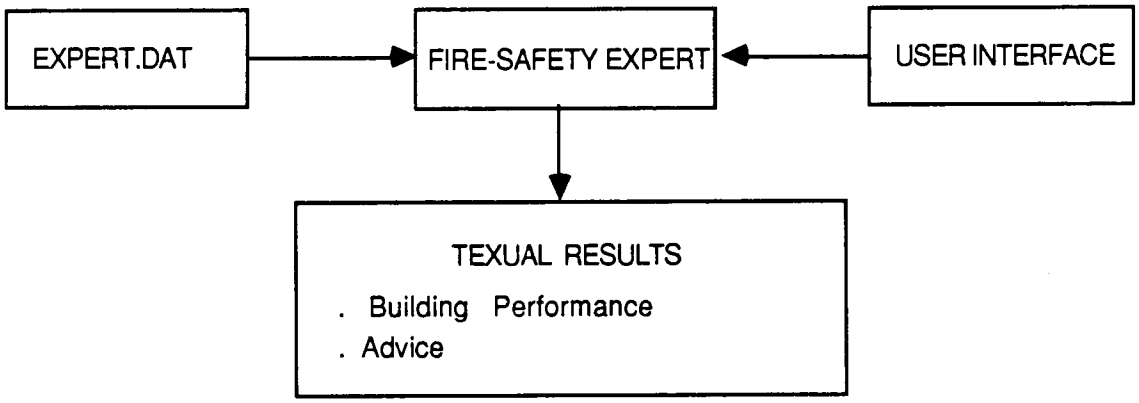

Figure 2.10: Fire-Safety Expert

Chapter 3

Energy Expert

The second module of the integrated Computer-Aided Architectural *Design* system is *Energy Expert*. *Energy Expert* collects data from the AutoCAD drawing files, interactively from the user, and from the site database. Using this data, it analyzes the energy performance of the building and gives directives to make the building more energy efficient. At the beginning of the program the user is prompted for the location of the building site. If the location of interest to the user is not in the site database then the user needs to enter the site data or use the nearest location in the database as the site. Using this data, *Energy Expert* calculates a quick and approximate energy performance of the building for each of the four seasons in a year and displays the energy performance graphically. It also gives directives to the user to improve the design. These aspects make the program ideal to use at the schematic design phase.

Energy Expert is based on the method described in *Energy Graphics* [10]. This tool was developed by Booz Allen & Hamilton Inc. to assist federal personnel in designing or procuring the design of new energy efficient federal buildings. The method of computing energy performance is expected to meet this goal and also

the New Building Energy Performance Standards prepared for the U.S. Department of Energy.

In the next section some important terminology is given. Section 3.2 presents the *Energy Expert* methodology. Details of data collection and representation are described in Section 3.3. Calculation details are explained in Section 3.4. Section 3.5 describes the rule-based expert systems and also presents the structure of the advisory system used to give advice to the designer to modify the design. The form of the graphical output is presented in Section 3.6. The last section is a summary of this chapter.

3.1 Terminology

Before going into details of *Energy Expert* it will be necessary to understand some terminology.

Heat, a form of energy, is measured in British Thermal Units *(Btu)*. A *Btu* is defined as the amount of heat necessary to raise the temperature of 1 pound of water by 1 degree Fahrenheit. Heat flow is measured in *Btu/hour* and denoted as *Btuh*.

The heat flow from air on one side of a body (such as a wall) to air on the other side, is *air-to-air resistance (R)*. The reciprocal of *air-to-air resistance (1/R)* is the *U value* of the body. Thermal transmission is usually measured in terms of *U values*.

Insolation is the amount of solar radiation reaching a body. It is measured in terms of $Btu/(h \cdot ft^2)$. *Insolation* is also referred to as *radiation*.

Air change is the replacement of a quantity of air in a volume within a given period of time. It is expressed in number of changes per hour. If a house has 1 air change per hour, all the air in the house will be replaced in a 1 hour period [16].

Each functional space can be allocated an allowable temperature range. The minimum temperature of the range is the *minimum allowable interior temperature* (T_{lo}). The maximum temperature of the range is the *maximum allowable interior temperature* (T_{hi}).

3.2 Methodology

In this section we will describe the methodology of *Energy Expert*. *Energy Graphics* assumes that the thermal performance of a building (heat gain and loss) invariably plays a major role in energy performance. In *Energy Expert* the thermal performace of a building is considered as the sole measure of energy performance. Hence the thermal performance is used to estimate overall energy performance of a building. There are four basic components of thermal performance:

1. Internal heat gain
2. Solar heat gain
3. Envelope heat gain and loss
4. Ventilation heat gain and loss

In *Energy Graphics*, these four factors are considered in the analysis of the energy performance of a building and in finding solutions to control it. We will briefly discuss all four components:

1. **Internal Heat Gain:** Heat generated by the sources inside a building is the heat gain for the building. Occupants, lighting, and equipment which generates heat are the major sources of internal heat generation. In most cases, the designer cannot control the occupancy and equipment but he can use artificial or natural lighting according to need.

2. **Solar Heat Gain:** A space receives heat from solar radiation. The amount of heat received depends on the amount of radiation reaching the space. This depends on the available insolation, the amount of exposed area in each direction and the transmission from the surface to space. Insolation depends on the geographical location of the site, so it cannot be changed to control the solar heat gain. However, the exposed areas and transmissions can be designed according to the need for heat gains and losses.

3. **Envelope Heat Gain and Loss:** Heat is transmitted through the exterior building envelope. Envelope heat gain or loss depends on the exterior and interior temperature difference, exposed areas of the exterior walls and roof, and the transmission through the building envelope. Exterior temperatures cannot be changed, but the architect can design the exposed areas and transmission features of a building to have the required energy consumption.

4. **Ventilation Heat Gain and Loss:** Heat gain and loss due to ventilation primarily depends on the difference between the exterior and interior temperature, and the ventilation rate. The temperature differences for a site are determined by the climate (environment) and are therefore uncontrollable. However, the ventilation rate is controllable.

Energy Expert collects data from the site database (site.dat), the AutoCAD files, and from the user. Using this data, an approximate energy analysis is calculated. This analysis is displayed graphically on the screen. Based on the user given fuel rate, an approximate cost of heating and cooling is calculated. The program also gives directives to improve the design. In the next section, we will discuss each of these points in detail.

3.3 Data Collection

To perform the energy analysis of a building, *Energy Graphics* (and consequently, *Energy Expert*) requires certain information about the buiding design and the environment of the site. At most times, the climate of the site is the principal factor in determining the amount of energy needed to maintain a required interior environment. For example, if the exterior temperature is less than 50° F throughout most of the year, then heating is required to maintain a comfortable interior temperature, whereas if the exterior temperature is often around 90° F then cooling the interior space is required. The methods for obtaining the two types of necessary information, are described in the following subsections.

3.3.1 Site Information

We have created a database for 70 cities in the U.S. which contains the necessary environmental information about those cities. There are five important characteristics of a site, namely the temperature of the city (dry bulb temperature), insolation (direct and diffused sunlight), earth temperature, wind patterns, and humidity (wet bulb temperature).

In the prototype system we have restricted ourselves to the two most significant determinants: the temperature of the site and its insolation. In the actual system, all the five determinants would need to be considered.

Theoretically, the climatic information should be detailed and available for all 365 days of the year. But carrying out lengthy computation on so much data would be too time consuming. Since we need a quick and approximate energy analysis, the year is divided into four distinct seasons, Winter, Spring, Summer, and Fall. Information about an average day in every season is collected. We have chosen average days in January, April, July and October. The information about the midnight, 4 a.m., 8 a.m., noon, 4 p.m. and 8 p.m. of each average day is stored. Thus, data for the seven hours in each season is maintained. The database contains information about the temperature and insolation for the selected seven hours of each season of a year for each of the 70 cities.

The temperature data (the highest and lowest temperatures in the chosen months) was obtained from Climatological Data, published by National Oceanic and Atmospheric Administration. It is assumed that at 4 a.m. in the morning the temperature is the lowest, at 4 p.m. in the afternoon it is the highest, and the temperature rises from 4 a.m. until 4 p.m. (and falls from 4 p.m. to 4 a.m.). Based on this assumption, temperatures during all the seven hours are calculated for each season. The actual calculations are given on the next page [2]:

1. Winter

   Value = (WinterMaxTemperature − WinterMinTemperature)/4.0

Temperature at midnight	=	WinterMinTemperature + Value
Temperature at 4 a.m.	=	WinterMinTemperature
Temperature at 8 a.m.	=	WinterMinTemperature
Temperature at noon	=	WinterMaxTemperature − Value
Temperature at 4 p.m.	=	WinterMaxTemperature
Temperature at 8 p.m.	=	WinterMaxTemperature − (2 × Value)

2. Spring

   Value = (SpringMaxTemperature − SpringMinTemperature)/4.0

Temperature at midnight	=	SpringMinTemperature + Value
Temperature at 4 a.m.	=	SpringMinTemperature
Temperature at 8 a.m.	=	SpringMinTemperature
Temperature at noon	=	SpringMaxTemperature − Value
Temperature at 4 p.m.	=	SpringMaxTemperature
Temperature at 8 p.m.	=	SpringMaxTemperature − Value

3. Summer

   Value = (SummerMaxTemperature − SummerMinTemperature)/4.0

Temperature at midnight	=	SummerMinTemperature + Value
Temperature at 4 a.m.	=	SummerMinTemperature
Temperature at 8 a.m.	=	SummerMinTemperature + Value
Temperature at noon	=	SummerMaxTemperature − Value
Temperature at 4 p.m.	=	SummerMaxTemperature
Temperature at 8 p.m.	=	SummerMaxTemperature − Value

4. Fall

Value = (FallMaxTemperature − FallMinTemperature)/4.0

Temperature at midnight	=	FallMinTemperature + Value
Temperature at 4 a.m.	=	FallMinTemperature
Temperature at 8 a.m.	=	FallMinTemperature
Temperature at noon	=	FallMaxTemperature − Value
Temperature at 4 p.m.	=	FallMaxTemperature
Temperature at 8 p.m.	=	FallMaxTemperature − (2 × Value)

The Turbo Prolog predicate giving the temperature information is as follows:

```
place(city, state, latitude, longitude, timezone, winter_highest,
      winter_lowest, spring_highest, spring_lowest, summer_high-
      est, summer_lowest, fall_highest, fall_lowest).
```

The insolation information was directly obtained from the National Bureau of Standards (NBS, BSS 96), "Hourly Solar Radiation Data for Vertical and Horizontal Surfaces on Average Days in the US and Canada" [15]. Insolation is the rate of heat gain per square foot (Btu/Ft^2) of exposed area. We have limited ourselves to the typical orientations, namely horizontal and vertical (surfaces). The vertical orientation is further divided into the north, northeast, east, southeast, south, southwest, west, and northwest directions. Consequently, a list of solar radiations contains radiation values for the nine directions: north, northeast, east, southeast, south, southwest, west, northwest, and horizontal. At midnight at any location, insolation is zero. So we have collected data for these nine directions at 4 a.m., 8 a.m., noon, 4 p.m. and 8 p.m.

The Turbo Prolog predicate giving this information for a season is as follows:

```
radiation(city, state, season, L4, L8, L12, L16, L20)
where:
L4 = a list of solar radiations in 9 directions at 4 a.m.
L8 = a list of solar radiations in 9 directions at 8 a.m.
L12= a list of solar radiations in 9 directions at 12 p.m. (noon)
L16= a list of solar radiations in 9 directions at 4 p.m.
L20= a list of solar radiations in 9 directions at 8 p.m.
```

In the prototype the user is prompted for the city and state name. If the specified city is not found the program displays all the cities in that state and in the neighboring states. The user needs to choose from the list, the city which has the closest climate to the specified city.

However, in the actual system the user should be able to insert his own data about the site in the database, as the microclimate of a site may be different. For example, the average temperature of a building site (microclimate) may be different from that of the city. The temperature of a site in a city varies because of topography, nearby thermal masses and reflectivity of the surroundings. Insolation information of the building site may be different from the insolation of the city because of shading from nearby trees and adjacent buildings, altitude, pollution, etc.

3.3.2 Building Design Information

The procedure for obtaining the approximate energy analysis described in *Energy Graphics* [10] suggests a grouping of available spaces (e.g. rooms) based on similar requirements of schedules, and/or allowable interior temperature ranges, and/or internal heat generation. As a result of grouping, the procedure performs

calculations on all spaces within a group at one time rather than separately on each one. Thus, grouping saves considerable calculations. The classification of these groups remains the same until the schematic design phase is complete. In *Energy Expert*, we are considering only residential buildings or small commercial buildings. Spaces in such a building require the same range of allowable interior temperature. Hence, presently a building is treated as one group. The discussion here is based on the assumption that the building comprises one group only.

The following information about the building is collected either from the drawings or the user:

- Floor and roof areas
- Floor to roof height
- Allowable interior temperature range
- Exterior wall areas in each direction
- Window areas in each direction

Using this data, the energy performance of each energy group in each season is calculated.

3.4 Calculation Details

As stated earlier, heat gain and loss from the four main sources are calculated. In this section we will describe the computation of: (1) heat gain and loss from these four sources, (2) total heat gain and loss in each season, (3) total allowable

heat gain and loss in each season, and (4) seasonal and annual heating and cooling costs.

Before calculating the energy performance, certain assumptions about the shape and orientation of the building are made. They are are listed below:

Shape : *Energy Graphics* assumes that a building is a cube with five sides exposed to the sun. Since this is not usually the case in practice, we assume that the shape of a building is a polyhedron with n walls.

Orientation : *Energy Graphics* assumes that the sides of the building face north, east, south and west. To get a more precise analysis we allow the sides of the building to face any of the eight directions north, northeast, east, southeast, south, southwest, west, and northwest. The building roof is assumed to be horizontal. In the actual system the assumption about the slopes of the roof can be changed.

Transparency : *Energy Graphics* assumes that each of the four sides of the building have 50% glass (transparency) and 50% wall. However, the transparency of external walls (i.e. windows) is the main factor in the heat gain or loss. If the purpose of the system is to give directives based on the existing design, then the use of exact areas for the walls and windows is needed. We extract the information about the sizes and orientations of the windows, exterior walls and roof from AutoCAD drawings or from the user.

Transmittance : Transmittance, or the U value of a wall or a window, depends on the material and its thickness. (For a definition of U value refer to Section 3.1.) *Energy Expert* obtains the U values of walls, roof, and glass from the

AutoCAD drawings or the user. The default U value of wall and roof is 0.10 $Btu \cdot hr/ft^2$ and that of glass is 0.8 $Btu \cdot hr/ft^2$.

Ventilation Rate : Ventilation is the flow (introduction) of outside air through uncontrolled (infiltration) or controlled (mechanical) means. Ventilation rate is measured in cubic feet per minute *(cfm)*. As stated earlier, air change is the replacement of a quantity of air in a volume within a given period of time. It is expressed in number of changes per hour. It is a unit to measure infiltration. A designer can control the rate of infiltration through the design and construction of the building envelope. In a well constructed building ventilation from infiltration should be at a low rate of 1/3 to 1/2 air changes per hour. Ventilation by mechanical means can increase ventilation to 10 or more air changes per hour [10]. After discussions with experts in the field, we assume the ventilation rate due to the infiltration to be 1/3 air changes per hour. The minimum ventilation rate by mechanical means is assumed to be 1 cfm per person and the maximum rate by mechanical means to be 5 air changes per hour. *Energy Graphics* suggests that these figures are 0.5 air changes per hour, 5 cfm per person, and 10 air changes per hour, respectively.

After assuming these basic factors we are now ready to describe the calculation details.

3.4.1 Internal Heat Gain

The major criteria of the internal heat gain in a building are occupants and kitchen appliances. Heat release per occupant of a residence is assumed to be 225 *Btuh*. The number of occupants is given by the user. If this number is not known then

the rule of thumb is to assume that the number of occupants is equal to twice the number of bedrooms [9]. Using this rule, an approximate number of occupants is estimated. In a residential building about 1200 *Btuh* is the accepted value of heat released by kitchen appliances. It is assumed that the heat gains from the kitchen appliances are only during the day.

Consequently, during the day we assume that

$$\text{Internal Heat Gain} = \text{No of Occupants} \times 225 + 1200\ Btuh$$

and during the night

$$\text{Internal Heat Gain} = \text{No of Occupants} \times 225\ Btuh$$

3.4.2 Solar Heat Gain

The equation to calculate the solar gain is as follows [10]:

Solar Heat Gain = (Insolation × Exposed Area of Walls of Each Side × Transmission of Building Walls) + (Insolation × Exposed Area of Windows of Each Side × Transmission of Windows) + (Insolation × Roof Area × Transmission of Roof).

Solar heat gain depends on the environment. By making appropriate changes in the design, the gain can be minimized during periods of net heat gain and maximized during periods of net heat loss, if the gain and loss are during sun light hours.

The database of *Energy Expert* has data for solar radiation (insolation) of the 70 cities mentioned earlier. This data is for the horizontal orientation and 8 directions of vertical orientation. The roof is assumed to have the horizontal direction. The wall areas facing these eight directions are denoted as WSouth, WSouthwest, WWest, WNorthwest, WNorth, WNortheast, WEast and

WSoutheast . The respective glass areas are GSouth, GSouthwest, GWest, GNorthwest, GNorth, GNortheast, GEast, and GSoutheast and the roof area is ARoof. The U values (transmission) for walls, roof and glass are WU, RU and GU. Suppose the insolation at 4 a.m. in winter for the surfaces in these 9 direction is RW4_South, RW4_Southwest, RW4_West, RW4_Northwest, RW4_North, RW4_Northeast, RW4_East, RW4_Southeast and RW4_Hor. Then the solar heat gain at 4 a.m. in the winter is as follows [10]:

Solar Heat Gain =

(RW4_South × (WSouth×WU + GSouth×GU)) +
(RW4_Southwest × (WSouthwest×WU + GSouthwest×GU)) +
(RW4_West × (WWest×WU + GWest×GU)) +
(RW4_Northwest × (WNorthwest×WU + GNorthwest×GU)) +
(RW4_North × (WNorth×WU + GNorth×GU)) +
(RW4_Northeast × (WNortheast×WU + GNortheast×GU)) +
(RW4_East × (WEast×WU + GEast×GU)) +
(RW4_Southeast × (WSoutheast×WU + GSoutheast×GU)) +
(RW4_Hor × ARoof×RU)

The solar gain is calculated for the selected seven hours in each season.

3.4.3 Envelope Heat Gain and Loss

The envelope heat gain and loss are calculated for each of the selected seven hours of each season. The basic equation for the envelope heat gain/loss is as follows [10]:

Envelope Heat Gain/Loss = (ΔT × Exposed Area of Walls of Each Side × Transmission of Building Walls) + (ΔT × Exposed Area of Windows of Each Side × Transmission of Windows) + (ΔT × Roof Area × Transmission of Roof).

Temperature difference (ΔT) is the difference between the exterior dry bulb temperature (T_o) and the interior dry bulb temperature (T_i), i.e. $\Delta T = T_0 - T_i$.

ΔT is calculated for the seven selected hours for each season. The notation for wall areas, window areas, roof area, and their respective U values (trasmission) is given in the previous section. Let us denote the maximum allowable interior temperature as T_{hi} and the minimum allowable interior temperature as T_{lo}. Exterior temperatures at the seven selected hours in each season have already been calculated based on the information obtained from the site database. To calculate the interior temperature T_i for each hour, the following three conditions are checked [10]:

1. If $T_o > T_{hi}$ then

   $T_i = T_{hi}$

2. If $T_o < T_{lo}$ then

   $T_i = T_{lo}$

3. If $T_{hi} \geq T_o \geq T_{lo}$ then

   $T_i = T_o$

For convenience, we will also discuss the calulations of another quantity, ΔT_{hi}, which will be used in the ventilation heat gain and loss calculations. ΔT_{hi} is the temperature difference between the maximum allowable interior temperature (T_{hi}) and the assumed interior temperature (T_i), i.e. $\Delta T_{hi} = T_{hi} - T_i$. ΔT_{hi} is calculated for all the 28 hours in the year (7 hours for each of the four seasons). Again the above three conditions are checked [10]:

1. If $T_o > T_{hi}$ then $T_i = T_{hi}$ and $\Delta T_{hi} = 0$.

2. If $T_o < T_{lo}$ then $T_i = T_{lo}$ and $\Delta T_{hi} = T_{hi} - T_{lo}$.

3. If $T_{hi} \leq T_o \leq T_{lo}$ then $T_i = T_o$ and $\Delta T_{hi} = T_{hi} - T_o$.

Thus, the greatest possible value of ΔT_{hi} is the difference between the maximum and minimum allowable interior temperatures, and the minimum value of ΔT_{hi} is zero.

3.4.4 Ventilation Heat Gain and Loss

The basic equation to calculate the ventilation heat gain/loss is as follows:

Ventilation Heat Gain/Loss = ΔT × Ventilation Rate × Ratio of Specific Heat to the Specific Volume of the Air

Ventilation rate is assumed to be a constant rate at midpoint between the ventilation due to infiltration and the maximum mechanical ventilation rate.

Ventilation Rate by Infiltration = Volume × Air Changes/60min
= Volume × 0.33 × 1/60

Min Ventilation by Mechanical Means = Occupancy × 1

Max Ventilation by Mechanical Means = Volume × 5 × 1/60

Calculations for the temperature difference ΔT, have been described in Section 3.4.3. The ratio of specific heat to specific volume of the air (a constant) is assumed to be 1.08. Using these values the ventilation heat gains and losses at

the selected seven hours of each season are calculated. In the final system other factors such as humidity should also be considered while calculating ventilation heat gain and loss when humidity is a problem. In such a case,

$$\text{Ventilation Heat Gain/Loss} = \text{Ventilation Rate} \times \Delta H \times 4.5$$

where 4.5 is the constant used to convert $minutes \cdot ft^3$ to $hours \cdot lb$ and ΔH is the average heat content per pound of outside air for a given site.

3.4.5 Allowable Heat Gains From Envelope and Ventilation

When heat gains in a building are such that the interior temperature remains within the allowable range, no artificial cooling or heating is required. This helps to cut down on the energy costs. Therefore, such allowable heat gains are not avoided. Using ΔT_{hi}, explained in Section 3.4.3, allowable envelope and ventilation heat gains are calculated. This section shows the methods for calculating allowable heat gains.

1. **Allowable Envelope Heat Gains:** Computation of the allowable envelope heat gain is similar to that of envelope heat gain and loss except that instead of ΔT, ΔT_{hi} is used. Thus,

   $$\text{Allowed Envelope Heat Gain} = \Delta T_{hi} \times \text{Exposed area} \times \text{Transmission}$$

   Allowable envelope heat gain is calculated for all the selected seven hours in each season.

2. **Allowable Ventilation Heat Gain:** Computation of the allowable ventilation heat gain is similar to that of ventilation heat gain and loss except that instead of ΔT, ΔT_{hi} is used. Thus,

$$\text{Allowed Ventilation Heat Gain} = \Delta T_{hi} \times \text{Ventilation Rate} \times 1.08$$

Allowable gain from ventilation is calculated for all the selected seven hours in each season.

3.4.6 Total Heat Gain and Loss

Total heat gain/loss in each season for each of the chosen seven hours is calculated by adding the respective internal heat gain, solar heat gain, envelope heat gain/loss and ventilation heat gain/loss. The *total allowable heat gain* in each season for each of the seven hours is calculated by adding the corresponding envelope and the ventilation allowable gains. The results form the basis for the energy performance analysis.

3.4.7 Cost Estimations

Heating and cooling equipment typically operate at different coefficients of performance. Therefore, heating and cooling costs are computed separately. *Energy Expert* calculates seasonal and annual heating and cooling costs based on the approximative method described in *Energy Graphics* [10].

The *total heat gain/loss* and *total allowable heat gains* for each of the selected seven hours in a typical day of each season have already been calculated (Refer to Section 3.4.6). The time interval between the two consecutive selected hours is four hours. (The seven hours form six such intervals.) For each four-hour period an average of the beginning and ending rates of the total heat gain/loss is noted.

It is multiplied by 4 to get the *average heat gain/loss* in that period. Similarly, the *average allowable heat gains* in each interval are calculated.

1. **Heating Costs:** To calculate heating costs, only heat losses are considered. The *average heat losses* in all the six intervals are added to get the heat loss in a typical (average) day (24 hours) of each season. Then,

$$\text{Seasonal Heating Cost} = 0.7 \times 90 \times \text{Fuel Cost} \times \text{Heat Loss in 24 Hours}$$

   where 0.7 is the assumed coefficient of performance of the equipment [10] and 90 is the approximate number of days in the season.

   Heating costs in the four seasons are calulated and then the annual heating cost is found by adding them together.

2. **Cooling Costs:** To calculate cooling costs, *average heat gains* and *average allowable heat gains* in each of the six intervals of each season are considered. If the *average heat gain* in an interval is greater than the corresponding *average allowable heat gain* then the difference between the two is noted as the *excess heat gain* in that interval. Otherwise the *excess heat gain* is zero. After this step, the *excess heat gains* in a typical (average) day (24 hours) of each season are computed. Then,

$$\text{Seasonal Cooling Cost} = 0.5 \times 90 \times \text{Electric Cooling Cost} \times \text{Excess Gain in 24 Hours}$$

where 0.5 is the assumed coefficient of performance of the equipment [10], 90 is the approximate number of days in the season.

Cooling costs in the four seasons are calulated and then the annual cooling cost is obtained by adding them together.

3.5 Expert Advice

Energy Expert is a rule-based expert system. This section first describes principles of a rule-based expert system and then presents the structure of advising in *Energy Expert.*

3.5.1 Rule-Based Expert Systems

An expert system is a computer program which solves a complex problem in a knowledge rich domain in the real world, making decisions normally made by an expert. An expert system should not just make a decision, it should also explain the rationale behind the decision.

A typical expert system is composed of three parts: a knowledge base and a database, an inference mechanism, and a user interface. Most commonly used knowledge representations include production rule systems, frames, semantic nets and decision trees.

Rules in an expert system provide a formal way of representing recommendations, directives or strategies [29]. Rules in a rule-based expert system are IF-THEN structures. For example:

IF the outside temperature is more than the inside temperature

THEN heat cannot be removed from inside to outside without artificial means.

The same rule can also be represented as:

Heat cannot be removed from inside to outside without artificial means :-

Outside temp > Inside temp.

In a rule-based expert system, the rules are checked against the knowledge (a collection of facts) about the current situation. When a condition given by the IF portion (antecedent) of a rule is matched with the facts, the action (consequent) specified by the THEN part is performed. This is called *firing* a rule.

The rules in a rule-based system can be used in two ways: forward chaining and backward chaining. These are called *inference methods*. In a forward chaining system, for a set of data, applicable production rules are found, new conclusions are asserted in the knowledge base and again the rules are matched with the modified knowledge base. The process continues until no more applicable production rules can be found. Conversely, in a backward chaining system a possible diagnosis is a goal and an attempt is made to satisfy the goal. Antecedents of the goal become subgoals which the system tries to satisfy. The process is repeated until the goals are directly satisfied by the current knowledge (facts) about the case.

At the schematic design stage, the evaluation of the building is done for its energy performance, fire-safety, circulation and structural soundness. Using standard methods, the performance of the current alternative building design can be calculated for each of these criteria. This provides the knowledge about the design. A set of rules about each criterion was constructed after discussions with experts in the field. Since the knowledge about the performance of the building

alternative (design solution) is calculated in this system, forward chaining is the most appropriate system for the *evaluators*. In a forward chaining system, when a rule is fired, new facts may be added to the knowledge base. Matching the IF portions of rules with the new facts can produce a *forward inference chain*. The *forward inference chain* can be used to give the reasoning behind the decision.

3.5.2 Advice in *Energy Expert*

Energy Expert computes the energy performance of a building in each season. It is also designed to analyze the energy performance of a building and to advise the designer on the possible ways of improving the design. The suggested modifications to the building design make the building more energy efficient and help to cut down on the heating and cooling costs.

In the next few paragraphs, first we will present the structure of the advisory system of *Energy Expert*. Then we will present an example to illustrate the function of the system.

The Advisory System

Initially, *Energy Expert* prompts the user to enter a range of comfortable (allowable) temperatures in the building. The objectives of the advisory system are such that the interior building temperature is to be maintained within the comfort range while keeping the energy costs as low as possible. Based on this range, the seasonal energy performance is tested for six possible conditions. These conditions are numbered from 1 to 6. The conditions and the respective numbers (identifiers) are as follows:

1. Heat loss all day.

2. Some heat loss and some allowable heat gain.

3. Some heat loss and some allowable gain and some excess heat gain.

4. All allowable heat gain.

5. Some allowable heat gain and some excess heat gain.

6. All excess heat gain.

The main rules are based on the above six conditions. The rules are structured in such a way that only one rule gets fired. The logical structure of the calling routine is as follows:

```
rule(rule_no) :- condition(rule_no),
                 action(rule_no).
```

where rule_no is a unique identifier for the rule, its corresponding condition and action.

Once a rule is fired, the system tries to satisfy the goal action. The structure of the predicate action(rule_no) can be expressed as follows:

```
action(rule_no) :- get_subconditions(rule_no, subcond_no),
                   possible_solutions(rule_no, subcond_no, [soln_nos]),
                   get_directives(rule_no, subcond_no, [soln_nos]).
```

where rule_no stands for the rule_no of the fired rule, and subcond_no stands for the number (identifier) of a matched subcondition.

Now we will present a discussion about the three predicates: get_subconditions, possible_solutions and get_directives.

get_subconditions : The logical structure of this predicate looks as follows:

```
get_subconditions(rule_no, subcond_no) :- subcondition(subcond_no).
```

where rule_no stands for the rule_no of the fired rule, and subcond_no stands for the number (identifier) of a matched subcondition.

It is assumed that the advice is such that the usage of artificial cooling and heating will be minimum. For example, even though the outside temperature is higher than the inside temperature, the inside heat can be removed from inside to outside by using artificial means. However, this solution increases the cost of cooling. Naturally, while improving the design to cut down the cost, this alternative cannot be used. To achieve the objectives of keeping the low energy costs, three subconditions have been developed:

1. If there is an excess heat gain and the exterior temperature is above the maximum allowable interior temperature ($T_o > T_{hi}$) then heat cannot be removed from inside to outside without artificial means.
2. If there is an excess heat gain and the exterior temperature is less than the maximum allowable interior temperature ($T_o < T_{hi}$) then heat can be removed from inside to outside.
3. If most of the heat loss is during the night time then the increase in solar heat gain should be avoided.

If a subcondition is satisfied, subcond_no is instantiated to the corresponding subcondition number (identifier), else it is instantiated to zero.

possible_solutions : After trying to satisfy the goal get_subconditions, the system tries to satisfy the second goal, possible_solutions. The knowledge base contains clauses possible_solutions(rule_no, subcond_no, [soln_nos]) that are designed so that given the combination of a rule identifier (rule_no) and subcondition number (subcond_no), a list of possible solution numbers ([soln_nos]) is provided. Thus, this predicate provides a link between the fired rule, the matched subcondition and the corresponding solutions.

The knowledge base of *Energy Expert* contains 10 possible solutions to improve the building design. The solutions and their corresponding numbers (identifiers) are as follows:

1. Increase in internal heat generation.
2. Decrease in internal heat generation.
3. Increase in solar heat gain.
4. Decrease in solar heat gain.
5. Increase in envelope heat gain/loss.
6. Decrease in envelope heat gain/loss.
7. Increase in ventilation heat gain/loss.
8. Decrease in ventilation heat gain/loss.
9. Store heat.
10. Store cold.

In *Energy Expert* the predicate storing the above solution numbers and the corresponding solutions looks as follows:

```
soln(soln_no, solution).
```

For example,

```
soln(1, increase_internal_heat_generation).
```

The text of the solution has to be connected by underscores in order to satisfy Prolog syntax requirements. Presently, the solution is displayed in this form. However, this output could be converted to a more readable form in future extensions of the system.

get_directives : After trying to satisfy the goal possible_solutions, the system tries to satisfy the third goal get_directives. The logical structure of this routine can be expressed as follows:

```
get_directives(rule_no, subcond_no, [soln_nos])
    {For each soln_no
        directive_list(soln_no, [directives]),
        {For each directive
            modification_list(directive, [modifications]).
        }
    }
```

In this routine, for each of the solution numbers in the list [soln_nos], a list of directives ([directives]) is obtained. For each of the directives given in the list [directives], a list of possible building modifications is represented by the list [modifications].

The knowledge base of *Energy Expert* contains the clauses directive_list. The structure of directive_list is as follows:

```
directive_list(soln_no, [directives]).
```

where the list of directives corresponding to a soln_no is given by [directives]. At present the knowledge base contains a total of 31 different directives. The following is an example for a soln_no and the corresponding directives:

```
directive_list(3, [increase_exposed_surfaces,
                   consider_active_solar_heating,
                   increase_absorptivity]).
```

The knowledge base also contains of the clauses for possible modifications to achieve the given directives. The structure of the predicate modification_list is as follows:

```
modification_list(directive, [modifications]).
```

where the list of modifications corresponding to a directive is given by [modifications]. At present, the knowledge base contains a total of 82 different directives. An example for a directive and the corresponding modifications is as follows:

```
directive_list(increase_exposed_surfaces,[avoid_shading,
                                          slope_roof_towards_south,
                                          use_clear_glazing]).
```

The solutions, and their corresponding directives and modifications are displayed on the screen. This is the advice (ADVICE in Figure 2.6) about the possible

changes in the building design for better energy performance. Most expert systems diagnose a problem in a given domain, but they may not necessarily give solutions to solve the problem. *Energy Expert* analyzes the approximate energy performance of a building, and based on the performance, it gives the directives to improve the building design, and also suggests the possible modifications in the design.

A future extension to *Energy Expert* would be to include many more directives and modifications in the knowledge base of the system. Also, there can be some directives and modifications that may conflict with each other when combined. Hence, a conflict resolution strategy and some heuristics will need to be added to pick up directives and modifications that provide maximum benefit in the given situation.

An Example

Now we will present a brief example to illustrate the advisory system. *Energy Expert* calculates energy performance of a building in each of the four seasons. Based on the user given range of the comfortable (allowable) temperatures in the building, the energy performance is checked for the possible six conditions. (Refer to the list of six conditions on page 50.)

Suppose the energy performance of a building design is such that in the Fall there is some heat loss, some allowable heat gain and some excess heat gain. In this case, condition 3 is satisfied. Hence, the corresponding action is taken. Thus, the calling routine looks as follows:

```
rule(3) :- condition(3),
           action(3).
```

While processing action(3), the goal get_subconditions(3, subcond_no) is checked first, in order to find the corresponding subcondition number, i.e. subcond_no. Suppose that in this example the exterior temperature is less than the maximum allowable temperature, therefore, subcondition 2 is satisfied. (Refer to page 51 for the subconditions and their numbers). Hence, subcond_no is instantiated to 2.

The system then tries to satisfy the goal:

possible_solutions(3, 2, [soln_nos]).

In this example, the solution numbers for action 3 and subcondition 2 are given by the list [6,8,9]. The solution numbers (soln_no) and the corresponding solutions are as follows:

6. Decrease in envelope heat gain/loss.
8. Decrease in ventilation heat gain/loss.
9. Store heat.

When possible_solutions succeeds, the system tries to satisfy the next goal get_directives. For each soln_no in the solution list [6,8,9] the corresponding directives are retrieved from the knowledge base by matching the soln_no in the clauses directive_list. For example, for the soln_no 6, the corresponding directives are given as follows:

directive_list(6, [decrease_exposed_surfaces,
increase_thermal_resistance,
decrease_temperature_difference, $\cdots$]).

For each directive, corresponding modifications are retrieved from the knowledge base by matching the directive in the clause modification_list. For example, the directive decrease_exposed_surfaces and the corresponding modifications are as follows:

```
modification_list(decrease_exposed_surfaces, [provide_large_overhangs,
                                              reduce_floor_to_floor_height,
                                              cover_roof_with_sod, ···]).
```

The above three solutions, with the corresponding directives and modifications are displayed as advice on the screen.

3.6 Graphical Output

Architects are often called right brain thinkers because they appreciate graphical results more than quantitative output. Based on this observation, the energy performance is shown graphically instead of numerically. The design of the graphical output was limited by the tool used to implement the project (Turbo Prolog) and the time available to implement it. However, the graphs are sufficient to provide information about energy consumption at the schematic design phase.

Energy Expert displays the seasonal energy performance on the screen for each of the four seasons, one at a time. The screen is divided into three windows (Figure 3.1). Window 1 (upper left window) displays the line graph of the total energy gain and loss against the seven chosen hours in a season. Window 2 (upper right window) shows two pie charts of cooling costs and heating costs in that sea-

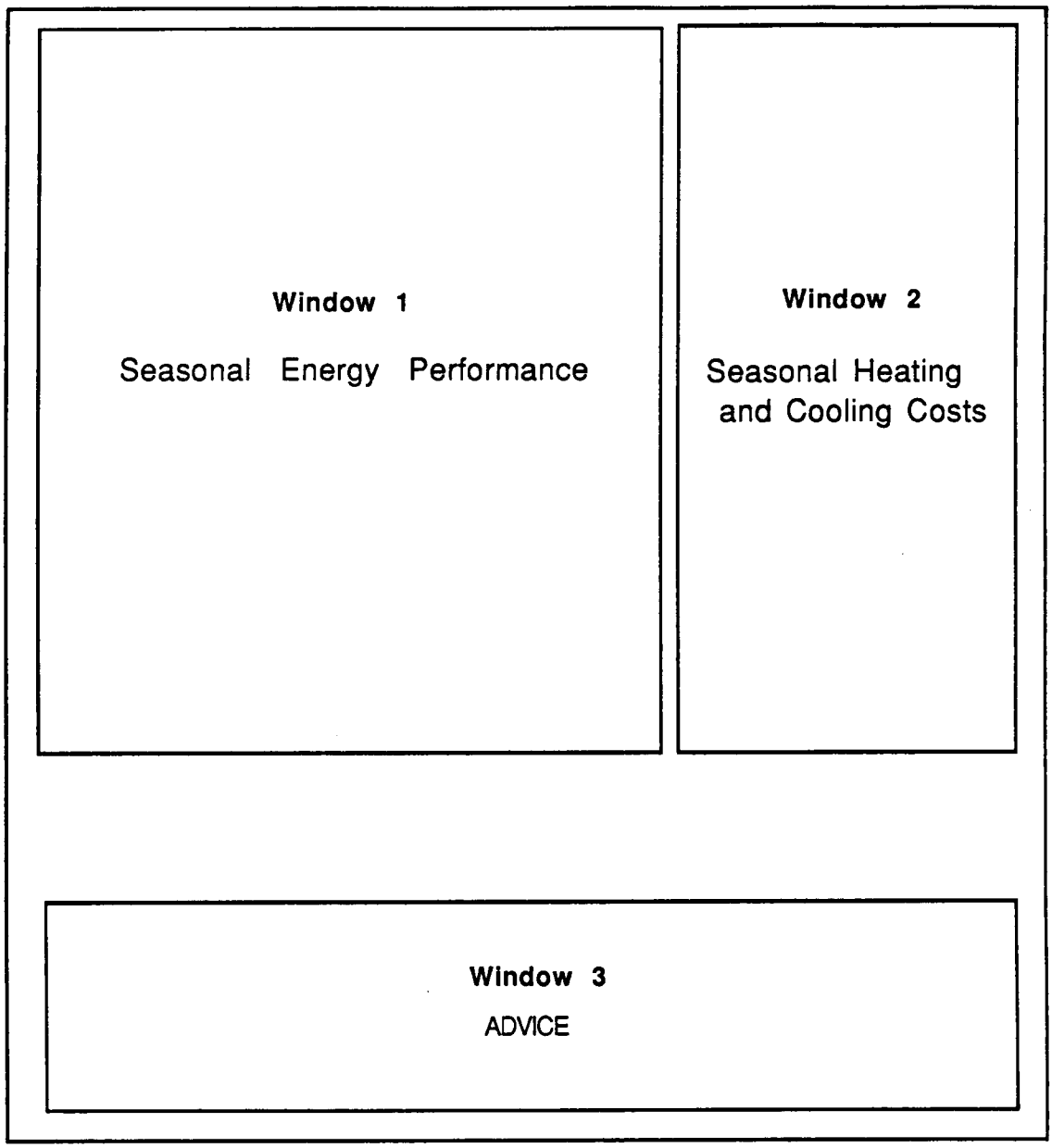

Figure 3.1: Graphical Output: The Three Windows

son. Window 3 (lower window) displays the advice (the solutions, their directives and modifications) for the design improvement. The user needs to press the space bar to observe the building performance in the next season. Figure 3.2 illustrates these three windows in detail.

3.7 Summary

This chapter presented a description of *Energy Expert* which is one of the *evaluators* in *CAAD*. *Enery Expert* is designed to be used at the schematic design stage to obtain a quick and approximate energy analysis, and to give advice about the possible modifications in the building design to improve the energy performance. The output of the system is graphical so that it can be appreciated better at the schematic design stage. Ideally the program should store the information gathered from the user and the directives in the database for the next iteration of the design. At present this feature is not provided. Based on the logical structure of *Energy Expert*, expert systems for evaluating fire-safety, structural and circulation performance of the building can easily be developed.

In *CAAD*, the principles of expert systems have been applied at the schematic design stage. The construction of expert systems for the final design stage is an ambitious project. In *CAAD*, instead of building an expert system for the final design stage, CALPAS3, an existing program has been used. The use of CALPAS3 in *CAAD* is described in the next chapter.

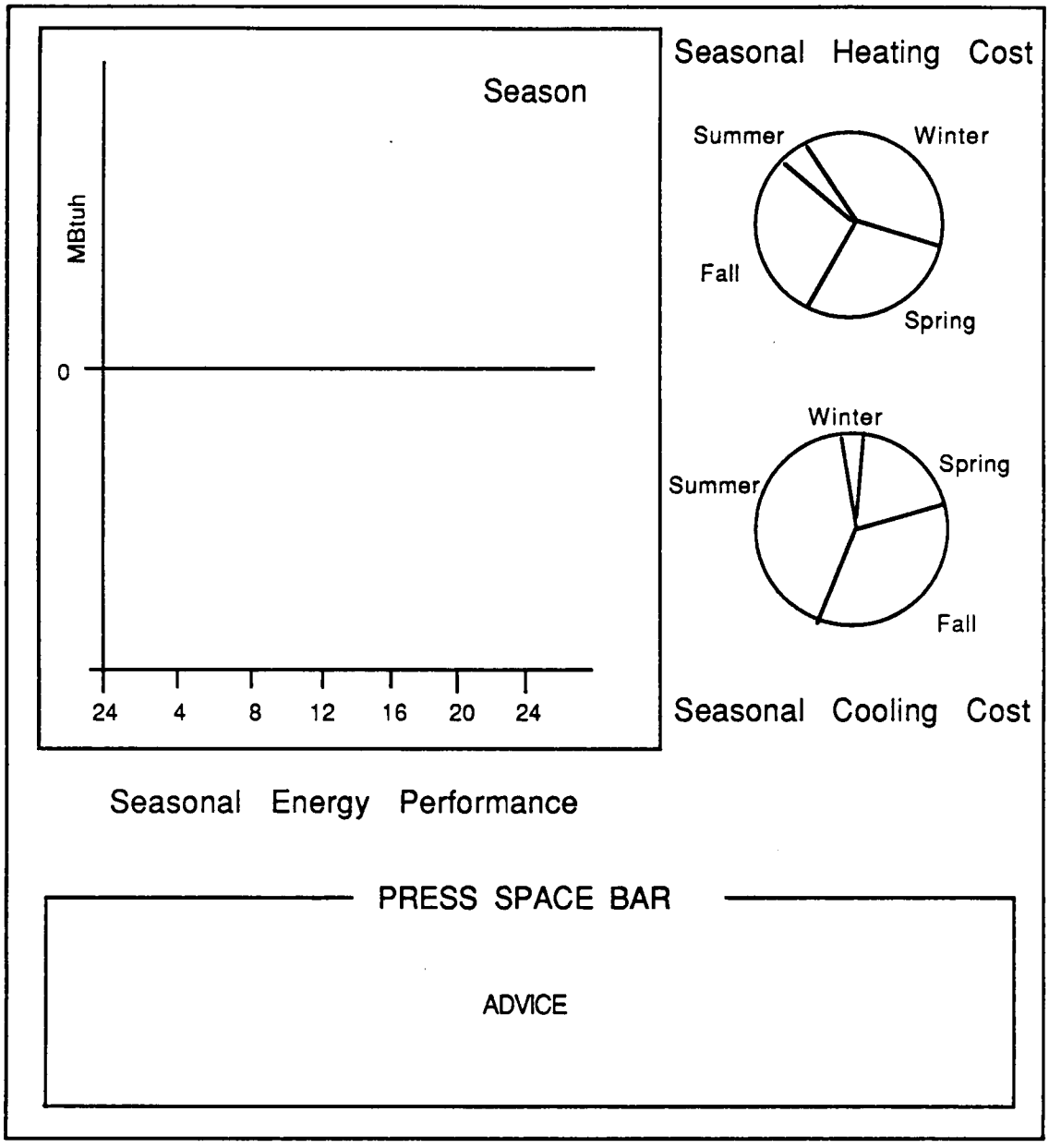

Figure 3.2: Graphical Output

Chapter 4

CIC

After finalizing a schematic design, the architect prepares the final design. At this stage an accurate energy analysis of the building is done. Some states such as California have building bylaws for energy performance of a building. Checks for the fire-safety requirements and a structural analysis are also carried out at this stage. If necessary, the design is improved to satisfy the requirements.

An ideal integrated Computer-Aided Architectural *Design* system should provide all these checks. There are some independent programs developed by different groups. For example at the University of Michigan, a fire-egress program has been developed for Apollo workstations [24]. This program runs only on that particular machine. The Berkeley Solar Group has also developed a sophisticated program, CALPAS3, for the accurate energy analysis of a building [3]. The program is available by dial up and also on the IBM PC and its compatibles.

Though these programs are sophisticated, very often they require a complicated and lengthy data to be entered by the user in a definite format. For example, to run CALPAS3 the user needs to prepare an input file in a definite format. In addition, the user needs to know about the CALPAS3 editor to edit

the file. Each command has several arguments. The user needs to be familiar with these arguments. For example, in the SITE command, the argument LAT should be used for latitude, LONG for longitude and TZN for the time zone. The latitude and the longitude values need to be real numbers where as the TZN value must be an integer. The user needs to remember these things or at least refer to the manual for almost every command. Consequently, many architects prefer to have the program run by computer professionals. However, if an interface were developed between the drafting package and CALPAS3, most of these difficulties would be overcome. Some data can be read from the drawing files and the remaining information could be collected from the user through simple questions and a menu.

In the prototype for *CAAD* we have developed an interface between CALPAS3 and AutoCAD. We will refer to this program as *CIC* (Create Input file for CALPAS3). In the next section we will describe CALPAS3. The implementation details of *CIC* are given in Section 4.2. Section 4.3 is a summary of this chapter.

4.1 CALPAS3: Description

CALPAS3, a computer simulation program, analyzes the energy performance of passive solar, conventional residential, and small commercial buildings. The output of this program is a report which consists of hourly, daily, monthly, and yearly heating and cooling loads, spaces and storage temperatures, heat gains and losses, and annual heating and cooling costs [3].

Originally this program was developed by Phil Niles and collaborators at the California Polytechnic State University for use in modeling the performance of passive solar homes for the California Solar Handbook published in 1980. Later

the Berkeley Solar Group implemented and enhanced the program [3].

CALPAS3 has been available nationwide since 1980 via dial-up time sharing. The PC version was made available in November 1984. The educational package is available for use in colleges and universities.

CALPAS3 can model conventional and passive solar residential buildings which may have attached sunspace or a green house. CALPAS3 considers walls, water walls, slabs, and underslab rockbed as thermal masses. A building can have windows and walls of any type and orientation. Windows can have seasonally variable shading devices such as shutters, overhangs and fans. A building can have evaporative cooling. In CALPAS3 hourly weather data is available for 300 cities in the U.S.A.

4.1.1 Reporting Features

The output of CALPAS3 is a report containing building performances during the entire year and each month of the year. It also reports the daily and hourly performance during user-specified hours. However, the program is not designed to give an advice to improve the design.

The annual report gives the total operating costs and the total conventional energy used during the year. It also shows peak hourly loads of heating and cooling for the entire year.

A monthly report gives the energy flows for each month. This report also contains the total energy flows for the entire year. These energy flows are conduction and infiltration gain and losses, solar gain, storage gains and losses, internal gain, and the peak heating and cooling loads. The report also consists of the total

ventilation flows, outdoor temperatures and isolation.

A daily report specifies the same quantities as those in the monthly report for each day specified by the user.

An hourly report is about all the hours of the days specified by the user. The report is about the same quantities as those in the monthly report. In addition, energy flows for each individual thermal mass element, storage temperatures, wind speed, and directions on an hourly basis for each specified day are provided.

4.1.2 Basics of CALPAS3

The major elements of CALPAS3 are described in the CALPAS3 manual. We will describe them in brief:

1. **Building Envelope:** CALPAS3 can analyze a building having unlimited number of walls and windows of any orientation. The solar radiation effects on each surface are calculated using the absorptivity of the surface and the exterior film coefficient. A constant air change rate or varying rates, and interior and exterior temperature differences are used to calculate infiltration.

2. **Thermal Mass Elements:** The building can have a maximum of five thermal mass elements. They are:

   (a) floor slab

   (b) interior walls

   (c) exterior walls

   (d) mass wall with glazing

   (e) floor slab over a rockbed

3. **Solar Gain:** CALPAS3 calculates solar gain from the windows. CALPAS3 calculates hourly glazing transmittivities of each window using the mean declination day of each month. The windows can be chosen from the given eight types of glass with either single or double glazings. These windows are standard windows described in ASHREA [3]. An optional argument to provide the ground reflectivity for each window for each month is also provided. CALPAS3 takes into consideration overhangs, fins or movable shutters while calculating the solar gain. It uses the solar data from its database of 300 cities.

4. **Internal Heat Gain:** A residential or commercial schedule is used by default for calculating internal heat gains. An optional command is provided to change these values.

5. **Ventilation Heat Gain and Loss:** Using the ASHREA algorithm, cooling due to natural ventilation from wind and/or by the stack effect[1], or a fan is calculated. A simple evaporative cooler can also be considered. In some cases, such as the stack effect, the rate of ventilation depends upon air temperature of the house. In these cases if the house temperatures cannot be taken as a specified ventilation set point, CALPAS3 repeats the calculations until it obtains the actual air temperature/flow rate combination which would exist under the prevailing conditions.

6. **Heating and Cooling:** CALPAS3 calculates heating costs if the building heating is by a fuel fired furnace, an electric heater, or a heat pump. It calculates the cooling costs when cooling is by an air conditioner. Heating

[1]The stack effect relies on thermal forces set up by temperature differences between indoor and outdoor air. It can occur through open windows.

or cooling is used when building temperatures exceed the respective thermostat set points. Seasonal coefficients of efficiencies and performance are considered. The user needs to provide the cost for fuel in the input file if he wants to have annual costs of heating and cooling.

7. **Sunspace:** In CALPAS3, sunspace is an independent thermal zone which can be added to the building. A sunspace has its own windows, walls, slabs and all other elements of a building. The sunspace can be modeled together with the house, using air flows or by conduction through the mass wall. It can also be used to heat up the under-floor rockbed in the house. In a sunspace, ventilation and heating can be used to limit the temperatures. Seasonal shadings can be used to control the energy flow.

8. **Control:** CALPAS3 figures out the rational movements of the occupants in calculating heat flows. According to the day and mean outdoor temperature, the summer and winter seasons are determined. Each season has three thermostat set points namely heating, cooling and the desired temperature. When the inside temperature of the building exceeds these extreme set points, heating or cooling is provided accordingly. A proportional control strategy is used to prevent instability. Night set back temperatures are allowed for the house heating thermostat.

4.2 Implementation Details

If an interface is developed between a CAD system and an energy analysis program, then the user can use this integrated system as an analytical design tool. The user can design and draft his building using the CAD system. At different steps during

the design process, he can use the interface to reach the energy analysis. Using these results, the design can be improved to get allowable energy consumption levels. This process can be repeated until the user is satisfied with his design. Such a system can be a useful tool in design studios. This concept is the backbone of the program *CIC*.

CIC is a program module that has been developed in Turbo Pascal on an IBM PC AT running the DOS operating system. The basic purpose in *CIC* is to extract information from the drawings prepared using AutoCAD and to utilize this information in forming an input file for CALPAS3, the energy analysis program (Figure 4.1).

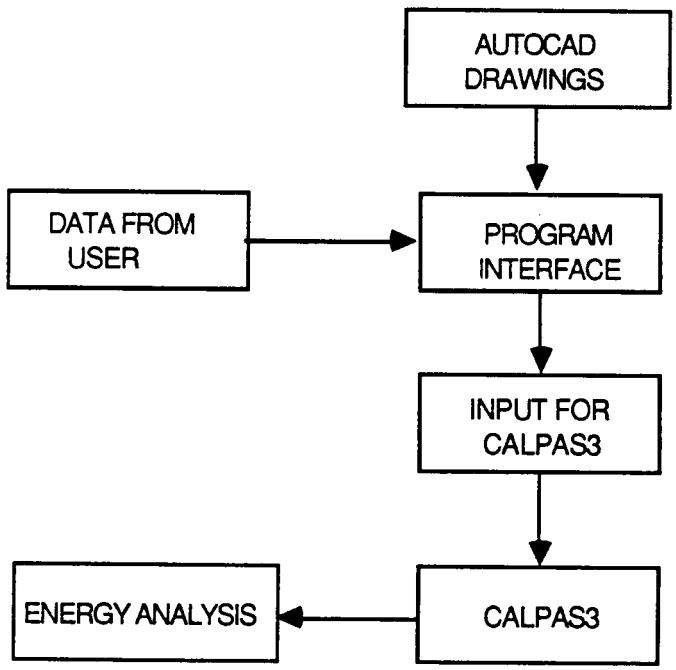

Figure 4.1: An Interface between CALPAS3 and AutoCAD

Information which is necessary and not given in the drawings is obtained interactively from the user. However, there may be a user who does not use AutoCAD and would still like to use CALPAS3. In such situations, when the user is is prompted with the following question:

Do you have a DXF file ? (Y/any key)

If the user enters any key except Y, the information to be extracted from the AutoCAD drawing file is obtained interactively from the user.

CIC is divided into four modules. They are listed below:

1. The Site Database
2. User Interface and Warning Messages
3. Extraction of Information from the AutoCAD Drawings or interactively from the user
4. Use of Extracted Information to Create New Information

We now present a brief discussion of the four modules.

4.2.1 The Site Database

CALPAS3 requires exact information about the site to calculate accurate energy analysis. A database of site information for 70 cities in the U.S. is stored in the file *site.dat*. As mentioned in Chapter 3, this database is used by the program *Energy Expert* and hence is in the form of Prolog predicates. The data about the sites, required by *CIC* is a subset of the data stored in *site.dat*. *CIC* requires the database in the form of a text file. A Turbo Pascal program called *fetch.inf* converts the necessary information from *site.dat* into a Turbo Pascal readable form

(i.e. text file) and stores it in the file *site1.dat*. Thus *site1.dat* is the database for *CIC*. A typical line of *site1.dat* is shown below:

City State Latitude Altitude TimeZone

The user is prompted for the city name. If the city name is not found in the database, the user needs to enter the data for the location.

4.2.2 User Interface and Warning Messages

CIC retrieves information from the user. As the user is expected to be naive, the user friendliness of *CIC* was one of the main considerations in its development. Before describing the user interface section, we will briefly discuss some of the main principles of user interface design.

User Interface Design Considerations

James Foley says:

> When a person uses an interactive graphics system to do real work, he wants the system to virtually disappear from his consciousness so that only his work and its ramifications have a claim on his energy. [8]

This can easily be applied to computer interfaces in general.

The user interface can be accomplished through a combination of hardware and software techniques. In this project, we concentrate on the software techniques.

One of the major and accepted principles of a user interface in an integrated system is its consistency in design. Consistency here refers to the dialogue design, command vocabulary and structure, error handling techniques, and use of the auxiliary factors such as colors and sounds. Consistency reduces the retention time

for the user. The user is likely to make less mistakes in such a system. Consistency also increases the user's self-confidence and his confidence in the system.

In an ideal system, the user can never make a mistake. However, in practice, this may not be possible. The designer needs to minimize the possibility of user errors. The system should also provide a capability to allow the user to undo an action that happens to be not what was wanted [17]. When the user makes a mistake, instead of displaying an error number, a short guiding message is always preferred.

The form of dialogue, in a user interface can be a crucial factor. There are several types of dialogues which can be embedded in a design. A few important ones are given below:

1. Questions and answers
2. Form filling
3. Query language
4. Menu selection
5. Hybrid dialogues
6. Function keys
7. Command language
8. Graphics interaction
9. Natural language
10. Parallel dialogues

The designer can select any or a combination of the above types to construct the dialogues.

Implementation of the User Interface

We considered *question and answers, form filling* and *menu selection* for the dialogues in *CIC*. Implementation of the *form filling* technique, although attractive was not possible because of the contraints of the language (Turbo Pascal). Hence, a combination of the *menu selection* and *questions and answers* techniques was implemented.

Questions and Answers: For consistency, throughout the program this technique is used in only two forms. In the first type the answer of a question is either "YES" or "NO". If the answer is "YES" the user needs to enter "Y" or "y". Any other key is taken as "NO". For example:

Do you have DXF file of floor plan ? (Y/any key)>

In the second type of question, the user is prompted to enter a value for a given quantity. These values are either integer or real numbers, for example:

Enter the U value of the wall >

If the user makes a mistake in answering, an error message is given and he is prompted with the same question again. In some cases,value possible answers are displayed to help the user in choosing the correct one.

As stated earlier, every CALPAS3 command has several arguments. After collecting the answers for all the arguments in a command, a chance is given to the user to check the entered data. If he is not satisfied, he can alter them. At this time the *menu selection* technique is used. For example after executing the command ROOF the following message appears on the screen :

The following values have just been entered :

Area of the roof : 1000.20 Sq. Ft.

U value of of the roof : 0.6

Absorptivity of the roof : 0.9

Do you want to change these values? (Y/any key) >

If the user enters "Y", the system displays:

1 Area of roof

2 U value of roof

3 Absorptivity of roof

4 Display entered values

5 Continue (no further changes)

Enter appropriate number>

Menu Selection: The menu form is designed so that the user can observe the entered data. If the entries are satisfactory, he may continue further. The options are always labeled numerically. The second to the last choice is always to display the entered values and the last choice is always to continue. Unfriendly label allocation such as A,B,C,D or "Wordstar" like commands such as "O" to copy a file or "Y" to erase it are strictly avoided.

Error checks are provided for the most likely errors. In such a case an error message is printed followed by the prompt for reentry. The process repeats until a valid answer is entered. This aspect can further be improved. If the user does

not enter the right answer within a couple of tries, the system should assume a default answer and carry on with the next question.

The error checks provided are of three types:

1. **Illegal Type of Value:** The user enters a decimal number instead of an integer. Or a letter is entered instead of a real or integer number.

2. **Illegal Values:** Sometimes the answers have to be specific, or there is a range of possible answers. It is possible that the user makes an error in entering legal values. Error checks are provided to capture these errors. For example, possible months can be 1 to 12 only. February cannot have more than 29 days even in a leap year. In most cases, the roof area cannot be less than floor area.

3. **Mistakes due to Human Memory:** Sometime the user needs to remember some information, e.g. drawing file names. It is possible that the user misspells or forgets the file name. For such a case we have developed a routine which uses BIOS and displays the DOS directory. The user can then choose the correct file name. But it is also possible that the file may not exist at all. In such a case the system cancels (undoes) the command and provides the alternative question.

CIC uses different colors and sounds to make the interface interesting. We start and end the program with beeps. Some systems give a beep when the user makes a mistake. But experience shows that it becomes very irritating after a few mistakes and may leave the user with a bad impression. So errors are not pointed out by beeping. Instead, error messages are displayed in a different color. Three colors are used in *CIC* to indicate the type of interaction in the program.

1. Color No. 1 (Yellow): for dialogues i.e. questions and answers and for menus.

2. Color No. 2 (Magenta): for error messages and warnings.

3. Color No. 3 (Green): for displaying the entered data.

To make the tool user friendly a possible range of answers is provided when needed. For example:

Enter no. of glazings (1 or 2) >

Apart from the error messages some warning messages are also given. When CALPAS3 calculates hourly energy consumption and costs it takes as much as an hour for execution. Many times the average users do not need such detailed reports. They need monthly and yearly reports. Before starting on such sections the user is warned that the section requires about one hour of execution time. He is then asked whether he would still like to go for it. If the user is willing to go for the hourly calculations, he is asked to reconfirm his reply. If the user enters information for only two walls, a warning message is given. (We assume that the building has at least three walls.) If the user still wants to continue with less than three walls, he is prompted with the next question. These warnings ensure that the user action is not a mistake.

4.2.3 Extraction of Information from AutoCAD Drawings

AutoCAD with its *advanced features (ADE-1, ADE-2, ADE-3)* provides two options for the extraction of drawing file information [1]:

1. Attributes

2. Drawing Interchange Files

Attributes is text information which can be associated with each occurrence of an object in the drawing [1]. The programmer can extract this information by writing a program in a suitable language. At present there is no AutoCAD database of building elements (objects) which can be used during the final design stage. As described in Section 2.1.1, examples of such objects are walls, windows and doors. At a later stage, when routines are written to draw these entities, *attributes* can be used to attach the information to the entities.

Software Arts, Inc. has provided the *Data Interchange Format.* Many software packages such as VisiCalc and Lotus 1-2-3 provide this format. This assists in interchanging data between different programs on different systems. AutoCAD also provides a *Drawing Interchange File Format* (*DXF* format) [1]. *DXF* files are standard ASCII files.

Using "DXFOUT", an AutoCAD command, drawing files can be converted into *Drawing Interchange File Format* before using *CIC*. *CIC* extracts the necessary information from these files.

A *Drawing Interchange File* is divided into five sections:

1. **Header Section:** This section contains general information about the drawing.

2. **Table Section:** This section contains information about line types, layers, styles, and views, etc.

3. **Block Section:** A *block* is a set of entities grouped together into a compound object [1]. This section contains block definitions.

4. **Entity Section:** This section contains drawing entities.

5. **End of the file.**

Each line of a *DXF* file is made up of two groups, a *group code* and a *group value*. The *group code* is an integer between 0 and 79. The nature of the *group value* depends on the corresponding *group code*. If the *group code* is between 0 and 9 the *group value* is a string. If the *group code* falls between 10 and 59 then the *group value* is a floating point number, and if the *group code* is between 60 and 79 the *group value* is an integer. Each *group code* is used for a specific item. For example, for *group code 6* the respective *group value* is a line type name. *Group code 10* is used to give the X coordinate of the point as the *group value*, whereas *group code 20* gives the Y coordinate of the point as the *group value*. Coordinates are real numbers and angles are always expressed in decimal degrees with zero degrees oriented to the east of the origin.

All the sections of a *DXF* file have a definite format. Therefore, it is possible to scan the sections and extract the needed information by another program. *CIC* uses two routines, *Reader* and *Convert* to scan these sections and extract the coordinates of the given item on the given layer of the drawing. These items are floor plans, roof plans, walls, and windows.

4.2.4 Use of Extracted Information to Create New Information

As stated in the previous section the routines *Reader* and *Convert* extract coordinates of walls, plans, and windows. Using these X and Y coordinates the respective areas are calculated. A plan, a wall, and a window in elevation is considered as a polygon. This polygon may be convex or concave. A circular plan or elevation is approximated as a polygon with several sides.

If a polygon is regular then calculating the area is a very simple and fast process. But often area calculations of an irregular polygon may take a lot of execution time. The method we have used is very quick. We will now discuss this method.

Let us consider a polygon *ABCDEFA* (Figure 4.2) with 6 (in general, n sides). Join all vertices of the polygon to the vertex say A, forming 4 (in general, n-2) triangles.

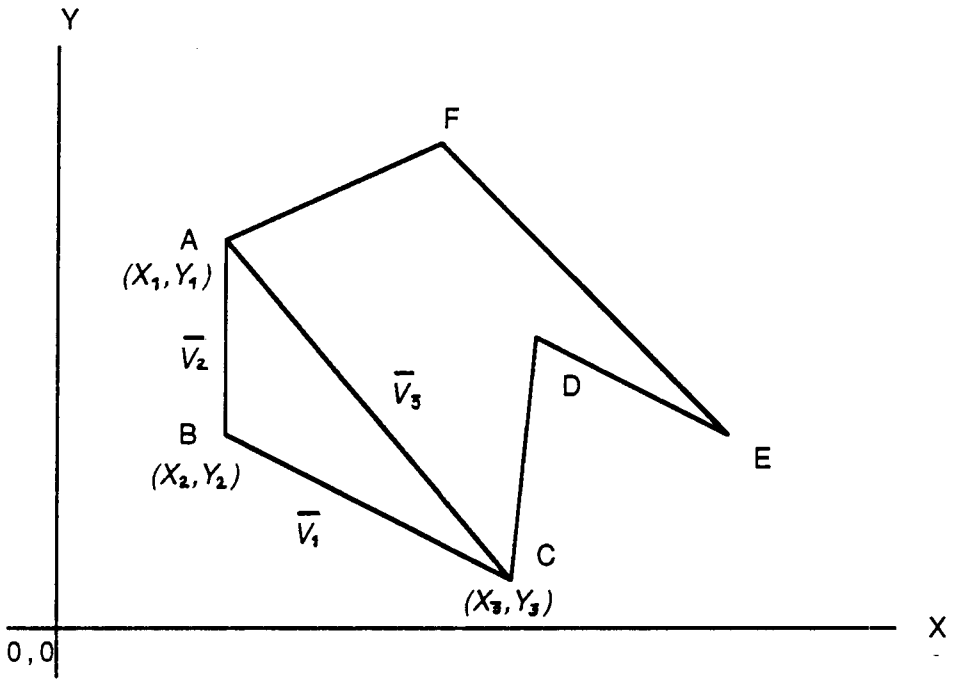

Figure 4.2: Polygon *ABCDEFA*

Assume,

PArea	denotes the area of the polygon
TArea	denotes the area of a triangle of the polygon, say ABC.

If the triangle ABC is expressed by its sides $\vec{V_1}, \vec{V_2}, \vec{V_3}$, given by coordinate pairs $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3)$ respectively,

$$\begin{aligned} \text{Area of the triangle} &= 1/2(\vec{V_2} \times \vec{V_3}) \\ &= 1/2((X_2 - X_1)(Y_3 - Y_1) - (X_3 - X_1)(Y_2 - Y_1)) \end{aligned}$$

and

$$\text{PArea} = \sum_{i=1}^{n-2} \text{TArea}_i$$

4.3 Summary

In summary, this chapter presents the idea of an interface between a CAD system and an Energy Analysis Simulation System. *CIC* is a tool to create an input file for CALPAS3. The tool can be used in design studios in schools or in the professional field. The user may use it for a particular design repetitively as he modifies the design. It can play an important role as an analytical tool in an integrated Computer-Aided Architectural *Design* system.

Chapter 5

Conclusions

In this thesis, *CAAD*, an integrated Computer-Aided *Design* system for architects is designed. The system is useful for the design phase of an architectural design project. A prototype model has been developed on an IBM PC AT running the DOS operating system to examine the feasibility of the system.

On the whole, *CAAD* provided an interesting exploration into the possibilities of developing an integrated CAD system for architects. It is certainly feasible and worthwhile to pursue further development of the system. *CAAD* also shows that expert system technology can be applied to the knowledge rich domain of architecture, without restraining the architect's creativity.

Section 5.1 gives a summary of the project. A discussion about future directions is presented in Section 5.2.

5.1 Summary of *CAAD*

CAAD aids in both stages of the architectural design phase: schematic design stage and final design stage.

At the schematic design stage, AutoCAD, an existing popular drafting system was chosen as the basic drafting tool. At this stage, a few routines are written in AutoLISP to create AutoCAD commands to draw quick sketches of architectural drawing entities such as windows, doors, walls, and staircases.

Evaluators are used at the schematic design stage. These *evaluators* are rule-based expert systems. *Energy Expert*, has been developed as an example of these *evaluators*. *Energy Expert* collects data from the drawing files, site database (which is created as a part of *CAAD*) and the user. Using this data, *Energy Expert* calculates the approximate energy performance of a building. Based on this performance it gives directives for further improvements. This module was developed in Turbo Prolog. The results are displayed graphically on the screen. The module needs to be developed further by adding more rules, directives, and appropriate heuristics. This module can be used as a model to develop other *evaluators* such as *Structural Expert.*

Routines to create AutoCAD commands to draw detailed architectural entities at the final design stage, have not been developed in the present project. Instead, AutoCAD without any such supporting routines was used as an architectural drafting package in the final design stage. In a final system, these routines need to be developed in AutoLISP.

An expert system that assists in the final design stage should tackle problems such as extracting accurate and detailed knowledge (information) about the the building design. There is also a need to develop routines to carry out accurate building analysis based on the knowledge. The process is a huge project in itself. However, sophisticated programs are available on the market to analyze a building accurately for its energy consumption, structural soundness, etc. Such available

programs can be linked with the current *CAAD*. As an example of such a module, an accepted energy analysis program, CALPAS3 [1] has been linked with the system. *CIC*, a program in Turbo Pascal has been developed to collect data from the AutoCAD drawing files, the user and the site database, and create an input file for CALPAS3. Using this file as the input, CALPAS3 creates the reports about the energy performance of the building. *CIC* is user friendly.

The *CAAD* system as it has been developed so far can serve as a basic model for futher stages of development.

5.2 Future Directions

CAAD has been developed as a prototype model for Computer-Aided *Design* in architecture. To make *CAAD* marketable much more work is needed.

Development of entities for architectural drafting will require extensive programming. Entities for drawing all the standard architectural objects need to be developed as AutoCAD commands. Examples of such entities are doors, windows, and staircases. Each entity should be a tree structure in which more specific entities are placed deeper in the tree. Section 2.2 gives the detail description of these entities.

At the schematic design stage, current *CAAD* offers expert systems for evaluating a design for simple energy and fire-safety performance. More rules can be added to allow the expert systems to handle more realistic problems. The systems have a knowledge base of directives to make the building more energy efficient. More directives can be written to make the knowledge base richer. That may necessitate using heuristics to choose the appropriate directives. Currently, *En-*

ergy Expert displays a graph of seasonal energy performance, two pie charts of the seasonal costs of cooling and heating along with advice of how to make the building more energy efficient. Although this program is capable of finding many possible modifications in the design, it displays the first two items of the list of modifications. This is due to a lack of space on the screen. Arrangements could be made to display all the modifications. At the schematic design stage, a rule-based expert system for quickly evaluating structural soundness of the building design needs to be developed. *Fire-Safety Expert* needs to be enhanced.

The current database consists of information for 70 cities in the U.S.A. The database should be expanded. At present, the final design stage of *CAAD* offers only the interface between CALPAS3 and AutoCAD. Sophisticated programs are available on the market for accurate structural analysis. One of these programs can be interfaced with the architectural drafting package. The author does not know of any commercial program that carries out an accurate fire-safety check at the final design stage. A program which gets data from the drawing files and the user and checks the fire-safety of the building needs to be developed and interfaced with the architectural drafting package.

In this thesis *CAAD* has been developed only for residential buildings. Further work is needed to make *CAAD* workable for all types of buildings such as schools, universities, recreational buildings, and commercial buildings.

The expert system module has been developed in Turbo Prolog. After working with standard Prolog, programming in Turbo Prolog becomes a tedious job. The predicates and all variable types need to be declared. This tool does not offer several functions such as set_of, and bag_of which are found in standard Prolog. User defined operators are also not permitted. The 'OR' operation has a

constraint of only one 'OR' in the rule condition. These drawbacks make the programming task difficult. For further development it is recommended that the code be translated into Quintus Prolog on an IBM PC AT.

At present, three languages Turbo Prolog, Turbo Pascal and AutoLISP have been used. It would be worthwhile to explore the possibility of using only one language throughout the project without sacrificing the advantages gained by using three different languages. Using one language would make the system easier to maintain.

CAAD runs on an IBM PC AT. Efforts need to be made to run *CAAD* on a variety of commonly used machines.

BIBLIOGRAPHY

Bibliography

[1] AUTOCAD Reference Manual, AutoDesk Inc., Sausalito, Ca., 1986.

[2] Bovill Carl, "UCSD Pascal Intrinsic Units Used to Create a File Drive Early Design Energy Analysis Program With Input Error Checking Automatically Provided by the UCSD Pascal Compiler," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[3] CALPAS3 User Manual, Berkeley Solar Group, Berkeley, Ca., 1984.

[4] Cole Ivan, Mark Lansdale, and Bruce Christie, "Dialogue Design Guidelines," (Chapter 10), *Human Factors of Information Technology in the Office* (Ed. Bruce Christie), John Wiley and Sons, Chichester, 1985.

[5] Cristensen Craig, "Energy Mapping and Analysis Procedures (EMAPS) as Building Diagnostics," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[6] D'Ambra, Charles A., "Evaluation of the Income Required to Qualify for an Energy Efficient Home Mortage," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[7] Flemming Ulrich, Gerhard Schmitt, "The Computer in the Design Studio. Ideas and Exercises that Go Beyond Automated Drafting," *ACADIA Workshop, '86 Proceedings* (Ed. James A. Turner), University of Houston, Texsas, 1986.

[8] Foley, James D., Victor L. Wallace, and Peggy Chan, "The Human Factors of Computer Graphics Interaction Techniques," *IEEE Computer Graphics and Applications*, November 1984, Vol. 4, pp. 13-48.

[9] Handbook of Fundamentals, American Society of Heating, Refrigerating and Air-Conditioning Engineers, New York, 1972.

[10] Hart, Kimball G., John M. Kurtz, William I. Whiddon, "Energy Graphics, a New Approach to Energy-Conscious Design," Booze Allen & Hamilton Inc., U.S. Department of Energy, Washignton D.C., 1980.

[11] Hood, John D., "Easy AutoCAD," McGraw-Hill, Inc., New York, 1988.

[12] "Integrated Interactive Computing Systems," (Ed. Dogano P., and E. Sandwall), North-Holland Publishing Company, Oxford, 1983.

[13] "Introductory Readings in Expert Systems," (Ed. Donald Michie), Gordon and Breach Science Publishers, New York, 1982.

[14] Koenigsberger, O.H., T.G. Ingersoll, Alan Mayhew and S.V. Szokolay, "Manual of Tropical Housing and Building, Part one: Climatic Design," Orient Longman, New Delhi, 1975.

[15] Kusuda T., and Ishii K., "Hourly Solar Radiation Data for Vertical and Horizontal Surfaces on Average Days in the United States and Canada," National Bureau of Standards Building Science Series 96, Washington D.C., 1977.

[16] Levy, Emanuel M., Deane Evans, and Cynthia Gardstein, "Passive Solar Construction Handbook," Rodale Press, Emmaus, Pa., 1983.

[17] Loffman, Regis S., "User Interface For A Partially Incomplete System Software Environment With Non-ADP Users," Thesis, The University of Tennessee, Knoxville, Tn., 1987.

[18] Merritt Dennis, "Forward Chaining in Prolog," *AI Expert*, San Francisco, Ca., November 1986.

[19] MicroCAD Manual, Imagimedia Technologies, Inc., San Francisco, CA., 1984.

[20] Milne Murray, "Ten Principles for the Design of User-Friendly Design Tools," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[21] Mitchell, William J., "Computer-Aided Archtectural Design," Petrocelli/Charter, New York, 1977.

[22] NFPA 101 Life Safety Code 1985, National Fire Protection Association, Ma, 1985.

[23] Ozel Filiz, "Fire Safety Codal Evaluation and Computer Aided Design," *Architectural Publications of Design & Technology Research*, The American Institute of Architects, Los Angeles, Ca., 1985.

[24] Ozel Filiz, "Using CAD in Fire safety Research," *ACADIA Workshop, '85 Proceedings* (Ed. Patricia G. McIntosh), Arizona State University, Arizona, 1985.

[25] Rees Jeff, "Energy Models: Exploring the Mini - and Microcomputers Applications," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[26] Schmitt, Gerhard N., "Microcomputer Based Integrated Energy Design Expert Systems: A powerful Tool," *The First National Conference on Microcomputer Applications for Conservation and Renewable Energy (NCMACRE) Proceedings* (Ed. Donald E. Osborn), The University of Arizona, Tucson, 1985.

[27] Selby, Samuel M., "Standard Mathematical Tables," The Chemical Rubber Co., Ohio, 1977.

[28] TurboProlog, Owner's Handbook, Borland International Inc., California, 1986.

[29] Waterman, Donald A., "A Guide to Expert Systems," Addison- Wesley Publishing Company, Massachusetts, 1978.

[30] Watson, Donald, "Designing & Building A Solar House," Gardenway Publishing, Vermont, 1977.

[31] Winston, Patrick Henry, "Artificial Intelligence," Addison Wesley Publishing Company, Massachusetts, 1984.

[32] Yessios Chris, "What Has Yet To Be CAD," *ACADIA Workshop, '86 Proceedings* (Ed. James A. Turner), University of Houston, Texsas, 1986.

APPENDICES

APPENDIX A

The appendix gives a brief description of the CALPAS3 commands implemented in *CIC*. The commands are capitalized and underlined. The arguments in the commands are capitalized and indented.

TITLE

- Title of the each page of the CALPAS3 output file
- Required command
- Legal values: string of maximum 47 characters

SITE

- Gives information about the site
- Optional command

LAT

- Latitude of the site
- Optional argument
- Default value: latitude of the weather station
- Legal values: real numbers from -89.99 to +89.99
- Units: degrees south when negative value
  degrees north when positive value

LONG

- Longitude of the site
- Optional argument
- Default value: longitude of the weather station
- Legal values: real numbers from -180 to +180
- Units: degrees east when negative value
  degrees west when positive value

TZN

- Time zone of the site based on the Greenwich mean time
- Optional argument
- Default value: time zone of the weather station
- Legal values: integers from 0 to 23
- Units: unitless

AZMSOUTH

- Overall building orientation
- Optional command
- Default value: 0 (i.e. due south)
- Legal values: real numbers from -180 to 180
- Units: degrees east due south when negative value
  degrees west due south when positive value

GREFLECT

- Ground reflectivity of each window for each month
- Optional command
- Default value: 0.2
- Legal values: real numbers from 0 to 1
- Units: unitless

HOUT

- Outside film coefficient
- Optional command
- Default value: 2.64
- Legal values: 1.47 or greater real number
- Units: $Btu/ft^2\,^\circ F$

HOUSE

- Gives information about the building
- Required command

FLRAREA

- Floor area
- Required argument
- Legal values: real numbers 1 or greater
- Units: ft^2

VOL

- Volume of the building
- Optional argument
- Default value: FLRAREA×8.0
- Legal values: real numbers 0 or greater
- Units: ft^3

CHIR

- Heat capacity of light
- Optional argument
- Default value: FLRAREA×1.5 + 1000
- Legal values: real numbers 1 or greater
- Units: $Btu/°F$

ROOF

- Gives characteristics of the roof
- Optional command

AREA

- Area of the roof
- Optional argument
- Default value: FLRAREA
- Legal values: real numbers 0 or greater
- Units: Ft^2

AZM

- Orientation(azimuth) of the roof
- Optional argument
- Default value: 0 (i.e. due south)
- Legal values: real numbers from -180 to +180
- Units: degrees east due south when negative value
  degrees west due south when positive value

TILT

- Tilt of the roof
- Optional argument
- Default value: 0 (i.e. horizontal)
- Legal values: real numbers from 0 to 90
- Units: degrees

UVAL

- U value of the roof
- Optional argument
- Default value: 0.05 (approximates an R-19 roof)
- Legal values: real numbers 0 or greater

- Units: $Btu/ft^2{}^\circ F$

INSIDE

- Material in contact with inside roof surface
- Optional argument
- Default value: AIR
- Legal values: AIR or EXWALL

WALL

- Wall charactristics
- Optional command
- The following arguments are to be repeated for each wall

NAME

- Name of the wall
- Optional argument
- Default value: CALPAS3 makes its own name
- Legal values: string of 10 characters

AREA

- Area of the wall
- Required argument
- Legal values: real numbers 0 or greater
- Units: Ft^2

AZM

- Wall orientation (Azimuth)
- Required argument
- Default value: 0 (i.e. due south)
- Legal values: real numbers -180 to 180
- Units: degrees east of due south when negative value
  degrees west of due south when positive value

TILT

- Tilt of the wall
- Optional argument
- Default value: 90° (i.e. vertical)
- Legal values: real numbers from 0 to 180
- Units: degrees

ABSRP

- Absorptivity of the wall
- Optional argument
- Default value: 0.5 (i.e. approximate for a wall fully exposed to sun)
- Legal values: real numbers between 0 and 1
- Units: unitless

INSIDE

- Material in contact with the inside surface of the wall
- Optional argument
- Default value: AIR
- Legal values: AIR or EXWALL
- Units: unitless

SLAB

- Slab characteristics
- Optional command
- SLAB is used to model massive floors
- It can be used to represent thermal storage found in passive houses.

AREA

- Area of the slab
- Required argument
- Legal values: real numbers 1 or greater
- Units: Ft^2

THKNS

- Thickness of the wall
- Required argument
- Legal values: real numbers 1 or greater
- Units: inches

Material

- Slab material
- Required argument
- Legal values: names from the Table No. 5 of [3] or
  user defined material
- Units: unitless

HTAHS

- Heat transfer coefficient from inside surface of the wall
- Optional argument
- Default value: 1.3
- Legal values: real numbers 0 or greater
- Units: $Btuh/Ft^2\,^\circ F$

RSURF

- Resistance at the inside surface of the wall
- Optional argument
- Default value: 0
- Legal values: real numbers 0 or greater
- Units: $Ft^2\,^\circ F/Btuh$

GLASS

- Window (glass) characteristics
- Optional command
- The command is repeated for each opening

NAME

- Name of the glass
- Optional argument
- Default value: CALPAS3 makes its own name
- Legal values: string of maximum 10 characters
- Units: unitless

AREA

- Area of the glass
- Required argument
- Legal values: real numbers 0 or greater
- Units: Ft^2

AZM

- Glass orientation (Azimuth)
- Required argument
- Legal values: real numbers from -360 to +360, (0 is due south)
- Units: degrees east due south when negative value
  degrees west due south when positive value

TILT

- Tilt of the glass
- Optional argument
- Default value: 90° (i.e. vertical)
- Legal values: real numbers from 0 to 180
- Units: degrees

NGLZ

- Number of glazings
- Optional argument
- Default value: 2
- Legal values: 1 or 2
- Units: unitless

UVAL

- Heat transfer coefficient of glass to the outside temperature
- Optional argument
- Default value: 0.6 for double glazing and
  1.07 for single glazing
- Legal values: real numbers 0 or greater
- Units: $Btuh/Ft^2\,°F$

GLSTYP

- Type of glass
- Optional argument
- Default value: 1
- Legal values: refer to the Table No. 5 [3]
- Units: unitless

SHADING

- Shading overhang and fins for the window currently being described
- Optional command
- 'Left' and 'right' are defined from outside of the window

WHEIGHT

- Window height
- Required argument
- Legal values: real numbers 0.01 or greater

- Units: feet

WWIDTH

- Window width
- Required argument
- Legal values: real numbers 0.01 or greater
- Units: feet

OHDEPTH

- Overhang depth
- Optional argument
- Default value: 0
- Legal values: real numbers 0 or greater
- Units: feet

OHWD

- Distance from top of the window to overhang
- Optional argument
- Default value: 0
- Legal values: real numbers 0 or greater
- Units: feet

OHLX

- Distance overhang extends to left of the window
- Optional argument
- Default value: 0
- Legal values: real numbers 0 or greater
- Units: feet

OHRX

- Distance overhang extends to right of the window
- Optional argument
- Default value: 0
- Legal values: real numbers 0 or greater
- Units: feet

OHFLAP

- Height of the flap extending down from the end of overhang
- Optional argument

- Default value: 0
- Legal values: real numbers 0 or greater
- Units: feet

DAYTIMES

- Specifies beginning and end hours of the day in each season
- Optional command
- The command is used in the shutter control algorithm of CALPAS3

WDBEG

- Beginning of winter day
- Optional argument
- Default value: 8
- Legal values: integers from 1 to 24
- Units: o'clock

WDEND

- End of winter day
- Optional argument
- Default value: 19
- Legal values: integers from 1 to 24
- Units: o'clock

SDBEG

- Beginning of summer day
- Optional argument
- Default value: 8
- Legal values: integers from 1 to 24
- Units: o'clock

SDEND

- End of summer day
- Optional argument
- Default value: 19
- Legal values: integers from 1 to 24
- Units: o'clock

OPCOST

- Specifies energy prices and equipment operating efficiencies
- Optional command

ELPRICE

- Price of electricity
- Required argument
- Legal values: real numbers 0 or greater
- Units: $dollars/KWh$

ACCOP

- Air conditioner seasonal coefficient of performance
- Optional argument
- Default value: 2
- Legal values: real numbers 0 or greater
- Units: unitless

Heating

- Heating type
- Required argument
- Default value: FUEL
- Legal values: FUEL or ELECTRIC
- Units: unitless

If HEATING = FUEL then

FUELPRICE

- Cost of FUEL
- Required argument
- Legal values: real numbers 0 or greater
- Units: $dollars/1000Btu$

HTEFF

- Seasonal furnace efficiency
- Optional argument
- Default value: 0.6
- Legal values: real numbers 0 to 1
- Unit: unitless

If HEATING = ELECTRIC then

HTCOP

- Seasonal coefficient of performance of the heating system
- Optional argument
- Default value: 1
- Legal values: real numbers 1 or greater
- Unit: unitless

SOLARPRINT

- Period of days for which a report of solar factor is generated
- Optional command
- command is repeated for any number of periods

FIRSTDAY

- First day to print
- Required argument
- Legal values: MMM-DD where

  MMM = first three letters of the month name

  DD = day
- Units: unitless

LASTDAY

- Last day to print
- Optional argument
- Default value: FIRSTDAY
- Legal values: MMM-DD where

  MMM = first three letters of the month name

  DD = day
- Units: unitless

REPORT1 =

REPORT2 =

REPORT3 =

- Reports to be printed
- Optional command
- Default value: all the three repots are printed
- Legal values: WALL, GLSWNTR, GLSSMR
- Units: unitless

PRINTDAILY

- Period of days for which a detailed report on energy balances conditions and/or temperatures is desired
- Optional command
- command is repeated for any number of periods

FIRSTDAY

- First day to print
- Required argument
- Legal values: MMM-DD where
  MMM = first three letters of the month name
  DD = day
- Units: unitless

LASTDAY

- Last day to print
- Optional argument
- Default value: FIRSTDAY
- Legal values: MMM-DD where
  MMM = first three letters of the month name
  DD = day
- Units: unitless

REPORT1 =

REPORT2 =

REPORT3 =

REPORT4 =

- Reports to be printed
- Optional command
- Default value: all the three repots are printed
- Legal values: HSEB, SSEB, RSEB, COND
- Units: unitless

PRINTHOURLY

- Period of days for which a detailed hourly report of solar factor is generated
- Optional command
- command is repeated for any number of periods

FIRSTDAY

- First day to print
- Required argument
- Legal values: MMM-DD where

  MMM = first three letters of the month name

  DD = day
- Units: unitless

LASTDAY

- Last day to print
- Optional argument
- Default value: FIRSTDAY
- Legal values: MMM-DD where

  MMM = first three letters of the month name

  DD = day
- Units: unitless

REPORT1 =

REPORT2 =

REPORT3 =

REPORT4 =

REPORT5 =

- Reports to be printed
- Optional command
- Default value: all reports are printed
- Legal values: HSEB, SSEB, RBEB, COND, TEMP
- Units: unitless

END

- The terminating command of CALPAS3 input file
- Required argument.

APPENDIX B

AutoLISP routine to draw a two-flight staircase using AutoCAD.

```
; Function to convert angle in degrees to radians.
(defun dtr (a)
        (* pi (/ a 180.0))
)

; Function to acquire information about the staircase interactively from the user.
(defun gpuser1 ()
        (setq sp (getpoint "\nEnter base point of the staircase: "))
        (setq ep (getpoint "\nEnter width of the staircase from the base point: "))
        (setq l (getpoint "\nEnter length of the staircase from the base point: "))
        (setq trade (getdist "\nEnter trade width from the base point: "))
        (setq width (distance sp ep))
        (setq plength (distance sp l))
        (setq hangle (angle sp ep))
        (setq pangle (angle sp l))
        (setq hwidth (/ width 2))
)
```

```
; Function to draw the staircase-well.
( defun drawoutwell ()
        ( command "pline"
            ( setq p ( polar sp hangle width))
            ( setq p ( polar p pangle plength))
            ( setq p ( polar p (+ hangle ( dtr 180 )) width ))
            ( polar p (+ pangle ( dtr 180)) plength)
            "close"
        )
)

; Function to draw a step of the staircase.
; pdist is the distance of the trade from the base point
( defun drawoutstep (pdist)
        ( command "pline"
            ( setq p ( polar sp pangle pdist))
            ( polar p hangle width )
            ""
        )
)

; Function to draw the hand-railing line.
( defun drawoutrailing (tlength)
        ( setq p ( polar sp pangle hwidth))
        ( command "pline"
            ( setq p ( polar p hangle hwidth))
            ( polar p pangle tlength )
            ""
        )
)
```

```
; Function to draw steps of the staircase.
( defun drawoutsteps ()
        ( setq pdist (+ hwidth 0.0))
        ( setq off 0.0)
        ( while (≤ pdist (- plength hwidth))
            ( drawoutstep pdist)
            ( setq off (+ off 1.0))
            ( setq pdist (+ pdist trade))
        )
        ( setq off (- off 1.0))
        ( setq tlength (* off trade))
        ( drawoutrailing tlength)
)

; Driving routine.
( defun C:STAIR2 ()
        ( gpuser1)
        ( setq sblip (getvar "blipmode"))
        ( setq scmde (getvar "cmdecho"))
        ( setvar "blipmode" 0)
        ( setvar "cmdecho" 0)
        ( drawoutwell)
        ( drawoutsteps)
        ( setvar "blipmode" sblip)
        ( setvar "cmdecho" scmde)
)
```

VITA

Bharati E. Jog graduated from Sir J. J. College of Architecture, University of Bombay with a Bachelor degree in Architecture in 1981. Upon completing her undergraduate degree, she worked with an internationally known contemporary architect. At the same time, she built few buildings independently as an architect.

She joined the University of Arizona in 1983 from where she received her Master of Architecture with a concentration in Computer Applications in Architecture in August 1985. In September 1985 she entered the Department of Computer Science at the University of Tennessee. She received her M.S. in Computer Science from UT in March 1988.

Since January 1988, she is an Assistant Professor in the School of Architecture and Interior Design at the University of Cincinnati, Ohio.