An Expert System Based Engineering Design Aid

by

Yanming Hou

A Thesis Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

Department of Biosystems Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

ISBN 0-612-16161-7

Name _____

Dissertation Abstracts International and *Masters Abstracts International* are arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation or thesis. Enter the corresponding four-digit code in the spaces provided.

Agricultural Engineering
SUBJECT TERM

0539
SUBJECT CODE

UMI

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Nom _____

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

_____ ☐☐☐ U·M·I

SUJET — CODE DE SUJET

Catégories par sujets

HUMANITÉS ET SCIENCES SOCIALES

COMMUNICATIONS ET LES ARTS
Architecture 0729
Beaux-arts 0357
Bibliothéconomie 0399
Cinéma 0900
Communication verbale 0459
Communications 0708
Danse 0378
Histoire de l'art 0377
Journalisme 0391
Musique 0413
Sciences de l'information 0723
Théâtre 0465

ÉDUCATION
Généralités 515
Administration 0514
Art 0273
Collèges communautaires 0275
Commerce 0688
Économie domestique 0278
Éducation permanente 0516
Éducation préscolaire 0518
Éducation sanitaire 0680
Enseignement agricole 0517
Enseignement bilingue et multiculturel 0282
Enseignement industriel 0521
Enseignement primaire. 0524
Enseignement professionnel 0747
Enseignement religieux 0527
Enseignement secondaire 0533
Enseignement spécial 0529
Enseignement supérieur 0745
Évaluation 0288
Finances 0277
Formation des enseignants 0530
Histoire de l'éducation 0520
Langues et littérature 0279
Lecture 0535
Mathématiques 0280
Musique 0522
Orientation et consultation 0519
Philosophie de l'éducation 0998
Physique 0523
Programmes d'études et enseignement 0727
Psychologie 0525
Sciences 0714
Sciences sociales 0534
Sociologie de l'éducation 0340
Technologie 0710

LANGUE, LITTÉRATURE ET LINGUISTIQUE
Langues
- Généralités 0679
- Anciennes 0289
- Linguistique 0290
- Modernes 0291

Littérature
- Généralités 0401
- Anciennes 0294
- Comparée 0295
- Médiévale 0297
- Moderne 0298
- Africaine 0316
- Américaine 0591
- Anglaise 0593
- Asiatique 0305
- Canadienne (Anglaise) 0352
- Canadienne (Française) 0355
- Germanique 0311
- Latino-américaine 0312
- Moyen-orientale 0315
- Romane 0313
- Slave et est-européenne 0314

PHILOSOPHIE, RELIGION ET THÉOLOGIE
Philosophie 0422
Religion
- Généralités 0318
- Clergé 0319
- Études bibliques 0321
- Histoire des religions 0320
- Philosophie de la religion 0322

Théologie 0469

SCIENCES SOCIALES
Anthropologie
- Archéologie 0324
- Culturelle 0326
- Physique 0327

Droit 0398
Économie
- Généralités 0501
- Commerce-Affaires 0505
- Économie agricole 0503
- Économie du travail 0510
- Finances 0508
- Histoire 0509
- Théorie 0511

Études américaines 0323
Études canadiennes 0385
Études féministes 0453
Folklore 0358
Géographie 0366
Gérontologie 0351
Gestion des affaires
- Généralités 0310
- Administration 0454
- Banques 0770
- Comptabilité 0272
- Marketing 0338

Histoire
- Histoire générale 0578
- Ancienne 0579
- Médiévale 0581
- Moderne 0582
- Histoire des noirs 0328
- Africaine 0331
- Canadienne 0334
- États-Unis 0337
- Européenne 0335
- Moyen-orientale 0333
- Latino-américaine 0336
- Asie, Australie et Océanie 0332

Histoire des sciences 0585
Loisirs 0814
Planification urbaine et régionale 0999
Science politique
- Généralités 0615
- Administration publique 0617
- Droit et relations internationales 0616

Sociologie
- Généralités 0626
- Aide et bien-être social 0630
- Criminologie et établissements pénitentiaires 0627
- Démographie 0938
- Études de l' individu et de la famille 0628
- Études des relations interethniques et des relations raciales 0631
- Structure et développement social 0700
- Théorie et méthodes. 0344
- Travail et relations industrielles 0629

Transports 0709
Travail social 0452

SCIENCES ET INGÉNIERIE

SCIENCES BIOLOGIQUES
Agriculture
- Généralités 0473
- Agronomie. 0285
- Alimentation et technologie alimentaire 0359
- Culture 0479
- Élevage et alimentation 0475
- Exploitation des pâturages 0777
- Pathologie animale 0476
- Pathologie végétale 0480
- Physiologie végétale 0817
- Sylviculture et faune 0478
- Technologie du bois 0746

Biologie
- Généralités 0306
- Anatomie 0287
- Biologie (Statistiques) 0308
- Biologie moléculaire 0307
- Botanique 0309
- Cellule 0379
- Écologie 0329
- Entomologie 0353
- Génétique 0369
- Limnologie 0793
- Microbiologie 0410
- Neurologie 0317
- Océanographie 0416
- Physiologie 0433
- Radiation 0821
- Science vétérinaire 0778
- Zoologie 0472

Biophysique
- Généralités 0786
- Medicale 0760

SCIENCES DE LA TERRE
Biogéochimie 0425
Géochimie 0996
Géodésie 0370
Géographie physique 0368
Géologie 0372
Géophysique 0373
Hydrologie 0388
Minéralogie 0411
Océanographie physique 0415
Paléobotanique 0345
Paléoécologie 0426
Paléontologie 0418
Paléozoologie 0985
Palynologie 0427

SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT
Économie domestique 0386
Sciences de l'environnement 0768
Sciences de la santé
- Généralités 0566
- Administration des hôpitaux 0769
- Alimentation et nutrition 0570
- Audiologie 0300
- Chimiothérapie 0992
- Dentisterie 0567
- Développement humain 0758
- Enseignement 0350
- Immunologie 0982
- Loisirs 0575
- Médecine du travail et thérapie 0354
- Médecine et chirurgie 0564
- Obstétrique et gynécologie 0380
- Ophtalmologie 0381
- Orthophonie 0460
- Pathologie 0571
- Pharmacie 0572
- Pharmacologie 0419
- Physiothérapie 0382
- Radiologie 0574
- Santé mentale 0347
- Santé publique 0573
- Soins infirmiers 0569
- Toxicologie 0383

SCIENCES PHYSIQUES
Sciences Pures
Chimie
- Généralités 0485
- Biochimie 487
- Chimie agricole 0749
- Chimie analytique 0486
- Chimie minérale 0488
- Chimie nucléaire 0738
- Chimie organique 0490
- Chimie pharmaceutique 0491
- Physique 0494
- PolymÈres 0495
- Radiation 0754

Mathématiques 0405
Physique
- Généralités 0605
- Acoustique 0986
- Astronomie et astrophysique 0606
- Électronique et électricité 0607
- Fluides et plasma 0759
- Météorologie 0608
- Optique 0752
- Particules (Physique nucléaire) 0798
- Physique atomique 0748
- Physique de l'état solide 0611
- Physique moléculaire 0609
- Physique nucléaire 0610
- Radiation 0756

Statistiques 0463

Sciences Appliqués Et Technologie
Informatique 0984
Ingénierie
- Généralités 0537
- Agricole 0539
- Automobile 0540
- Biomédicale 0541
- Chaleur et thermodynamique 0348
- Conditionnement (Emballage) 0549
- Génie aérospatial 0538
- Génie chimique 0542
- Génie civil 0543
- Génie électronique et électrique 0544
- Génie industriel 0546
- Génie mécanique 0548
- Génie nucléaire 0552
- Ingénierie des systèmes 0790
- Mécanique navale 0547
- Métallurgie 0743
- Science des matériaux 0794
- Technique du pétrole 0765
- Technique minière 0551
- Techniques sanitaires et municipales 0554
- Technologie hydraulique 0545

Mécanique appliquée 0346
Géotechnologie 0428
Matières plastiques (Technologie) 0795
Recherche opérationnelle 0796
Textiles et tissus (Technologie) 0794

PSYCHOLOGIE
Généralités 0621
Personnalité 0625
Psychobiologie 0349
Psychologie clinique 0622
Psychologie du comportement 0384
Psychologie du développement 0620
Psychologie expérimentale 0623
Psychologie industrielle 0624
Psychologie physiologique 0989
Psychologie sociale 0451
Psychométrie 0632

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION

AN EXPERT SYSTEM BASED ENGINEERING DESIGN AID

BY

YANMING HOU

A Thesis/Practicum submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

DEDICATION

To my parents

"Your love, support, encouragement, and expectancy are
the inexhaustible spiritual resources"

Abstract

An Expert System-based Basic Engineering Design Aid (BEDA) has been developed and furthermore verified for wood-frame building design. General rules were researched, extracted, and encoded to automate the creation and analysis processes of engineering design. Code constraints and loading parameters appropriate for Canadian wood design procedures have been used in the development. Other design codes and procedures can be incorporated to permit its use in different jurisdictions.

The basic idea of the research was to create a knowledge-based software package. It streamlines the design process and makes it faster, easier, and more economical. The package is not limited to a single design stage or a narrow application. It is appropriate for the whole design process, from initial sketches, through analysis, to final design document. BEDA applies common rules of engineering design that complement the input of the design engineer. Its working style is designed to mirror the practice and customs of human designers. Both machine reasoning and human judgement are employed to control design processes.

BEDA used wood-framed buildings as the vehicle for system development. It has such function units as External Load Calculation, Graphical Editor, Finite Element Analysis, and Member Design. The end user can start a design from sketching or existing design examples. Examples can be brought in from a built-in case base. The case base contains current available templates of structural elements. External graphic software packages can be activated from within BEDA for sketching. BEDA retrieves the user's sketch into its knowledge base for automatic decomposition and load calculation. After the user's verification, BEDA calls a finite element analysis tool to analyze the internal stresses and the displacements of each structural element. Based on the analysis and other available information, a tentative design is selected. BEDA supports detailed designs of individual elements. The design of a building with BEDA is an iterative process of structural sketching, load calculation, and successive member design. Final design output will include drawings, materials lists, and specifications.

BEDA used several commercially available software packages to accomplish its sophisticated functions, Paradox for database management, AutoSketch (or AutoCad) for graphics editing, and Borland Pascal for special programming. Level5 Object was used as the development shell and a server which integrate all function units into an package. With the built-in case base, BEDA is not only a decision supporter but also a tutorial tool for new designers.

ACKNOWLEDGEMENT

I would like to give my sincere thanks to the following:

Dr. M.G. Britton, my main advisor. He introduced me to and guided me through this project. His concern, advisement and encouragement have enabled me successfully complete this program.

Dr. Q. Chang and Dr. M. Evans, my committee members. Their helpful discussions and valuable suggestions greatly increased the depth and width of this study.

Dr. B.A. Engel of Purdue University. His careful reviewing and constructive suggestions enhanced the quality of this thesis.

The Department of Biosystems Engineering and the School of Graduate Studies. Their financial support guaranteed the smoothness of my study.

My wife, Jie Zhou, the most important person in my life. Her love, understanding, encouragement, support, and sacrifice have been critical and invaluable to my study. Every word in this thesis comes to physical existence through her fingertips.

Table of Contents

1. INTRODUCTION

1.1 Overview

In October of 1989, a study on the feasibility of applying artificial intelligence (AI) to engineering was submitted to PRECARN Associates by a consortium of three Canadian aerospace companies and four universities [PRECARN Associates, 1989]. The report suggested that Canadian engineering creativity and productivity could be doubled by the development of artificial intelligence based software for use in design. It is expected that expert system techniques could automate a major part of routine works in the design process. Engineers would then be able to concentrate on more creative design activities. Initiating the feasibility study reflected the earnest desire from both industry and academe to computerize, or more exactly, to automate the design process. Given the wide variability of engineering practice and its effects on industry, the prospect is absolutely exciting.

The research work presented in this dissertation has evolved from the PRECARN study.

1.2 The Problem

Applying artificial intelligence to engineering design demands insights into both human's design process and the way in which artificial intelligence works. Unfortunately, general principles governing the engineering design are not yet clearly understood, but are required for use as guidelines in developing knowledge-based design (KBD) expert systems.

Engineering design is a sophisticated problem solving process. There is no fixed logic with its reasoning method. Opportunistic thinking and the trial and error method are commonly used.

> "... despite modern technical advances, good engineering is still as much a matter of intuition and nonverbal thinking as of equations and computation." [Ferguson 92]

Engineering design is seldom a straightforward process, especially for a product consisting of more than one component. Design of the product as a whole and designs of its components are interrelated and inter-constrained. Intelligence, experience, and personal preference play decisive roles in decoupling locked design loops.

A number of knowledge sources contribute to the solution of a design problem. Design orientations, design theories, industry codes, environmental conditions, and available analysis tools are all important factors. They present in polymorphous forms and act in diversified ways. It has been an annoying problem to organize them into a unified knowledge base and activate the appropriate part at the right time.

Engineers communicate through engineering drawings. An engineering design normally starts from sketches. Engineers conceive initial design ideas and sketch them out. After calculations, analyses, and revisions, sketches eventually evolve into final drawings. While most graphic software packages function only in the drawing process, none of analysis software use graphics as subjects for analysis. Graphic software handle graphics, where as analysis software work in algorithmic languages. The deficiency of links between them is mostly due to the difficulty of representing graphics in algorithmic languages.

Engineering design seldom starts from nothing. A successful design is usually based on experience with both successes and failures. It is a prevailing practice to use existing designs as reference in a design [Petroski, 1992]. There is no right or wrong design, only a better or worse one. Perfect design is never attained. Rooms and demands for improvement and advancement are always there. A reasonable desire is that the design aid can prompt engineers with successful examples and warn them of possible errors. Huge storage capacity of computers provides the physical possibility for developing a design case resource. The question is whether a case base can be built and managed in a similar manner to that used for a database. A database contains numbers and values. A case base contains text and graphics.

Historically, most engineers were trained for certain ways of design. Commonly accepted design procedures may even be fostered as design custom after years of practice. It is easier and maybe efficient to follow design routines, but creativities are hindered.

On one side, knowledge based systems are expected to provide new design approachs and promote creative designs. On the other side, it is desirable that the system still function in a way similar to common engineering practice. So that engineers can easily adapt to the new tools. No matter how a system works inside, its interface must be user-friendly and follow normal engineering expressions as much as possible.

Expert systems have been developed to address specific engineering problems. However, current applications of AI techniques are limited to clearly defined problems with a neat knowledge source and simple pattern of problem solving. There is no widely accepted approach to handle engineering design in a general, universal, and inclusive way. Existing techniques can readily encode equations and computations into computer programs. Some parts of the intuition and nonverbal thinking, such as expertise, may also be computerized. But many parts, such as sudden inspiration and aesthetic perception, are still beyond our understanding.

1.3 Appealing Developments of AI Technology

Engineering design is a dynamic, iterative, and opportunistic process, affected and restricted by polymorphous knowledge sources. A promising AI technique for handling such a situation is the blackboard model of problem solving.

The blackboard system provides a highly structured, opportunistic way of problem solving. It consists of three major components: Knowledge sources, blackboard data structure, and a control module, as shown in Figure 1.1 (Nii 1989):

The domain knowledge for solving a problem is partitioned into a number of knowledge sources. These knowledge sources are kept separate and independent. A blackboard system allows integrations of knowledge in diverse types and forms. A knowledge source can be added or modified without affecting the working styles of the system and other knowledge sources.

The function of a knowledge source is to contribute information in the process of problem solving. For engineering design, all resources contributing to the design process can be treated as knowledge sources. They are naturally partitioned.

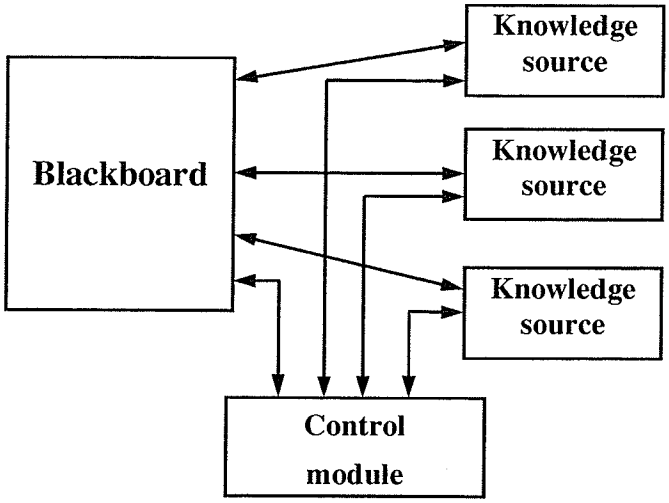


Figure 1.1 The conceptual structure of the blackboard modal

The blackboard is a global data store. It keeps all the data that reflect the state of problem-solving, such as input data, partial solutions, alternatives, final solutions and possibly control data. The blackboard has a hierarchical data structure. Communications and interactions among knowledge sources take place through the blackboard.

Knowledge sources respond opportunistically to changes on the blackboard, take a subset of information, update it, and produce changes on the blackboard. The process is monitored and controlled by a control module. Problem-solving activities occur in an iterative sequence until an acceptable solution is found or the system cannot continue further for lack of knowledge or data. The sequence is dynamic and opportunistic rather than fixed and preprogrammed. The control strategy for information manipulation parallels the information flow pattern in a normal engineering design process.

Pursuits of full automation without human participation lead many expert systems to a dead end. Under the structure of blackboard model, human designers become an integrated part

of the system and act both as a knowledge source and a part of the control module. They participate in the design process through the blackboard. The blackboard provides a window for them to exert their creativity and intelligence.

The blackboard model of problem solving is a general frame for constructing the knowledge based system. Although a macro link among knowledge sources of diverse types is provided, no solution is spelled out for the polymorphous information to communicate with the blackboard. The situation is like that bridges are built between isolated islands, no vehicle or carrier is yet available for cargo transportation.

Object-oriented knowledge representation and information manipulation provide a solution to this problem. Entities of domain knowledge can be represented as objects. An object is a combination of data and code. It works like a courier. Its datafield, the data part, carries information from various knowledge sources. Its methods, the code part, define the way it communicates with surrounding objects.

Knowledge presents differently at different design stages and from different knowledge sources. It can all be transformed to and represented as objects. With objects as universal information carriers, the information manipulation in engineering design is reduced to the manoeuvre of objects.

Help files, patterned after those found in Microsoft Windows, are an important component of the system. They display explanations of the terminology and assist in the use of the system. Information can be in the form of text, graphics, or other multimedia elements. Hypertext capabilities and context-sensitive links help the end user navigate through the content easily.

The Windows Help is a display engine. Contents displayed are compiled into Help files. When design cases are compiled into the same format, an on-line case resource is readily available for the user.

1.4 Summary

Techniques of artificial intelligence have advanced to an unprecedented stage in both theory and application. The development is motivating a new way of thinking in the construction of engineering design aid systems. This situation is the initiative for our development of BEDA (Basic Engineering Design Aid), a prototype, knowledge based design aid for engineering analysis and engineering design.

There are three main objectives in the research:

a. to build a prototype knowledge-based system, BEDA, for aiding engineering analysis and engineering design,

b. during the process of BEDA development, to explore the common rules in engineering design and their implementation in building expert systems, and

c. to explore a method of accomplishing typical procedures of design processes through expert system techniques.

2. LITERATURE REVIEW

2.1 Overview

This chapter has been organized in a manner that reflects the project objectives: to understand the essence of engineering design, to explore the method of AI techniques, and to build a knowledge-based design aid. It presents a general review of the research conducted by previous researchers in engineering design and AI technology applications.

2.2 Engineering Design

Diversified definitions are given to engineering design by different researchers. Engineering design is a complicated activity. It is worth explorations from different perspectives.

> "Engineering is traditional design activity, but it is more than that. Engineering is the solution of people and resource problems. Engineering is the management of production and services. And engineering is control, . . . " [Britton, 1993]

Given the very wide range of perspectives and the limit on available time, it was decided to concentrate on philosophical and morphological studies of engineering design. The philosophy of engineering design is the general and abstract understanding of the process. The morphology of engineering design is the common procedures and structures that have gained acceptance.

2.2.1 Philosophy of engineering design

It is easy to find definitions of engineering design. Understanding is not as easy to obtain.

> " ... although we speak freely of technology, it is unlikely that we have the vaguest notion philosophically of what it is or what is befalling us as it soaks deeper into our life. " [Koen, 1985]

The shortfall of understanding has no implication of professional disqualification. It denotes a devotion to specialties and the lack of general comprehension. The essence of engineering is felt by engineers, but not well defined and expressed.

The occurrence of such a situation is not surprising. It may be attributed to two reasons. First engineering design is very sophisticated. Secondly not enough research has been conducted.

> " . . . little significant research to date has sought the philosophical foundations of engineering. . . . Not only is there little research into the theory of engineering, . . . , but engineers themselves are chronically averse to writing about their world. . . ."

> "Because the pairing of engineers with their completed design is so enduring and the paring with their use of method so fleeting, . . . " [Keon, 1985]

Coping with endless technical problems is engineer's daily duty. Engineers are usually fully occupied by production projects. Little time is left for them to ponder or write about the philosophical foundation of engineering activities.

The lack of understanding unavoidably leads to some pessimistic assertions about engineering design. An argument cited in the PERCARN study was:

> " Only when problems cannot be structured must the problem solver 'resort' to design. Design, therefore, typically deals with ill-structured problems, which are more difficult to represent and solve."

> "Design is sometimes characterized as search through a very large and complex problem space . . . The resulting space is explosive and possible solutions are too numerous to list. " [PRECARN Associates, 1989]

Subjects dealt with by engineers vary from small housewares to huge civil structures, from simple tools to sophisticated equipment. The diversity and complexity of engineering design can easily conceal the common features of different problem solving approaches. However, searches for the essence of engineering are persistent and fervent. Not only because engineering greatly affects society, but also because engineering is a part of our human nature.

> "I believe, and I argue in this essay, that the ideas of engineering are in fact in our bones and part of our human nature and experience." [Petroski, 1992]

> "I think that engineering is what human beings, deep down, want to do. Not only the thing, but one of the most basic and satisfying things. Engineering is an activity that is fulfilling-existentially." [Florman, 1987]

Human nature has been forging the evolution of technology and society. Pursuits of newer and better products never end. Engineers will continuously face new problems and new challenges. The variation of problems presented to engineers demands a systematic research of engineering activities and a philosophical exploration of their essence.

Definitions of engineering range from pragmatic to abstract. Florman [1987] defined engineering as:

> "Although engineering is serious and methodical, it contains elements of spontaneity. Engineering is an art as well as a science, and good engineering depends upon leaps of imagination as well as painstaking care. Creativity and ingenuity, the playfulness of original ideas-these are also a part of the engineering view." [Florman, 1987]

Florman revealed two characteristics of engineering. As an art, it involves creativity, ingenuity and leaps of imagination. As a science, it is serious, methodical, and depending upon painstaking care. Engineering is an organic combination of these two extremes.

Petroski emphasizes the participation of humans in the design process and the importance of learning from previous designs. Back to the beginning of the engineering history, designs may have started without any previous examples as reference.

> "The earliest engineering structures may have been designed by trial and error, . . . " [Petroski, 1992]

Whether the trial was a success or an error, it gave later designers something to learn from. Then engineering design became a process of improvement based on existing cases. From this point of view, he stated:

" Engineering, like poetry, is an attempt to approach perfection." [Petroski, 1992]

The pursuing of perfection is a process of learning and improving. Unlike poets, who can go back to a poem hundreds of times after its first publication, engineers seldom have the chance to revise a completed project. Petroski especially stressed the importance of learning from failures:

> "I believe that the concept of failure-mechanical and structural failure in the context of this discussion-is central to understanding engineering, for engineering design has as its first and foremost objective the obviation of failure."
>
> "The engineer . . . learns more from his mistakes and those of others than he does from all the masterpieces created by himself and his peers. " [Petroski, 1992]

The definition given by Koen is the one I like the most. His definition of engineering was:

> " ... the strategy for causing the best change in a poorly understood or uncertain situation within the available resources; ... " [Koen, 1985]

This definition expressed a view of system engineering. Defining all possible states of a design problem as a state space, engineering design can be viewed as a process of changing from one state to another. A solution is then the evolution of best changes. But the best is by no means absolute. It is an optimum under certain criteria, economical, technical, or combined with other factors. Under such considerations, engineering design is an optimization process.

A poorly understood or uncertain situation means that something is not well defined and engineering judgment is required. It puts certain constraints on the design process, or the optimizing process.

The best change is based on available resources that contribute to the solution. With the alteration of available resources, the best change will be different. From the viewpoint of system

engineering, available resources are the input to the optimization process. The solution is the system output.

2.2.2 Morphology of engineering design

Fox [PRECARN, 1989] characterized the design process as design-in-small and design-in-large. Design-in-small implies those tasks performed by individual engineers when they turn requirements into specifications and drawings. Conventional expert system models are useful in dealing with such problems. Design-in-large is the all-encompassing process shaped by each small piece of the design task and normally carried out by large coordinated teams of engineers. A broadly generalized knowledge-aid-design tool is required to facilitate this type of work.

At Brighton Polytechnic in England, a joint venture was conducted by the Departments of Civil and Mechanical Engineering and the Information Technology Research Institute to investigate ways of using expert systems in engineering design [Thompson, 1989]. In their opinion, design operations can be broken down into synthesis and analysis, which are followed iteratively until a satisfactory design is found. The flow chart of the engineering design process, as they describe it, is shown in Figure 2.1.

In the first step, designers use initial design specifications, with design literature and their own expertise, to synthesize an initial design solution. This solution takes the form of fixed design parameters that specify particular features, or constraints.

The next step is the analysis and evaluation of the design solution. Very often some initial specifications are found to be unacceptable at this point. Designers need to reconsider them or replace them with more appropriate values. Modification is repeated until an acceptable design is obtained.

Miles and Moore [1989] expressed a similar opinion, but in different terms:

> "The design of any structure can be said to consist of two stages. The first being the conceptual design stage, in which the overall form of the structure is decided upon. The second stage consists of a more detailed structural analysis, during

which calculations are carried out to verify the conceptual design choice, and determine component dimensions."

This definition is similar to the previous one. The conceptual design stage matches the synthesizing process. The detailed structural analysis embraces the analysis and evaluation process.

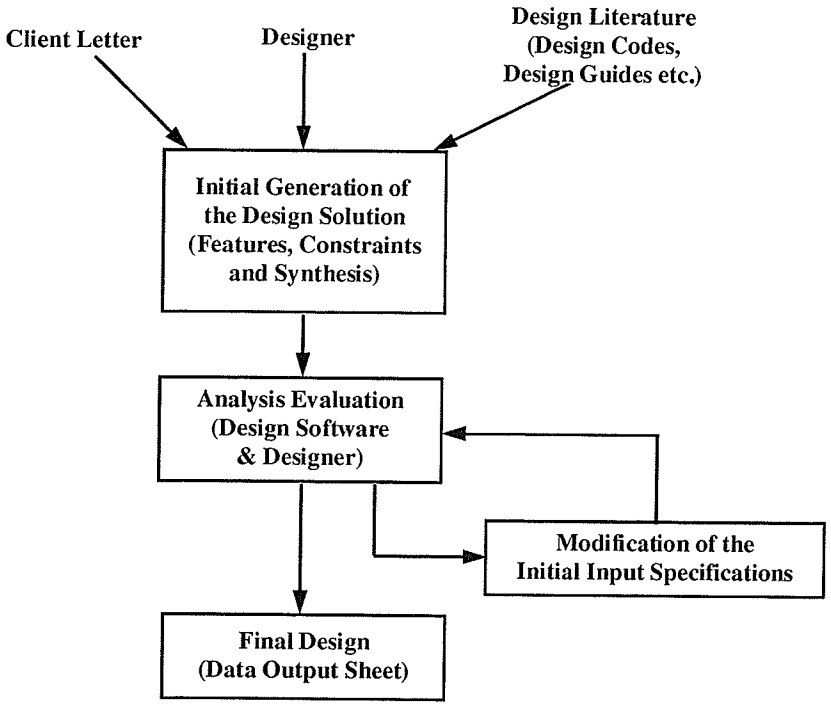


Figure 2.1. The flow chart of the engineering design process [Thompson, 1989]

A three-stage model of engineering design was proposed by McCarthy and Nouas [1989]. It includes preliminary design, detailed design and analysis, and assessment of the results. The suggested design cycle is shown in Figure 2.2.

This definition differs from previous ones in that evaluation is separated from analysis. That separation is reasonable. Although assessment and analysis are sometimes conducted simultaneously, logically they are two different functions.

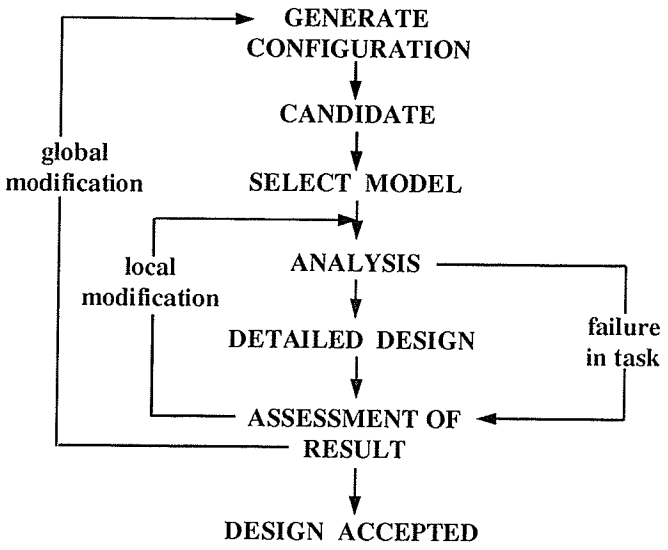


Figure 2.2. The design cycle of engineering products [McCarthy and Nouas, 1989]

2.2.3 Summary

Defining the nature of engineering is still in its exploratory and diversified stage. The understanding will improve as more researches are conducted.

Engineering design is a stereoscopic process. It can be examined from different perspectives. Over emphasizing a vision from one viewpoint may tend to ignore visions from other angles.

2.3 Artificial Intelligence (AI) Technology Applications

The following section is a brief review of AI techniques in agriculture and structural engineering.

2.3.1 AI in agriculture

As little as five years ago, the application of AI techniques in agriculture was still rare. Traditional software was used for most computer-aided farming applications. Peart [1989] stated:

> " Where are we in applications? We are still learning the tools, we have some good applications developed, but we are also still learning about the types of problems that can be solved with ES and decision support systems" [Barrett et al., 1989]

Pursuing efficient production, harvesting, and processing is a characteristic of the agricultural industry. The potential of AI techniques was recognized by agricultural engineers in the late 80's. Since then the applications have been permeating to agricultural industry in an ever faster pace.

The early applications of AI system were quite simple.

> "The application of this remarkable science to agriculture began with the development of simple, narrowly focused computer programs designed to help agriculturists make decisions." [Barrett et al., 1989]

Up to 1992, most publications on expert systems in agriculture still claim their works as prototype. Most systems were developed with expert system shells such as PC Plus [Schaper and Lund, 1992], KES™ [Deer-Ascough et al., 1992], and PennShell [Heinemann et al., 1992].

From 1992, more sophisticated systems started to appear. Most of them were developed with software packages rather than simply ES shells. Universal programming languages were integrated into systems to overcome the limitations of ES shells and to increase flexibility. Conventional languages used include Pascal [Batchelor and McClendon, 1992], Basic [Gupta and Suryanto, 1993], and C++ [Evans et al., 1990; Beck et al., 1994].

AI systems in agriculture fall into different categories, including planning [Randhawa et al., 1992; Beck et al., 1994], operational management [Evans et al., 1990; Schaper and Lund, 1992; VanDevender et al., 1994], diagnostics [Deer-Ascough et al., 1992], and environmental control [Gauthier, 1992]. A very interesting application was described by Batchelor and McClendon [1992]. It used the blackboard model to link two separate systems and to solve the conflicts between them:

> "A blackboard was used to link an irrigation expert system with an insect pest management expert system to determine the most economical rescheduling alternatives for conflicting irrigation and insecticide scheduling recommendations. The blackboard was demonstrated for several combinations of insect and water stress levels. The blackboard was also used to study the effect of rainfall probability on rescheduling conflicting irrigation and insecticide applications. . . . " [Batchelor and McClendon, 1992]

AI systems in agriculture have rapidly developed from prototype and shell constrained programs into sophisticated packages. Most of the state-of-the-art computer techniques are currently used in system development, including object-oriented programming, graphical user interfaces, and relational databases. It can be expected that newer ways will be adopted as AI techniques advance. As Holt [1989] stated:

> "ES become functional but are never really finished. Once a prototype is developed, further development, changes, evaluation, refinement and maintenance proceed indefinitely." [Barrett et al., 1989]

2.3.2 Artificial Intelligence in structural engineering

There are increasing numbers of publications on AI applications in structural engineering. While some systems were developed with AI languages [Miles and Moore, 1989; McCarthy and Nouas, 1989], most were developed with existing expert system shells [Adeli and Al-Rijleh, 1987; Thompson, 1989; Embleton, 1990]. Current AI systems are often developed with an integrated package, a shell plus a programming language.

Expert systems developed with AI language are often the result of cooperation between the engineering field and the information field. The objective is to combine the engineering experience of engineers with the software capability of computer experts. Developers work hard to fill the gap between the engineer's lack of mastery of AI language and the computer expert's lack of understanding of the engineering design process.

Most expert systems developed with existing shells were narrowly focused. Some were oriented to the design, or selection, of a special component, such as a column base plate

[McCarthy and Nouas, 1989], timber roofs [Thompson, 1989], and roof trusses [Rowlinson, 1989]. Others were developed for the automation of a single design step, such as the estimation of snow loads and wind loads [Embleton, 1990], the design of structural elements [Ghosh and Kalyanaraman, 1993], and fire protection analysis [Olynick, 1994]. The development of EXSEL, an expert system for designing steel structural elements, was typical of such systems:

> "The code requirements, expert knowledge, basic strength of materials, and the handbook information used . . . are represented appropriately using production rules, C functions, and a relational data base. An inference mechanism . . . is used to implement the system. " [Ghosh and Kalyanaraman, 1993]

These systems were developed in diversified styles and operation routines. As the editorials of Journal of Computing in Civil Engineering stated:

> "They have existed as algorithmic "islands of automation," giving little support to the more ill-structured aspects of the design process." [Garrett and Maher, 1991]

> "... not a single CADD system is compatible with another. The confusion generated by this deficiency is incredible. . . . Information exchange between systems at this point is extremely difficult, time consuming, and, as a result, practically nonexistent. " [Thornton and Velivasakis, 1991]

The need for eliminating problems in communication and compatibility encouraged research on integrated systems. Tyson [1991] gave some interesting guidelines for constructing integrated systems:

> "1. Engineering judgment is an essential element of the design process. Software should help engineers develop judgment and implement their own decisions. ...
>
> 2. Software should include a data base that allows thorough interfacing between analysis, design and drawing development operations. ...
>
> 3. Software . . . should include special purpose routines as well as a general purpose analysis program . . .
>
> 4. Software should work on microcomputers so it is accessible to small design office and a variety of project types. " [Tyson, 1991]

According to Tyson, engineers should not be excluded from the design process. Computer software can be a design aid. Engineers still implement engineering judgment and make decisions. Design aids for different functions of a design process, analysis, design, and drawing, should be thoroughly interfaced. This opinion represents a typical expectation from industry.

A model of communication for automated engineering design was presented by Morse and Hendrickson [1991]. The blackboard architecture was recommended for building the conceptual framework. It suggested four approaches of communication: procedural, rule-based, object-oriented, and frame-based. Among the four approaches, the object-oriented technique was claimed the best.

A practical approach for system integration was demonstrated in a prototype system called Builder [Jonathan et al., 1991]. It attempted to use CAD results for later analysis and design. The entities of a drawing were represented as semantic networks. The semantic network representation was then used for later analysis.

> "The semantic intent of the drawing originates with the user. Because it is menu-driven, the Draw program forces the user to be explicit about the drawing semantics. For example, instead of drawing two lines to symbolize a wall, the user must select Walls from a menu. . . . The system then uses two lines to symbolize in the drawing that it knows to be a wall; ..." [Jonathan et al., 1991]

In recent years, case-based systems have gained attention in engineering design [Sycara et al., 1991; Arciszewski and Ziarko, 1991; Bailey and Smith, 1994]. A case-based system retrieves and reuses previous successful designs. Failure cases may be retrieved for information purposes. This provides designers either something to start with or references. It can be used as a design advisor or a brainstorming assistant.

Now a major concern in AI development is to improve the behavior of AI systems. Some current fields of activity include: neural networks [Elkordy et al., 1993; Flood and

Kartam, 1994], genetic algorithms [Koumousis and Georgiou, 1994], and fuzzy systems [Chen et al., 1994]. Efforts have been devoted to understanding and simulating computational principles of the learning process, natural selection, and fuzzy logic. Such principles provide new approaches for dealing with nonlinear, discrete, and time-variant problems, or problems without appropriate mathematical descriptions.

2.3.3 The principle of blackboard model

The blackboard model of problem solving consists of three major components: knowledge sources, a blackboard data structure, and a control module, as shown in Fig. 1.1.

Knowledge sources respond opportunistically to changes on the blackboard. With or without a unique internal structure, a knowledge source is in general composed of two parts: a precondition part and an action part. The precondition part prescribes special situations on the blackboard. When such situations appear, the action part is activated. The action part specifies what and how a knowledge source contributes to the problem solving activity. It produces changes to the blackboard that lead incrementally to a solution, or a set of acceptable solutions, to the problem.

The blackboard is a global expression of the problem-solving state. It has a hierarchically organized data structure. Initial information or raw materials belong to the lowest hierarchy. Final solutions are on the highest hierarchy. Intermediate results stay on hierarchies in between. Incomplete information belongs to lower hierarchy levels whereas more completed information is assigned to higher levels. Output of a lower hierarchy is the input to a higher hierarchy.

The evolution of problem-solving states on the blackboard activates knowledge sources. Knowledge sources get input from the blackboard and produce changes onto the blackboard. Communication and interaction among knowledge sources take place through the blackboard.

The control module monitors and controls the evolution on the blackboard. Various kinds of information are made available to the control module. Proper reasoning steps are applied at

each stage of solution formation. Problem-solving activities occur in an iterative sequence. They are dynamic and opportunistic rather than fixed and preprogrammed.

The knowledge organization and control scheme make the blackboard model especially suitable for solving complex problems with multi-phase, multi-resource, and ill-defined structures.

2.3.4 Conclusion

The application and development of AI technology in both agriculture and engineering are proceeding rapidly and steadily. Increasingly complicated AI systems are under development. New approaches and new ways of thinking are being invented and adopted.

Programming of AI systems is gradually breaking through the limitations of pure shell or AI language use. More systems are developed with integrated programming environment, such as a shell complemented by a database and universal language subprograms. Object-oriented programming and graphical user interfaces are becoming common tools in AI system development.

For the most part, early systems addressed narrow applications with simple procedures of problem solving and well-structured knowledge sources. Integrated systems were investigated to solve problems of communication and compatibility. The blackboard architecture was recommended in most cases for being the system frame. Object-oriented programming was claimed as the best way of information exchange. Case base reasoning, neural networks, fuzzy logic, and genetic algorithms are being added to AI systems to increase their functionality. The pace of development has been increasing with time. With the emergence of new computer technologies, such as the information highway, multimedia, and CD-ROMs, we can expect that the development of AI systems will enter an era of even more promise and potential.

3. SYSTEM MODEL OF THE DESIGN AID

3.1 Overview

A knowledge-based system, BEDA (Basic Engineering Design Aid), has been developed as an engineering design aid. In the construction of BEDA, engineering design was considered as a multi-input, multi-constraints, and multi-output optimization process. The blackboard architecture was adopted as the framework of the expert system. Information flows iteratively within the system in a pattern of self-organizing and self-stabilizing. This chapter introduces the system definition, blackboard framework, and control strategy of BEDA. They form the theoretical basis, upon which the system model of this engineering design aid is based.

3.2 Engineering Design As An Optimizing Process

Physically, the design process of any engineering product, be it a building, a machine, or an electrical device, includes three levels: system design, subsystem design, and component design. With a sophisticated product, there may be several levels of subsystems. With a simple product there may be neither subsystem design nor system design. Functionally, design has two phases: conceptual design versus analytical and detail design. System design focuses more on conceptual development. Component design mainly contains analytical and detail assessment. System design and conceptual design may be considered as higher levels of design, whereas component design and analytical design as lower levels of design. Higher level designs regulate lower level designs, while lower level designs constrain higher level designs. The whole design procedure is an iterative process, which cycles from higher level to lower level, and back to higher level. These cycles continue, until a compatible solution is reached.

From the viewpoint of system engineering, engineering design can be visualized as a multi-input, multi-output, and multi-constraint optimizing process. The basic model is shown in Fig. 3.1.

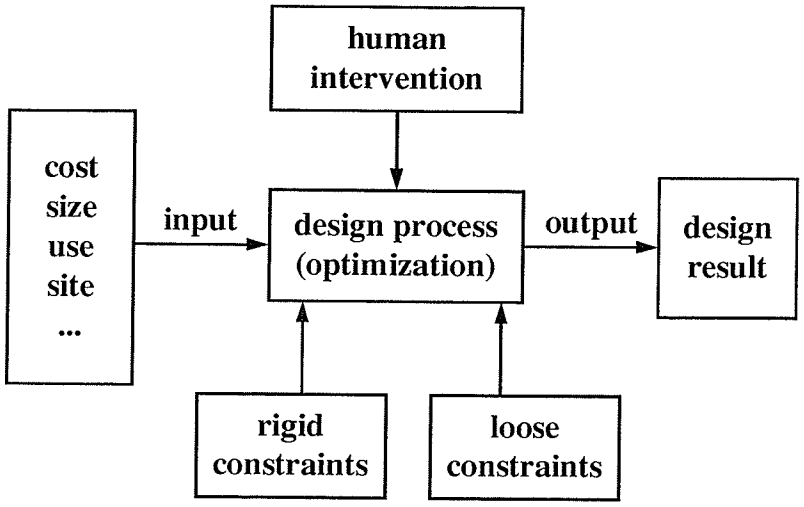


Fig. 3.1 Engineering design as an optimizing process

Input includes: use, size, site, cost, and any other initial specifications.

Output includes: final design results, both graphical and textual, such as drawings, materials lists, and specifications.

Optimizing objects would be to:

minimize the consumption (of money, materials, labor, and other elements)

maximize the capacity (of load, lifespan, functionality, space, and utilizations)

The optimizing process is under two categories of constraints: rigid and loose. Rigid constraints include design codes, physical laws, environmental requirements, and design theory. Loose constraints involve personal preference, experience, aesthetical attainment and/or even cultural background. Rigid constraints must be strictly followed. Loose constraints are more human factor related and can be loosely followed. Under rigid constraints, an optimizing process may have no single solution. Loose constraints narrow the options down to a unique one.

The interaction between the system and its constraints continues as the optimizing process proceeds. It is initiated in each design step to form a local optimal solution. Step by step, a full scale optimal solution or suboptimal solution is achieved.

3.2.1 Definition of the optimizing process

Optimization represents the best course of action from available alternatives. It conforms to the principle of engineering design: to gain maximum utility with minimum consumption.

As mentioned above, the objective of engineering design is:

1. minimizing the consumption (of money, materials, labor, and other real elements), or

2. maximizing the capacity (of load, lifespan, functionality, space, and utilizations).

The two indexes are interchangeable in a design process. In most cases, minimizing the consumption will maximize the capacity. Alternately, if the capacity is maximized, it follows that the consumption is minimized. One index may be easier to implement than the other in different situations. Circumstances dictate which approach is more desirable.

It is suitable to express the objective in a quantitative form through the formulation of an objective function $I(x_1, x_2, ..., x_n)$, which is to be minimized

Min $I(x_1, x_2, ..., x_i)$, i=1,2,...,n.

In engineering design, the expense of labor, materials, space, and other physical items can be evaluated in their equivalent cost index. The objective function can then be expressed with a weighted linear equation:

$$I=a_1M_n+ a_2S_p+ a_3M_t+ a_4L_b+...$$

M_n, S_p, M_t and L_b are the equivalent cost indexes corresponding to money, space, materials, and labor. a_1, a_2, a_3, and a_4 are weighting coefficients. In different situations, the importance of the labor, materials and space may vary. Weighting coefficients embody the diversity.

Many rigid constraints in an optimizing process can be expressed mathematically. The expression may take several forms:

i) Algebraic equality constraints

$$h_i(x_1, x_2, ..., x_n)=0 \qquad i=1,2,...,m_1$$

ii) Algebraic inequality constraints:

$$g_j(x_1, x_2, ..., x_n) <= 0 \quad j=1,2,..., m_2$$

iii) Differential or integral equation equality constraints:

$$dx_i/dt=f_i(x_1,x_2, ..., x_n) \quad i=1,2,..., m_3$$

$$x_i(0) = x_{i0} \qquad t_0 <= t <= t_z$$

Suppose that the prices of all items remain stable during a design process, type iii) constraints will disappear. The optimizing process then becomes a typical linear programming problem.

Optimization problems expressed in the above form have been extensively studied in optimizing theory [Radford and Gero, 1988]. Solution techniques are well documented and widely accepted.

Engineering design does not fit into a simply defined pattern of optimization. A single design stage may be a typical optimization problem in the traditional sense. But the whole design process must be optimized in a more sophisticated way.

Besides gaining an optimal solution at each design step, we want the design process to go along a general optimal path from step to step. The optimizing objective is achieved by applying the objective function under rigid and/or loose constraints. Though constraints may take different forms at different design stages or in different designs, three design criteria should be met: safety, function, and aesthetics.

Constraints are embodied by production rules and functional procedures in BEDA. Two rules were cultivated for the optimizing process. The first is the rule of load-resistance compatibility, i.e., factored resistance > factored load. It addresses the criterion of safety. The second is the rule of space-compatibility. It means no dimensional conflict among components and partially addresses the criterion of function. The criterion of aesthetics is addressed by the human designer, who is considered an integrated part of BEDA.

3.2.2 Two rules employed in the optimizing process

Rule 1: load-resistance compatible rule

This rule requires that the factored resistance be not less than the factored load applied to a system or its components.

The requirement is basic for any design. Its mathematical expression depends on the application field of the design. With electrical engineering, the resistance and the load are voltage and/or current related. With mechanical or civil engineering, the load and the resistance are more likely force related. There are well-established standards relating to load and resistance in engineering design.

In wood design, for example, the approach of limit states design is adopted (see section 5.3.2 for detail). If the factored load does not exceed the factored resistance, a safe design is ensured. This rule was easily encoded into BEDA.

Rule 2: space compatible rule

The rule requires that there be no space conflict among components of a design. This minimum functional requirement can be ensured with fixed algorithms.

Take two rectangular objects on a plane as an example:

The position of an object can be identified by the coordinates of its four corners (Fig. 3.2).

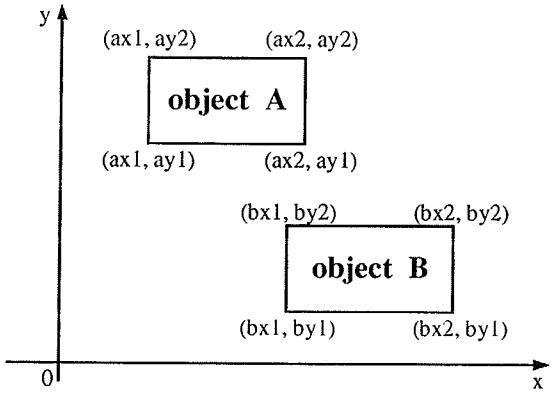
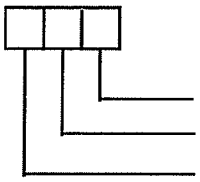


Fig. 3.2 Objects identified by coordinates

Each coordinate is represented by three characters:



1: the smaller coordinate, 2: the bigger coordinate,
x: X dimension, y: Y dimension,
a: object A, b: object B,

In either the X or Y direction, if the smaller coordinate of one object is bigger than the bigger coordinate of the other object, there is no conflict between the two objects.

The logic can be expressed as:

If (AX1 > BX2) or (BX1 > AX2) or (AY1 > BY2) or (BY1 > AY2)

then there is no conflict between object A and object B.

If the above precondition is not met, there is conflict between the two objects.

For the purpose of installing, dismantling, or usage, operational space is often needed for an object. It is the extra space other than what the physical object occupies. In such cases, the rule requires that there be no conflict not only between any two objects, but also between any object and the operational space of other objects. Conflicts among operational spaces are usually not a problem since operational space can be shared. The position of an object, including its operational space, can still be expressed by the coordinates of its four corners (Fig. 3.3).

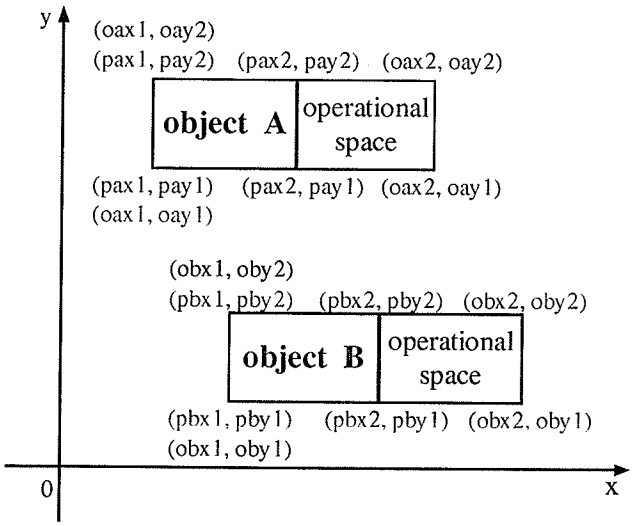
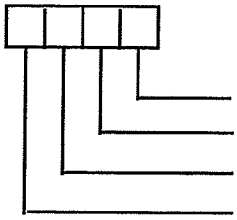


Fig. 3.3 Objects with operational space

Each coordinate is represented by four characters:



1: the smaller coordinate, 2: the bigger coordinate,
x: X dimension, y: Y dimension,
a: object A, b: object B,
p: physical space, o: operational space.

The relationship becomes:

If [(OAX1 > PBX2) and (PAX1 > OBX2)] or

[(OBX1 > PAX2) and (PBX1 > OAX2)] or

[(OAY1 > PBY2) and (PAY1 > OBY2)] or

[(OBY1 > PAY2) and (PBY1 > OAY2)]

then there is no conflict between object A and object B.

If the above precondition is not met, then there is conflict.

The same logic can be generalized to three-dimensional space and objects with more complex shapes.

3.2.3 Conclusion

The optimization approach is an initial exploration of the engineering design process. Its development is for guiding the construction of the BEDA system. The approach embodies basic features of engineering design.

As an optimizing process, rigid constraints and optimizing rules can be readily encoded into an expert system. They form a part of the knowledge base or the rule set. Loose constraints, such as expertise, can be partially computerized.

The success of a design is evaluated by three criteria: safety, functionality, and aesthetics. The load-resistance compatible rule secures the safety criteria. The functionality criteria is partially shielded by the space compatible rule. Additional rules are necessary for its full coverage. The third criteria, aesthetics, should be addressed by the human designer.

3.3 Framework of BEDA System - the Blackboard Architecture

The blackboard model of problem solving provides a workable organization of diverse knowledge resources, supports cooperations among various subsystems, and allows system users to get involved in the design process. Under the blackboard architecture, knowledge sources can then work opportunisticly under a unified framework. The model is suitable for solving complicated problems, such as engineering design.

In BEDA, the blackboard architecture is shaped as shown in Fig. 3.4.

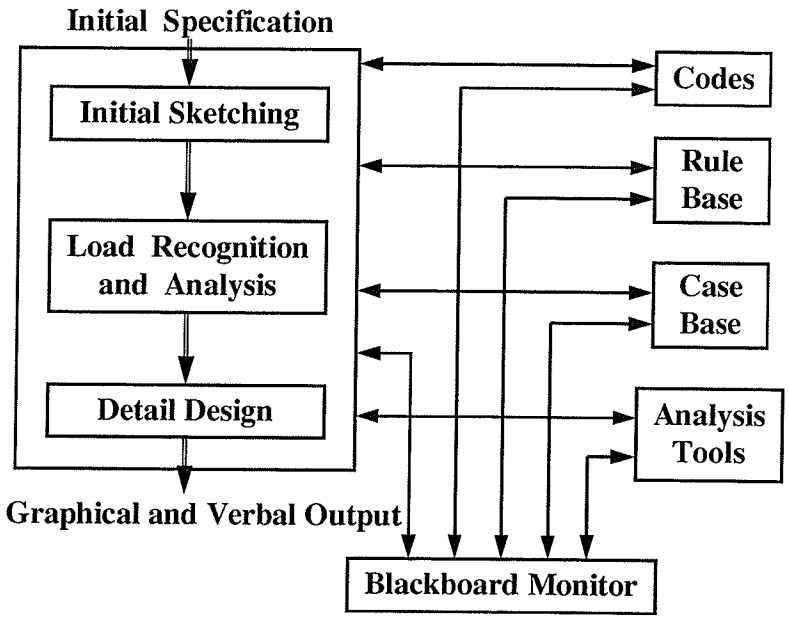


Fig. 3.4 The blackboard architecture of BEDA

Knowledge sources include design codes, rule bases, conventional analysis tools, and case bases. Intermediate results are kept in the global data store. The computer monitor with the user-system interface acts as the blackboard monitor. Knowledge sources are activated at different design stages to contribute to the data evolution process. The case base can make contributions to the initial sketching. The rule base and analysis tools work for load recognition and analysis. Design codes and the rule base are active in detail design stage. Under the

coordinations and regulations of the blackboard monitor, the global data structure evolves from the initial specification to the final graphical and verbal output.

The blackboard data structure can be viewed as a three-dimensional information space. Initial information, including the client's requirements, design codes, and other technical or economic input, is on the x-axis. Hypotheses and/or tentative plans, complete or incomplete, are on the y-axis. Final solutions, plausible or implausible, are on the z-axis.

The information mapping from the x-axis to the y-axis is a process of creation and synthesis. The process is completed in the initial sketching and graphic editing stages. The mapping from the y-axis to the z-axis is a process of analysis and validation. It is finished in the load analysis and detail design stages. Fig. 3.5 shows the blackboard data structure.

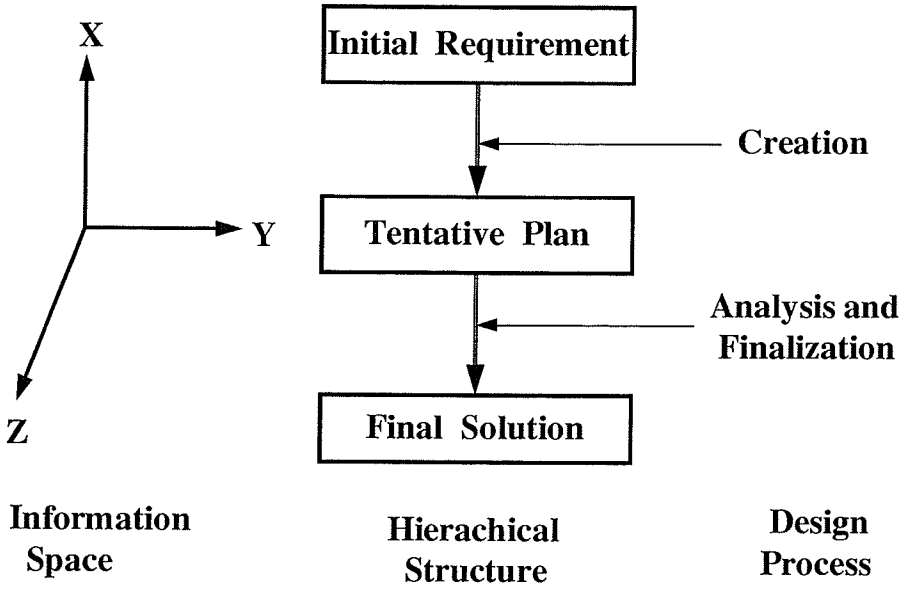


Fig. 3.5 The blackboard data structure of BEDA

The control module consists of a blackboard monitor and a scheduler. The monitor is the interface between the system and human designers. It monitors the problem solving process and the state of knowledge sources being utilized. The problem solving process is marked by flags on the blackboard. These flags act as stimuli to activate appropriate knowledge sources. When

the current state of problem solving conforms to the precondition part of a knowledge source, the action part of that knowledge source is activated. When more than one knowledge source contributes to the design process, the scheduler selects the optimal activity. Therefore, the process of problem solving is continuous and smooth.

Knowledge sources activate changes on the blackboard. For example, the Member Design Unit of BEDA is a knowledge source. Its action part is the member design process based on the limit states design, see section 5.3 for details. The appearing of results from finite element analysis is a situation specified in its precondition part. When the situation appears, the member design function is activated.

For engineering design, the analysis and validation process has been well studied and can be fully computerized. The creation and synthesis process involves more human factors. From our current understanding of human intelligence, a creative process can not be fully computerized. Some parts of the process can only be finished by human designers. A human designer can propose hypotheses and conceive tentative solutions from available information. The mechanism needs further exploration before it can be replicated mechanically. So a reasonable objective is the optimal man-machine combination in a design process.

A virtue of the blackboard model is its ability to bring human factors into the system. In BEDA, the human designer becomes an integrated part. The system user, i.e., the designer, functions as both a knowledge source and a scheduler. As knowledge sources, human designers complete the knowledge base of the expert system. As decision makers, human designers encode their own styles into the design results. The blackboard system treats the designer as an equal among knowledge sources in the sense of their contribution to the problem solving process. When faced with an unsolvable situation, the system calls for the participation of human designers. As a control component, the designer functions in parallel to the built-in machine scheduler. The human scheduler has priority over the machine scheduler and can override, or complement, decisions made by the machine scheduler.

The blackboard architecture makes the design process transparent to human designers. It provides them a stage to employ creativity and intelligence. If the human designer cannot lead the system out of an impasse, the design process fails.

3.4 The Control Strategy of BEDA

In BEDA, information manipulation is controlled by two controllers, the human designer and the system scheduler. The dual control mechanism is a conjugation of human intelligence and machine power. It propels the evolution of any design process. The function process is demonstrated in Fig.3.6.

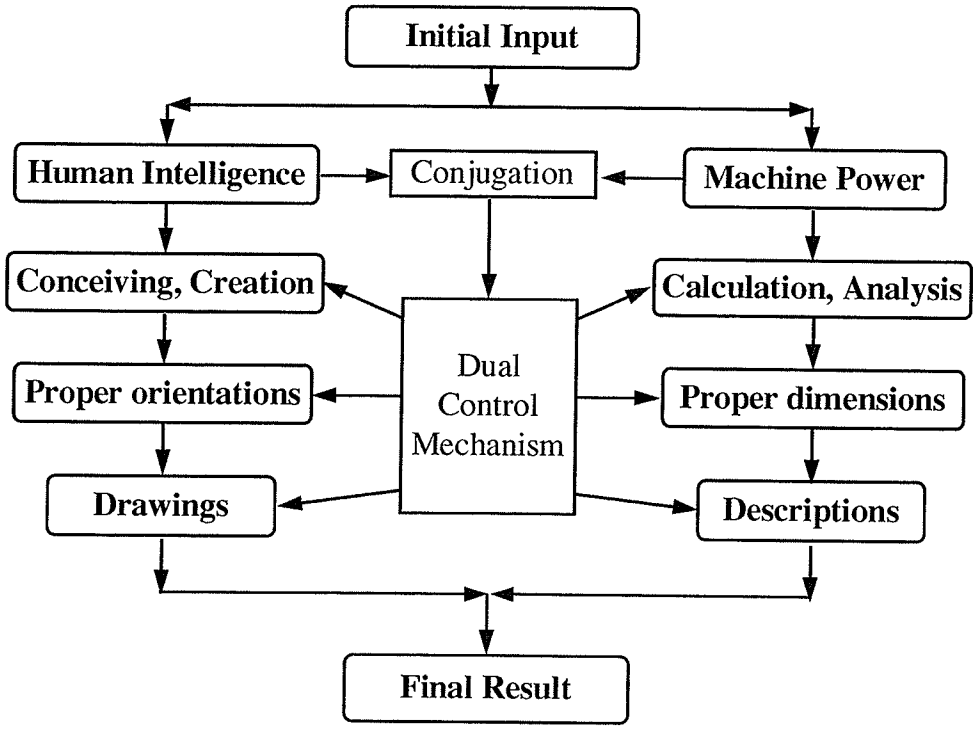


Fig.3.6. Functions of the dual control mechanism

Human intelligence shows importance in creative processes, such as deciding the structural shape of a design. Machine power is more active in analysis and calculation processes

to determine the dimensional shape of structural members. However, there are overlaps between the two control factors.

Engineering design is an iterative process which proceeds from undetermined to determined and then to optimal. Information flow within BEDA follows a self-stabilizing and self-organizing pattern cultivated from a cognitive model [Laszlo, 1969]. The model describes the acting mode of human spiritual phenomena, which is logically homogeneous with the thinking pattern of designers in engineering designs. The pattern is enforced by the control mechanism as the control strategy of BEDA.

The cognitive model symbolizes a path of information flow as shown in Fig.3.7. Information flows continuously along the sequence of E-P-C-R-E.

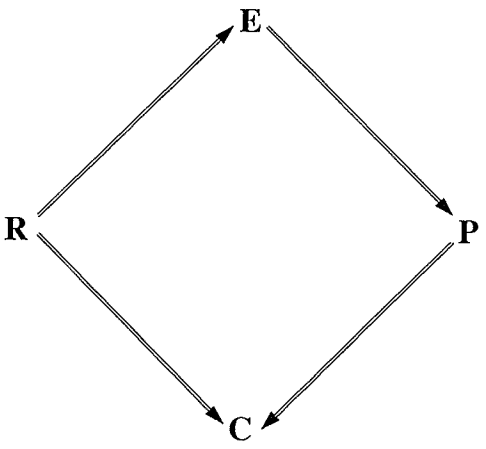


Fig. 3.7. The information flow chart

E is the information source and the data storage. P acts as a selector and a filter. Information conforming to system standards is picked up, filtered, and sent to C. Unit C contains system modes, or rules. As long as unit P induces information complying with the rules, corresponding reactions arise from the system. C is also the control centre of the whole information flow scheme. It analyses, manipulates and refines information accepted from P, and then sends it to R. R is the reviewing and adjusting unit. Under transforming conditions of the

system, R is a particular reaction to the whole state and orients to all potential information in unit E. It rechecks the adaptability of information to the system standards and the information manipulating process. If the results meet the requirement of the system, the information will be sent to E.

As the process continues iteratively, noise contaminated information is filtered and refined. It becomes more and more harmonized with system standards. The result, in turn, eases the information manipulation process so that more satisfactory output is produced. In this sense, the system is self-stable.

The other characteristic of the system is self-organization. Standards of the system are stable but not unchangeable. Unsuitable information introduced by P will engender tentative reactions in the system. The system can accordingly adjust its standards and structure. When the input fits the system standards or rules, the information flow is maintained. Otherwise the system will adapt itself through unit C.

Originally unit E contains badly structured information combined with noise. Through continuous manipulation and processing, it gradually becomes an orderly and transparent knowledge base. Necessary conclusions and reactions are drawn from the ultimate version of E.

In BEDA, E represents the state space of problem solving. C represents the design theory, physical laws, and the envisagement of design plans. P is the matching mechanism between state flags on the blackboard and the precondition part of knowledge sources. R is the examining and rechecking function, which is fulfilled jointly by the human designer and the system scheduler. See Fig. 3.8.

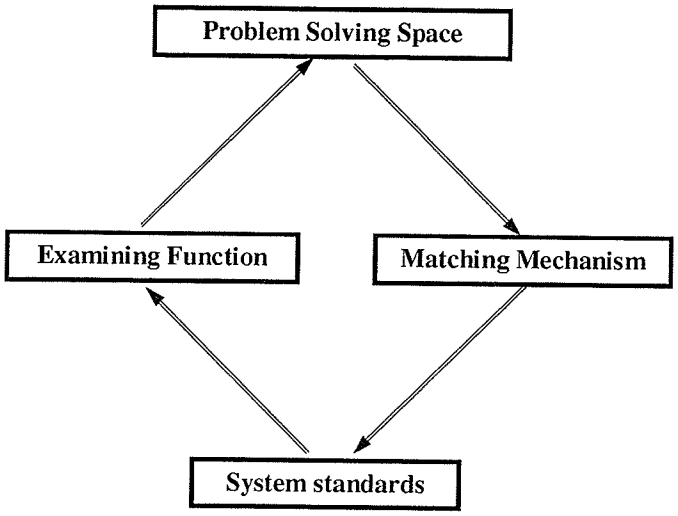


Fig. 3.8. Information flow chart within BEDA

The self-stability of the system is critical to its use in the design process. At the beginning, E contains initial information, such as the client's requirements and unprocessed data. As the design process proceeds, specific details evolve from tentatively defined plans. At the end, an acceptable solution is obtained.

The self-organization allows progresses from the general case, as in design codes, to specific cases within the design objective. It reveals the fact that engineering design is a continuous adapting process between the concept and the reality, between the system and the components, and between the general plan and the member details.

4. THE GENERAL SYSTEM STRUCTURE OF BEDA

4.1. Overview

The prototype expert system, BEDA (Basic Engineering Design Aid), applies the design process to wood-framed buildings for the purpose of system development. The principle can be adapted to other types of design. BEDA runs in the MSWindows environment on personal computers (minimum requirement 33MHZ 386 machines with 4M RAM). It is intended to provide a complement to engineers doing design and analysis work.

BEDA has been developed using an integrated approach of AI programming and conventional programming. Level5 Object, an Expert System development environment, and Borland Pascal were the fundamental development tools. Other software packages, including Paradox, AutoSketch, and HelpAuthor, were used to facilitate various design functions. Knowledge representation, information management, and logic inference were all written in object-oriented programming. The blackboard model of problem solving forms the system frame for BEDA. It links all knowledge sources and restrains information flows within a design process.

The goal is to create an intelligent software package, which will streamline the design process to make it faster, easier, and more economical. As well, it will permit more complete analysis without usual time requirements. The package is not developed for a single design stage, such as AutoCad for graphics or finite element analysis software for strength analysis. It is aimed at broad applications. The system includes, in fact requires, human experts in the design process. With a built-in case base, BEDA provides not only a decision support but also a tutorial tool for new designers.

The general structure of BEDA and its working principle are herein presented.

4.2. The Working Process of BEDA System

4.2.1. General description of the wood-frame based BEDA

BEDA is designed to accommodate typical features of engineering design. Its working process is patterned after human design routines. Without losing the generality, BEDA utilizes the procedure of wood-frame building design as a vehicle for system development. The process is divided into several design steps in the expert system (Britton et al., 1992). Its flow chart is shown in Fig.4.1.

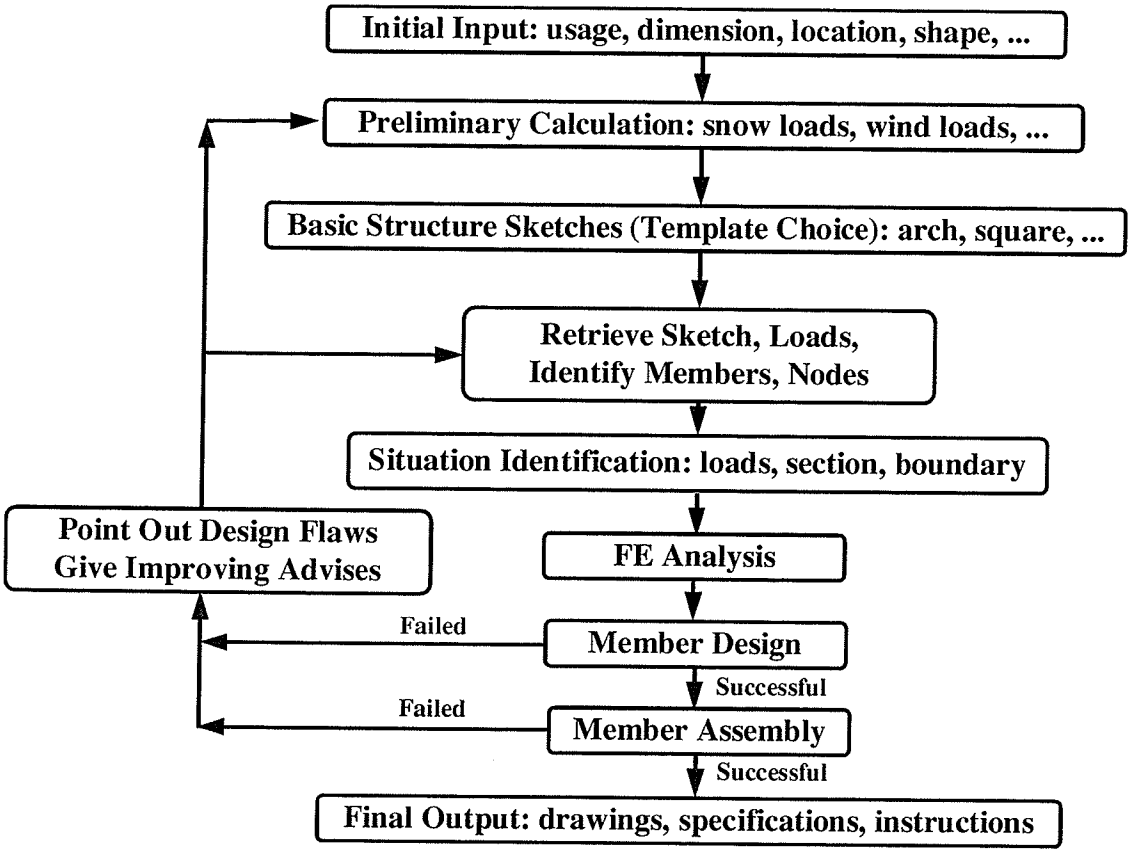


Fig.4.1 The flow chart of the design process

Initial input to BEDA are basic requirements, functional or geometrical, about the design subject. External loads, snow loads and/or wind loads in the instance of a wood-framed building, sustained by the subject are then estimated. Using graphic software connected with BEDA,

designers can sketch a tentative design, retrieve an existing design case, or revise a previous sketch. BEDA characterizes all elements of the sketch, identifies their situations, and calls up a finite element analysis program to analyze internal loads and displacements of each structural member. Based on the analysis, it carries on detailed member designs and recommends member dimensions. Requirements for iterative cycles as the design develops are accommodated. Final design output includes drawings, element lists, and material lists.

Main steps of the design process are described below.

4.2.2. Main steps of the design process

Sketching a Design

Sketching is commonly the first step in a design process. Designers translate their thoughts into tentative plans. Analyses, calculations, and revisions are then carried out to reach a final design solution.

Sketching is done using external graphic software connected with BEDA. After a sketch has been entered, the result is retrieved into the system. BEDA decomposes the sketch into lines, arcs, and other simple geometric elements. The sketch is then represented as a collection of objects in the knowledge base. See section 5.2.2 and 6.1 for details about objects. Each object represents a structural member. The datafield of an object stores member's dimensions, orientation, and types of connection with others. The methods of the object specify how the object is manipulated in the system, such as storing, loading, drawing, printing and so on. As the design process advances, such data as material type and section shape are added to each object. A sketch is continuously refined as information develops in datafields.

Load Calculation and Structural Analysis

This functional unit is currently written for wood-framed structure design. In order to broaden the application, different function units have to be developed for design subjects other than buildings. But the system structure and its working principle will remain the same.

External loads applied to a building are calculated according to the building's location, geometric shape, and environmental factors. The calculations of snow loads and wind loads strictly follow specifications of the Supplement to the National Building Code of Canada (NRC, 1990). Basic data for load estimation are stored in the database. Calculation conventions are stipulated in the knowledge base. All input and output are in a user-friendly format. The user can easily examine different scenarios and compare the results.

After external loads are determined, a finite element analysis program can be called to determine the stress and deformation of structural members under any load combination. The results are used later for detailed member design.

Detailed Design of Structural Members

Wood member design is written as object related methods and production rules in BEDA. Mechanical and physical features of woods of different species and grades are stored in the database (CSA, 1989). A designer specifies material types and service conditions to be considered. BEDA will either recommend a member size for the given loads, or the loads sustainable by a member of specified size. Results are then added to the object representing that member.

Result Output

Since BEDA is written in object-oriented programming, results of each design step are automatically added to the datafield of corresponding objects. Objects are in a hierarchical structure. Virtual commands are shared up and down the hierarchy by all polymorphous objects. Design information can be outputted whenever it is necessary. Final recommendations by BEDA include: drawings, lists of structural elements, and material lists. See Fig a26 in Appendix A for example. The designer can either accept proposed results as the final design or go back to any stage to refine the design.

It should be noted that the level of refinement of formatted output is not intended for final use at this stage. Commercially available programs linked to BEDA would be required to meet design office needs. Capability of making these links is demonstrated but the links have not been incorporated.

4.3. The Construction of BEDA

4.3.1. General description

Engineering design applies analysis and synthesis in an iterative and dynamic way, in which information flows in forms of data, drawings, and descriptions. Some parts of the work are logic inferences suitable for AI programming. Other tasks are routine analysis and calculations that can be done by conventional programming. The polymorphous information flows and knowledge transformations are not easily implemented by either AI programming or conventional programming alone.

BEDA is developed as several function units, with each accommodating a specific function of engineering design: External Load Calculation, Graphics Processing, Finite Element Analysis, and Member Design. Various software packages are employed in the construction of BEDA: Paradox for database management, AutoSketch (or AutoCad) for graphics editing, and HelpAuthor for case base development. Routine analysis, calculations, and information manipulations are written in Borland Pascal. Level5 Object is used for typical AI procedures and as a server to coordinate all function units. Fig.4.2 shows a graphic representation of functional interactions of BEDA system.

BEDA is constructed following the scheme of the blackboard model of problem solving. Its central unit features three components: scheduler, interface, and blackboard. The scheduler coordinates the functioning of all units and controls information flows. The blackboard is a data structure where the system and all units store their input and output data. Through the interface, a user can either trace data evolution or participate in system control. All function units contribute as knowledge sources.

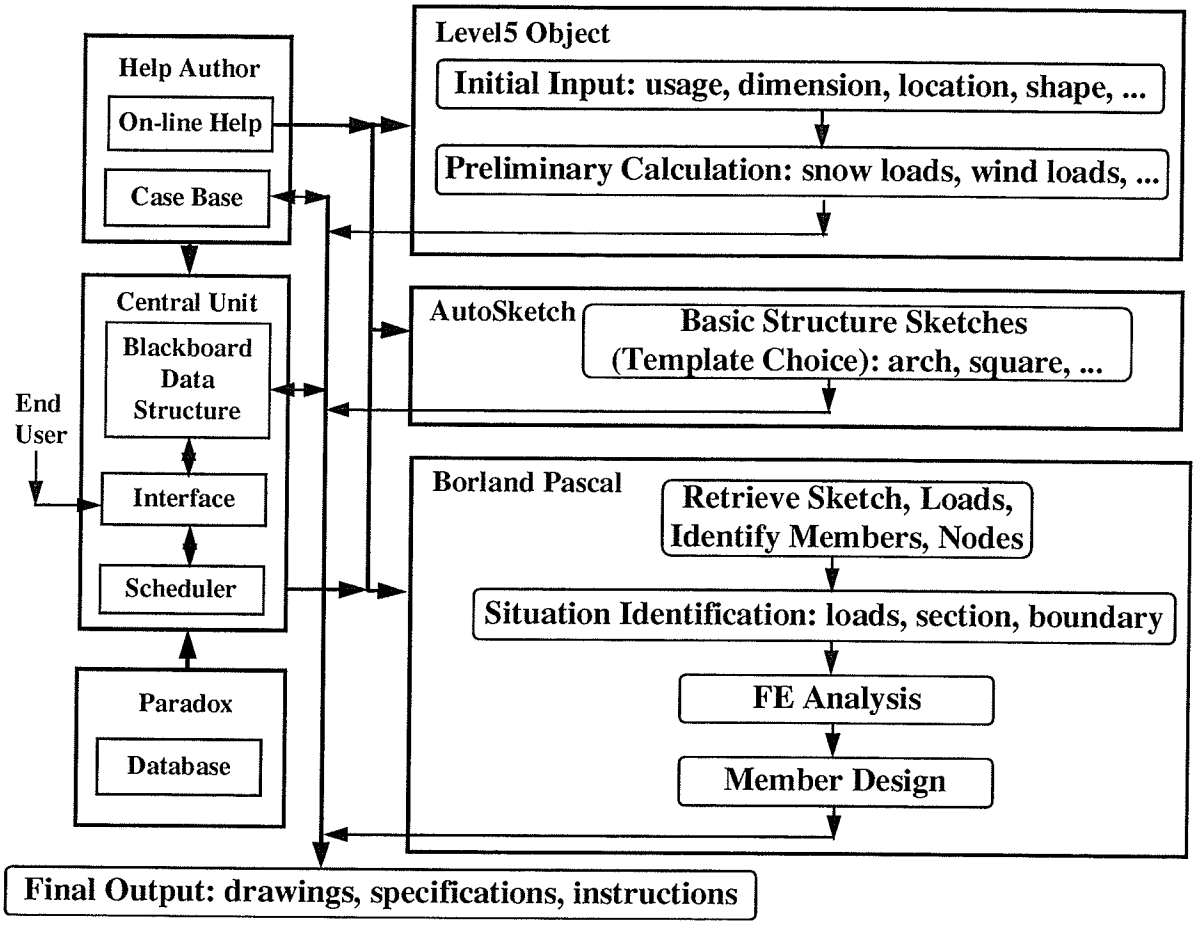


Fig.4.2. The interactions of function units in BEDA

Typical logic inferences and a part of the user machine interface were developed in Level5 Object. Level5 supports backward-chaining, forward-chaining, and procedural inference strategies. Through its database and file management functions, Level5 can share data resources, intermediate information, and final results with external programs. Its built-in communication engine allows the activation of external programs through Level5 object rules or methods. Under the system control, other function units are connected, executed, or closed.

AutoSketch is connected with BEDA for design concept development. Currently BEDA accepts and manipulates graphics in *dxf* format therefore results are saved in this form. Other graphical software can be used to substitute for AutoSketch as long as it can produce *dxf* files. Details of graphics manipulation are given in section 5.2.

Units for engineering calculation, finite element analysis, and member design are developed in Borland Pascal. Information flow within them is object-oriented. Input from other units, such as AutoSketch and Level5 Object, is first transformed to object representations by subunit DXFDEAL and L5ODEAL. The object-oriented information manipulations are detailed in Sections 5.2, 5.3, and 6.1.

A case base is developed with HelpAuthor, in a style similar to standard Windows on-line help. It contains available templates and successful cases for each design step. Bad examples and design failures are also stored for warning references. Together with on-line help, it guides end users through the design process. See section 6.2 for detail about case base and on-line help.

Design data from design codes are stored in a database developed in Paradox. Data are physically stored in *dbf* files. Weather data and physical features of building materials are among the data in database.

4.3.2. Blackboard data structure

Engineering design with BEDA is a process of information transformation, from initial requirements to final solutions. The blackboard reflects the process of data evolution, although real changes happen inside each function unit. Ignoring functioning details in each unit and classifying program units according to their development tools, we can illustrate the blackboard data structure as Fig.4.3 shows.

In Fig.4.3, dash-lined squares are knowledge sources that produce the data change. The large solid-lined square is the blackboard, a global data store for BEDA. It can be viewed as a data structure of three hierarchies. In the first layer are the initial user input and raw system materials. Intermediate results from function units are in the middle layer. Final results go to the final layer.

Initial user input goes through the system interface, which allows a user to change attribute values in the system knowledge base. Information from the case base and data base is in the category of raw material that is put into the blackboard in forms of *cas* files and *dbf* files,

respectively. A *cas* file is retrieved directly from the case base, while all *dbf* files are connected to the system through Level5's database management function.

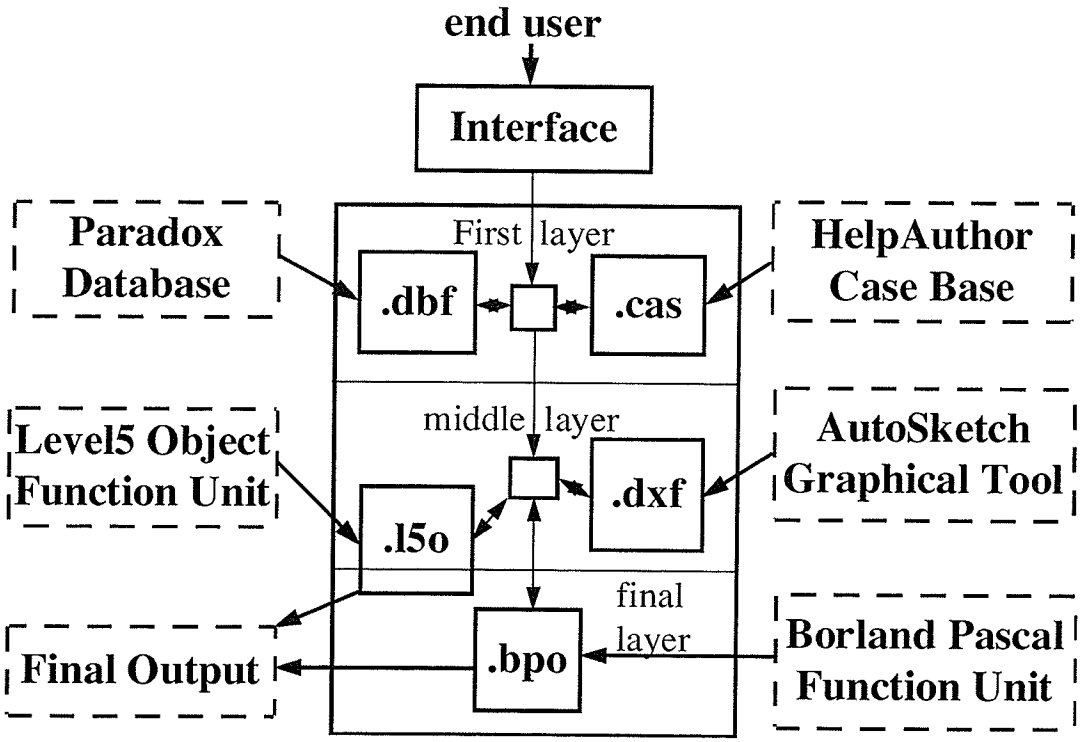


Fig.4.3. The blackboard data structure

Design results from AutoSketch and the Level5 unit are sent to the blackboard as *dxf* files and *l5o* files. They form the input to the Borland Pascal unit. Most of the final output originates from the Borland Pascal unit in form of bpo files. Some parts of results from the Level5 unit are included in the final output.

4.3.3. The information flow in BEDA

Information flow in BEDA, from initial requirements to final solutions, is an iterative process. It reflects the nature of engineering design. Knowledge may go through a few cycles in different function units until an acceptable solution is reached.

Knowledge and information may have different forms. However they are all represented and manipulated as objects in the blackboard data structure. This can be accomplished because both Level5 Object and Borland Pascal support object-oriented programming.

Final results of a design are the appropriate dimensions, orientations, and materials of all structural members. In BEDA, each structural member is represented as an object, with a datafield specifying its attributes and methods stipulating the way it is manipulated. The design subject, a wood-frame building in this instance, is identified as a collection of objects. In general, any entity in an application is identified as an object or an set of object.

The initial sketch of a design is transferred to a collection of graphical elements, lines, arcs, etc., from its *dxf* file. At the begining, each graphical element is identified as at least one object with an almost empty datafield. During the design process, knowledge is contributed from various knowledge sources, manipulated by each function unit, checked by the control mechanism, and distributed to the datafields of the objects. Specific functional details are outlined in Chapters 5 and 6. The process is governed by design theories and common sense logic. When the datafields of all objects are properly filled, the design process is finished.

4.3.4. The blackboard scheduler

BEDA adopts a dual control mechanism through an automatic scheduler and the human designer. Information flow in BEDA is initiated and controlled by the dual control mechanism. It starts a selected function unit at the proper design stage and coordinates the operation of all function units.

The automatic scheduler is a rule based decision supporting module. It has a set of built-in standards to evaluate the design process. States of problem solving are marked by flags on the blackboard. Based on flag changes, function units are activated or stopped. The scheduler keeps checking information changes on the blackboard. According to the evolution of the problem solving process and current states of function units, it initiates or advises for proper system

actions. If changes tend to lead incrementally to a solution to the problem, the activities will be intensified. If the trend is toward a dead end or an unacceptable solution, it will be adjusted.

The human designer or the system user is considered an integral part of the control mechanism. BEDA is not a black-box styled, fully automatic system. It emphasizes the participation of human designers in the control of the problem solving process. Human controllers complete the system control mechanism. They have a higher priority than the machine scheduler and can override decisions made by it. Whenever the machine scheduler fails in an inference process, the system will call for the contribution of human designers. All human-machine interactions take place through the special built user-machine interface, detailed in Section 6.3.

4.4. Conclusion

BEDA is a knowledge-based system for engineering analysis and design. It allows designers to maintain control of the design process, but utilize the work reduction capabilities of the computer.

The blackboard model of problem solving was adopted as the basic system framework. It permits the use of diverse knowledge sources. Commercial software packages are utilized to accommodate various functions of BEDA system.

BEDA is composed of several functional units. Each implements a special function of engineering design. Some can operate as a stand-alone design tool. All units are self-contained and independent from each other. This makes it easy to revise, upgrade, or generalize the functions of BEDA. A single unit can be partly revised to reflect the change in design theory and practice without affecting other units.

BEDA runs under the MSWindows environment on DOS computers. The system was tested for design of wood-framed buildings.

5. FUNCTIONAL UNITS OF BEDA

BEDA consists of several functional units. The working procedures and the construction of these individual units are discussed in detail in following sections. Discussions relate specifically to the BEDA for wood-frame buildings.

5.1. The Precalculation Unit

5.1.1. Introduction

The Precalculation Unit of BEDA has been developed in Level5 Object, an expert system shell. It was created to calculate external loads applied to the building under design. The calculations were based on the user's initial input about the building, including the basic shape, the location, and the usage.

External loads include, but need not be restricted to snow load, wind load, and earthquake load. Procedures for estimating snow load and wind load were encoded into the system. The specific objectives for creating this unit were:

(1) to encode the procedures for estimating snow load and wind load as production rules in the knowledge base of BEDA, complying with the National Building Codes of Canada (NBCC) (NRC, 1990) and the Supplement of NBCC (SNBCC) (NRC, 1990);

(2) to store meteorological data, including the ground snow load and the reference velocity pressure of wind, across Canada in the database of the expert system;

(3) to automate the process of identifying situations from the input, tracing data from the database, and activating appropriate procedures to calculate snow load and wind load; and

(4) to present results with descriptions and graphics through a windows-styled user interface.

The development of this function unit was to liberate designers from the tasks of searching values from tables and calculating the resulting parameters. These are the most tedious and error prone works in external load estimations. Meteorological data across Canada were stored in the database of BEDA. Load calculation procedures were encoded as production rules

in the knowledge base. Once users specify the location and the basic geometry of the structure under design, the expert system will calculate external loads complying with NBCC (NRC, 1990).

5.1.2. Snow load estimation

Snow loads are the environmental effects on the building due to snow accumulation on the roofs. They are one of the main causes in the failure of agricultural buildings in Canada. Snow loads on roofs vary dramatically with geographical location (climate), site exposure, and shape of the building.

Snow load estimation for building design is specified in Section 4.1.7 of NBCC (NRC, 1990) and Commentary H of SNBCC (NRC, 1990). In the design codes, the snow loading, S, on a roof or other building surface subject to snow accumulation is calculated as:

$$S = S_s(C_b C_w C_s C_a) + S_r$$

where

S_s = the ground snow load (kPa)

S_r = the associated rain load (kPa)

C_b = the basic roof snow load factor

C_w = the wind exposure factor

C_s = the slope factor

C_a = the accumulation factor

Ground snow load S_s and the associated rain load S_r for 644 geographical locations across Canada are tabulated in Design Data for Selected Locations in Canada in SNBCC (NRC, 1990). The basic roof snow load is set at 80 per cent of the ground load, i.e., $C_b = 0.8$. Other roof snow load factors, C_w, C_s, and C_a, are functions of site exposure, geometry of the building, and load cases. The calculations of C_w, C_s, and C_a are specified in SNBCC (NRC, 1990) from Figure H-1 to Figure H-6.

To design a building, three different cases of snow loading need to be considered. Case 1 is uniformly distributed snow load identified as S_1, Case 2 the nonuniformly distributed load S_2, and Case 3 the unusual situation of snow load S_3.

In a traditional design process, the designer needs to identify a location, which is the nearest to the building site, from the 644 locations listed in the table of Design Data for Selected Locations in Canada. The ground snow load S_s and the associated rain load S_r are then decided for that location. Roof snow load factors, C_b, C_w, C_s, and C_a for the three load cases are calculated following specifications in SNBCC (NRC, 1990). Some calculations are very complicated, such as those for arch or curved roofs, which involve lengthy formula deductions and integrations.

5.1.3. Wind load estimation

Wind loads are the specified pressure or suction due to wind on part or all of a surface of a building. Wind load estimation for building design is specified in Section 4.1.8 of NBCC (NRC, 1990) and Commentary B of SNBCC (NRC, 1990). In the codes, the wind loading, P, on a building surface is calculated as:

$$P = qC_eC_gC_p$$

where

P = the specified static external pressure acting statically and in a direction normal to the surface either as a pressure directed towards the surface or as a suction directed away from the surface (kPa),

q = the reference velocity pressure (kPa),

C_e = the exposure factor

C_g = the gust effect factor

C_p = the external pressure coefficient averaged over the area of the surface considered.

The net wind load for the building as a whole is the algebraic difference of the loads on the windward and the leeward surfaces. In some cases it may be calculated as the products of the

external pressures or suctions and the areas of the surfaces over which they are averaged (NRC, 1990).

The net specified pressure due to wind on part or all of a surface of the building is the algebraic difference of the external pressure or suction and the specified internal pressure or suction (NRC, 1990). The specified internal pressure due to wind is calculated from:

$$Pi = qC_eC_gC_p$$

where

P_i = the specified static internal pressure acting statically and in a direction normal to the surface either as a pressure directed towards the surface or as a suction directed away from the surface (kPa),

q = the reference velocity pressure (kPa),

C_e = the exposure factor

C_g = the gust effect factor

C_{pi} = the internal pressure coefficient.

The reference velocity pressure, q, with a probability of being exceeded in any one year of 1 in 10, 1 in 30, and 1 in 100 are tabulated in the Design Data for Selected Locations in Canada in SNBCC (NRC, 1990) for 644 geographical locations across Canada. Wind load factors, C_e, C_g, C_p, and C_{pi}, are functions of site exposure, geometry of the building, and other factors. Their calculations are specified in SNBCC (NRC, 1990) from Figures B-7 to Figure B-12.

Traditionally, the designer identifies a building site from the 644 locations in Canada and then determine the reference velocity pressure q for that location. Wind load factors, C_e, C_g, C_p, and C_{pi}, are calculated following specifications in SNBCC (NRC, 1990). Wind load estimation is an extremely dreary exercise with manual calculation. Wind load is different for different surfaces: end walls, side walls, and roofs. Its calculation is also different for main structural members, secondary structural elements or cladding, and the building as a whole. Wind effects

are decided not only by the location, site exposure, and geological status of the building, but also the openings on the building, wind directions, and other factors.

5.1.4. Construction of the function unit

5.1.4.1. Expert system shell

A commercial expert system shell, LEVEL5 Object (LEVEL5 for short) (INFORMATION BUILDERS, 1993), was used to develop the function unit for external load estimation. LEVEL5 features object-oriented programming, production rule language, relational database model, and interactive user interface. A built-in inference mechanism provides a combination of backward-chaining, forward-chaining, and procedural inferring strategies. Through its database management and file management function, LEVEL5 can share with external programs data resources, intermediate knowledge and final results.

In LEVEL5, knowledge is represented in a class structure, with attributes defining its characteristics, instances holding actual values, and methods or rules regulating its behaviours. When a knowledge base is developed , any entity in the application is identified as a class or an attribute of a class. A class can inherit other classes. When a class is inherited, its attributes, instances, and methods are all inherited.

Two types of methods can be attached to an attribute: the WHEN-NEEDED method and WHEN-CHANGED method. A WHEN-NEEDED method specifies the way of evaluating the attribute when its value is needed in the referring process. A WHEN-CHANGED method specifies a series of actions which will be triggered by the value change of the attribute.

The inference mechanism supports both backward-chaining and forward-chaining. Backward-chaining starts from a hypothesis or a goal. It infers backward along a chain to find the evidence supporting the hypothesis or the goal. Forward-chaining starts with known facts or conditions to infer new facts. A chain of inference consists of a series of rules, demons, and methods. Rules and demons are all IF-THEN statements, expressing the logic relationships between causes and effects. A method contains a sequence of procedural statements. The

WHEN-CHANGED method is executed when its attribute changes value. The WHEN-NEEDED method is fired to determine the value of attributes. Backward-chainings use WHEN-NEEDED methods and rules. Whereas forward-chainings use WHEN-CHANGED methods and demons.

5.1.4.2 Knowledge representation

Knowledge is represented as classes and class-related methods in BEDA. Classes represent declarative knowledge. Their methods and production rules represent procedural knowledge.

Class and Attributes

Snow load is defined as a single class in the knowledge base. All parameters for snow load estimation are identified as attributes under the snow load class. Attributes are the subjects being manipulated by methods and production rules.

The Snow Load class includes the following attributes:

S_1, S_2, S_3: final goals,

S_s, S_r, C_a, C_b, C_w, C_s: parameters,

roof shape, roof exposure: query items,

OK_{in}, OK_{out}: control flags.

Query items are the basic information queried from the end user by the expert system. OK_{in} and OK_{out} are control flags used by the system to control the inference process. Each control flag or query item has a corresponding interface item on the user-input windows. See Fig.5.1, the input window for snow load estimation, for an example. Through the interface, a user can respond to a query or change the state of a control flag.

In the knowledge base, a class is introduced by the reserved word "CLASS" and followed by its attributes. Attributes are initiated with the reserved word "WITH". The Snow Load class and its attributes in the expert system is expressed as follows:

CLASS Snow Load
WITH snow load S_1
WITH roof exposure
WITH Roof shape
WITH OK_{in} SIMPLE

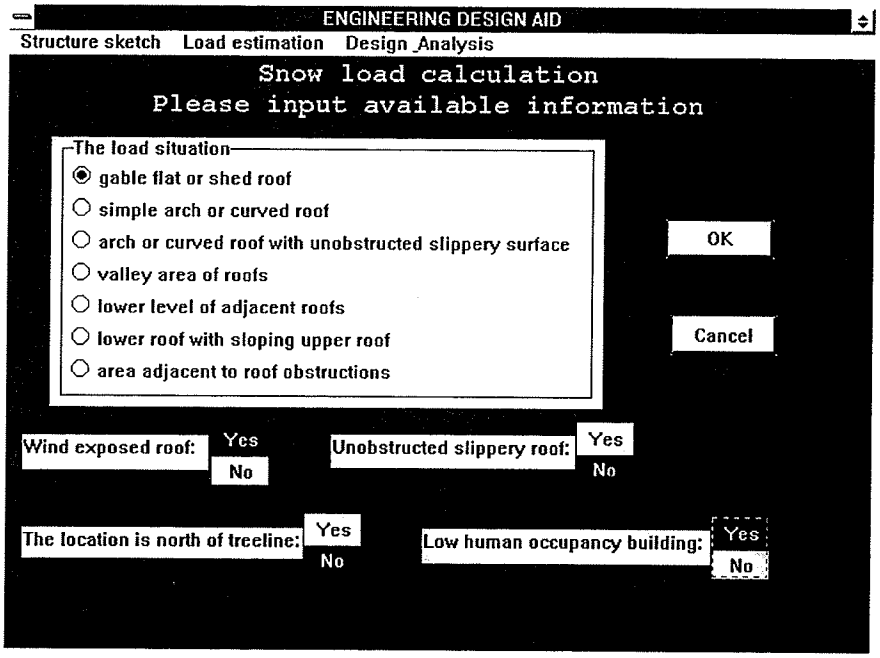


Fig.5.1. The input window for snow load estimation

An attribute can be of such type as STRING, NUMERIC, SIMPLE, COMPOUND, COLOR, PICTURE, RECTANGLE, or TIME. It can be assigned an initial or a default value. Each attribute has several inference facets. These facets designate the strategy used by the inference mechanism to process the attribute and the way in which the attribute is queried from end users. In the knowledge base, features of an attribute are specified with further descriptions. For example:

WITH Load situation COMPOUND
gable flat or shed roof,
simple arch or curved roof,
arch or curved roof with unobstructed slippery surface,
valley area of roofs,
lower level of adjacent roofs,
lower roof with sloping upper roof,
area adjacent to roof obstructions;
INIT gable flat or shed roof

As illustrated, the attribute "Load situation" is of the COMPOUND type. The type of an attribute is specified by the capitalized word immediately following the name of that attribute. Load situation can take the value of "gable flat or shed roof", "simple arch or curved roof" or any of the other choices listed under the attribute. The reserved word "INIT" assigns an initial value "gable flat or shed roof" to Load situation.

Attribute Related Methods

The procedural knowledge in BEDA is represented as production rules in the rule base. These production rules codify the operational logic, rules-of-thumb, and cause-and-effect relationships that comprise the actions of an application.

All production rules are expressed as attribute related methods in BEDA. A method is activated when its related attribute changes value or the value of that attribute is under pursuit in the inference process.

For example, the WHEN-NEEDED method for Case 1 snow load, S_1, is fired when BEDA pursues the value of S_1. In the knowledge base, the method is expressed as:

WITH snow load S_1 NUMERIC
 WHEN NEEDED
  BEGIN
   IF Load situation IS simple arch or curved roof THEN
    snow load $S_1 := S_s * C_b * C_w + S_r$
   ELSE
    snow load $S_1 := S_s * C_b * C_w * C_s + S_r$
  END

The attribute, snow load S_1, is introduced by the reserved word "WITH". Its type is "NUMERIC". A WHEN-NEEDED method is introduced by the reserved word "WHEN NEEDED" and positioned right after the type specification of the attribute. The logics of the method are in the "IF THEN ELSE" format and bracketed with the reserved words "BEGIN" and "END". Whenever the value of S_1 is needed in the inference process, this method is activated and S_1 is calculated accordingly.

Similarly, a WHEN-CHANGED method is introduced by the reserved word "WHEN CHANGED". It has the same format and position as a WHEN-NEEDED method in knowledge representations. For example, the WHEN-CHANGED method for attribute OK_{in} is written as:

```
WITH OKin  SIMPLE
 WHEN CHANGED
  BEGIN
   PURSUE snow load S1
   PURSUE snow load S2
   PURSUE snow load S3
    ......
  END
```

OK_{in} is an attribute of "SIMPLE" type. It takes the value of 'True' or 'False'. When its value changes during the inference process, the WHEN-CHANGED method is fired. The "PURSUE" command forces the system to pursue the value of an attribute.

Database

The database is an integral part of the knowledge base in BEDA. It stores the design data from SNBCC (NRC, 1990), including S_s, S_r, and q. BEDA has a relational database developed in Paradox. It consists of 12 *dbf* files, each containing design data for one province or territory of Canada.

In the knowledge base, database is a special class. It has the same structure as its *dbf* files, with each attribute representing a field of the files. All data management functions associated with files are incorporated into the knowledge base. BEDA treats a database class the same as other classes in the knowledge base.

The 12 *dbf* files form 12 instances of the database class. A set of methods is developed and attached to the database class. BEDA can automatically look through the database and search for suitable design data during the inference process.

5.1.5. Procedures for snow load estimation

Snow load estimations in BEDA strictly follow specifications of NBCC (NRC, 1990) and SNBCC (NRC, 1990). The three cases of snow load, S_1, S_2, and S_3, constitute the inferring goals in snow load estimations. The relationship between the goals and their preconditions is summarized in the flow chart shown in Fig.5.2:

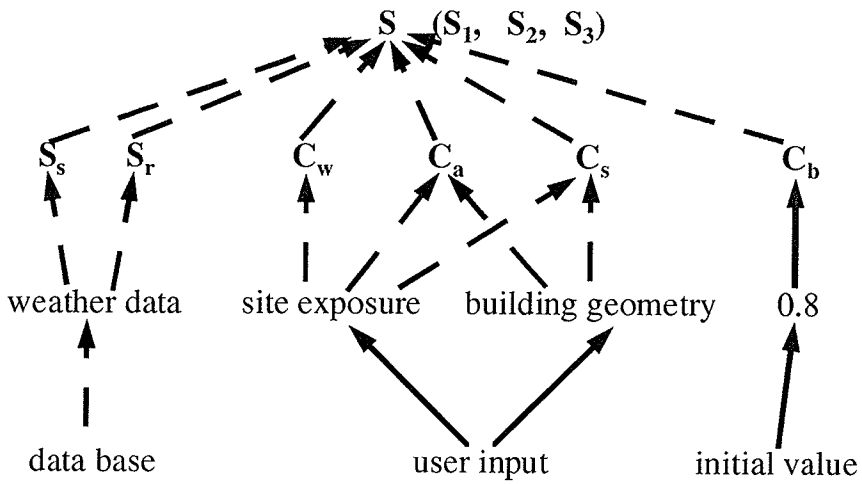


Fig.5.2. The knowledge tree of snow load estimation

This chart can be viewed as the knowledge tree for snow load estimation. It reflects the paths of information flow in the system. Its base level is the information source. The top level is the goal. Attributes on any level are achieved based on attributes on the level below. In Figure 5.2, a solid line indicates a direct value assignment from a lower level to an upper level. Dashed lines indicate that the upper attribute is evaluated, using a predefined method, based on lower attributes.

Initially, snow load calculation goes downwards from upper levels to lower levels. The inferring goals of the system are the snow loads, S_1, S_2, and S_3. Their WHEN NEEDED methods force the system to trace S_s, C_b, C_w, C_s, C_a, and S_r. WHEN NEEDED methods for S_s, C_b, C_w, C_s, C_a, and S_r are fired thereafter. The value of C_b is assigned when it is first defined in BEDA. Appropriate values for S_s and S_r are traced from the database according to user input about the location. Calculations of C_w, C_s, and C_a go down further to the user input about

building geometry and site exposure. When all basic information is obtained, the calculation process goes backwards from lower levels to upper levels until the three snow loads are achieved.

When necessary, BEDA will query the user for information input. The following is a list of items queried by the system:

Geographical location (2 items):

Province

Location

Site exposure (3 items):

Is the roof wind exposed?

Is the building north of treeline?

Is the building a low human occupancy building?

Geometry of the building (roof) (3 items):

The roof slope

Is the roof unobstructed slippery?

Load situation

gable flat or shed roof,

simple arch or curved roof,

arch or curved roof with unobstructed slippery surface,

valley area of roofs,

lower level of adjacent roofs,

lower roof with sloping upper roof,

area adjacent to roof obstructions.

When the user specifies the Load situation, the system may query a few more items. Depending on the roof type, information such as roof span for arch or curved roof, elevation difference between adjacent roofs, and/or width of a roof obstruction, may be requested.

When the input is over, the user can mouse-click an OK button on the user interface. The button is attached to the attribute OK_{in} of the Snow Load class. Clicking it changs the value of OK_{in} and fires its WHEN CHANGED method. The Pursue commands in the method then trigger WHEN NEEDED methods associated with snow loads S_1, S_2, and S_3. It starts the goal driven, backward-chaining process of snow load calculations.

When the calculations are complete, an output window (see Fig.a10) will be activated to display the results. An OK button on the window, which is attached to the attribute OK_{out} of the Snow load class, can be clicked to terminate the execution of the functional unit.

5.1.6. Procedures for wind load estimation

Procedures for estimating wind loads are similar to those for estimating snow loads. They strictly follow specifications of NBCC (NRC, 1990) and SNBCC (NRC, 1990). The relationship between the goals and their preconditions is summarized in Fig.5.3.

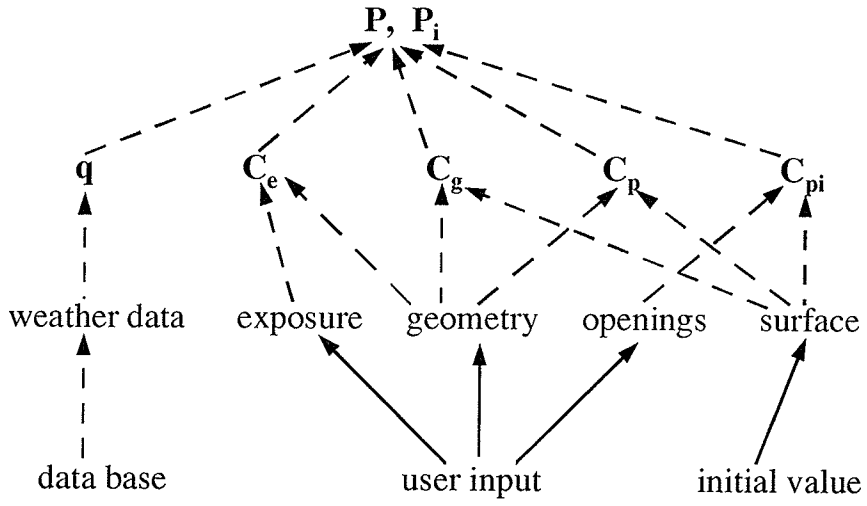


Fig.5.3. The knowledge tree of wind load estimation

The base level of the flow chart is the information source. The top level is the goal. Again a solid line indicates a direct value assignment from a lower level to an upper level. A dashed line indicates that the upper attribute is evaluated, using a predefined method, on the basis of lower attributes.

The following is a list of items queried by BEDA for wind load estimations:

Geographical location (2 items):

Province

Location

Design situation (4 items):

building as a whole,

main structural member for strength,

main structural member for deflection or vibration, secondary structural element or cladding

Openings on the building

large or significant opening on one wall,

nonuniformly distributed openings

small uniformly distributed openings

airtight

Geometry of the building (4 items):

The roof slope

Building length

Building width

Building height or wall height

Human occupance

high

low

After the input is over, the user can mouse-click an OK button on the user input display. It starts the goal driven, backward-chaining process of wind load calculations. Results are displayed on an output window, as shown in figure 5.4. An OK button on the window can be clicked to terminate the execution of this functional unit.

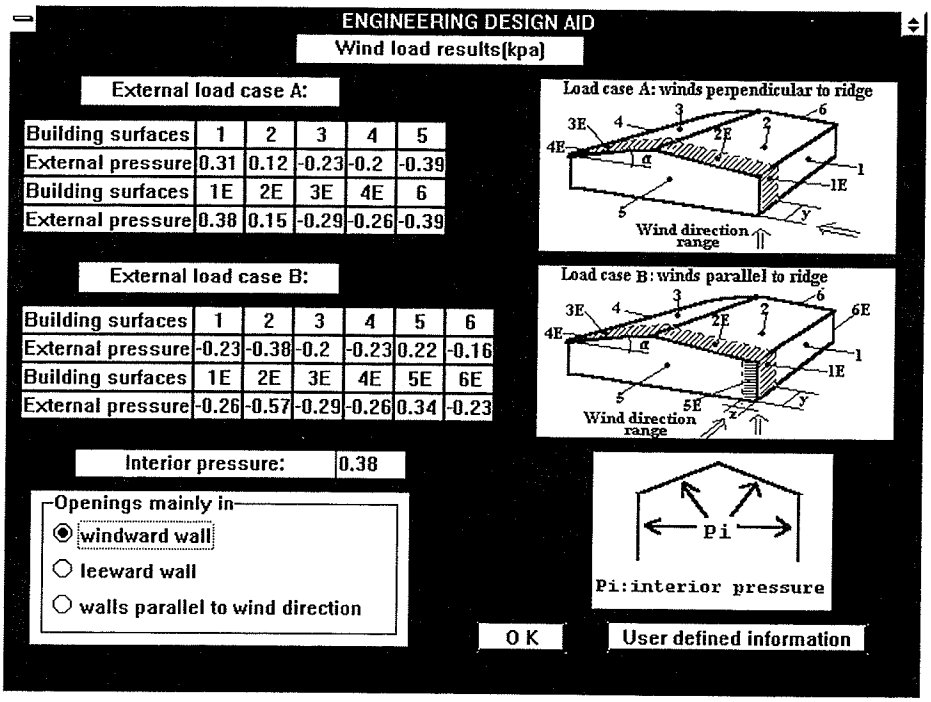


Fig.5.4. The output window for wind load results

5.1.7. Summary

External load estimations have been encoded as a functional unit of BEDA. The unit is knowledge based and written in Level5 Object. Users specify the graphical location, site exposure, and geometry of the building. The system accomplishes tedious tasks of searching tables and calculating parameters. The interface was designed to be user-friendly. Minimum user input and clear presentation are attained.

5.2. From Graphical Design to Structural Analysis

5.2.1 Overview

Engineering design involves sequential alternations between sketching and analysis. Sketching is a creative process, in which designers conceive tentative configurations of a design. Analysis is a stage to verify contemplations and to determine parameter values. Analysis often results in revisions to the sketch. Once a sketch is revised, it normally needs another round of analysis.

Computer software has been developed for each of the two design stages. In the last 10 years, the technique of Computer Aided Design (CAD) has experienced rapid development. Two independent fields of CAD have evolved. One is the Computer Aided Graphics Design (CAGD). Software for CAGD includes such programs as AutoCAD, DesignCAD, and AutoSketch. The other field is Computer Aided Analysis (CAA), which includes many Finite Element Analysis programs. CAD techniques have reached a stage where sophisticated graphics or analysis works can be done easily with computer programmes.

CAD techniques bring great advantages to engineering design. Many CAGD programs support interactive graphical design. Graphics can easily be drawn, edited, and even stored for future use. Results can readily be printed out with printers or drawn with plotters. Some Computer-Aided Analysis Programmes have adopted well advanced mathematic methods, such as Finite Element Analysis, Boundary Element Analysis, and other special techniques. Complete analysis can be preformed for large and complicated structures.

Unfortunately, the two sequential procedures of a design process form two isolated fields in CAD. Direct linkages between the two are rarely seen [Garrett and Maher, 1991]. The process of Computer Aided Graphics Design stops once graphics are complete. Its results cannot be directly used by analysis programs. Users must re-input all information of the drawing into the analysis program. Just consider the identification of coordinates of all the graphical elements, the data identification and input is obviously a time-consuming and error-prone process.

Information is represented differently in CAGD and CAA. Basic information manipulated by graphic software is in the form of geometrical elements, which are intuitive and close to the physical shape of a design subject. Programmes for analysis mainly deal with variables and numbers, which provide more accurate descriptions and are more appropriate for calculations. The difference is a major obstacle to linking the two fields.

BEDA adopted an object-oriented approach for linking the graphical design process with the structural analysis process. An intelligent link was built to convert graphics into collections of objects, which can then be directly input to analysis programmes. A user-machine interface

was developed for identifying connection types, boundary conditions, and external loads of the structure.

The object-oriented knowledge representation is not limited to linking graphical design with structural analysis. It is the way of integrating all functional units in BEDA.

5.2.2 Object-oriented knowledge representation

Object-oriented knowledge representation is similar to the frame-based knowledge representation in artificial intelligence. An object is an abstract representation of the physical element. Any subject, a building, a bridge, or a car, can be identified as an object and/or a collection of objects. A more complete discussion of objects is given in Section 6.1. For information beyond, the reader is referred to references such as Borland Pascal with Objects, User's Guide [Borland, 1990].

In engineering design, the subject under design is the focus of all working stages. At the graphical design stage, the subject is represented as a collection of geometrical elements, such as lines, curves, or circles. In finite element analysis, it is represented as a collection of elements, such as line segments, triangles, or rectangles. Except for their physical meanings, all elements are similar objects. Each object forms a basic element for manipulation, represents a piece of the subject, and presents itself with unique coordinates and orientations.

A distinguishing feature of objects is the integration of data and code. Data regulate the flow of code. Code manipulates the shapes and values of data. In programs, an object is very much like a record attached to a few blocks of code. The record is the datafield of that object, which wraps related parameters under the object's title and describes its characteristics. Attached code blocks, called methods, are special procedures or functions to regulate its behaviours. With datafield and methods, an object models the physical element, or elements, it represents.

Certain programming languages, such as C++ and Borland Pascal, support object-oriented programming. Within BEDA, the interface for linking graphical design and structure analysis is written in Borland Pascal.

In BEDA, the basic object used for linking graphical design and structural analysis is TMember, (see Section 6.1). TMember is an abstract representation of structural elements. Each element of a design subject can be identified as a TMember type object. The whole structure being designed is represented as a collection of such objects. Borland Pascal provides a special type of object, called collection, for holding objects. A collection type object is a dynamically sized object storage. It has an initial size but can grow at run time to accommodate stored data. It can host objects of different types and sizes. Methods can be built in it to manipulate those objects held within the collection.

For wood-frame building design, a collection type object, named BUILDING, is defined to host structural elements represented by TMember type objects.

5.2.3 From *DXF* type of drawings to objects

The first step to link graphics design with analysis programs is to translate a graphic into a collection of objects, with each object representing a structural element. Objects are easy to manipulate for analysis programs. They are used as direct input to the analysis unit in BEDA.

In principle, any graphics can be represented as objects. Results from most graphical software can be saved as disk files. Any file is formed under specific rules. When these rules are known, graphic files can be transformed into objects accordingly.

It is much easier to transform vector styled graphics than bitmap styled ones. A vector styled graphic is represented and stored as a set of graphical elements, which can readily be transform to objects. Bitmap files represent a whole picture as a stack of neighbouring points. There is no obvious distinction between elements. In the future, artificial intelligence may be employed to write production rules to distinguish graphical elements, following a process similar to that of human visions.

BEDA has the capability to read in a file, identify graphical elements, and represent them as objects. The file must be in *DXF* format. No other graphic format has been utilized at this time.

DXF, a "Drawing Interchange" file format [Autodesk, 1988], is one of the most popular file formats for interchanging vector styled drawings among graphic programmes. Many graphic software packages, including AutoCAD, DesignCAD, and AutoSketch, support DXF format for drawing file representation.

A *DXF* file is a standard ASCII text file with a suffix ".dxf". Its HEADER section and TABLES section specify the linetype, text style, user coordinate system, and other general information. Definitions of drawing elements, the information most concerned for object representation, are contained in the BLOCKS section and the ENTITIES section.

Basic graphic elements in a DXF file include, but are not limited to LINE, ARC, CIRCLE, and POLYLINE. Each element is described by a few lines of text. The first two lines identify its name and code. Subsequent lines describe its features.

To transform a drawing into objects, BEDA reads in its DXF file line by line. When a graphic element is met, a TMember type object is created. Geometrical information of the element, its length and the coordinates of its endpoints, is transferred to the datafield of the object. Information useful for engineering analysis will be stored. Information about visual effects, such as linetype and layer, will be ignored. All objects created from a graphical file are stored in the BUILDING, a collection type object.

For simple elements, like LINE and ARC, one object is created for each element. For complicated elements, POLYLINE and BOX, one object is created for each segment of the element.

In addition to BUILDING, another collection type object called NODES will be created when a DXF file is processed. Each endpoint or vertex of an element is identified as a TNode type object, just as an element is a TMember type object. Details about TNode are outlined in Section 6.1.2. Endpoints or vertexes of all graphical elements are hosted in the NODES.

5.2.4 Identifications of additional information

Apart from geometrical features of a structure, additional properties, such as boundary conditions, section properties, and external loads, must be identified for structural analysis. These properties are determined by human designers according to structural choices, material supply, and environmental exposure. User-machine dialogue windows were developed for information input. These windows were created with ObjectWindows [Borland, 1992], an object-oriented application framework for Microsoft Windows.

Dialogue windows were represented as TDialog type objects. TDialog is an object type specially defined for communications between users and the system. Its datafield holds the items for data input and output. Its methods specify how the dialogue window is presented on the screen and how data items are manipulated.

The dialogue window for boundary condition input was called TBndrcdtn. Its appearance on the screen is shown in Figure 5.5. Its definition is as follow:

```
TBndrcdtn = object(TDialog)
  BND, BLS: ListBox;
  XCH, YCH, ZCH: CheckBox;
  XID, YID, ZID: Edit;
  BAD, BDL: Button;
  Nodalgrp: TCollection;
  constructor Init;
  procedure SetupWindow;
end;
```

The first line defines TBndrcdtn as an object of TDialog type. Items for data input-output are defined in its datafield. BND and BLS are ListBox on the dialogue window. BND lists all nodes of the design subject. BLS lists all information input by the user. XCH, YCH, and ZCH represent the degree of freedom, on the X, Y, and Z dimensions respectively, of a node highlighted in BND. XID, YID, and ZID are used to input displacements of that node. BAD and BDL represent buttons on the dialogue, which add and delete user specified information. Boundary conditions of the whole structure are stored in Nodalgrp, a collection type object.

"Constructor Init" is the method to create TBndrcdtn type object. Procedure SetupWindow specifies the appearance of TBndrcdtn on a screen.

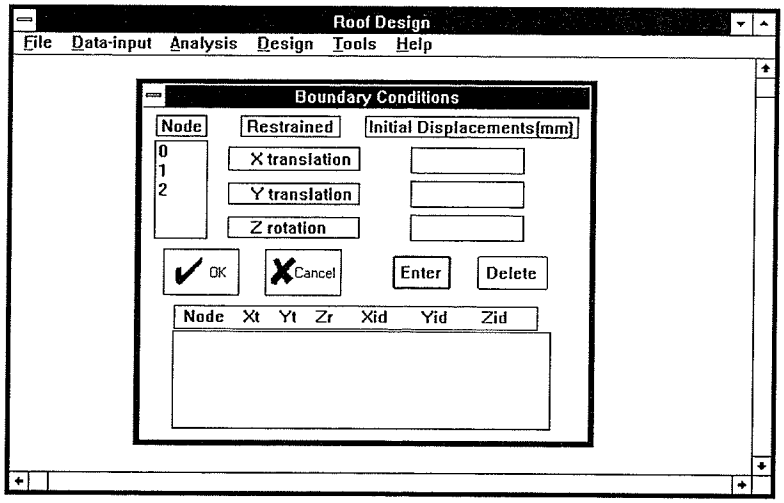


Fig.5.5. The dialogue window for boundary condition input

Similar dialogue windows were developed for user input of load situations and section properties (see Fig.a20 and Fig.a21 in Appendix A). External loads can be applied to the structure as concentrated and/or distributed loads. Load information is finally stored in a collection type object called Forcegrp. Section properties are directly transferred to each TMember type object and stored in BUILDING.

5.2.5 Finite element analysis

A two dimensional finite element analysis unit was developed to verify the feasibility of conducting analysis in an object-oriented approach. The unit takes objects as input. Output is also object-oriented. BEDA system can accommodate more complicated analysis package, such as three dimensional and nonlinear analysis. Proper interface is needed to transfer input and output into object-oriented style.

When the unit is started, a dialogue window TStrinfo is presented on the screen, see Figure 5.6. The user can specify the scale between the drawing and the real structure. The type of structure, truss, frame, or mix-typed structure, can also be specified. If the structure is of mix-

typed, all structural members will be listed on the screen. The user can then specify which members are pin-connected, and which are fixed. Information is transferred to the datafields of those objects representing these structural members.

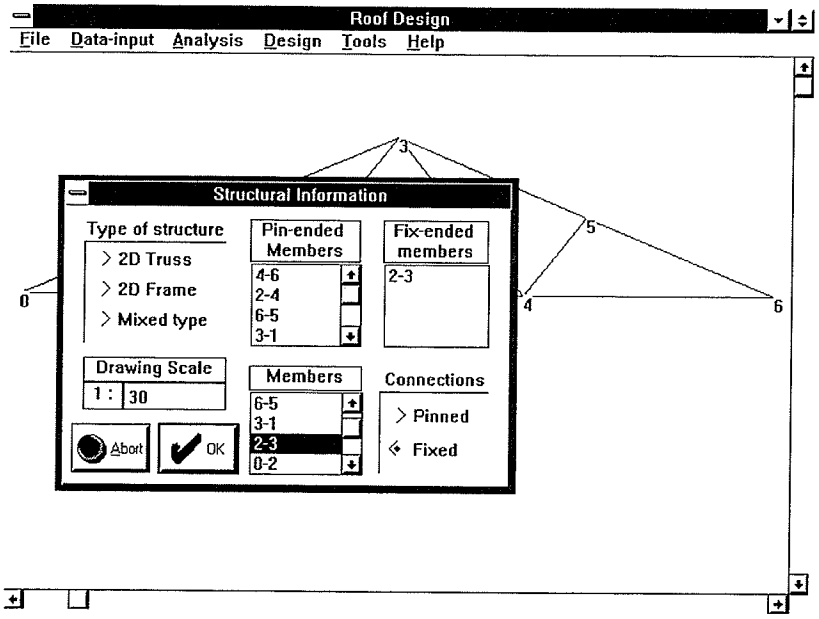


Fig.5.6. The input window for finite element analysis

The program uses an element stiffness matrix in an expanded form [Segerlind, 1984] for finite element analysis. For pin-connected members or truss structures, the area moment (Ix) is set to zero in the element stiffness matrix.

Nodal displacements are incorporated into the system of equations from object Nodalgrp. External loads are brought in by the object Forcegrp and decomposed into concentrated forces and moments. They are directly added to the global force vector {F}. Gaussian elimination is used to decompose the global stiffness matrix {K} and the global force vector {F}. System equations are solved using back substitution.

Finite element analysis results are presented to the user through dialogue TFearslt, see Figure 5.7. It consists of a set of displays. Each shows full information about one structural element, including internal forces, nodal displacements, and its geometrical and mechanical

features. Users can go through all elements one by one, or specify the one in which they are interested.

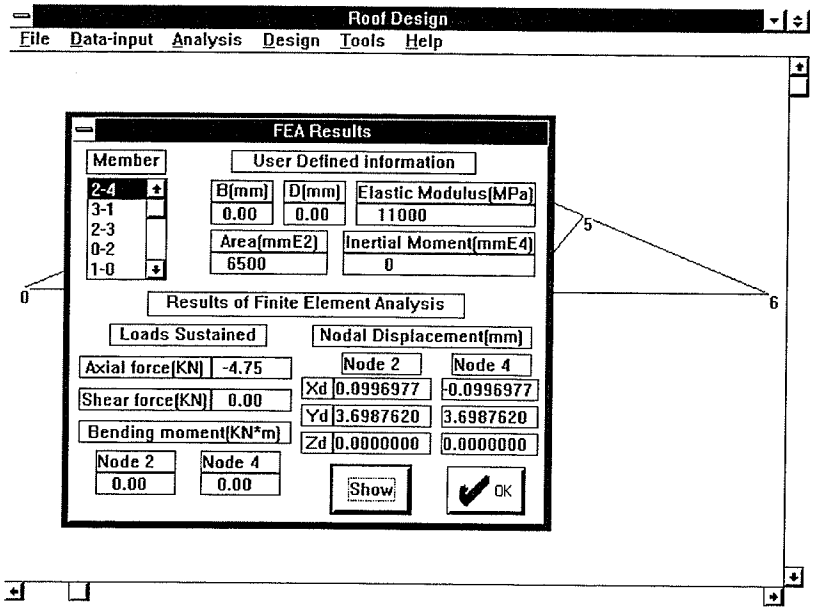


Fig.5.7. The output window for finite element analysis

Simultaneously, results are transferred to and stored in the datafield of each object held by BUILDING. The next design stage, the Member Design Unit, is also created in object-oriented programming. Finite element analysis results can be directly used as input.

5.2.6 Conclusion

An object-oriented link between graphical design and structural analysis has been developed. Direct use of graphics for finite element analysis has been exhibited. It eliminates the tedious re-input of geometrical information to analysis programs.

Objects, or object collections, form the basis for knowledge representation. The object-oriented approach unifies the information manipulation at different design stages. The principles herein demonstrated can be generalized to accommodate graphic formats other than *DXF* and more complicated analysis methods.

5.3. Members Design

5.3.1 Overview

The Member Design Unit of BEDA was developed in Borland Pascal. Its function is to select the appropriate type and size for structural members, based on previous structural and load analysis.

This function unit can perform two kinds of member designs: automatic and specific. Automatic member design will select the type and the size for all structural members. Member selections are based on results from the finite element analysis unit. Specific member design will recommend the type and the size for a single structural member specified by the user. The user designates loads sustained by the member.

Figure 5.8 illustrates the flow chart of the member design process. The design procedure follows the conventions of the Wood Design Manual published by the Canadian Wood Council (CWC, 1990). It conforms to the limit states design process outlined in CAN/CSA-O86.1-M89 (CSA, 1989). As specified, the design criterion is:

Factored resistance >= factored load effect

Factors of safety are applied to both the resistance side and the load side of the design equation. Both sides should have the same units (i.e., kN, kN*m). The load effect should be the least favourable effect considering all possible load combinations.

In developing the unit it was decided that the database should not be a simple mechanical tabulation of the design data from the Wood Design Manual (CWC, 1990). Only basic physical and mechanical features of wood were stored. This made the database incisive and universal to any wood members. The unit has built-in procedures to retrieve relevant data and to produce reference codes for design. The calculating power of computers is employed to avoid tedious retrieval of tabulated design data.

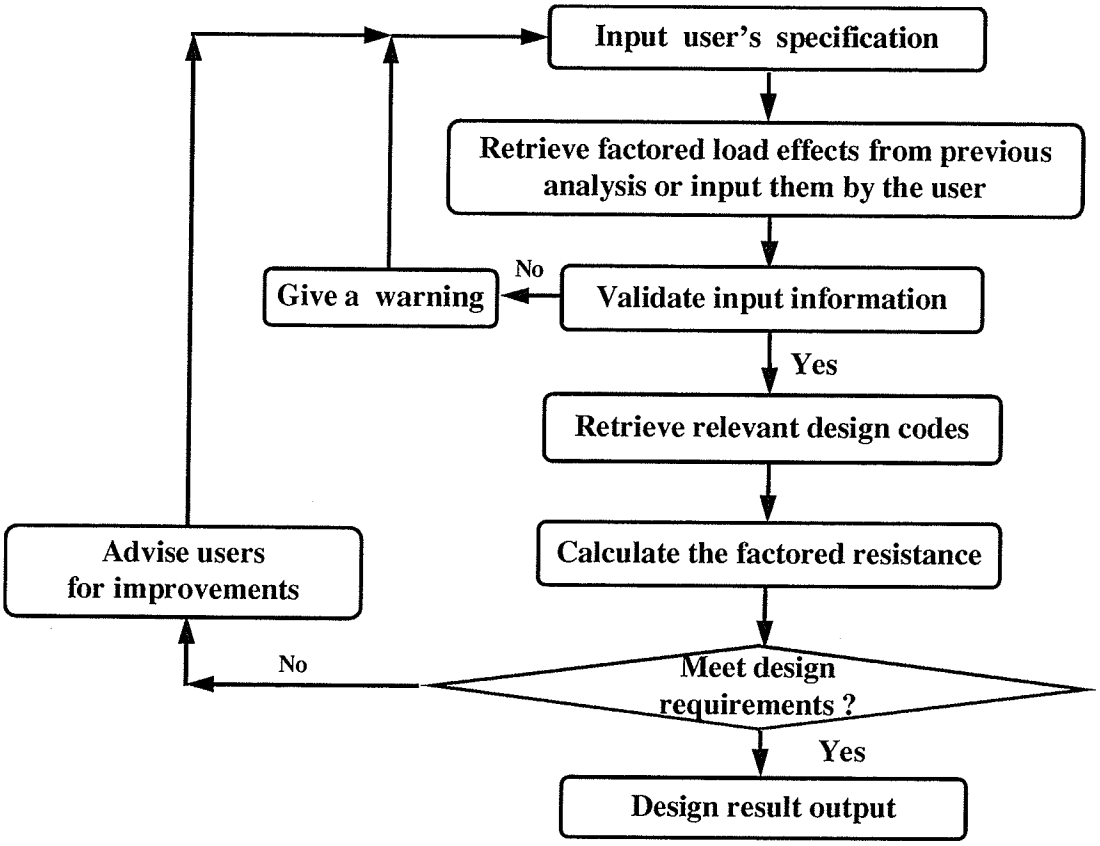


Fig.5.8. The flow chart of the member design process

Specified strengths f_b, f_v, f_c, f_{cp}, f_t and the Modulus of Elasticity (E) are basic features of wood and universal to all wood members. Their values are stored in the database. The section modulus (S), section area (A), and moment of inertia (I_x) are in the database too, since their values are slightly different from those directly calculated from member sizes.

All adjust factors specified by CAN/CSA-O86.1-M89 are determined according to the states of the member and the service environment. Their values are decided by rules and subroutines in the software.

Data input and output are in window-style and multi-choice forms. The user provides design information by selecting the most suitable choices or assigning proper values to input items. If the user does not make choices, optimal values have been set as default choices.

Initially, the unit selects the proper depth for single member with narrowest width. If no suitable member exists for that structural member, the program will choose wider member or multiple members according to the user preference.

5.3.2 Automatic member design

Automatic member design usually starts after the completion of finite element analysis. Its design procedures follow the flow chart shown in Fig.5.8.

When the user clicks the item 'design' in the pull-down menu, a user input window is displayed on the screen, see Figure 5.9.

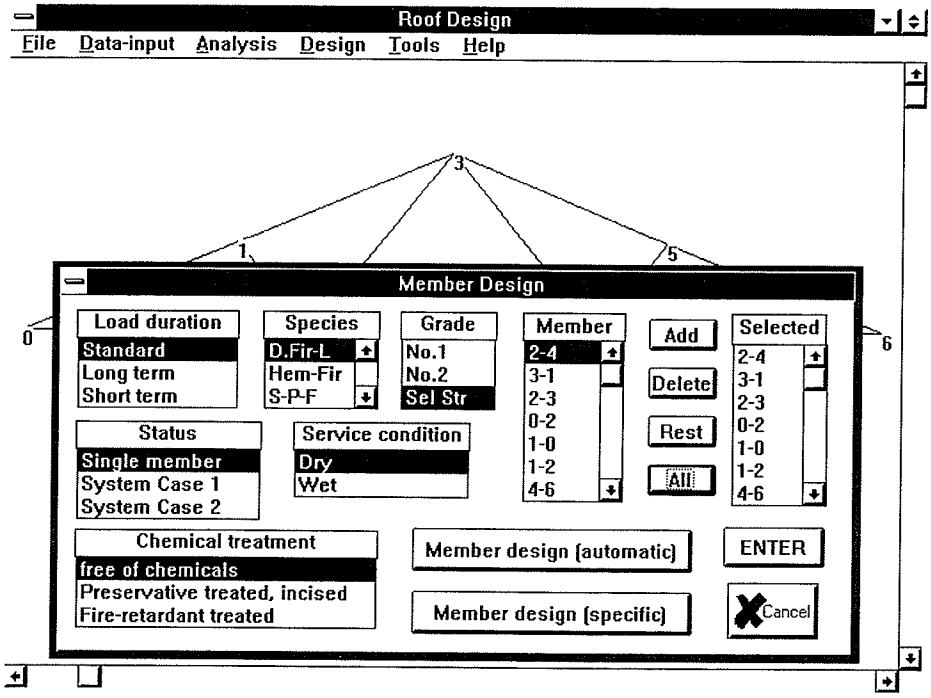


Figure 5.9. The user input window for member design

The user can specify load duration, species, grade, status, chemical treatment, and service condition for a member or a group of members according to usage, service environment, and market supply. All members of the structure are listed in the ListBox titled 'Member'. Through buttons Add, Delete, Rest, and All, the user can add or delete members listed in the ListBox

titled 'Selected'. Clicking the Enter button will apply the specified condition to the member or members in the 'selected' ListBox.

Member design is based on loads applied to each member. Since the system is developed with object-oriented programming, loads applied to each member are inherited from finite element analysis. In the datafield of TMember object, there are four items, F_x, F_v, F_{m1}, and F_{m2}, which in turn specify the axial force, the vertical shear, and bending moments applied to a member. Results of finite element analysis are distributed to F_x, F_v, F_{m1}, and F_{m2} of each TMember object and taken directly as input to the member design unit. Factored loads M_f, V_f, F_f are then induced from them.

According to the specified condition, BEDA will determine all factors for estimating load resistance, including the modification factors. Specified strengths, Modulus of Elasticity (E), section modulus (S), section area (A), and moment of inertia (I_x) are all retrieved from the database. Factored resistance M_r, V_r, and E_sI_r are then calculated, see CAN/CSA-O86.1-M89 (CSA, 1989) for detail.

The criteria for member selection are:

$$M_f/M_r + F_f/F_r <= 1$$

$$V_f <= V_r \text{ and}$$

$$EI <= E_sI_r$$

where

M_f = factored bending moment, N*mm

V_f = factored shear force, N

F_f = factored reaction force, N

M_r = factored bending moment resistance, N*mm

V_r = factored shear force resistance, N

F_r = factored reaction force resistance, N

EI = actual value for maximum deflection under specified loads

E_sI_r = required value for maximum deflection under specified loads.

The above criteria are applied sequentially to select proper width (B) and depth (D) for each structural member. Member selection starts from width B= 38. The least depth (D) that meets design criteria will be selected for a structural member. If the maximum available depth (D) does not meet design requirements for a structural member, the system will prompt the user to select either multiple members or a wider member, see Figure 5.10. The selection process proceeds iteratively until proper widths and depths are chosen for all structural members.

When the process is finished, design results are distributed to the datafield of all TMember objects. Items WdB, WdD, MemNo, Area, I_X, and Elm are then updated for each datafield. Here, WdB, WdD, MemNo, Area, I_X, and Elm respectively represent width, depth, number of multiple member, section area, moment of inertia and Modulus of Elasticity of a member. Results are presented on the output window shown in Figure 5.11.

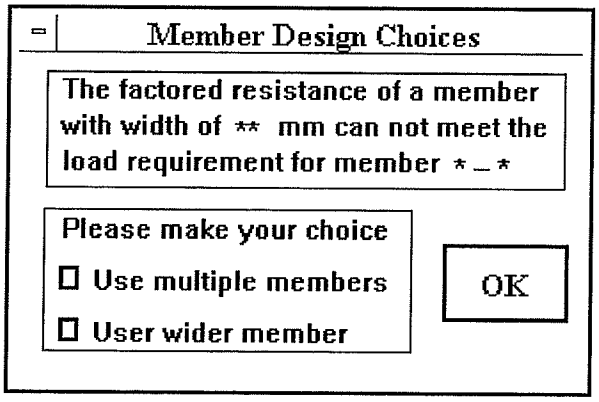


Figure 5.10. The window for member design choice

User defined information and intermediate results are also displayed for user reference. All information can be output to a printer.

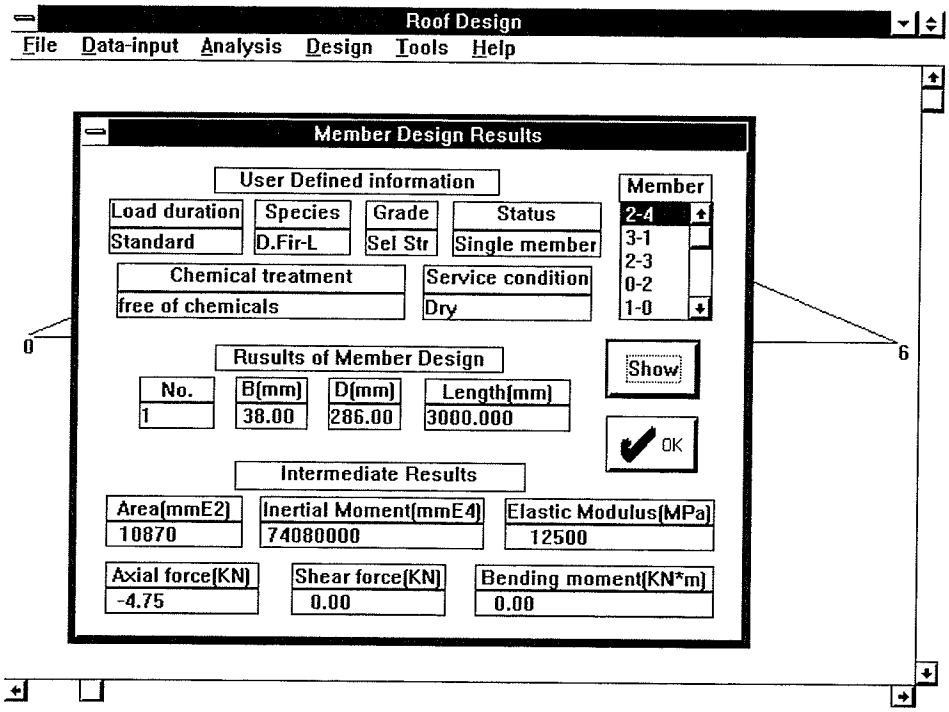


Figure 5.11.The output window of automatic member design

5.3.3 Specific member design

The specific member design function is for the user to design important or critical members in a structure. Less important members can be left for automatic design. The user is, therefore, allowed to have control of the design process and also provided with the convenience of machine design.

Specific member design follows the same procedures as automatic member design. Similar data and design criteria are utilized, but specific member design provides the user a chance to override the load condition and/or the constraints to a member. A member or a few members can be designed as unique structural elements, such as beams or columns.

Sometimes a user may feel it is necessary to revise results inherited from previous function units, such as the internal forces calculated from finite element analysis. The system provides a query for users to redefine the external loads applied to a member. For a beam, the query is as shown in Figure 5.12.

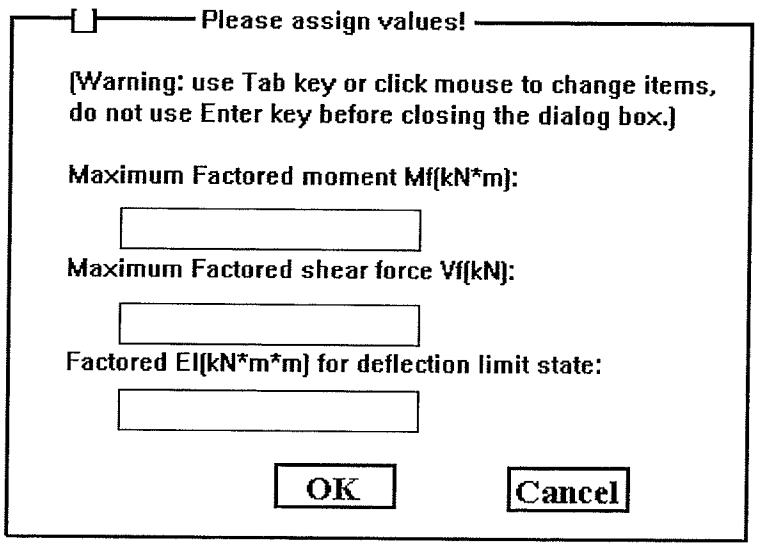


Figure 5.12. Load effect input query

The user needs to specify values for the maximum factored moment M_f, the maximum factored shear force V_f, and the factored EI.

After the input of proper information, the system will automatically trace relevant data, calculate the factored resistance, and select proper member sizes. Results are displayed in the output window shown in Figure 5.13.

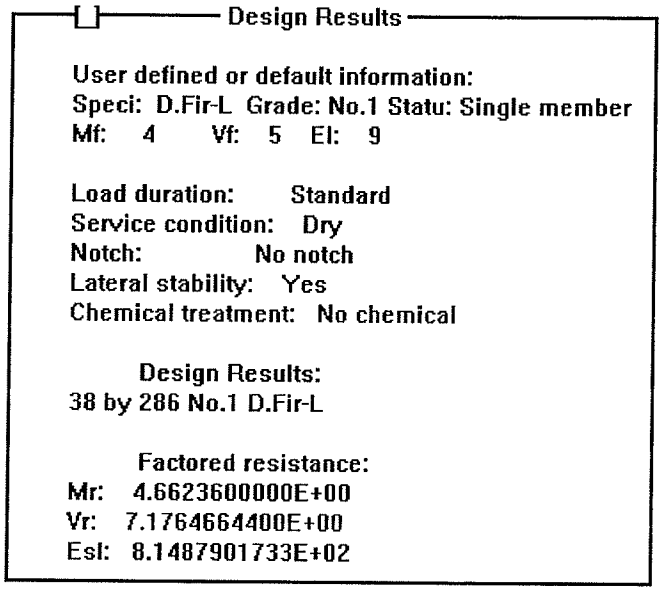


Figure 5.13. Result output window

If no single bending member can meet the factored resistance requirement, the window shown in Figure 5.14 will appear.

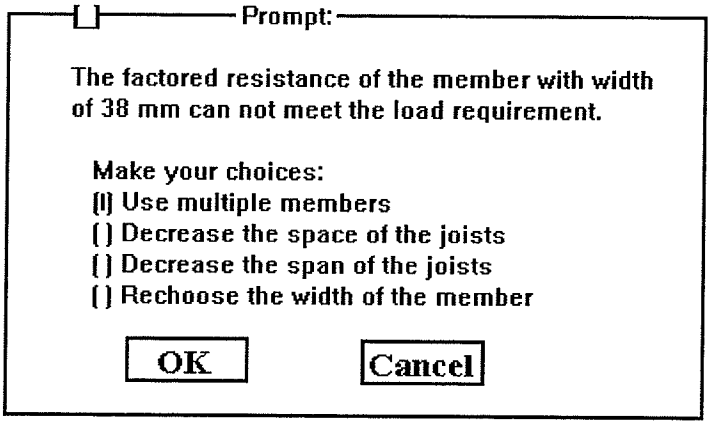


Figure 5.14. Member specification prompt

Based on the user's choice, BEDA will start another round of selection process.

5.3.4 Summary

An interactive member selection unit has been developed. It permits either stand alone or system integrated use. Design procedures are based on CAN/CSA-O86.1-M89 and follow limit states design. BEDA provides both automatic design function and specific design function. The user can control the design process and take advantage of the machine design power.

This function unit has been validated by comparing computer generated design output with that published in the Wood Design Manual (CWC, 1990). All design functions work as they were intended.

6. SPECIAL PROGRAMMING TECHNIQUES

6.1 Object-oriented Information Manipulation

6.1.1 Overview

Engineering design involves polymorphous, multiple-phased information manipulations. Knowledge appears in different forms, including drawings, variables, data, finite elements, equations, and specifications. Sketching, calculation, creation, and revision are among the diverse ways of information manipulations.

A review of the literature reveals that much effort has been devoted to make design processes faster, more efficient, and more error resistant. Applying AI techniques to engineering design bears the hope of achieving these goals. Up to now AI applications in engineering have been limited to narrowly defined expert systems, which address either a single design stage or the design of a single component. One main obstacle in automating design processes is the lack of an efficient and united method for knowledge representation and information manipulation. It is difficult to manage different forms of knowledge and various steps of a design process.

An object-oriented approach of knowledge representation and information manipulation is presented here. It represents a design as an object or a collection of objects. An object works like a courier, in that its datafield carries the information from various knowledge sources and its methods define the way it communicates with its environment.

With objects as information carriers, information manipulation in a design process is streamlined to the manoeuvre of objects. A dual control mechanism was built in the system to regulate and guide the information flow. Knowledge representation and information manipulation in BEDA all feature functional requirements of engineering design.

6.1.2 Object-oriented knowledge representation

To accommodate variations and differences in information manipulation, knowledge is transformed to and from objects at each design stage. Objects form the basis for knowledge representation and information manipulation.

The fundamental object in BEDA is TMember. It represents basic elements in a design. Each structural member is identified as a TMember type object. The whole design subject is represented as a collection of such objects. Features of a structural member are described in the datafield of its Object.

The TMember Object is defined as:

```
TMember = object(TObject)
  EP1, EP2, Memno: integer;
  WdB, WdD, Lngth, Area, Ix, Elm: real;
  Fx, Fv, Fm1, Fm2: real;
  Ldrtn, Species, Grade, Status, Srvcdtn, Chmctrt: PChar;
  constructor Init(a1, a2: Integer);
  procedure Load(var S: TStream);
  procedure Store(var S: TStream); virtual;
end
```

Here, EP1 and EP2 are node numbers for the two endpoints of a structural element. They are of integer type. The real type items, WdB, WdD, Lngth, Area, I_x, and Elm, respectively represent width B, depth D, length, area, area moment, and elastic modulus of the same structural element. F_x, F_v, F_{m1}, and F_{m2} are real type identifiers for axial force, vertical shear, and bending moments.

When an element is dealt with by BEDA for the first time, the system creates a TMember type object for it by calling TMember's 'Init' method, a procedure prefixed with the word 'constructor'. Integer type variables a1 and a2 are replaced by real node numbers of the element. Methods 'Load' and 'Store' define the way in which TMember is retrieved from and saved to a storage medium. Variable S is a TStream type object for information storage.

At the beginning, EP1 and EP2 are specified as a1 and a2. No other parameter is defined. As the design progresses, more and more parameters will be assigned appropriate values. When the design process is complete, all parameters should have been properly assigned.

The basic shape of a structure and the orientations of its structural members are first sketched by human designers using CAD programs. A special subunit is built in BEDA to handle graphic files. A graph can be retrieved and then decomposed into a collection of graphical elements. When a graphical element is identified, its corresponding object is created. A member is identified by its two endpoints EP1 and EP2.

Structural members are connected with each other through endpoints, also called nodes. When a graph is decomposed by the system, all its nodes are identified and represented by the kind of object defined as Tnode.

```
TNode = object(TObject)
  x1, x2: real;
  XCH, YCH, ZCH: boolean;
  Xds, Yds, Zds: real;
  constructor Init(x1, y1: Integer);
  procedure Draw(ADC: HDC; i: Word); virtual;
end
```

In its datafield, X1 and Y1 specify coordinates of the nodes. Attribute XCH, YCH, and ZCH describe constraints to its movement in the X, Y, and Z directions. If a movement is constrained, a TRUE value is assigned. Otherwise it is FALSE. Xds, Yds, and Zds are displacements in X, Y, and Z directions.

All nodes of a graph are numbered and stored in a collection type object defined as Nodalgrp. EP1 and EP2 of a TMember object are simply references of node numbers. Their states are determined by their corresponding TNode objects.

Another important group of objects are defined for identifying external loads. Four kinds of external loads have corresponding objects in BEDA. All external loads are identified as objects and stored in a collection type object called Forcegroup.

The object for nodal force identification is defined as:

```
TNodalforce = object(TObject)
  EP1, Fx, Fy, Mt: PChar;
  constructor Init(ATP, AEP1, AFx, AFy, AMt: PChar);
  procedure Load(var S: TStream);
  procedure Store(var S: TStream); virtual;
end
```

EP1 specifies the node to which the force is applied. F_x, F_y, and M_t represent the horizontal, vertical, and bending force acting on node EP1. Point loads, uniformly distributed loads, and nonuniformly distributed loads are defined with similar objects. Parameters in the datafields are slightly different for each type of load.

Each type of object has some functions or procedures associated with it. Constructor Init is a procedure every object has. It specifies the way in which the object is created. Other procedures or functions specify how objects are manipulated. The Store and Load procedures describe the way in which objects are saved and retrieved. The Draw procedure specifies how an object is presented on a screen.

6.1.3 Object-oriented information manipulation

When a design is identified as objects, the design process is consolidated to object manipulations. Information is retrieved from various knowledge sources, processed by function units, distributed to objects, and stored in the datafields of objects. From this point of view, the object is a information carrier. It transfers information from one design stage to another and connects all function units.

The program for object handling was written in Borland Pascal. It creates, manipulates, and holds all objects. Information from other function units is directed to this unit.

A design forms its embryo at the stage of initial sketching. Users can use an editing window in BEDA to draw simple graphics. But the use of commercial CAD software, such as AutoCad, DesignCad, or AutoSketch, is recommended. This permits sketches to be saved as

DXF files. BEDA has an interface which can connect the system with external software and read in external files.

An object-oriented representation of a design can be created by BEDA from the user's sketch. The RETRIEVE subunit can retrieve drawings from both the user editing window and external DXF files. It identifies all graphical elements, creates TMember type objects, and stores the design as objects in BUILDING.

Information from a sketch only defines the orientation of structural members, i.e. only the parameter EP1 and EP2 are determined for each object. TNode type objects are created to identify all nodes in a design and are stored in NODES, a collection type object.

After the basic shape of a design is sketched, designers need to specify external loads, boundary conditions, and in some cases a temporary choice of section properties, although section properties are among final goals of the design process. The process is accomplished through a number of dialogue windows which can be activated from the user editing window of BEDA. If the sketch is from an external file, the RETRIEVE unit will redraw it on the user editing window. Therefore the user can have a visual perception of the design as the design process is going on.

External loads are identified and stored in a collection type object, ForceGrp. Results from precalculation, snow loads and wind loads, can be retrieved into the user editing window for reference. Boundary conditions are processed and distributed to the datafield of each TNode object.

Section properties are part of final goals of a design process. Following normal engineering design practice, a designer can give them an initial choice, which is subject to further analyses, verifications, and revisions. A dialogue window is developed for specifying section properties, see Fig.6.1. Information is directly distributed to the datafield of each TMember object, where parameter WdB, WdD, Area, I_X, and Elm are accordingly specified.

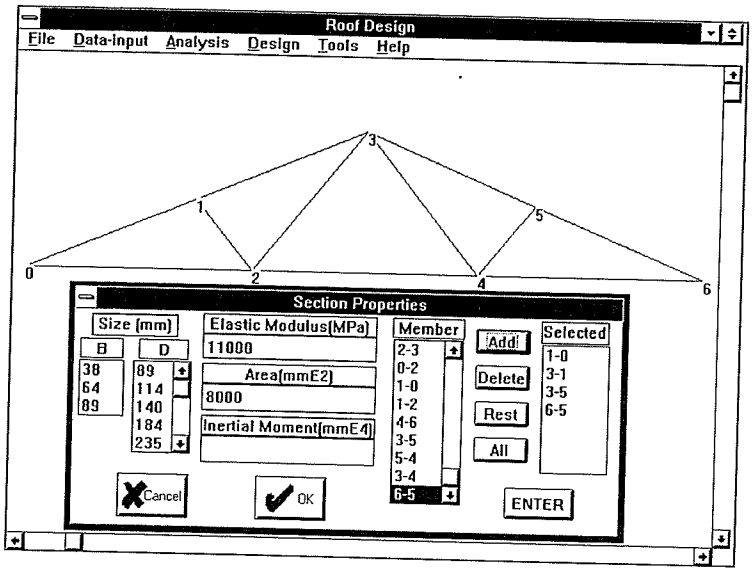


Fig.6.1. The dialogue window for specifying section properties

The initial sketch and designs must be justified by detailed strength and stress analysis. Function unit FEANLS was developed for finite element analysis. Within FEANLS, a design can be identified as a truss, a frame, or a mix-typed structure (see Fig.5.6). Results from FEANLS, including axial forces, shear forces, and bending moments applied to structural members, are directly allocated to the datafield of each TMember object as parameters F_V, F_X, F_{m1}, and F_{m2}.

Dimensions of structural members are determined according to internal forces applied to each member, following Limit States Design (CSA, 1989). Function unit MEMDSN was developed for member design. The user specifies species, grades, load durations, and other items listed in the dialogue window shown in Fig.6.2. The system will automatically determine the dimensions of all structural members. Results are spreaded to the datafield of each TMember object. They overwrite the previous choice of WdB, WdD, Area, and Memno.

Based on the member design results, unit FEANLS can be executed again to recalculate internal forces applied to all structural members. The results can then be sent to unit MEMDSN to calculate new dimensions for the structure. This iterative process continues until proper load-carrying ability is achieved for the whole structure.

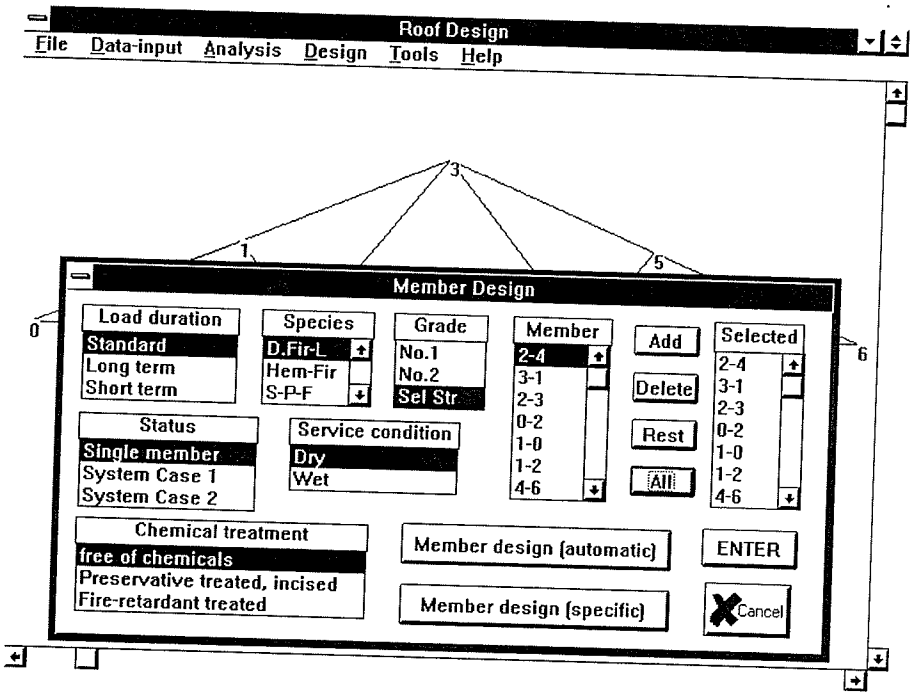


Fig.6.2 The input window for member design

If a compatibility between loads and resistances can not be obtained for the structure, the designer will have to go back to the sketching stage and revise the design. Results of finite element analysis will give experienced designers good clues for design revision.

6.1.4 Conclusion

Object-oriented approach in BEDA unites knowledge representation for various knowledge sources and integrate information manipulations in different design stages.

6.2 On-line Help and Case Base

6.2.1 Overview

On-line help and case resource are two standard functions within the BEDA system. Their operations follow the convention of Microsoft Windows on-line help. Actually, they are both developed with the HelpAuther, a tool provided by Microsoft for developing on-line help files. Both work in exactly the same way as standard MS Window help files. Both can be activated either from pull-down menus or through control buttons on the user interface.

The on-line help function displays explanations about the terminology that appears on the interface. It also prompts the user about how to use the BEDA system. The case resource gives the designer design cases which are related to the current design. Each case is accompanied by comments and suggestions which point out its advantages and disadvantages. Comments and suggestions belong to two groups. One group consists of comments carefully made by field experts and system developers. The other group is composed of personal comments made by end users for their own reference in future designs. The first group is permanently built in the system. The second group is temporary and changeable.

On-line help contents are stored in a binary file called Help file. Contents of the case resource are stored in Case base, a special knowledge resource of BEDA.

Microsoft Windows Help is used as a display engine. Information displayed includes text, graphics, and other multimedia elements. Advanced hypertext capabilities and context-sensitive links are available so that the end user can easily navigate through the help and case contents.

6.2.2 Basic functions

The on-line help and the case resource of BEDA are activated when the user clicks the 'help' or 'case' item on pull-down menus or corresponding buttons on the user interface. When either of the two functions is activated, a window appears to display the Help or case base contents (see Fig.6.3).

The mouse or the keyboard can be used for common actions under a help window: choosing commands, moving or resizing the display window, clicking buttons, interacting with dialog boxes, scrolling information, and choosing hot spots. Four standard buttons, Contents, Search, Back, and History, are provided as navigation controls. They appear in the button bar, helping users to access information in the Help file and the case base.

Clicking the Contents button will display the table of contents for the Help file or the list of cases for the case base. It serves as the home screen, from which all other information is accessed. The Search button shows an on-line index and allows the user to search for information

by key word, similar to searching through a book index. The Back button displays the topic previously displayed. The History button presents a list of topics (a maximum of 40) that have been viewed during the current session. Any topic in the list can be selected to view again. Clicking a name in the case list will bring in the graphic of that design case.

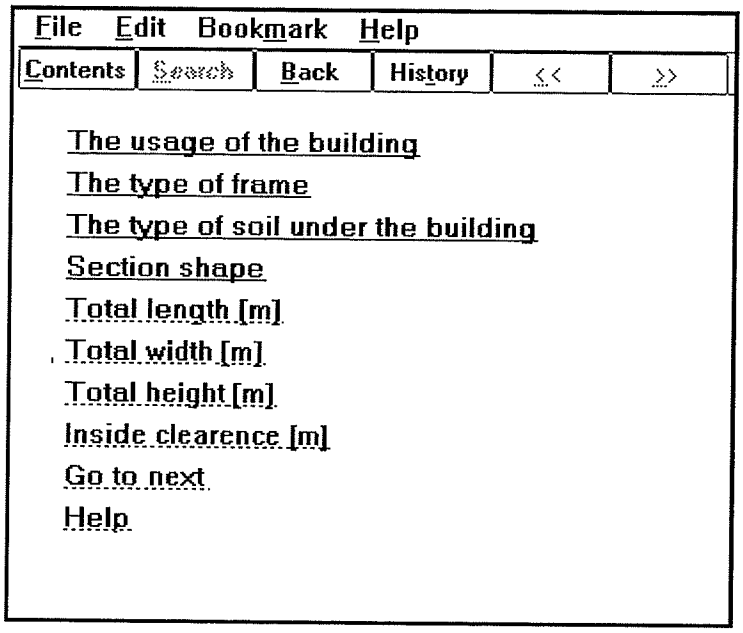


Fig.6.3 An example of the help window

The Help file and the Case base are made up of distinct units of information called topics. Each topic focuses on a specific piece of information such as a concept, a step-by-step procedure, a command summary, an illustrative graphic, a keyboard table, an example, or any other unit of information.

Topics are linked by hypertext and hyperregion jumps (called hot spots). The mouse pointer changes from an arrow to a hand when it is over a hot spot. Clicking a hot spot will clear the current topic and displays the topic indicated by the jump. After jumping to a topic, the user can return to the original one by clicking the Back button.

Within topics, certain terms or concepts often require further explanations. Instead of jumping away from the current topic, a pop-up window can be displayed on top of the current

topic. Pop-up windows are stripped-down windows with only a window border. They can display text and graphics, and can include hot spots. A pop-up window remains displayed until the mouse is clicked or any key is pressed.

Items shown in Fig.6.3 are all hot pots. Clicking the item *The type of soil under the building* will bring in an explanation window (see Fig.6.4). Choosing the item *Total length [m]* will bring a pop-up window, as shown in Fig.6.5.

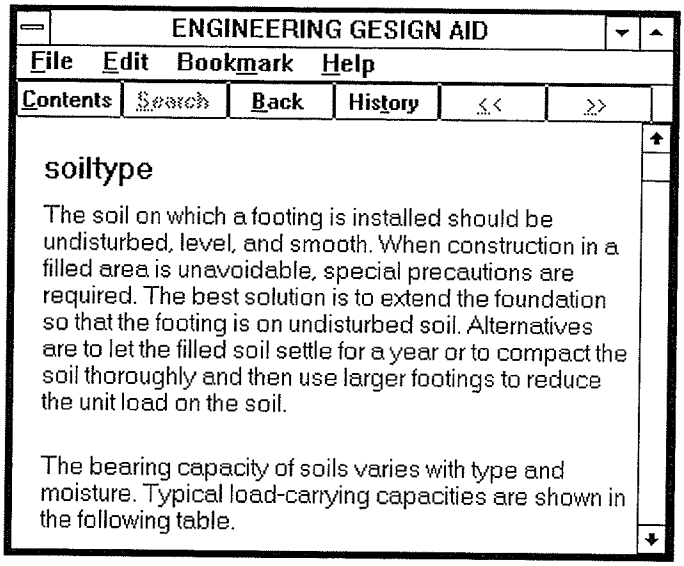


Fig.6.4 The explanation window for soil type

There are two special buttons with the case resource. The REMARK button brings in a dialog, through which comments can be added to the case base. Comments are displayed when the hot text COMMENTS, which appears with each corresponding case, is clicked. The other button is labelled as REDRAW. It activates an external graphic software to retrieve and edit the highlighted case. An edited case can be save as the user's design. When a graphic is displayed on the case window, it is in bitmap form. When brought into a graphic software, it will be in a form suitable to that software. However, if the end user wants to save the edited graphic as a user's design, it must be saved in DXF format for further analysis by BEDA.

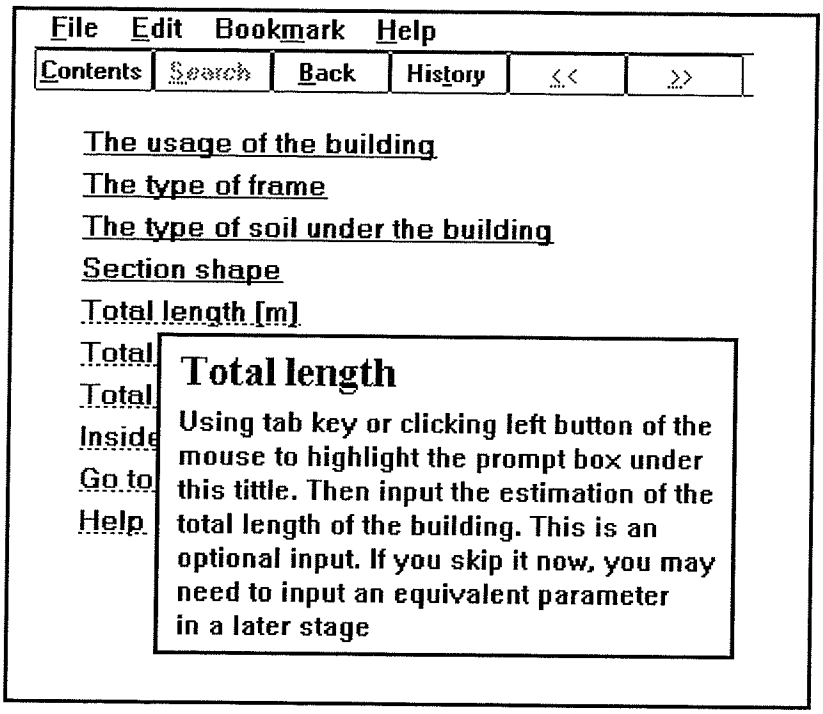


Fig.6.5 The pop-up window for total length

6.2.3 Help file and case base development

The Help file and the case base are built with the Microsoft HelpAuthor, an enhancement utility that makes it easier to create Help files for Windows version 3.0 and 3.1. It is composed of two parts: the Help Authoring Templates and the Help Project Editor. The development process involves several procedures: creation of topic files, creation of the Help project file, and compilation of topic files into a binary *.HLP* file.

A topic file contains the content and construction information for the Help file and the case base, including topic text, graphics, and special Help codes that pass organizational and build information to the Windows Help compiler. Within topic text, such features as cross-reference links, pop-up window hot spots, macro commands, references to graphics files, and other codes particular to the Help file and the case base are created.

Topic files are created in Microsoft Word and saved as RTF files. The Help Authoring Templates modify Word so that it is easier to create, edit, and format topic text, graphics, and/or

hot spots. After topic files are created, the Help Project Editor is used to create a Help project file, add topic files to the project, and edit project file information. The Help project file tells the compiler how to build the Help file.

Ultimately, the Help file and the case base are built from the topic files and the Help project file, using Windows Help compiler (HC31.EXE). The Help compiler displays error messages when it encounters problems in the creation. Programmers can use these messages to debug the Help file.

A compiled Help file is displayed using the Windows Help application (WINHELP.EXE). This application interprets and displays text, pictures, hypertext links, and keyword search index. It allows the user to explore the presented information interactively. In a completed file, all features designed into the Help file and the case base come into use. For example, users can browse topics as they would in a printed book or use hypertext links to jump around.

6.2.4 Connection with the system

Access to the Windows Help application was built in BEDA. In the part of BEDA developed in Level5 Object, the on-line help and the case base can be activated with the 'Help' and 'Case' buttons or the menu on BEDA's interface. The two buttons are respectively attached with attributes Helpbutton and Casebutton in the knowledge base of BEDA. If either button is clicked, its attached attribute will change status, which in turn will trigger a production rule to execute the Windows Help and display the on-line help or case resource.

Take the help function for the initial input as an example. When the 'Help' button is clicked (see Fig.6.1), the following production rule is triggered:

```
RULE
IF Helpbutton OF Initial Input
THEN ACTIVATE  "IPU,EXTERN,WINHELP.EXE"
 COMMAND  "C:\WINHELP\INITIAL.HLP"
```

ACTIVATE and COMMAND are two commands within Level5 Object. The function of ACTIVATE is to execute an external file through IPU, an internal protocol for communication

with MS-DOS and MS-Windows programs. WINHELP.EXE is the executable file for the Windows Help. COMMAND functions to specify the operand of the executable file. In this case, it is INITIAL.HLP stored in C:\WINHELP directory. INITIAL.HLP is the on-line help topic for the initial input.

In the part of BEDA developed in Borland Pascal, the Windows Help is activated with the WinHelp function. When either 'Help' or "Case' is chosen, a special procedure will be triggered to call the WinHelp function. The selected topic from the on-line help or the case resource is then displayed.

6.2.5 Conclusion

The Help file and the case base provide on-line information about BEDA and the design process supported by BEDA. They help the user go through design processes. Taking existing designs, either successful or failed cases, as template or reference will save engineers time and labour in detailed engineering design.

6.3 Interface

6.3.1 Overview

A user-machine interface is a bridge connecting the system and the user. Through it, users control and monitor the working progress of the system, and the system gets input and displays results. The interface of BEDA was built with ObjectWindows supplied with Borland Pascal and the Display Editor of Level5 Object.

Both ObjectWindows and the Display Editor are object-oriented development tools for Microsoft Windows. The interface is in Windows style. It is composed of a series of user friendly displays and dialogs. Its composition is shown as the dashed-lined area in Fig.6.6.

The mainline of a design process is controlled by the main frame windows in the blackboard interface. When a functional unit is in its active mode, the corresponding unit interface will be the active window. A user can tour through main windows and unit windows by clicking

control buttons or pull-down menus. Through the user-machine interface, users participate in the data input-output process, monitor the information evolution, and control the design process.

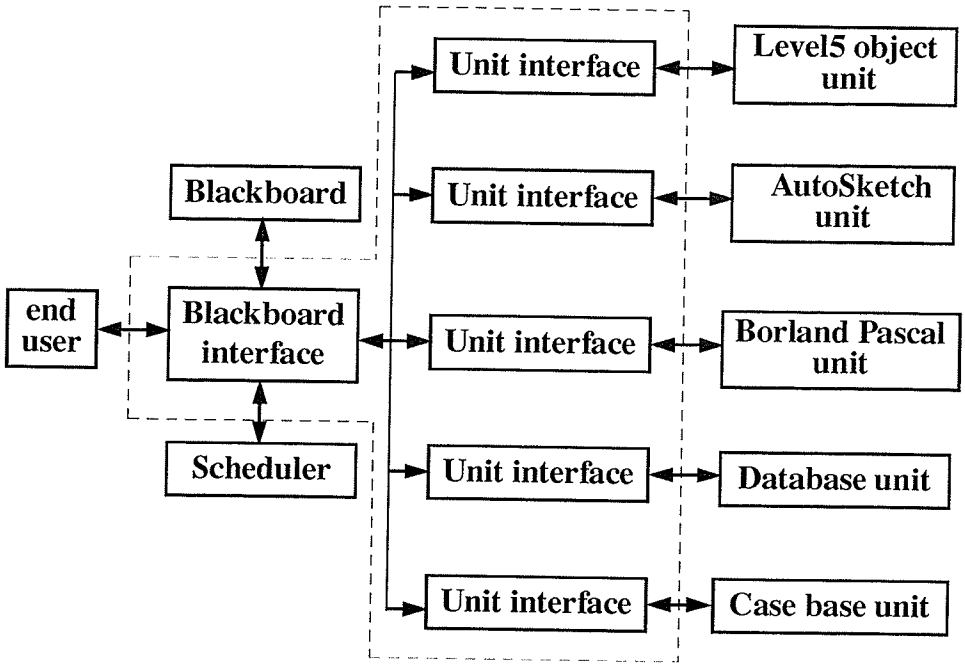


Fig.6.6 The composition of the user-machine interface

6.3.2 Interface elements

BEDA's interface is composed of window displays developed for each function unit. The main window is shown in Fig.6.7. For graphic sketching, the window is as shown in Fig.6.8. The rest of the function units share the window shown in Fig.6.9. When the system or a function unit is started, the corresponding window appears. It prompts for data input, result output, and design process control.

All windows were built following the convention of Microsoft Windows. A window is a framed, rectangular area on a computer screen, where communications between users and the machine take place. Each window contains menus for triggering system actions, controls for resizing the window, a title, and a working area for user editing or for child window displaying. Windows can overlap each other. The selected, front-most window is the active one.

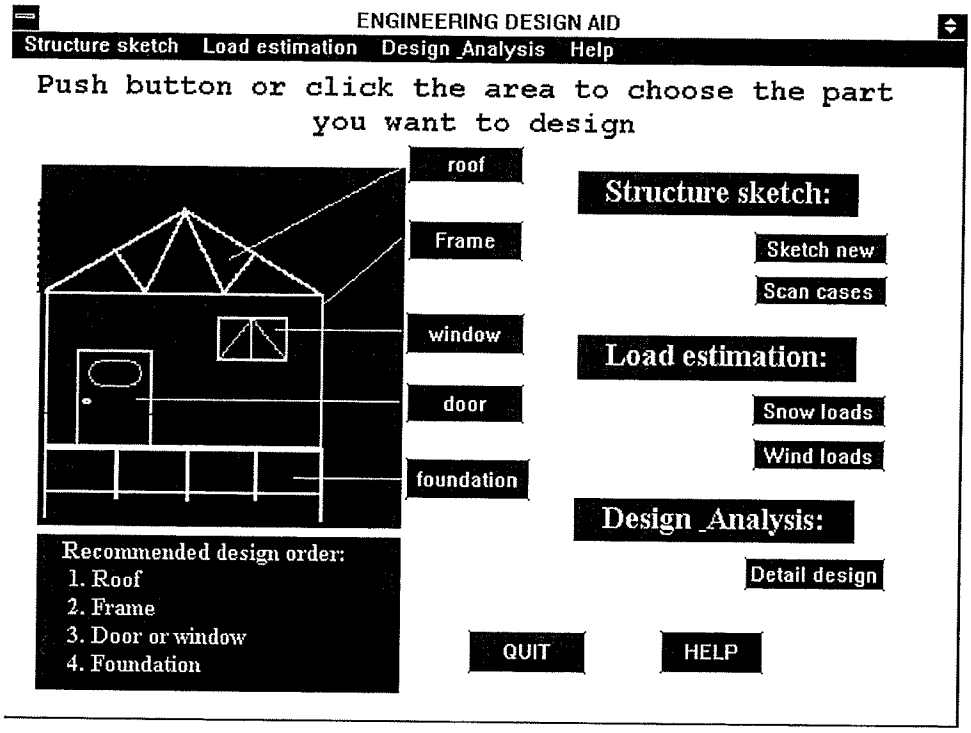


Fig. 6.7 The main window for system function selection

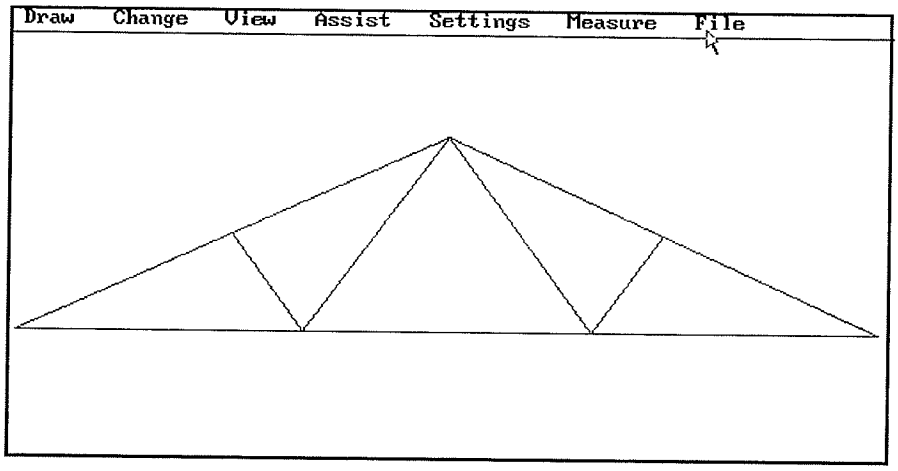


Fig.6.8 The main window for graphic sketching

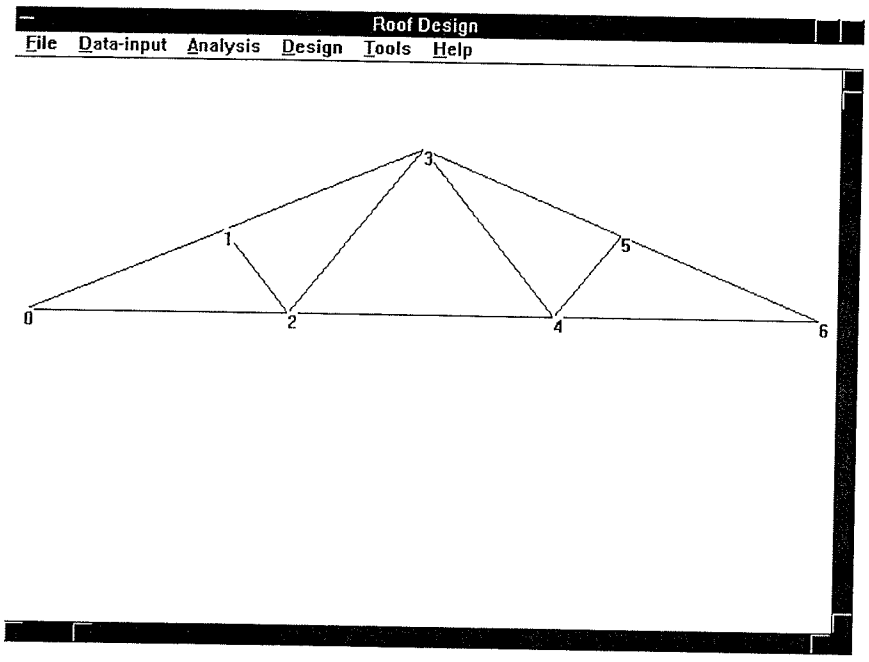


Fig.6.9 The main window for structure analysis and member design

Input or output information of a function unit is cast into a few child windows, called displays in Level5 Object and dialogs in ObjectWindows (see Fig.5.1 and Fig.5.11). A dialog or a display is an interactive window. Its working area contains a group of display items or control objects for user-machine communications. These items and objects are similar in Level5 Object and in Borland Pascal. Each is a small box on the screen. They can be classified into a few groups according to their functions: the group for data presentation, such as value boxes and list boxes; the group for graphical illustrations, such as picture boxes and bitmap boxes, and the group for process control, such as pushbutton and check boxes.

A basic function of dialogs and displays is for data input and output. The item for data input is called Edit Box in ObjectWindows and Promptbox in Level5 Object. The Static Box in ObjectWindows and the Valuebox in Level5 Object are used for displaying values. List Box and Radiobutton Group are for listing multiple choices.

Pushbuttons are display items used for working process control. They work in the same way as pull-down menus. Clicking a pushbutton will trigger a new function or activate a new dialog or display.

Descriptions and graphics are used in a display to prompt for data input, explain the meaning of displayed data, or label the action a display item may activate. Text is presented by Static Boxes in ObjectWindows. Level5 Object provides Textbox for the same function. Items for graphical illustrations are called Picturebox in Level5 Object and Bitmap Box in ObjectWindows. They can present bitmap graphics.

All display items can be accessed by the mouse, the Tab key, or Hot keys.

6.3.3 Interface construction

In both Borland Pascal and Level5 Object, the visible portion of an interface is separated from its program codes. The visible portion defines the visual appearance. Its program codes specify ways of data transfer between the interface and the system as well as the way in which the system reacts to user input.

There are two steps in the construction of an interface: the design of the visible portion and the writing of programs to handle it. The visible portion of an interface is created and edited with Resource Workshop of ObjectWindows in Borland Pascal. The Display Editor of Level5 Object has similar functions. They work like a drawing program in that a programmer can create, move, delete, size, color, and duplicate items for a display. All display items, such as buttons, text boxes, and scroll bars, are provided as bare-bone templates. Programmers can customize them by filling them with proper text or graphics and specifying their size, color, position and other graphical features. To design a display is the combination of deciding display items, planing the layout, and customizing display items.

In Level5 Object, entities of domain knowledge are identified and represented as attributes of class structure. A display item can be attached to an attribute. Once attached, the value of an

attribute can be prompted or displayed by its display item. The display is dynamically updated as the value of that attribute changes.

In Borland Pascal, a display is identified and represented as an object of dialog type. Display items are identified as control items in the datafield of the object. Ways of data transfer and manipulation are specified by methods of the object, see section 5.2.4 for the definition of dialog type objects.

Program code of an interface is the symbol representation of its visible portion. Through the program code, user's manipulations at the visible portion are transferred to the system, and system reactions are transferred to the visible portion. An interactive interface is composed of a visible portion and the corresponding program code.

6.3.4 Working principles of the interface

BEDA works in an event-driven manner. All its function units and corresponding displays are activated by events. An event will produce a message in the system. Messages are spread through the hierarchy of objects. Methods for message response were built within objects to handle messages in specified ways.

User interactions, such as key strokes or mouse clicks, are all treated as events. The termination of a piece of program or a function of BEDA is also an event. Messages from user interactions are generated by the Windows. Messages from the terminations of function units are created in ways specified by the programmer.

Each display item has a unique number, called handle, for identifying the item. Each control item has a unique control ID that is identical to the handle of its corresponding display item. By matching the handle and the control ID, a message from a display item is directed to a control item and manipulated by the corresponding procedure.

Ways of producing messages and responding to them are defined by the programmer. Windows and ObjectWindows will spread messages around. The mechanism of message

manoeuvring is hidden in the system. A message without a responding method is neglected by the system.

Messages are commands, which tell the system and objects that something happened somewhere. Apart from being a control panel, the interface of BEDA is also a channel of communication between the system and users. Each dialog has a data transfer buffer built in the system. When a dialogue is displayed, it extracts data from its transfer buffer to fill its display items. When users put new values to display items, the values are stored in the transfer buffer. The application can then take data from the buffer. Results of system execution are first written to the data transfer buffer and then displayed in a dialog.

Dialogues or displays are started and closed by events. Event responding methods are written to ensure that the right dialog is displayed and destroyed in the right time. The appearances and activations of different displays are given in Appendix A.

6.3.5 Conclusion

The user interface of BEDA is developed with object-oriented programming in Borland Pascal and Level5 Object. It is composed of a series of displays complying with the convention of Microsoft Windows. Each display has a visible portion and some program codes. Its visible part is designed to be user friendly. The corresponding program codes ensure proper interactions between users and the BEDA system.

7. CONCLUSIONS

The research reported herein led to the following conclusions:

1. The optimization model embodies basic features of engineering design and makes it applicable to automate the engineering design process.

2. Optimizing rules can be developed to address the design criteria of safety and functionality.

3. Object-oriented knowledge representation and information manipulations are appropriate for a knowledge based engineering design system.

4. Engineering drawings, with objects as information carriers, can be used directly in finite element analysis and the results of analysis can be used directly for automatic design.

5. The blackboard model of problem solving provides a functional system framework for the organization of diversified knowledge resources and facilitates the cooperations among various subsystems. Under the blackboard architecture, the human designer becomes an integrated part of the design system.

6. A knowledge-based system, BEDA (Basic Engineering Design Aide), has been developed as an engineering design aid, using an integrated approach of AI programming and conventional programming. It demonstrates a general design approach which was developed on the basis of light-frame building design.

7. BEDA is composed of several function units, each of which implements a special function of engineering design. The module structure of BEDA allow a single unit to be revised, to reflect the change in design theory and practice, without affecting other units.

8. Information flow pattern within a design system can be enforced by dual control utilizing the human designer and a machine scheduler.

9. A case based on-line help system can be integrated into a knowledge based design system.

8. SUGGESTIONS FOR FUTURE RESEARCH

Using artificial intelligence to automate engineering design is a promising but incomplete work. The research work presented herein is a prototype exploration of basic principles and feasible approaches for building intelligent engineering design aids.

The process of engineering design is still poorly defined. Better understanding of engineering design and further development of AI theory require deeper study by experts from many fields. Long term, interdisciplinary research is suggested into:

1. the philosophy and morphology of engineering design, and
2. the improved modelling of the engineering design process.

During this work, many interesting questions arose. Time did not permit the luxury of pursuing each question. Further work specific to BEDA is suggested in the following areas:

1. expansion of the rule base and the control mechanism to include design codes other than wood design, such as steel, concrete, and plastics,
2. expansion of the database to hold more information concerning engineering design, such as the physical features of steel and concrete,
3. development of more universal graphic processing function to accommodate graphics in formats other than *DXF*, which should be able to deal with graphic files coming directly from common CAD software packages such as AutoCAD and DesignCAD,
4. upgrade of the Finite Element Analysis (FEA) function. Since many sophisticated FEA packages are commercially available, the research can be focused on the development of interfaces which support the connection of BEDA with other software,
5. refinement of the member design unit, especially the specific design function. Now the specific design part can design structural members as beams, columns, and decking. Design procedure can be expanded to include the design of mechanical and electronic parts,

6. the on-line help and case base only has a basic shape. It can be enriched with more instructions and cases. Case management needs to be improved to facilitate the case aviation, case reasoning, and automatic case collection, and

7. the user interface needs to be refined to achieve better visual effect.

REFERENCE:

Adeli, H. and Al-Rijleh, M.M., 1987. A Knowledge-based expert system for design of roof trusses. *Microcomputers in Civil Engineering*, 2, pp.179-195.

Arciszewcki, Tomasz and Ziarko, Wojciech, 1991. Structural optimization: case-based approach, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.2, pp. 159-174.

Autodesk Inc.. 1988. AutoCAD: Reference manual, Sausalito, CA. USA.

Bailey, Simon F. and Smith, Ian F.C., 1994. Case-based preliminary building design, *Journal of Computing in Civil Engineering*, ASCE, Vol.8, No.4, pp. 454-468.

Barrett, J.R. and Jones, D.D., eds. 1989. Knowledge engineering in agriculture, American Society of Agricultural Engineers, St. Joseph, Michigan, USA.

Batchelor, W.P. and McClendon, R.W., 1992. A Blackboard approach for resolving conflicting irrigation and insecticide scheduling recommendations, *Transactions of the ASAE*, General edition, ASAE, Vol.35, No.2, pp. 741-747.

Beck, H.W., Jones, P.H. and Watson, D.G., 1994. A CD-ROM based agricultural information retrieval system, *Applied Engineering in Agriculture*, ASAE, Vol.10, No.1, pp. 127-132.

Borland International Inc., 1992. ObjectWindows 7.0: programming guide, Scotts Valley, CA, USA.

Borland International Inc.. 1990. Turbo Pascal version 6.0: programmer's guide, Scotts Valley, CA, USA.

Britton, M.G., Hou, Y. and Zhang, Q., 1992. Using artificial intelligence as a foundation for computer based design-aids, Presented at the Agricultural Institute of Canada Annual Conference, Brandon, Manitoba.

Cherneff, Jonathan, Logcher, Robert and Sriram, D., 1991. Integrating CAD with construction-schedule generation, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.1, pp. 64-84.

CSA 1989. Engineering design in wood (Limit states design). CAN/CSA-086.1-M89. Canadian Standards Association, Rexdale, Ont.

CWC 1990. Wood design manual. Canadian Wood Council, Ottawa, Ont.

Deer-Ascough, L., Jones, D.D., Barrett, J.R. and Okos, M.R., 1992. Soybean oil extraction diagnostic expert system, *Applied Engineering in Agriculture*, ASAE, Vol.8, No.4, pp. 545-552.

Elkordy, M.F., Chang, K.C. and Lee, G.C., 1993. Neural networks trained by analytically simulated damage states, *Journal of Computing in Civil Engineering*, ASCE, Vol.7, No.2, pp. 130-145.

Embleton, K.M., 1990. An Expert system approach to establish design snow and wind loads. Dissertation. The University of Manitoba.

Evans, M. 1988. A Knowledge-based model of distrubuted problem solving. Dissertation. The University of Manitoba.

Evans, M., Mondor, R. and Flaton, D., 1990. Using expert systems to generate fertilizer recommendations, *AI Applications*, Vol.4(2), pp. 3-10.

Florman, Samuel C., 1987. The civilized engineer, St. Martin's Press, New York, USA.

Flood, Ian and Kartam, Nabil, 1994. Neural networks in civil engineering. I: principles and understanding, *Journal of Computing in Civil Engineering*, ASCE, Vol.8, No.2, pp. 131-148.

Garrett, James H. Jr. and Maher, Mary Lou, 1991. Special section: knowledge-based systems in design and planning, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.1, pp. 1-3.

Gauthier, L., 1992. A Smalltalk-based platform for greenhouse environment control, *Transactions of the ASAE*, General edition, ASAE, Vol.35, No.6, pp. 2003-2020.

Ghosh, D.K. and Kalyanaraman, V., 1993. KBES for design of steel structural elements, *Journal of Computing in Civil Engineering*, ASCE, Vol.7, No.1, pp. 23-35.

Gupta, C.P. and Suryanto, H., 1993. A Knowledge-based system for insecticide management for rice crops, *Transactions of the ASAE*, General edition, ASAE, Vol.36, No.2, pp. 585-592.

Heinemann, P.H., Calvin, D.D., Ayers, J., Carson, J.M., Curran, W.S., Eby, V., Hartzler, R.L., Kelley, J.G.W., McClure, J., Roth, G. and Tollefson, J., 1992. Maize: a decision support system for management of field corn, *Applied Engineering in Agriculture*, ASAE, Vol.8, No.3, pp. 407-414.

Hou, Y., Britton, M.G. and Zhang, Q., 1994. An expert system based engineering design aide, Proceedings of 1994 CSCE Annual Conference, pp. 303-308, Winnipeg, MB, Canada.

INFORMATION BUILDERS., 1993. Level5 Object for Microsoft Windows: reference guide release 3.0, INFORMATION BUILDERS, Inc., New York, USA.

Koen, Billy V., 1985. Definition of the engineering method, American Society for Engineering Education, Washington, D.C., USA.

Koumousis, Vlasis K. and Georgiou, Panos G., 1994. Genetic algorithms in discrete optimization of steel truss roofs, *Journal of Computing in Civil Engineering*, ASCE, Vol.8, No.3, pp. 309-325.

Laszlo, Ervin, 1969. System, structure, and experience: toward a scientific theory of mind, Gorden and Breach, New York, U.S.A.

McCarthy, T.J. and Nouas, Z., 1989. An Experimental expert system for the design of column base plates. In: Topping, B.H.V., ed. *Artificial Intelligence Techniques and Applications for Civil and Structural Engineers*. Edinburgh, Civil-Comp Press.

Miles, J.C. and Moore, C.J., 1989. An Expert system for the conceptual design of bridges. In: Topping, B.H.V., ed. *Artificial Intelligence Techniques and Applications for Civil and Structural Engineers*. Edinburgh, Civil-Comp Press.

Morse, David V. and Hendrickson, Chris, 1991. Model for communication in automated interactive engineering design, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.1, pp. 4-24.

Olynick, D.M., 1993. An Expert system for the fire protection requirements of the national building code of Canada 1990. Dissertation. The University of Manitoba.

Petroski, Henry, 1992. To engineer is human: the role of failure in successful design, Vintage Books, New York, USA.

Phillips, Richard E., 1981. Farm buildings: from planning to completion, Doane-Western, St Louis, Missouri, USA.

PRECARN Associates.,1989. Knowledge aided design: a feasibility study on the application of artificial intelligence in engineering, submitted by Spar Aerospace limited, et al.. Nepean, Ontario, PRECARN Associated Incorporated.

Radford, Antony D. and Gero, John S., 1988. Design by optimization in architecture, building, and construction, Van Nostrand Reinhold Company, New York, USA.

Randhawa, S.U., Scott, T.M. and Olsen, E.D., 1992. Timber harvester: a microcomputer-based system for automatic selection of timber harvesting equipment, *Applied Engineering in Agriculture*, ASAE, Vol.8, No.1, pp. 121-127.

Rowlinson, S., 1989. Knowledge based systems: potential in design and management. In: Topping, B.H.V., ed. *Artificial Intelligence Techniques and Applications for Civil and Structural Engineers*. Edinburgh, Civil-Comp Press.

Schaper, L.A. and Lund, S., 1992. SUBERMAX: an expert system on the suberization phase of potato storage, *Applied Engineering in Agriculture*, ASAE, Vol.8, No.3, pp. 401-406.

Segerlind, Larry J., 1984. Applied finite element analysis, 2nd ed., John Wiley and Sons, New York, etc., USA.

Thompson, D.R., 1989. An Expert system for preliminary design of timber roofs. in: Topping, B.H.V., ed. *Artificial Intelligence Techniques and Applications for Civil and Structural Engineers*. Edinburgh, Civil-Comp Press.

Thornton, Charles H. and Velivasakis, Emmanuel E., 1991. State of automation in structural engineering- progress made and issues still pending, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.2, pp. 129-131.

Tyson, Thomas R., 1991. Effective automation for structural design, *Journal of Computing in Civil Engineering*, ASCE, Vol.5, No.2, pp. 132-140.

VanDevender, K.W., Costello, T.A., Ferguson, J.A., Huey, B.A., Slaton, N.A., Smith, R.J. Jr. and Helms, R.S., 1994. Weed management support system for rice producers, *Applied Engineering in Agriculture*, ASAE, Vol.10, No.4, pp. 573-578.

Walker, Terri C. and Miller, Richard K, 1990. Expert system handbook: an assessment of technology and applications, Fairmont Press, Inc., Lilburn, GA, U.S.A.

Wang, J. and Howard, H.C., 1988. Design-dependent knowledge for structural engineering design. In: Gero, J.S., ed. Artificial Intelligence in Engineering Design, Southampton, Computational Mechanics Publications.

Appendix A

Description of BEDA software

The knowledge based design aid, BEDA, is composed of several functional units. It starts from the precalculation part, which is written in Level5 Object and saved as a *KNB* file under the file name DSGNAID.KNB. A KNB file can be transferred into an executable file with a special editing tool commercially available from Information Builders Inc. Currently, DSGNAID.KNB can be executed from within the Level5 Object editing environment.

When users run DSGNAID.KNB from Level5 Object, the following display (Fig.a1) will appear on the screen.

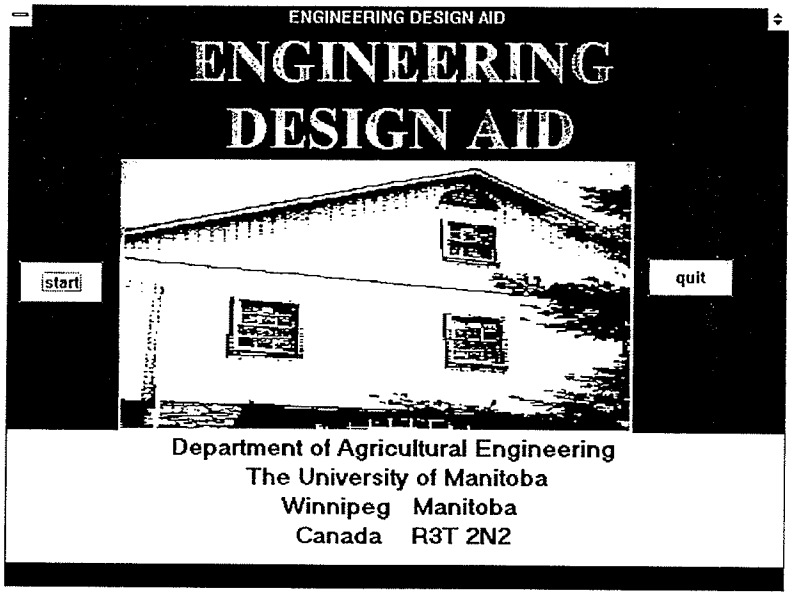


Fig.a1. The greeting screen of BEDA

Displays of BEDA follow the standard convention of Microsoft Windows, i.e. normal operations for MS Windows can all be applied to BEDA displays. Clicking the *Start* button on the above display will bring in the display shown in Fig.a2.

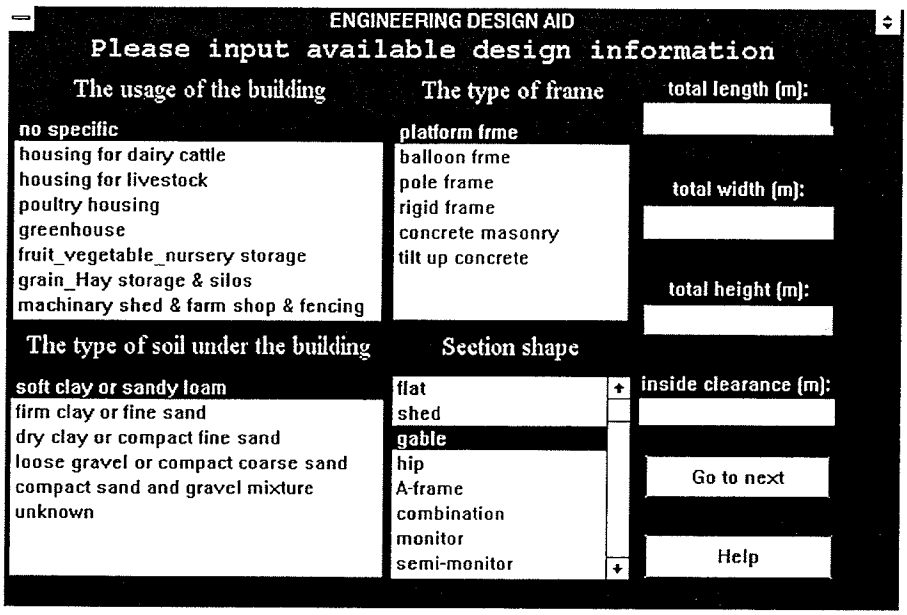


Fig.a2. The input window of general information

This screen is for the user to input general information about the building under design. The user can specify the usage, frame platform, soil type under, and section shape by clicking an appropriate item in the corresponding listbox. The total length, width, height, and inside clearance can be designated by putting right numbers into value boxes. Button *Go next* will activate the display shown in Fig.a3.

The screen shown in Fig.a3 is for users to specify the location of the building, which will be used by the system to trace data in database for calculating snow and wind loads. All together 644 towns in the 12 provinces and territories of Canada are given for the user to choose from. On the same screen, a summary of all user input is displayed for recheck. The user can bring back previous display by clicking *Reinput the information* or go to the display shown in Fig.a4 by clicking *Go to detail design.*

The screen shown in Fig.a4 is the main control window of BEDA system. From within this window, the user can start various function units of BEDA by clicking buttons on the screen.

These buttons are programmed to trigger corresponding software of those functions. The functions are classified as structure sketching, load estimation, and design analysis.

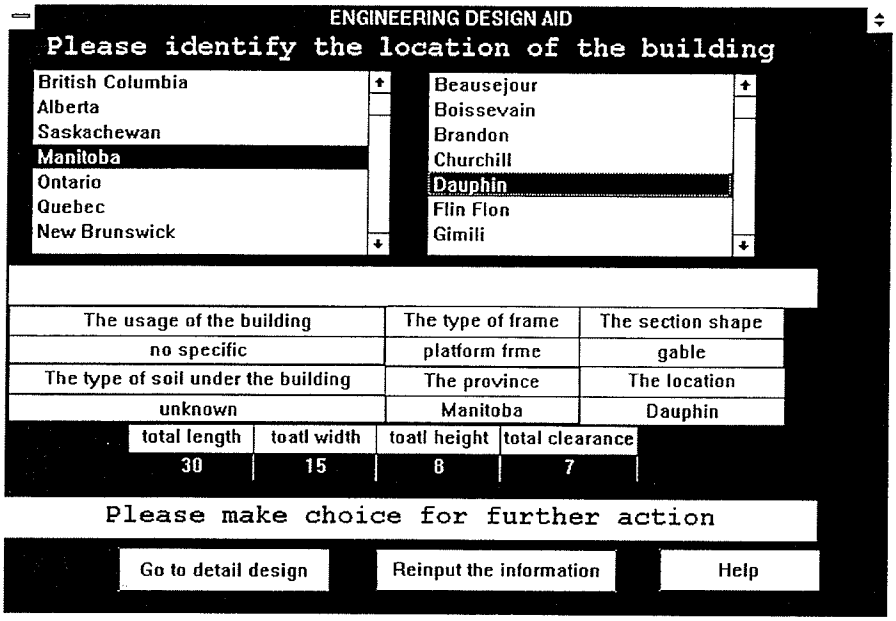


Fig.a3. The input screen for locations

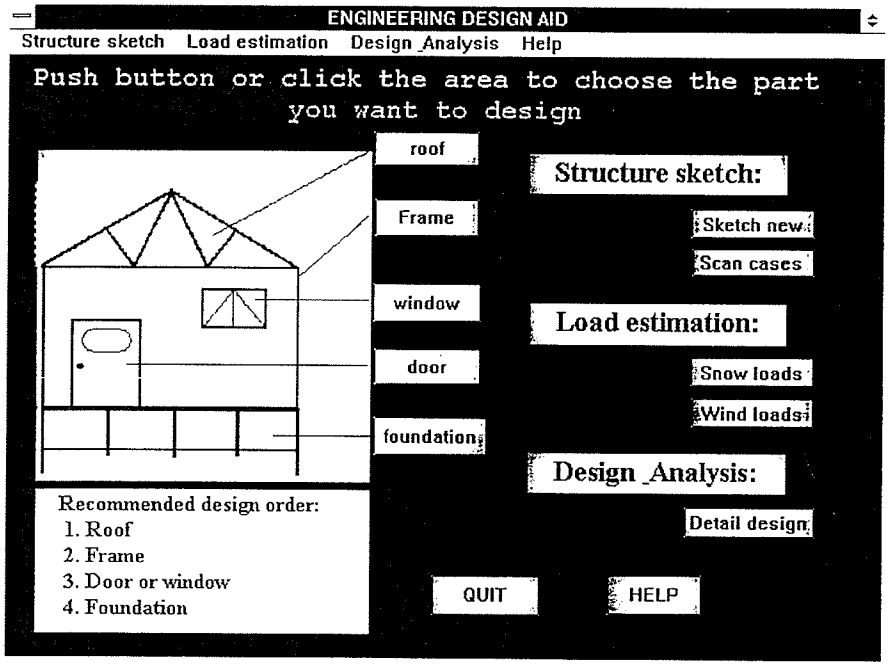


Fig.a4. The main control window of BEDA

Under the label *Structure Sketch*, button *sketch new* will start a graph editing tool while *scan cases* is designed to open existing design templates in the case base. Currently, button *sketch new* is attached to the commercial software AutoSketch. More work needs to be done to make *scan cases* fully functional. The standard user editing window for AutoSketch, shown in Fig.a5, will appear when *sketch new* is clicked. The user can create, edit, and save graphics by using all functions provided by AutoSketch.

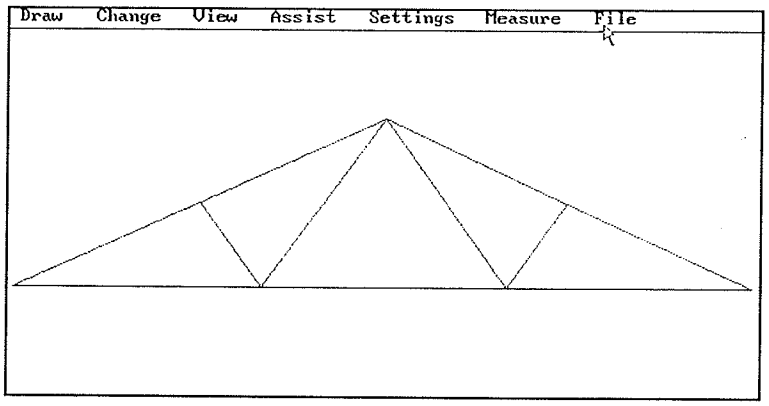


Fig.a5. The user editting window for AutoSketch

The load estimation part of BEDA is written in Level5 Object and contained within DSGNAID.KNB. A display, shown in Fig.a6, will appear when button *snow loads* is clicked.

The display queries about environmental states and load situation about the building. There is an input redundance between the load situation and the section shape queried in a previous display. The redundance is purposely designed to remind the user of his/her input history. If there is a conflict, BEDA will prompt the user with the screen shown in Fig.a7.

For different load situations, BEDA may further query some trivial details, such as the screen shown in Fig.a8 for *simple arch or curved roof* and the one in Fig.a9 for *lower level of adjacent roof*. If all queries are properly answered, BEDA will calculate the snow loads and display the results in the output window shown in Fig.a10.

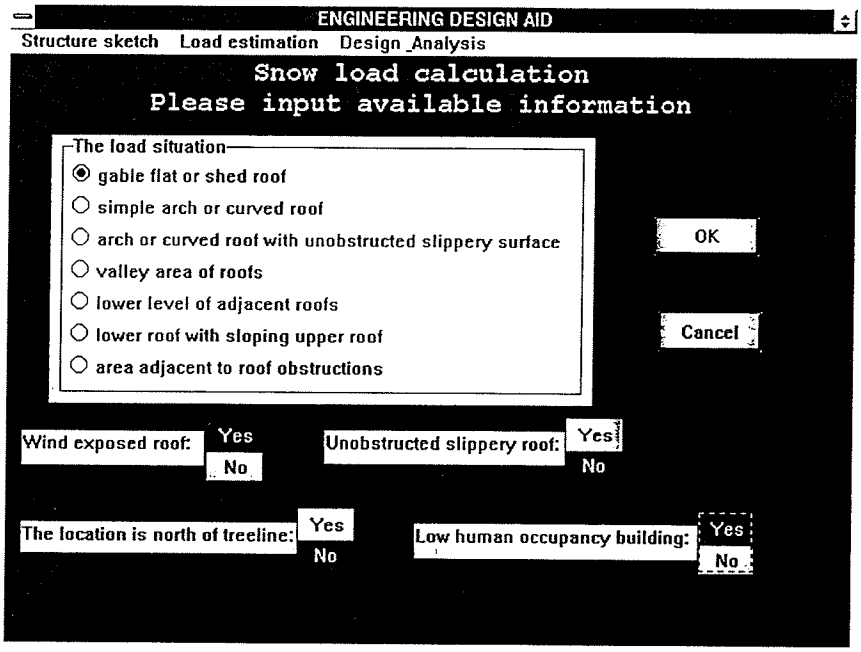


Fig.a6. The input window for snow load estimation

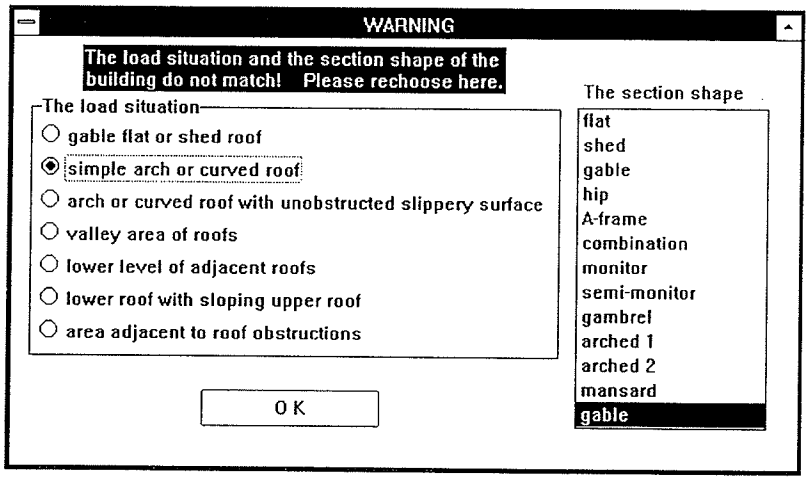


Fig.a7. The prompt screen for input conflict

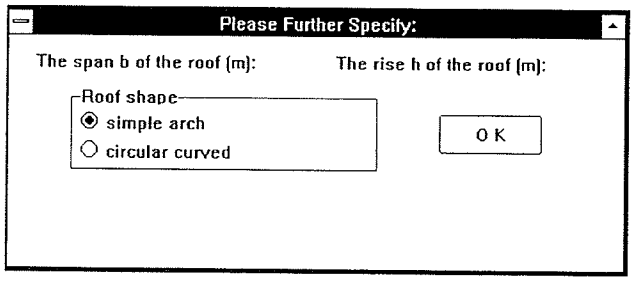


Fig.a8. The querry for simple arch or curved roof

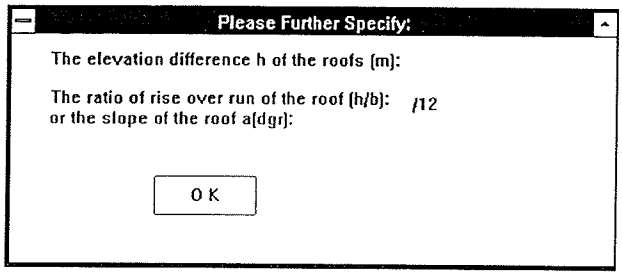


Fig.a9. The querry for lower level of adjacent roof

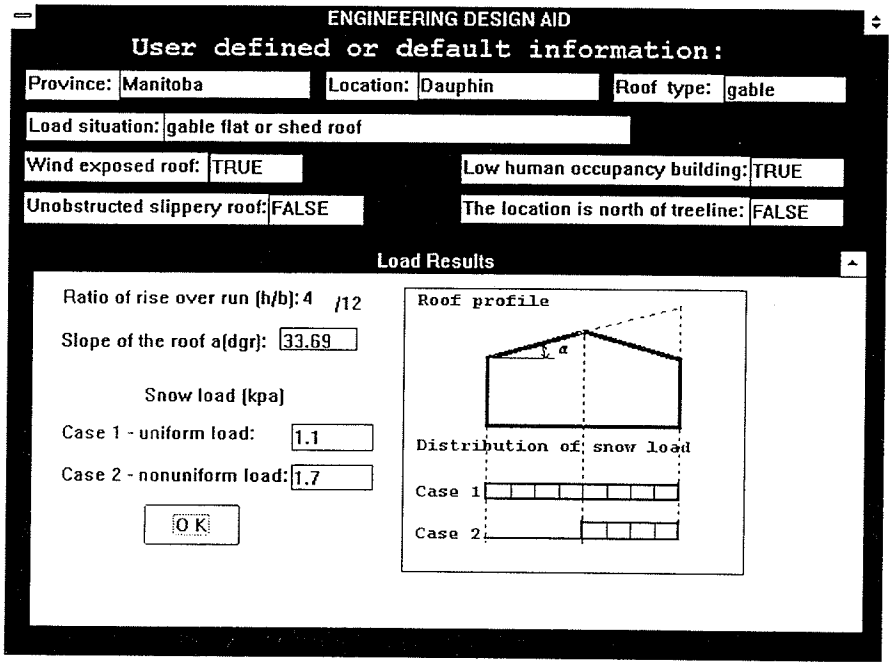


Fig.a10. The output window of snow load results

The upper part of the window shows a summary of user input. The lower part displays the results of snow loads. If the user wants to know snow load values at a certain position of the roof, clicking button *load at a point x* will bring the window shown in Fig.a11. After the user specifies the value of x and hits the *OK* button, snow loads at the specified point will be displayed in the window shown in Fig.a12.

The procedure for wind loads estimation is similar. A display, shown in Fig.a13, will appear when the user hits the button *wind loads* in the main control window shown in Fig.a4.

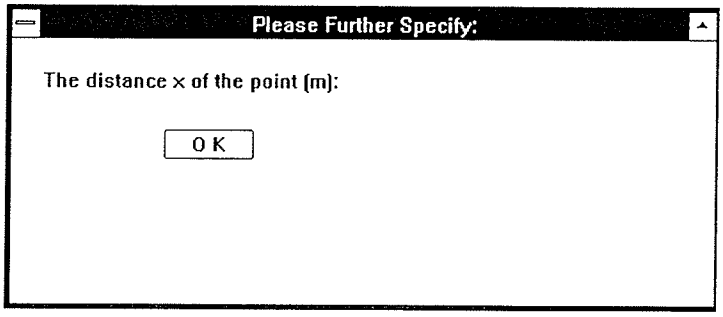


Fig.a11. The input window for load point

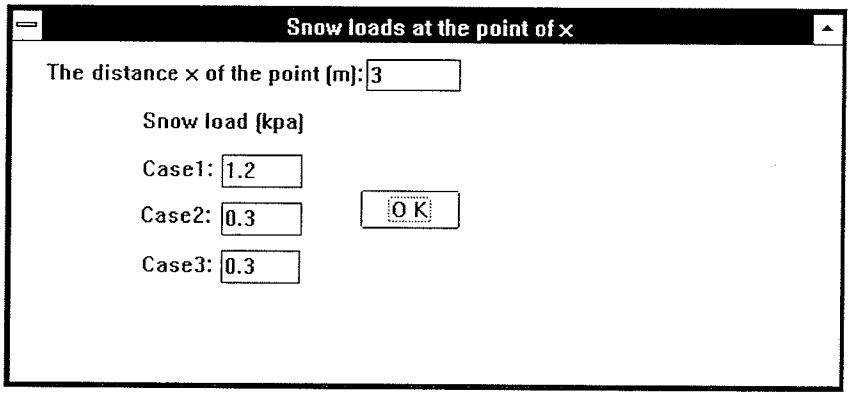


Fig.a12. The output window for snow loads at specified point

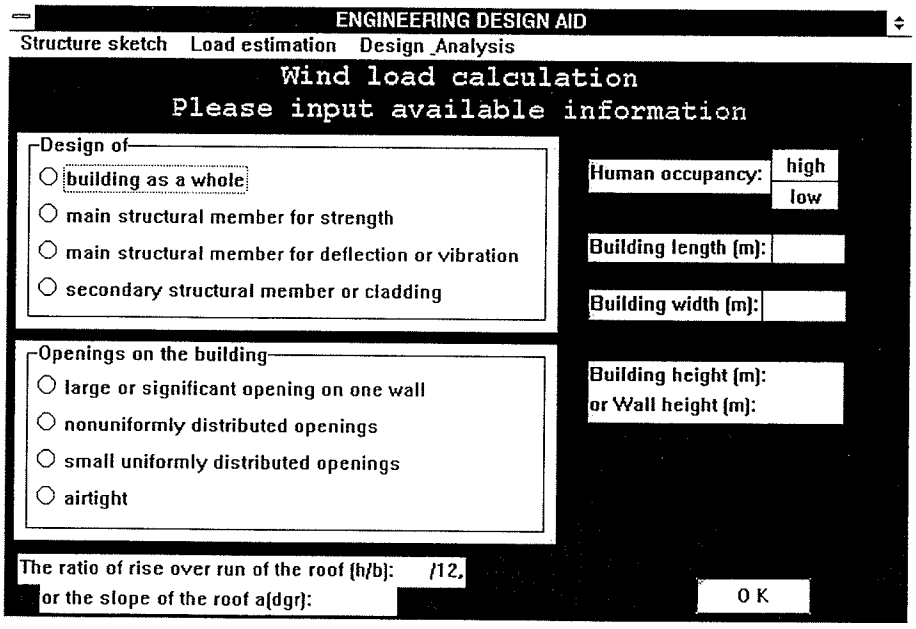


Fig.a13. The input window for wind load estimation

For the design of structural members, the output window will be like the one shown in Fig.a14. As the user specifies *side wall, end wall,* or *roof,* external wind values will change accordingly. The interior wind pressure changes with openings in different walls.

For the design of a *building as a whole*, the output window is shown in Fig.a15. A summary of user input will be shown if button *User defined information* is clicked.

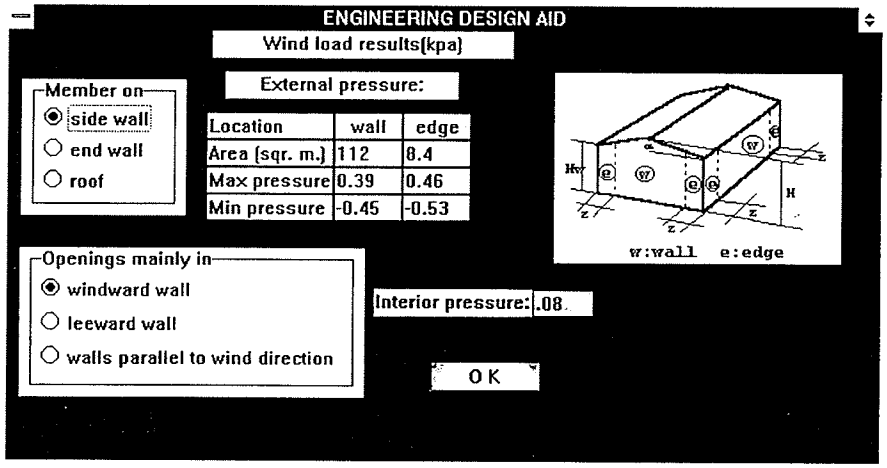


Fig.a14. The output window of snow loads for structural members

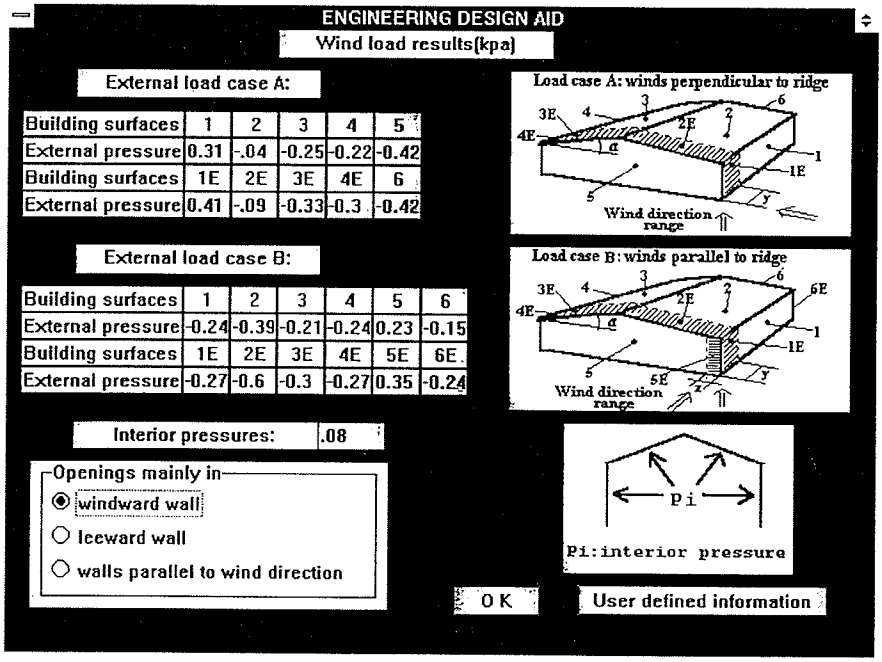


Fig.a15. The output window of snow loads for building as a whole

From the main control window shown in Fig.a4, the design and analysis part of BEDA can be started by clicking the button *detail design*. This part of BEDA is written in Borland Pascal and consists of several units of program codes. Its main program is DTLDSGN.PAS. The window shown in Fig.a16 will appear on the screen when DTLDSGN.PAS is started.

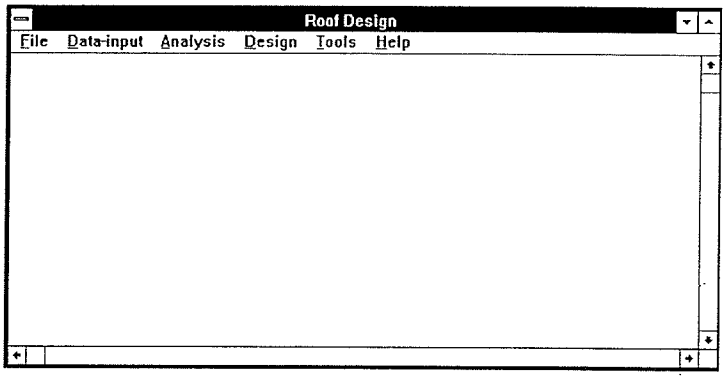


Fig.a16. The user editting window for design and analysis

In responding to a user's click of the pull-down menu, other units are called to finish various design and analysis functions. The user can open and retrieve a graph by clicking *open* from *the File* menu. The file retrieving function is accomplished by the program unit RETRIVE.PAS. It retrieves graphics saved as *DXF* file from any graphical editing software. It also automatically decomposes retrieved graphics into graphical elements, labels them, and reprints them on the screen, as Fig.a17 shown.

When the sketch of a structure is retrieved into BEDA, the user needs to specify loads applied to the structure. The load input function is started from the submenu *Load-input* under *Data-input* and realized by the program unit LOADDL.PAS. Previously calculated snow loads and wind loads can be retrieved by clicking *show snow load* and *show wind load*, see Fig.a18 and Fig.a19. User specified loads can be input through the window shown in Fig.a20, which is activated by clicking *loads input*.

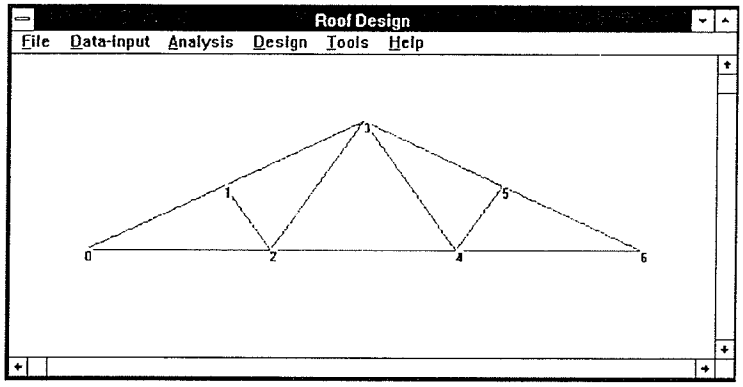


Fig.a17. The retrieved DXF graph from other software

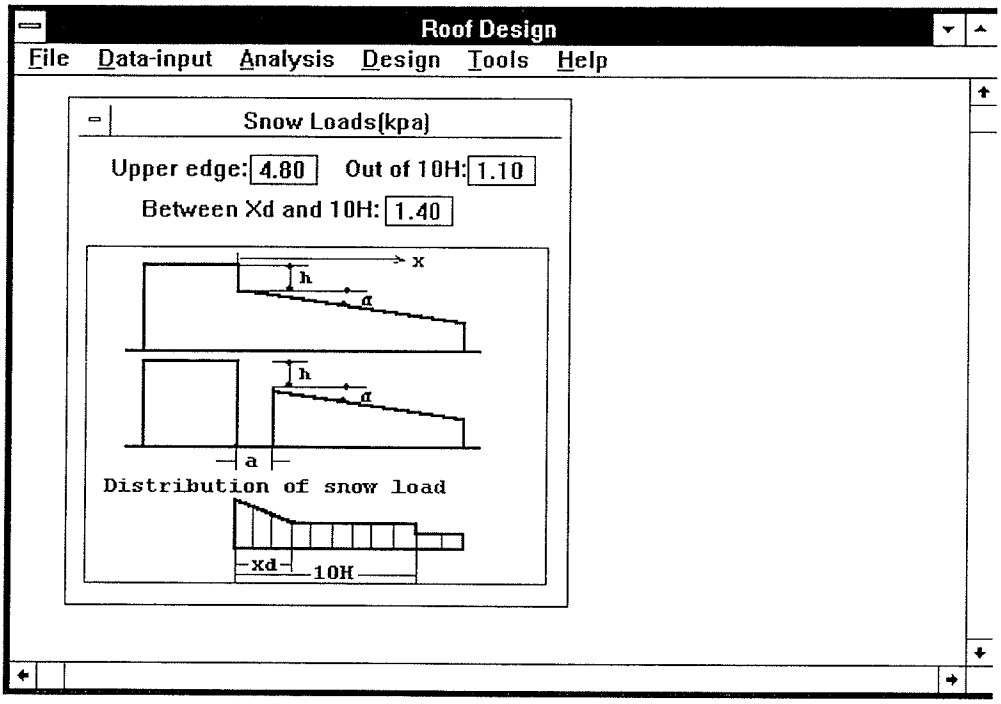


Fig.a18. The retrieved snow loads

The other submenu under Data-input is Structure-input. Its corresponding program unit is STNIDNTF.PAS. The window shown in Fig.a21 is activated by clicking *section properties*. It allows the user to specify structural properties of the structure. Boundary conditions of the

structure can be specified through the window shown in Fig.a22, which is activated by clicking *boundary conditions*.

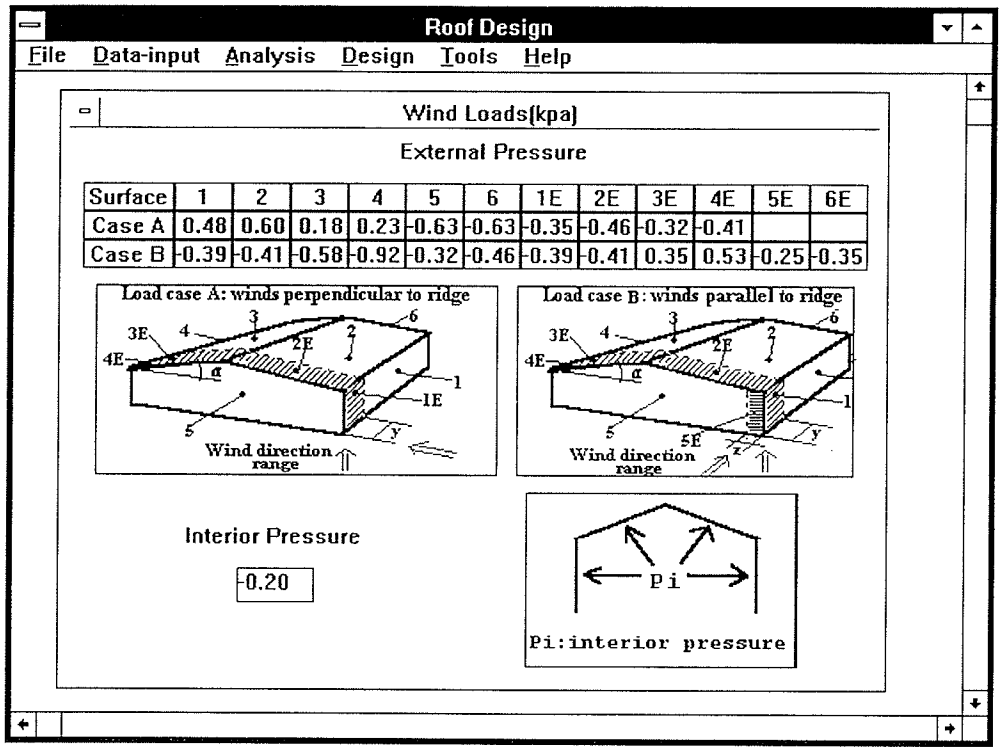


Fig.a19. The retrieved wind loads

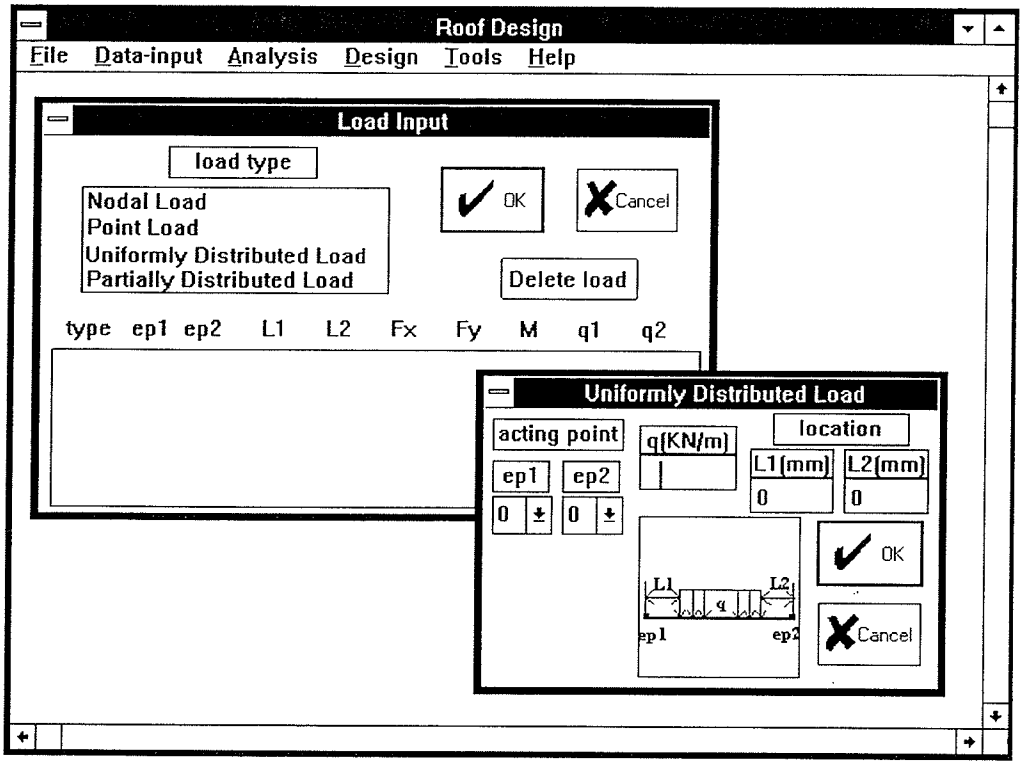


Fig.a20. The input window for external loads

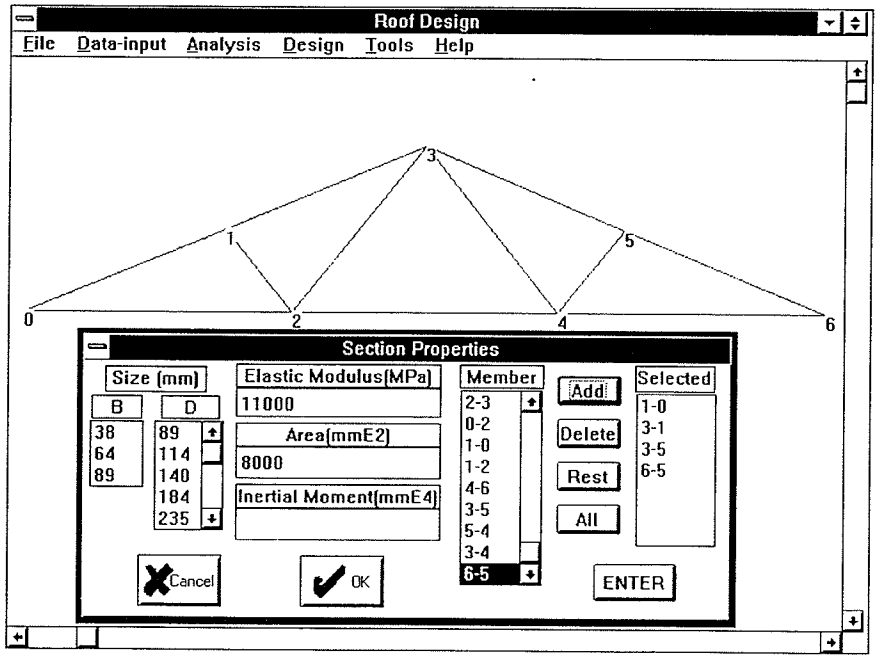


Fig.a21. The dialogue window for specifying section properties

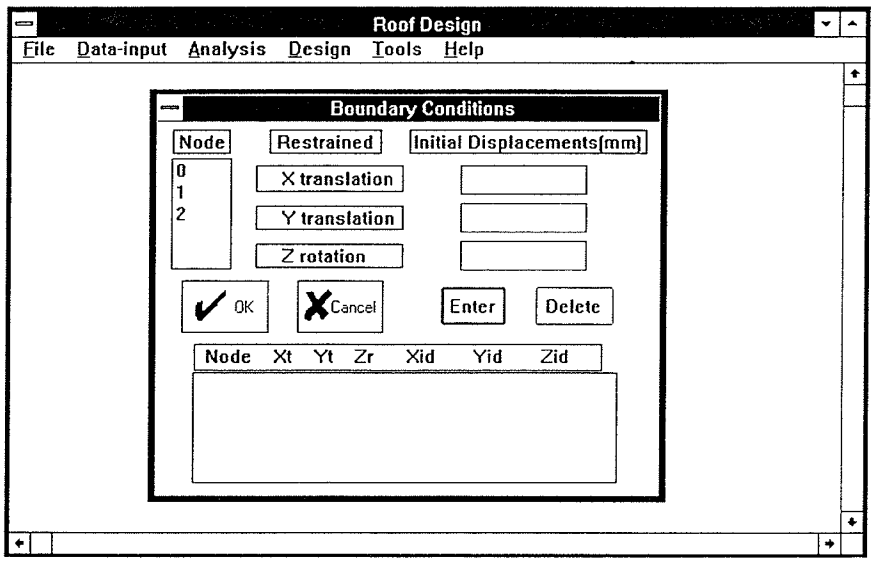


Fig.a22. The dialogue window for specifying boundary conditions

After above specifications, the user can analyze the structure by calling the finite element analysis program unit FEANLS.PAS. When the user clicks *Call FEA* under menu *Analysis*, the input window shown in Fig.a23 will appear. Results of the analysis will be presented on the output window shown in Fig.a24.

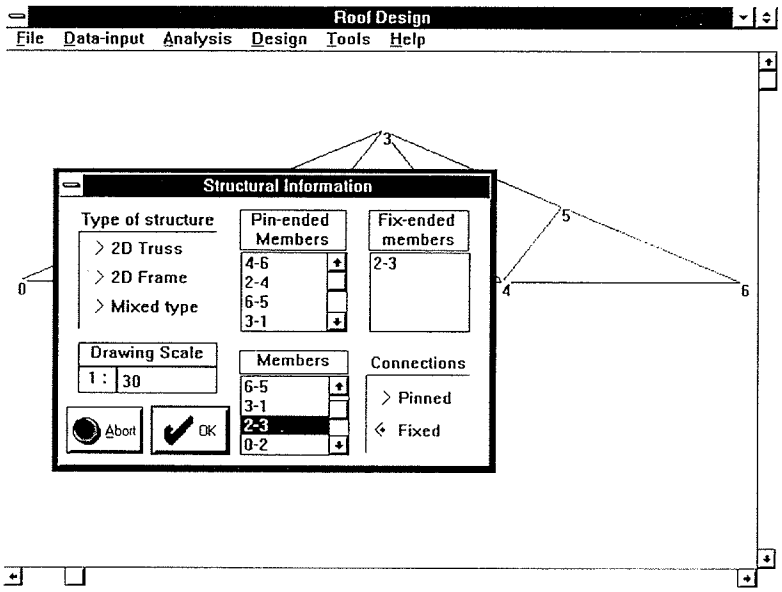


Fig.a23. The input window for finite element analysis

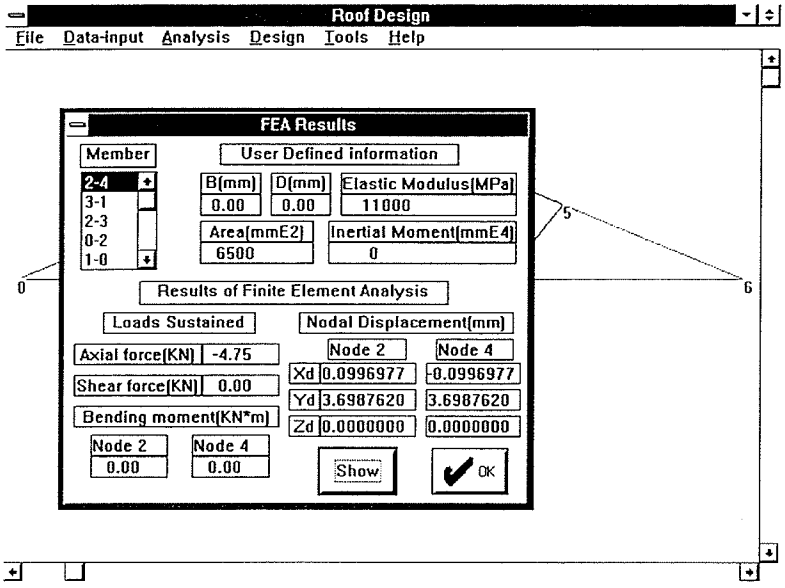


Fig.a24. The output window for finite element analysis

Based on the analysis, the user can go to detail member design by clicking *Member details* under *Design* menu. It will start the program unit MEMDSN.PAS. The input window for member design is shown in Fig.a25. The output window is shown in Fig.a26. Design results can also be printed out by clicking *print* under *File* menu. The printing function is fulfiled by the program unit PRNTEST.

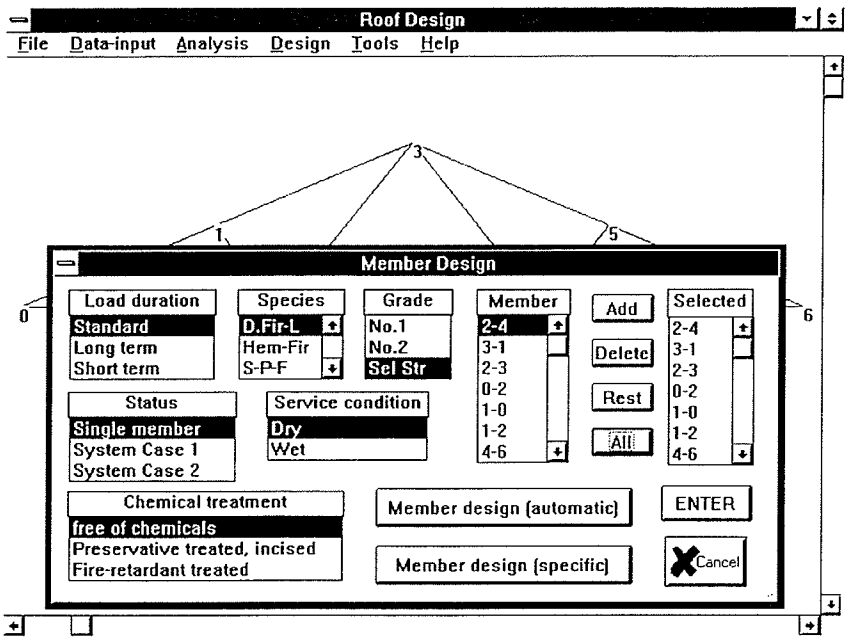


Fig.a25. The user input window for member design

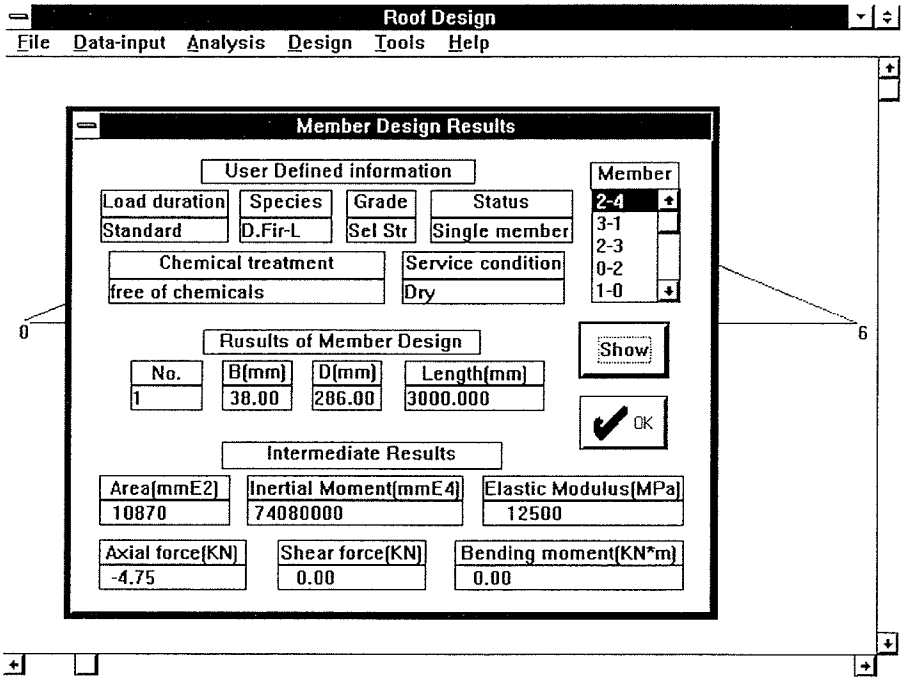


Fig.a26. The output window of automatic member design