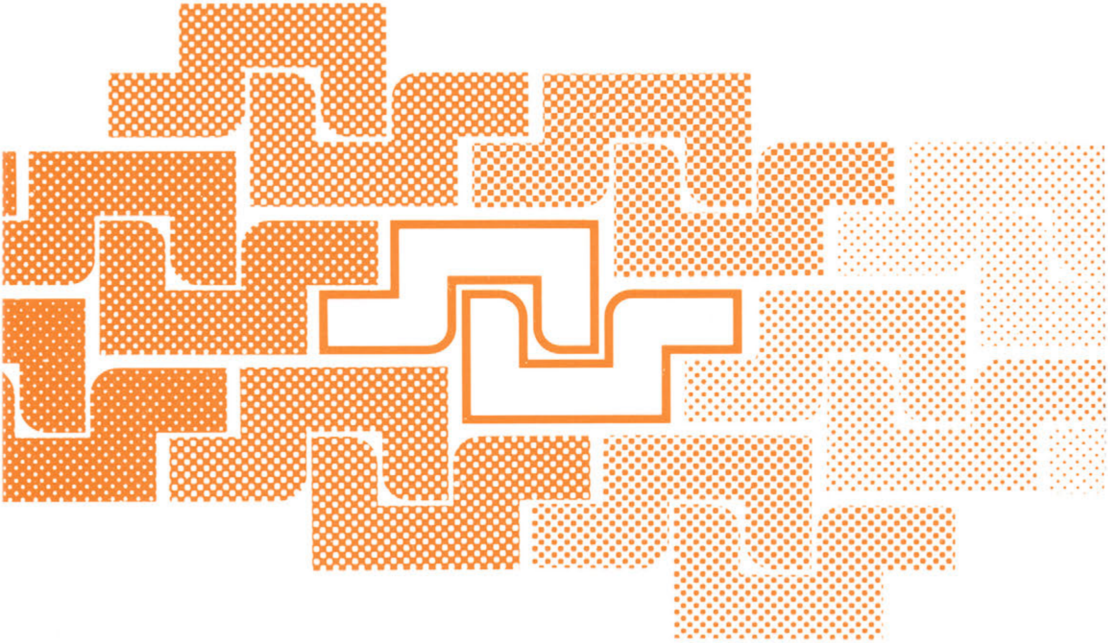


Optimization and Artificial Intelligence in Civil and Structural Engineering

Volume II
Artificial Intelligence in Civil and Structural Engineering

Edited by

B. H. V. Topping

NATO ASI Series

Series E: Applied Sciences - Vol. 221

Optimization and Artificial Intelligence in Civil and Structural Engineering

Volume II: Artificial Intelligence in Civil and Structural Engineering

NATO ASI Series

Advanced Science Institutes Series

A Series presenting the results of activities sponsored by the NATO Science Committee, which aims at the dissemination of advanced scientific and technological knowledge, with a view to strengthening links between scientific communities.

The Series is published by an international board of publishers in conjunction with the NATO Scientific Affairs Division

A	**Life Sciences**	Plenum Publishing Corporation
B	**Physics**	London and New York
C	**Mathematical**	Kluwer Academic Publishers
	and Physical Sciences	Dordrecht, Boston and London
D	**Behavioural and Social Sciences**	
E	**Applied Sciences**	
F	**Computer and Systems Sciences**	Springer-Verlag
G	**Ecological Sciences**	Berlin, Heidelberg, New York, London,
H	**Cell Biology**	Paris and Tokyo
I	**Global Environmental Change**	

NATO-PCO-DATA BASE

The electronic index to the NATO ASI Series provides full bibliographical references (with keywords and/or abstracts) to more than 30000 contributions from international scientists published in all sections of the NATO ASI Series.
Access to the NATO-PCO-DATA BASE is possible in two ways:

– via online FILE 128 (NATO-PCO-DATA BASE) hosted by ESRIN, Via Galileo Galilei, I-00044 Frascati, Italy.

– via CD-ROM "NATO-PCO-DATA BASE" with user-friendly retrieval software in English, French and German (© WTV GmbH and DATAWARE Technologies Inc. 1989).

The CD-ROM can be ordered through any member of the Board of Publishers or through NATO-PCO, Overijse, Belgium.

Series E: Applied Sciences - Vol. 221

Optimization and Artificial Intelligence in Civil and Structural Engineering

Volume II: Artificial Intelligence in Civil and Structural Engineering

edited by

B. H. V. Topping
Department of Civil and Offshore Engineering,
Heriot-Watt University,
Edinburgh, U.K.

Springer-Science+Business Media, B.V.

Proceedings of the NATO Advanced Study Institute on
Optimization and Decision Support Systems in Civil Engineering
Edinburgh, U.K.
25 June–6 July, 1989

Library of Congress Cataloging-in-Publication Data

Optimization and artificial intelligence in civil and structural engineering / edited by B.H.V. Topping.
p. cm. -- (NATO ASI series. Series E, Applied sciences ; vol. 221)
Contents: v. 1. Optimization in civil and structural engineering -- v. 2. Artificial intelligence in civil and structural engineering.
ISBN 978-90-481-4202-6 ISBN 978-94-017-2492-0 (eBook)
DOI 10.1007/978-94-017-2492-0
1. Civil engineering--Mathematical models. 2. Structural engineering--Mathematical models. 3. Artificial intelligence.
I. Topping, B. H. V. II. Series: NATO ASI series. Series E, Applied sciences ; no. 221.
TA153.068 1992
624'.01'5118--dc20 92-26741

ISBN 978-90-481-4202-6

Printed on acid-free paper

Contents

Part II

DIRECTOR

Dr B H V Topping, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom

ORGANISING COMMITTEE

Professor C B Brown, Department of Civil Engineering, University of Washington, Seattle, Washington, United States of America

Professor J S Gero, Department of Architectural Science, Univesity of Sydney, Sydney, Australia

Professor D Grierson, Department of Civil Engineering, University of Waterloo, Ontario, Canada

Professor P W Jowitt, Department of Civil Engineering, Heriot-Watt University, Riccarton, Edinburgh, United Kingdom

Professor A B Templeman, Department of Civil Engineering, The University of Liverpool, Lancashire, England

Dr B H V Topping, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom

PRINCIPAL LECTURERS

Professor C B Brown, Department of Civil Engineering, University of Washington, Seattle, Washington, United States of America

Professor D G Elms, Department of Civil Engineering, University of Canterbury, Christchurch, New Zealand

Professor J S Gero, Department of Architectural Science, Univesity of Sydney, Sydney, Australia

Professor L Gründig, Institut für Geodäsie und Photogrammetrie, Technische Universität Berlin, Berlin, West Germany

Professor D Grierson, Department of Civil Engineering, University of Waterloo, Ontario, Canada

Dr M H Houck, Purdue University, West Lafayette, Indiana, United States of America

A T Humphrey, GEC Marconi Research Centre, Great Baddow, Chelmsford, Essex, England

Dr D.G. Jamieson, Thames Water, Reading, Berkshire, England

Professor A Jennings, Department of Civil Engineering, Queeen's University of Belfast, Belfast, United Kingdom

Professor P W Jowitt, Department of Civil Engineering, Heriot-Watt University, Riccarton, Edinburgh, United Kingdom

Professor U Kirsch, Department of Civil Engineering, Technion, Israel Institute of Technology, Technion City, Haifia, Israel

Professor M.L. Maher, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, United States of America

Dr R N Palmer, Department of Civil Engineering, University of Washington, Seattle, Washington, United States of America

Professor G I N Rozvany, Department of Civil Engineering, University of Essen, Essen, West Germany

Professor A B Templeman, Department of Civil Engineering, The University of Liverpool, England

Professor G Thierauf, Department of Civil Engineering, University of Essen, Essen, West Germany

Dr B H V Topping, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom

Dr G N Vanderplaats, VMA Engineering, Goleta, California, United States of America

PARTICIPANTS

Professor A Adao-da-Fonseca, Universidade do Porto, Gabinete de Estruturas, Faculdade de Engenharia, Porto, Portugal

Dr G Akhras, Department of Civil Engineering, Royal Military College of Canada, Kingston, Ontario, Canada

Professor D Altinbilek, Department of Civil Engineering, Middle East Technical University, Ankara, Turkey

Dr T Arciszewski, Department of Civil Engineering, Wayne State University, Detroit, Michigan, United States of America

A M Azevedo, University of Porto, Faculty of Engineering, Gabinete de Estructures, Porto, Portugal

Dr B W Baetz, Department of Civil Engineering, McMaster University, Hamilton, Ontario, Canada

J Bäuerle, Institut für Geodäsie und Photogrammetrie, Technische Universität Berlin, Berlin, West Germany

S Bitzarakis, Department of Civil Engineering, National Technical University, Athens, Greece

Professor D G Carmichael, School of Civil Engineering, University of New South Wales, Kensington, New South Wales, Australia

Duane Castaneda, Department of Civil Engineering, University of Washington, Seattle, Washington, United Sates of America

S Coelho, LNEC - National Laboratory of Civil Engineering, Lisboa, Portugal

J B Comerford, Department of Civil Engineering, Bristol University, Bristol, England

J W Davidson, Department of Civil Engineering, University of Manitoba, Winnipeg, Manitoba, Canada

I Enevoldsen, University of Aalborg, Aalbourg, Denmark

J H Garcelon, Department of Aerospace Engineering, University of Florida, Gainesville, United States of America

Professor W J Grenney, Department of Civil & Environmental Engineering, Utah State University, Logan, Utah, United States of America

Professor D F Haber, Department of Civil Engineering, University of Idaho, Moscow, Idaho, United Sates of America

Professor P Hajela, Department of Aerospace Engineering, University of Florida, Gainesville, Florida, United States of America

T B Hardy, Department of Civil Engineering, Utah State University, Logan, Utah, United States of America

Professor M I Hoit, Department of Civil Engineering, University of Florida, Gainesville, Florida, United States of America

Dr B Heydecker, Transport Studies Group, University College London, London, England

D Howarth, Department of Civil Engineering, Heriot-Watt University, Riccarton, Edinburgh, United Kingdom

Dr S Hernandez, Department of Mechanical & Environmental Engineering, University of California, Santa Barbara, Santa Barbara, California, United States of America

J Jih, Dept of Aerospace Engineering, University of Florida, Gainesville, Florida,

United States of America

D Kallidromitou, National Technical University of Athens, Hydraulics Laboratory, Athens, Greece,

K Kayvantash, Baumechanik Instatik, Universität Essen, Essen, West Germany

Dr J O Kim, Department of Civil Engineering, University of Liverpool, Brownlow Street, Liverpool, England

Dr V K Koumousis, Department of Civil Engineering, National Technical University of Athens, Athens, Greece

Dr B Kumar, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom

Dr J Liu, Department of Civil Engineering, University of Liverpool, Brownlow Street, Liverpool

Professor I A Macleod, Department of Civil Engineering, University of Strathclyde, Glasgow

S Maheepala, Department of Civil Engineering, University of Newcastle upon Tyne, Newcastle upon Tyne

Professor M Malafaya-Baptista, Faculdade de Engenharia, Laboratorio de Hidraulica, Universidade do Porto, Porto, Portugal

Dr I M May, Department of Civil Engineering, University of Bradford, Bradford, England

Profesor P H McDonald, Department of Civil Engineering, North Carolina State University, Raleigh, North Carolina, United States of America

S Meyer, Department of Civil Engineering, Carnegie Mellon University, Pittsburg, United States of America

Dr T C K Molyneux, Department of Civil Engineering, University of Liverpool, Brownlow Street, Liverpool, England

Dr J Oliphant, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom

Dr M Oomens,Technische Universiteir, Civiele Techniek, Delft, The Netherlands

Professor G Palassopoulos, Military Academy of Greece, Athens, Greece

J C Rodrigues, Universidade de Coimbra, Departmento de Engenharia Civil, Coimbra, Portugal,

G Schoeppner, Department of Civil Engineering, Ohio State University, Columbus, Ohio, United States of America

Professor L M C Simoes, Departmento de Engenharia Civil, Universidade de Coimbra, Coimbra, Portugal

A Soeiro, Universidade do Porto, Departmento Engenharia Civil, Porto, Portugal

Dr D G Toll, School of Engineering & Applied Science, University of Durham, Durham, England

Dr F Turkman, Dokuz Eylul University, Department of Civil Engineering, Bornova, Izmir, Turkey

Dr G J Turvey, Engineering Department, Lancaster University, Bailrigg, Lancaster, England

N A Tyler, Transport Studies Group, University College London, London, England

Professor G Ulusoy, Department of Industrial Engineering, Bogazici University, Istanbul, Turkey

P Von Lany, Sir William Halcrow and Partners, Burderop Park, Swindon, Wiltshire, England

Dr S Walker, North West Water Authority, Warrington, England

Dr G A Walters, School of Engineering, University of Exeter, Exeter, England

Professor W E Wolfe, Department of Civil Engineering, Ohio State University, Columbus, Ohio, United States of America

C Xu, Department of Civil Engineering, Heriot-Watt University, Riccarton, Edinburgh, UK

PREFACE

This volume and its companion volume includes the edited versions of the principal lectures and selected papers presented at the NATO Advanced Study Institute on *Optimization and Decision Support Systems in Civil Engineering.*

The Institute was held in the Department of Civil Engineering at Heriot-Watt University, Edinburgh from June 25th to July 6th 1989 and was attended by eighty participants from Universities and Research Institutes around the world. A number of practising civil and structural engineers also attended. The lectures and papers have been divided into two volumes to reflect the dual themes of the Institute namely Optimization and Decision Support Systems in Civil Engineering. Planning for this ASI commenced in late 1986 when Andrew Templeman and I discussed developments in the use of the systems approach in civil engineering. A little later it became clear that much of this approach could be realised through the use of knowledge-based systems and artificial intelligence techniques. Both Don Grierson and John Gero indicated at an early stage how important it would be to include knowledge-based systems within the scope of the Institute.

The title of the Institute could have been: 'Civil Engineering Systems' as this would have reflected the range of systems applications to civil engineering problems considered by the Institute. These volumes therefore reflect the full range of these problems including: structural analysis and design; water resources engineering; geotechnical engineering; transportation and environmental engineering.

The systems approach includes a number of common threads such as: mathematical programming, game theory, utility theory, statistical decision theory, networks, and fuzzy logic. But a most important aspect of the Institute was to examine similar representations of different civil and structural engineering problems and their solution using general systems approaches. This systems approach to civil and structural engineering is well illustrated in the first paper in the first volume of these proceedings. The Decision Support aspect of the Institute was reflected by the knowledge-based systems and artificial intelligence approach. Papers discussing many aspects of knowledge-based systems and artificial intelligence in civil and structural engineering are included in this volume.

I should like to thank all the members of the organising committee for their assistance given so readily both before and during the Institute: Professor C B Brown, Department of Civil Engineering, University of Washington, Seattle, United States of America; Professor J S Gero, Department of Architectural Science, Univesity of Sydney, Australia; Professor D Grierson, Department of Civil Engineering, University of Waterloo, Canada; Professor P W Jowitt, Department of Civil Engineering, Heriot-Watt University, Edinburgh, United Kingdom; and Professor A B Templeman, Department of Civil Engineering, The University of Liverpool, United

Kingdom. The Director and Members of the Organising Committee would like to thank the NATO Scientific Affairs Committe for funding the Institute. Without the support of the Committee this Institute could not have been held.

I should also like to thank Professor A D Edwards, Head of the Department of Civil Engineering and Dean of the Faculty of Engineering at Heriot-Watt University for all the support he provided with this and other projects.

My sincere thanks are also due to the research students of the Department of Civil Engineering at Heriot-Watt University including: David Howarth, Chen Chao Xu, Jim Milne and John Hampshire who helped me considerably during the Institute. I should also like to thank Erik Moncrieff who was responsible for typesetting many of the papers in these proceedings and Asad Khan who so kindly came to my assistance during the final stage of preparation of these proceedings.

Finally I should also like to thank Dr L V da Cunha, Director of the NATO ASI Programme for all his help in the organisation of the Institute.

Barry H V Topping
Department of Civil Engineering
Heriot-Watt University
Edinburgh

Knowledge-Based Approaches to Engineering Design[1]

John S. Gero
The University of Sydney
New South Wales
Australia

Abstract This article introduces the fundamental notions of Knowledge-Based Systems before introducing a variety of categories of knowledge-based systems in design. It distinguishes between systems used for analysis and those used for synthesis. It concludes by discussing research which is likely to support future knowledge-based design systems.

1 Introduction

Design is one of the fundamental purposeful acts of humans. It commences with dissatisfaction with the current state either because the current state is unsatisfactory or because there is a perception that it can be improved. Thus, design is always goal driven. However, one of the distinguishing features of design (from, say, problem solving) is that in design part of the process involves determining and deciding the goals. The goals in design are called functions. These are the functions that are expected to be exhibited by the resultant artifact. The results of designing are not built, constructed or manufactured artifacts, only descriptions of artifacts. These descriptions describe the structure of the artifacts. Structure is the set of elements and their relationships which go to make up the artifact. Designing is concerned with transforming function to structure.

2 Computer-Aided Design

Computer-aided design has passed through a number of distinct phases. It commenced in the 1960s with a concern for graphical representation of objects being

[1] This lecture draws directly from Gero, J. S., (1989), 'Expert systems for design,' *Proc. ICESEA 89*, Wuhan, China.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 1–12.
© 1992 *Kluwer Academic Publishers.*

designed. This graphical genesis still manifests itself in today's computer-aided design systems. In the 1970s there was a concern for object modeling to support graphical representation. The aspect being modeled was geometry and topology. It is often simply called geometric modeling. There was the recognition that aspects other than geometric were also needed, so many systems allowed the inclusion of non-geometric attributes by attaching them to geometric entities.

By the end of the 1970s and the early 1980s geometric modeling had reached sophisticated levels. At the same time engineering analysis tools were finding their way into computer-aided design systems. The most prominent amongst these was the finite element analysis method.

However, with some exceptions, computer-aided design systems were not concerned with providing direct assistance to designers in their design decision-making processes. The exceptions derived their impetus from operations research techniques but did not find widespread acceptance. Recently, there has been renewed interest in using computers as direct aids to design decision making.

3 What are Knowledge-Based Systems

3.1 Knowledge Engineering

Knowledge-based systems are computer programs in which the knowledge is explicitly coded rather than implicitly encoded. They make use of knowledge engineering. Knowledge engineering is a subfield of artificial intelligence. It is concerned with the acquisition, representation and manipulation of human knowledge in symbolic form. Human knowledge is thought of as being reasoning (rather than the simple ability to acquire facts as you might find in an encyclopedia). Just as the industrial revolution can be considered to have automated mechanical power, and the computer revolution to have automated calculation, so knowledge engineering automates reasoning.

Feigenbaum (1977) defines the activity of knowledge engineering as follows:

> The knowledge engineer practices the art of bringing the principles and tools of artificial intelligence research to bear on difficult application problems requiring expert knowledge for their solution. The technical issues of acquiring this knowledge, representing it, and using it appropriately to construct and explain lines of reasoning are important in the design of knowledge-based systems ... The art of constructing intelligent agents is both part of and an extension of the programming art. It is the art of building complex computer programs that represent and reason with knowledge of the world.

The fundamental structure used to represent reasoning and, hence, knowledge, is symbolic inference. Inference is based on well established logic principles and has been extended to operate on symbols. The obvious advantage of inferencing is that it does not require an a priori mathematical theory such as is found in, say, hydraulics or structures. It can be used to manipulate concepts. Barr and Feigenbaum (1981), talking about the applicability of knowledge engineering in conceptual areas, state:

> Since there are no mathematical cores to structure the calculational use of the computer, such areas will inevitably be served by symbolic models and symbolic inference techniques.

3.2 Expert Systems

Expert systems have been defined as knowledge-based computer programs which use symbolic inference procedures to deal with problems that are difficult enough to require significant human expertise for their solution.

- Human experts can be compared with conventional computer programs (Lansdown, 1982).
- Human expertise arises from the possession of structured experience and knowledge in a specific subject area. These skills grow as more and more experience is gained.
- Human experts can explain and, if necessary, defend the advice they give and are aware of its wider implication.
- Human experts determine which knowledge is applicable rather than proceeding algorithmically.
- Human experts can, and frequently have to, act with partial information. In order to supplement this, they ask only sufficient and pertinent questions to allow them to arrive at a conclusion.

Conventional computer programs differ markedly from programs which act as experts.

- They are usually complex and difficult for anyone other than their designers to understand.
- They embody their knowledge of the subject area in terms designed for computational efficiency such that this knowledge is intertwined with the control parts of the program. Thus, the knowledge is implicit in the program in such a way which makes it difficult to alter or change.
- They cannot suggest to their users why they need a particular fact nor justify their results.

Thus, expert systems aim to capture the ability of human experts to ask pertinent questions, to explain why they are asking them, and to defend their conclusions. These aspects are unrelated to a specific domain of knowledge and apply to all experts.

Expert systems are computer programs which attempt to behave in a manner similar to rational human experts. They all share a common fundamental architecture even if the knowledge encoding mechanisms differ. An expert system will have the following components:

an inference engine this carries out the reasoning tasks and makes the system act like an expert

a knowledge base this contains the expert's domain specific knowledge and is quite separate from the inference engine

an explanation facility this interacts with both the knowledge base and the inference engine to explain why an answer is needed at a particular point or how a question can be answered; further it is used to explain how a conclusion was reached or to explain why a specific conclusion could not be reached

a state description or working memory this contains the facts which have been inferred to be true and those which have been found to be false during a particular session, as well as the facts provided by the user of another system

a knowledge acquisition facility this allows the knowledge base to be modified and extended

a natural language interface few expert systems have this yet.

4 Expert Systems in Design

4.1 Expert Systems for Design Analysis and Design Synthesis

Expert systems were originally developed to carry out diagnosis using classification concepts. They readily lend themselves to engineering analysis and evaluation that is, design analysis. Design analysis may be considered as the interpretation of a design description. The facts which describe an object and the knowledge by which properties of the object can be derived can be modelled as formal axiomatic systems. The advantage is that knowledge becomes amenable to formal proof procedures and the mechanism of logical inference (Kowalski, 1983).

A design possesses attributes other than those facts by which it is represented. These attributes can be described as derived, or implicit attributes, and a set of such attributes constitutes the semantic content of the design. The major operation in discovering meaning is interpretation by deductive inference, and the knowledge about interpretation can be formalised as inference rules.

For any system the issue arises as to how the process of interpretation should proceed. For a realistic set of inference rules the number of facts that can be derived is likely to be very large. There will therefore be more work involved in asking of a design: 'What attributes can be inferred from its description?'; than asking: 'Does the design have this particular set of characteristics?' The former suggests a data-driven approach which starts with a design description and an attempt is made to infer as much as the rules will allow. The latter is a goal-directed approach which begins with various attributes and tries to discover if the design possesses those attributes.

A system containing inference rules is of value even when there are no facts constituting a design description. The 'leaf nodes' of an inference tree correspond to requests for facts about the design and so can be handled interactively by means of prompts. They could also be regarded as entry points to other axiomatic subsystems which interpret computer databases. When incorporated into a general purpose inference system a dialogue is produced. The derived facts constitute the meaning of the total system.

The question arises whether the same architecture (inference rules with backward and/or forward chaining) used to carry out design analysis can be useful in design synthesis. It has been shown that it can in those cases where the set of design solutions is not large and where the components of the design and their relationships are known. For more complex problems a different architecture is needed.

Workable systems can be devised which operate on the basis of formal reasoning. This is particularly so in the case of interpreting the properties and performances of buildings where the theory by which interpretations can be made is well understood. This is generally the case, for example, when evaluating the performance of buildings for compliance with the requirements of building codes.

Expert systems of this type are also applicable to the synthesis of designs, particularly for those classes of design problem which can be subdivided into independent subproblems. But expert systems which are applicable to the more general class of design problem can also be devised.

Expert systems for design analysis are well-described in the literature (Sriram and Adey, 1986a; Sriram and Adey, 1986*b*; Sriram and Adey, 1987a; Sriram and Adey, 1987*b*; Sriram and Adey, 1987*c*; Gero, 1988a; Gero, 1988*b*; Gero, 1988*c*; Dym, 1985; Maher, 1987; Pham, 1988). Expert systems for design synthesis can also be found in the above references as well as in Rychener (1988), Gero (1985), and Gero (1987a). The foundations of the use of expert systems for design analysis and design synthesis are presented in Coyne *et al.* (1989).

4.2 Classes of Expert Systems Applications in Design

There appear to be two fundamental classes of applications of expert systems in design. These are:

1. expert systems for design analysis; and
2. expert systems for design synthesis.

The first class can be modeled by reasoning processes based on deductive reasoning, whilst the second class requires abductive reasoning.

4.2.1 Expert Systems for Design Analysis In this class a design description must have been previously produced. The function of design analysis is to transform the structure inherent in the description to a behaviour, so that the behaviour may be evaluated. The deductive knowledge in the knowledge base encodes this knowledge.

If the knowledge is in rules it has the following form:

If structure attributes **then** behaviour attributes.

Often, the connection between behaviour and function is also encoded in the form:

If behaviour attributes **then** function performed.

Some expert systems encode the connection between description and structure in the form:

If description attributes **then** structures attributes.

Figure 1 shows typical frameworks for the use of expert systems for design analysis.

4.2.2 Expert Systems for Design Synthesis In this class the function of the expert system is to aid the human designer to produce the design description or to produce the design description directly. Design synthesis is concerned with transforming expected behaviour derived from function to structure and a resulting design description. The abductive knowledge in the knowledge base encodes this knowledge.

If the knowledge is in rules, it has the following form:

If behaviour attributes **then** structure attributes.

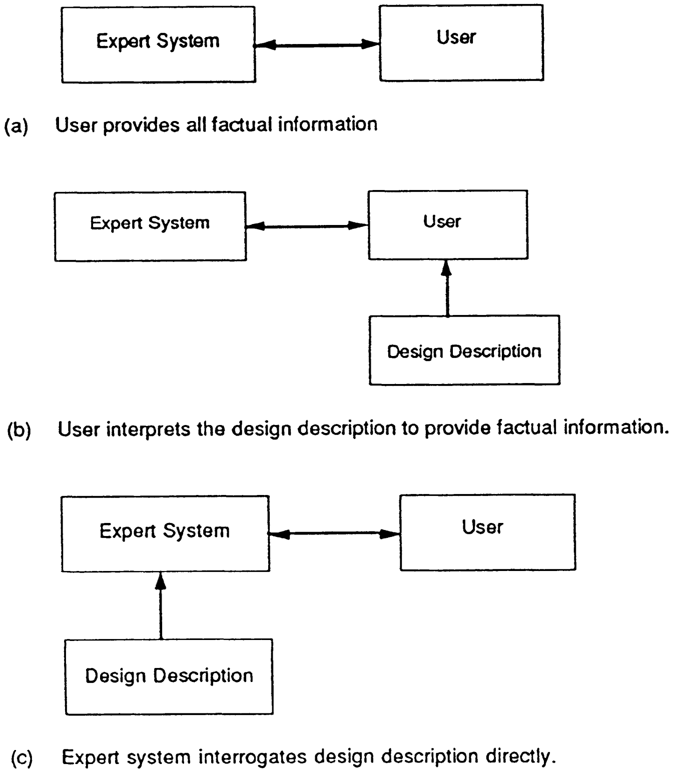
Often, the connection between function and behaviour is also encoded, in the form:

If function required **then** behaviour attributes.

Most expert systems encode the connection between structure and design description, in the form:

If structure attributes **then** design description.

Figure 2 shows typical frameworks for the use of expert systems for design synthesis.



(a) User provides all factual information

(b) User interprets the design description to provide factual information.

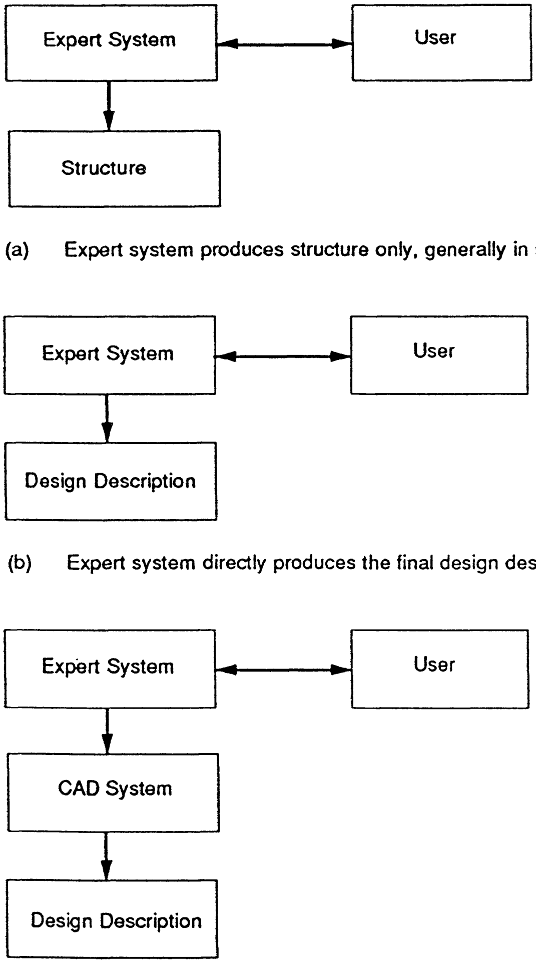
(c) Expert system interrogates design description directly.

Figure 1. *Frameworks for the use of expert systems for design analysis.*

5 Knowledge-Based Computer-Aided Design

The knowledge representation used in the expert systems technology fails to account for fundamental notions in design. Designing, whether with the aid of computers or not, involves transforming a description expressed in function terms to a fixed description expressed in structure terms. *Functions* are the requirements, specifications or goals. Part of designing involves determining the functions. *Structure* is the set of elements and their relationships that go to make up an artifact. When looking at the description of structure there is no explicit function evident. Similarly, function contains no structure. Since these two classes have no descriptors in common how can one be transformed into the other.

A designer's experience allows him to map function onto structure. This is



(a) Expert system produces structure only, generally in symbolic form.

(b) Expert system directly produces the final design description, generally in graphical form.

(c) Expert system produces the structure and drives a commercial CAD system to produce the design description in graphical form.

Figure 2. *Frameworks for the use of expert systems for design synthesis.*

how the abductive rules in expert systems encode this knowledge. However, such a direct mapping does not allow for any reasoning about the transformation process since it is a direct mapping. How does a designer incorporate new structures? It is suggested that both function and structure are translated into a homogeneous concept, namely, behaviour. Function is decomposed into *expected behaviour*. If this behaviour is exhibited by the structure then the function is produced. From the structure the *actual behaviour* can be deduced. In engineering the deductive process of producing the actual behaviour is called 'analysis.' Further it is suggested that function, structure, and behaviour are bound together into a single conceptual schema through experience.

This conceptual schema provides a framework for design activity. It can be accessed via function and it reminds the designer of appropriate structures. It can be accessed via structure and it introduces new functions into the design. It can be accessed via behaviour and structures which produce that behaviour found.

A conceptual schema, called 'prototypes,' for knowledge-based computer-aided design has been developed. A *prototype* is a generalization of groupings of design elements. It provides a framework for storing and processing design experience (Gero, 1987*b*). The prototype represents a class of elements from which instances of elements can be derived. It contains the necessary function and structure descriptions as well as behaviours and knowledge in a generic sense. Variables and methods are also provided. An instance is derived by inheriting any property, variable, and/or method from the generic prototype. A prototype may be related to other prototypes and an instance may need to inherit properties from instances of those prototypes. The system, therefore, constructs its own hierarchy.

A prototype needs to represent the function properties, structure properties, expected behaviours, the relationships to any other prototypes necessary and the knowledge required to find values for structure variables from the function description through behaviour. The function properties include the intended function, and the expected behaviours as attributes and variables. The structure properties include the vocabulary, their topology (these two produce a design description), configurational knowledge, as well as the actual behaviours as attributes and variables of the prototype. The vocabulary will include those elements that are essential to the existence of the prototype and those which are optional. The description will include typological properties as well as other attributes, such as dimensions, material, etc. Knowledge is required for every mapping from a property to another property. Knowledge is required to map from the behaviour attributes to the behaviour variables and to the description required. There will be constraints both on the function side and on the structure side. Constraints on the function side will generally be translated into requirements to be met, whereas constraints on the structure side will generally serve to prune the set of possibilities.

In addition to the elements described above, any design situation exists within a particular context. This context serves to define particular areas of interest. In

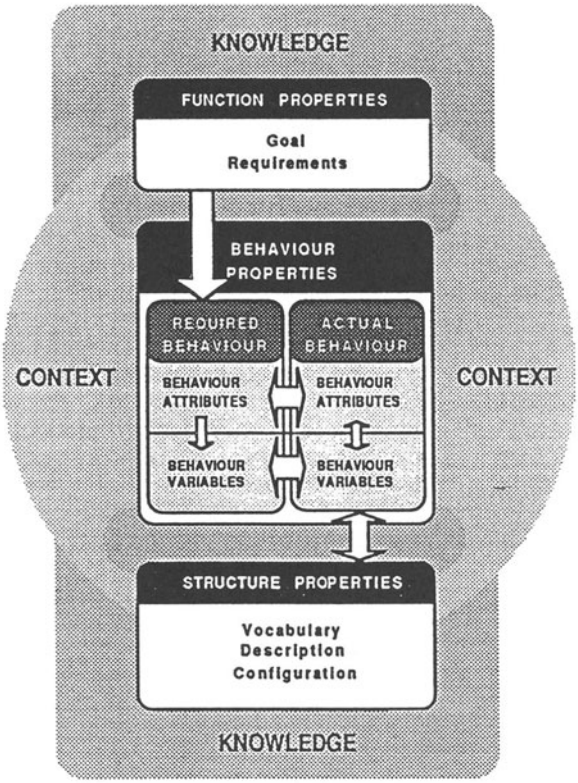


Figure 3. *The conceptual schema prototype (after Rosenman and Gero, 1989).*

some cases the context may merely define a set of function requirements whereas in other cases the context may define some, if not all, of the structure properties. For example, given that we want to design the engineering structure for a 50-storey high office building of square plan this will define a requirement with regards to wind loads and, in addition, may define the engineering structural system type and the material.

Figure 3 shows the model of the prototype schema consisting of function properties, behaviour properties and structure properties all existing within envelopes of knowledge and context. The function description is divided into function properties and behaviour properties where the function properties include the goal (or goals) and the requirements while the behaviour properties include the required (expected) and the actual behaviour attributes and variables.

The goal or goals are the intended function of the prototype. The requirements are divided into those requirements which must always be met and those which may have to be met depending on the particular problem at hand. The behaviour properties form the core of this model. The expected behaviours are derived from the function properties required whereas the actual behaviours are derived from the structure description. However, the selection of the type of behaviour to be derived from the structure description is dictated by the expected behaviours. For example, given the structure description of a door, we would not expect to derive its aromatic properties since this is not a property which has any bearing on its function. It is in this behaviour core, where there are commensurate elements, which allows us to evaluate the suitability of a prototype for a given design situation.

References

[1] Barr, A. and Feigenbaum, E. (Eds.), (1981), *Handbook of Artificial Intelligence*, Vol. 1, William Kaufmann, Los Altos.

[2] Coyne, R. C., Rosenman, M. A., Radford, A. D., Balachandran, M. and Gero, J. S., (1989), *Knowledge-Based Design Systems*, Addison-Wesley, Reading, Massachusetts (to appear).

[3] Dym, C. L. (Ed.), (1985), *Applications of Knowledge-Based Systems to Engineering Analysis and Design*, American Society of Mechanical Engineers, New York.

[4] Feigenbaum, E. A., (1977), 'The art of artificial intelligence: themes and case studies in knowledge engineering,' *IJCAI-77*, William Kaufmann, Los Altos, pp. 1014–1029.

[5] Gero, J. S. (Ed.), (1985a), *Knowledge Engineering in Computer-Aided Design*, North-Holland, Amsterdam.

[6] Gero, J. S. (Ed.), (1987a), *Expert Systems in Computer-Aided Design*, North-Holland, Amsterdam.

[7] Gero, J. S., (1987b), 'Prototypes: A basis for knowledge-based design,' *Working Paper*, Department of Architectural Science, University of Sydney, Sydney.

[8] Gero, J. S. (Ed.), (1988a), *Artificial Intelligence in Engineering: Design*, Elsevier/CMP, Amsterdam.

[9] Gero, J. S. (Ed.), (1988b), *Artificial Intelligence in Engineering: Diagnosis and Learning*, Elsevier/CMP, Amsterdam.

[10] Gero, J. S. (Ed.), (1988c), *Artificial Intelligence in Engineering: Robotics and Processes*, Elsevier/CMP, Amsterdam.

[11] Kowalski, R., (1983), *Logic for Problem Solving*, Elsevier-North Holland, Amsterdam.

[12] Lansdown, J., (1982), *Expert systems: their impact on the construction industry*, RIBA Conference Fund, London.

[13] Maher, M. L. (Ed.), (1987), *Expert Systems for Civil Engineers: Technology and Application*, American Soc. Civ. Eng., New York.

[14] Pham, D. T. (Ed.), (1988), *Expert Systems in Engineering*, IFS Publications/Springer-Verlag, Berlin.

[15] Rosenman, M. A. and Gero, J. S., (1989), 'A conceptual framework for knowledge-based design research at Sydney University's Design Computing Unit,' *Artificial Intelligence in Engineering: Design—II*, CMP, pp. 361–380

[16] Rychener, M. (Ed.), (1988), *Expert Systems for Engineering Design*, Academic Press, New York.

[17] Sriram, D. and Adey, R. (Eds.), (1986a), *Applications of Artificial Intelligence in Engineering Problems—Volume I*, Springer-Verlag, Berlin.

[18] Sriram, D. and Adey, R. (Eds.), (1986b), *Applications of Artificial Intelligence in Engineering Problems—Volume II*, Springer-Verlag, Berlin.

[19] Sriram, D. and Adey, R. (Eds.), (1987a), *Artificial Intelligence in Engineering: Tools and Techniques*, Computational Mechanics Publications, Southampton, England.

[20] Sriram, D. and Adey, R. (Eds.), (1987b), *Knowledge-Based Expert Systems in Engineering: Planning and Design*, Computational Mechanics Publications, Southampton, England.

[21] Sriram, D. and Adey, R. (Eds.), (1987c), *Knowledge-Based Expert Systems in Engineering: Classification, Education and Control*, Computational Mechanics Publications, Southampton, England.

Expert Systems for Engineering Design

M. L. Maher
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh
United States of America

Abstract The synthesis of design alternatives during the early stages of design is based largely on experience. The use of an expert system approach promises to capture some of this expertise and apply it in a systematic manner. The formalization of design knowledge in an expert system is facilitated by an expert system shell developed specifically for design applications. Example applications for structural design are presented.

1 Introduction

Design is a process of producing a description of a system or process to satisfy a set of requirements. Design proceeds through several levels of abstraction, where more information about the requirements as well as the evolving design description is available as the process continues. In this paper, the focus is on the early stages of design where the design knowledge is largely qualitative. During the early stages, or preliminary design, the major components and subsystems are identified and their composition is evaluated.

There are many books that provide definitions and elaborations of the design process; in structural engineering such books include [2], [3], [5], [6]. The design process can be considered as comprising different phases, synthesis being one of these phases. Although the phases may not be addressed hierarchically for the entire design cycle and are often carried out recursively, there is an inherent order in which designers approach a design problem. The following represents one formalism of the design process.

Formulation involves identifying the goals, requirements and possibly the vocabulary relevant to the needs or intentions of the designer.

Synthesis involves the identification of one or more design solutions within the design space elaborated during formulation.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 13–24.
© 1992 *Kluwer Academic Publishers.*

Evaluation involves interpreting a partially or completely specified design description for conformance with goals and/or expected performances. This phase of the design process often includes engineering analysis.

Formulation occurs at some level of abstraction and provides enough information to begin a synthesis process. Synthesis involves identifying the form of the design solution. Evaluation, during the early stages of design, is usually based on a subjective assessment of relevant criteria. Although synthesis and evaluation may be based on associated quantitative models, the designer typically reasons about these models in a qualitative manner.

The knowledge used during synthesis and evaluation of preliminary designs is not well articulated. Experienced designers resort to trial and error less frequently than novice designers when searching for an appropriate or satisfactory form, suggesting that the use of knowledge-based expert systems to represent 'experience' may improve design synthesis and evaluation.

2 Expert Systems

Knowledge-based expert systems (KBES) have emerged from research in artificial intelligence as practical problem-solving tools that can reach a level of performance comparable to that of a human expert in some specialized problem domain.

An expert system can contain from three to six of the components illustrated in Figure 1. All expert systems contain the following three basic components.

The **knowledge base** contains the knowledge specific to the domain of the problem to be solved. The knowledge in an expert system can be classified according to a spectrum ranging from *deep* to *surface* knowledge. Deep or causal knowledge is knowledge of basic principles, such as Newton's laws or static equilibrium. Surface or heuristic knowledge is knowledge developed through experience. Analysis procedures lie close to the deep knowledge end of the spectrum, while knowledge about combining and placing structural systems in a given building is closer to surface knowledge.

The **context** contains facts that reflect the current state of the problem. The organization of the context depends on the nature of the problem domain. The context builds up dynamically as a particular problem is being considered, and is used by the inference mechanism to guide the decision making process.

The **inference mechanism** manipulates the context using the knowledge base. Typically, the inference mechanism applies the knowledge base to the context using an approach suitable for a class of problems. The inference mechanism can embody a number of problem solving strategies, such as forward chaining, where the system reasons about the initial state of known facts until a goal state or conclusion is determined to be true or appropriate. The problem solving strategy serves as a formalization of the process used to solve a problem; it defines the focus of attention

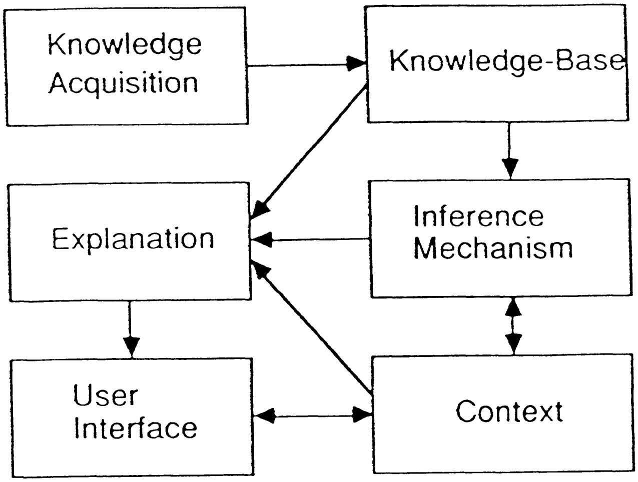


Figure 1. *Architecture of an expert system.*

at any point in the solution process. More detailed descriptions of problem solving strategies can be found in [1], [4], [7].

There are three other components that are not necessarily part of every expert system but are desirable in a final product.

The **knowledge acquisition module** serves as an interface between the expert(s) and the expert system. It provides a means for entering knowledge into the knowledge base and revising this knowledge when necessary.

The **explanation module** provides explanations of the inferences used by the expert system. This explanation can be *a-priori*, why a certain fact is requested, or *a-posteriori*, how a conclusion was reached.

The **user interface module** provides an interface between the user and the expert system, usually as a command language for directing execution. The interface is responsible for translating the input as specified by the user to the form used by the expert system and for handling the interaction between the user and the expert system during the problem solving process.

A number of formalisms have been developed to represent the knowledge in a domain. One such representation is the production system (PS) model [1]. The principal feature of expert systems based on the PS model is that a clear distinction is made between the *knowledge-base*, containing the model of an expert's problem-solving knowledge, and the *control strategy* or inference mechanism which manipulates the knowledge base. In addition to the heuristic surface knowledge, which consists of IF-THEN production rules encoding empirical associations based on experience, the knowledge-base can incorporate fairly deep knowledge comprised of algorithmic procedures. Whereas the knowledge-base is specific to a given domain, the control strategy is completely general.

The process of using a production system KBES is as follows. The user enters some known facts about the problem into the context, the part of the KBES that contains the knowledge about the particular problem at hand. Following its control strategy, the inference mechanism locates the potentially applicable rules—those whose condition portion is *matched* by the facts in the context—selects one of these and *fires* it, that is, causes its action to be executed. The result of any action is to add to or modify some aspect of the context; thus, new rules become candidates to be fired, and a cylce of matching and firing is repeated in an 'infinite loop' until a goal is satisfied or there are no more rules remaining to be fired.

3 EDESYN

EDESYN (Engineering DEsign SYNthesis) is a software environment for developing expert systems for design. The development of EDESYN was modelled after the expert system 'shell' concept. An approach to developing an expert system for structural design was implemented as HI-RISE [7]. This approach was generalized and expanded to facilitate the development of expert systems for design. The design method is implemented as an algorithm to serve as an inference mechanism. The design knowledge is structured to provide a formalism for developing a knowledge-base.

EDESYN solves a design problem by executing a plan that requires a series of goals to be satisfied. Each goal represents a decision in the design process. Thus, design solutions are formed by combining design decisions. Some salient features of EDESYN are:

1. The planning is performed during the design process. Goals are organized in the order best suited for the design problem under consideration.

2. Preliminary design is considered as the synthesis of design decisions at various levels of abstraction. During synthesis, plan generation, plan execution, and goal satisfaction techniques, combined with constraint-directed search, are applied.

3. All feasible solutions for a plan are generated. The alternatives are evaluated using heuristic criteria to identify the solution or set of solutions to be pursued further.

EDESYN consists of five main modules: design knowledge-base, synthesis processor, design context, user interface, and knowledge acquisition facility, as illustrated in Figure 2. When using EDESYN, the knowledge acquisition facility is

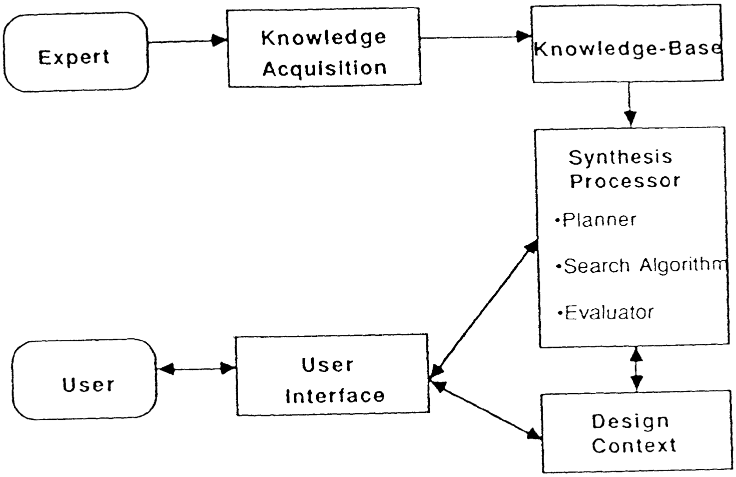


Figure 2. *Architecture of EDESYN.*

invoked first. During knowledge acquisition, the domain specific knowledge is read from files prepared by a domain expert. The domain specific knowledge is stored in the knowledge-base and the synthesis processor is invoked. The user then provides the problem specific information through the user interface to initialize the design context and guides the synthesis of design solutions to augment the context.

The **design knowledge-base** includes decomposition, planning, constraint, and evaluation knowledge. The decomposition knowledge is specified as systems and subsystems, where each system comprises a set of attributes. An attribute may be another system (i.e. subsystem), representing a synthesis node in a goal tree, or a simple attribute, representing a terminal node. The synthesis node is specified by another system. The terminal node is specified as a selection from a set of discrete alternatives or the evaluation of a Lisp function. The planning knowledge is associated with the system to identify the relevant attributes for the current design situation and the order in which the attributes should be considered.

An example of a system definition for designing the lateral load resisting system for a building is:

```
(system lateral
  3D-lateral one-of (core tube 2D-orthogonal)
  2D-lateral subsystem 2D-lateral
planning
  If stories < 5 Then 2D-lateral
end system)
```

The design of a lateral load resisting system is described by the 3D lateral system and the 2D lateral system. The 3D lateral system can be selected from a set of alternatives and the 2D lateral system must be synthesized. The planning rule indicates that buildings with less than 5 stories should only have one attribute, i.e. the 3D lateral system is not appropriate.

The constraints are specified in the knowledge base as elimination constraints, where each constraint is a combination of design decisions and design context that is not feasible. The constraints are used during the synthesis process to eliminate infeasible alternatives. Examples of constraints in the structural design knowledge base are:

```
IF
stories > 30
3D-lateral = 2D-orthogonal
THEN not feasible

IF
2D-lateral-x/material = steel
2D-lateral-y/material = concrete
THEN not feasible
```

The first constraint eliminates a 2D-orthogonal lateral system for buildings with more than 30 stories. The second constraint ensures that a concrete system is not built in the y direction if the lateral system in the x direction is defined to be steel.

The evaluation knowledge is specified by a set of criteria for each synthesis node or system. A criterion is described by a label, a weighting factor, a non-dimensionalizing factor, a normalization factor, and a function to determine the value of the criterion for a design solution. Example criterion for the lateral system are stiffness, compatibility, cost, and ease of construction. The value for each criterion is assessed using qualitative knowledge about structural systems since there is not enough information during preliminary design for a quantitative analysis. For example, stiffness could be assessed in a relative manner, where the designer knows that in most cases a braced frame structure is stiffer than a rigid frame structure.

The **synthesis processor** uses the design knowledge in the knowledge base to produce feasible design solutions consistent with the context. The overall algorithm is based on a constraint directed depth first search through the systems in the knowledge-base. The attributes of each system are assigned all legal values, where a legal value is one that does not get eliminated by the constraints. All feasible combinations are generated for each system, using the planning rules to define and order the attributes. After the alternatives for a system have been synthesized, the evaluation mechanism is invoked. The alternatives are compared for each criteria to produce a set of non-dominated solutions, which are then ranked using the preferences specified by the weighting factors. At this point, the solutions are presented to the designer along with the evaluation information and the designer chooses one solution for further consideration.

The **design context** initially contains the requirements and specifications associated with a particular design problem. For example, the intial context for a structural design problem includes the number of stories in the building, the occupancy, the structural grid, etc. The context expands as synthesis proceeds to include a tree of alternative solutions, where each node in the tree represents a solution for an attribute of a system. Along with the solution tree, a hierarchy tree is maintained to associate each attribute in the solution tree with the system for which it was generated.

The **user interface** is implemented using a multi-window, menu driven interaction style. During the design synthesis process, the user can view and change the design specifications, monitor the synthesis process as a tree of solutions is generated, and view a single solution in more detail.

The **knowledge acquisition** facility transforms the information provided by the expert to the frame based representation of the knowledge base. The expert provides the following design knowledge: preconditions, decomposition, constraints, evaluation criteria, and functions. The design knowledge is specified in a simple syntax and stored in files. Preconditions are specified as a set of names, default values, and allowable ranges. For example, one precondition may be wind load and its default value 30 psf, and its allowable range > 0.0. Decomposition knowledge includes the systems, subsystems, attributes, and planning rules. The constraints are specified as infeasible combinations of elements. Each evaluation criterion is sepcified by a name and a procedure for assigning a value using the goals and elements associated with the current solution. Functions are specified as Lisp functions that use the current state of the design solution to calculate the value of a parameter.

4 STRYPES and STANLAY

EDESYN has been used to develop two expert systems for structural design: STRYPES and STANLAY. STRYPES generates alternative combinations of structural systems and materials for a given building. STANLAY accepts a feasible combination of structural systems and materials for a given building and generates alternative layouts and approximates the load requirements for the structural components. The knowledge bases for each of these expert systems is described below.

The knowledge-base for STRYPES is described by the decomposition knowledge and the constraints for recomposition. The decomposition knowledge is illustrated in Figure 3. The generation of alternative structural system types and materials is decomposed into the lateral and gravity load resisting systems. For the lateral

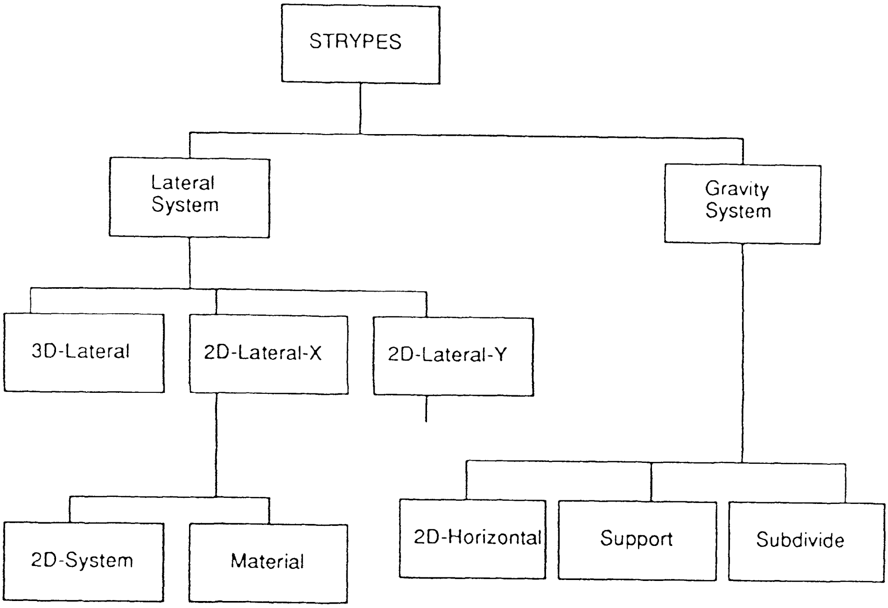


Figure 3. *STRYPES Decomposition Knowledge.*

Gravity-System

2D-Horizontal one-of (concrete steel-deck
panels waffle)
Support one-of (0-edges 2-X-edges 2-Y-edges
4-edges)
SubDivide one-of (none X-direction Y-direction)

Figure 4. *Gravity System in STRYPES.*

system, a selection of alternative 3D systems and 2D systems in each direction are combined. The 3D systems are selected from 2D orthogonal systems and a 3D core system.The 2D systems are selected from rigid frames, braced frames and shear walls. For the gravity system, a selection of alternative 2D-horizontal systems and support conditions are combined. For example, a possible gravity system is a reinforced concrete slab supported on 4 edges without intermediate floor beams. Another possible system is a steel deck supported on two edges with intermediate floor beams.

An example of a system definition in STRYPES is illustrated in Figure 4. The system represents the Gravity-System node in the decomposition tree. The alternative gravity systems are determined by combining selections from different 2D horizontal types and the number of edges supported and the decision to subdivide in one direction. The alternatives formed depend on the constraints and the design context. The use of a particular 2D horizontal type may depend on the lateral system and on the span of the structural grid. These constraints are generalized and stored in the knowledge-base.

The constraints on recomposition in STRYPES eliminate infeasible alternatives to reduce the number of solutions considered. Some constraints are based on design heuristics, eliminating alternatives that an experienced engineer would not consider. For example:

```
IF
lateral-system/3D-lateral = orthogoanl-2D
2D-lateral-system/2D-system = shear-wall
stories > 35
THEN not feasible.
```

This constraint eliminates the use of 2D shear wall systems for buildings with more than 35 stories. Other constraints eliminate unusual combinations of materials and systems. For example:

```
IF
2D-lateral-system/2D-system = shear-wall
2D-lateral-system/Material = steel
THEN not feasible.
```

This constraint eliminates shear walls made entirely of steel.

The decomposition knowledge in STANLAY is illustrated in Figure 5. The layout and load distribution is decomposed into three major decision groups: building parameters, lateral system, and gravity system. The building parameters system calculates and infers additional information about the building given the input conditions. The lateral system is considered by system and component type. The 2D-Panels system places the appropriate systems on the structural grid and distributes the lateral load to each panel. The 2D-Panels system generates alternative

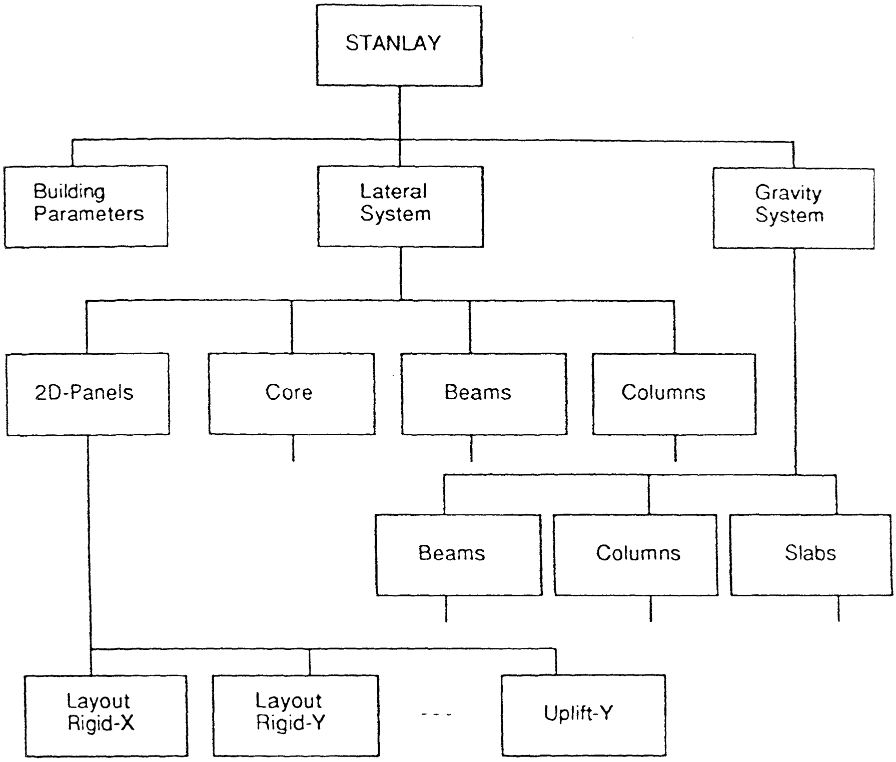


Figure 5. *STANLAY decomposition knowledge.*

placement schemes. The core system locates the walls around the service shaft and determines the lateral load acting on the core. The beams and columns systems distribute the loads to each of the components using approximate analysis techniques. The gravity system, similar to the lateral system, distributes the gravity loads to the components using approximations.

An example of a system definition in STANLAY is illustrated in Figure 6. The system represents the 2D-Panels node in the decomposition tree. The attributes of

```
2D-Panels

layout-rigid-X  one-of  (edges  edges+1   . . . )
layout-rigid-Y  one-of  (edges  edges+1   . . . )
- - -
Mover-X  function . . .
Mover-Y  function . . .
Uplift-X  function . . .
Uplift-Y  function . . .

Planning Rules:

IF  (2D-Lateral-  X = rigid-frame)
AND   (2D-Lateral-Y = rigid-frame)
THEN   (layout-rigid-X layout-rigid-Y   . . .)

IF  (2D-Lateral-X = braced-frame)
AND   (2D-Lateral-Y = rigid-frame)
AND  (TotalLength Y-Bays) > (TotalLength X-Bays)
THEN  (layout-braced-X layout-rigid-Y    . . .)
 - - -
```

Figure 6. *2D-panels system in STANLAY.*

the 2D-Panels system include layout information and load information. The layout attributes are selected and ordered by the planning rules. The load attributes, i.e. overturning moment in each direction (M_{over}) and uplift forces, are computed by Lisp functions. The layout attributes have values that represent alternative placement schemes, e.g. edges indicates that the panels are placed on the edges of the building only, edges + 1 places a panel in the center of the building in addition to the edges. The combination and use of the placement schemes are checked by constraints for consistency with building geometry and intended occupancy. Other constraints in STANLAY check the load attributes for each of the subsystems and components for appropriate magnitudes.

5 Conclusion

An expert system shell for preliminary engineering design has been developed. This shell allows an experienced designer to develop a knowledge-base by defining systems, constraints, evaluation criteria and planning rules. The inference mechanism is a constraint directed search for alternative combinations of systems that are consistent with the design context. The development of this shell is predicated on the observation that forward or backward chaining rule based tools do not facilitate the development of an expert system for design. The application of this shell to preliminary structural design illustrates the approach to developing a knowledge-base for design. The experience in developing these knowledge-bases has shown that incremental development and reorganization is relatively easy and facilitates the formalization of design knowledge.

References

[1] Brownstone, L., Farrell, R., Kant, E. and Martin, N., (1985), *Programming Expert Systems in OPS5*, Addison Wesley.

[2] Cowan, H. J. and Wilson, F., (1981), *Structural Systems*, Van Nostrand Reinhold.

[3] Fraser, D. J., (1981), *Conceptual Design And Preliminary Analysis of Structures*, Pitman.

[4] Giarratano, J. and Riley, G., (1989), *Expert Systems Principles and Programming*, PWS-KENT.

[5] Holgate, A., (1986), *The Art in Structural Design*, Oxford University Press.

[6] Lin, T. Y. and Stotesbury, S. D., (1981), *Structural Concepts and Systems For Architects And Engineers*, John Wiley and Sons.

[7] Maher, M. L. and Fenves, S. J., (1984), *HI-RISE: An Expert System For The Preliminary Structural Design OF High Rise Buildings*, Technical Report R-85-146, Carnegie Mellon University, Department of Civil Engineering.

Knowledge-Based Interfaces for Optimization Problems [1]

John S. Gero and Bala M. Balachandran
The University of Sydney
New South Wales
Australia

Abstract Optimization is a well understood process in design domains. Designers formulate their design problems as single or multicriteria optimization problems and then select an approximate optimization algorithm to search for the optimal values for the design variables. The formulation and algorithm selection procedures have been considered to be activities which relied on substantive human knowledge. This paper describes a computer system, OPTIMA, which formulates design optimization problems from a pseudo-English description into canonical algebraic expressions. It then recognises the formulation and selects appropriate algorithm(s) for its solution. Finally, it runs the selected algorithm(s) and sends the results back to the original descriptions. Areas of expert knowledge involved in carrying out the above tasks are identified. Such knowledge is explicitly encoded in the system. The basic philosophy and key features of the system are described and are illustrated with examples.

1 Introduction

The early uses of computers in engineering and architecture were for analytical purposes. Later it was realised that certain classes of design decision processes could be represented algorithmically and hence automated. The first of these processes made use of the iterative model of design. This was extended to direct design procedures based on optimization models. In this, however, analysis methods were not supplanted but continued to play a subordinate role in design. More recently the widespread availability of symbolic programming languages coupled with formal knowledge representation techniques has begun to allow us to incorporate specific knowledge into the computer system. This incorporation was not

[1] This lecture draws directly from Balachandran, M. and Gero, J. S., (1987), 'A knowledge-based approach to mathematical design modelling and optimization,' *Engineering Optimization* **12**, 2, 91–115.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 25–55.
© 1992 *Kluwer Academic Publishers.*

previously achievable with such ease because of the non-numeric nature of much of the knowledge.

A variety of automated design decision making systems have been developed based on optimization notions and are widely used to solve many different classes of problems (Gero, 1985). In using such systems the designer formulates the problem as a mathematical model, runs the model through the optimization system, and evaluates the results away from the computer. Here the computer is used only to carry out the optimization process. Such systems normally do not provide flexibility for design modifications. A knowledge-based approach aims to make optimization an easier tool for the designer to use. During the last decade much of applied artificial intelligence research has been directed at developing techniques and tools for knowledge representation. Many different representations have emerged to support the complex task of storing human expertise in a computable form.

The aim of this paper is to present a knowledge-based approach to design decision making processes. A computer system called OPTIMA, which utilises this approach will be demonstrated. The system includes a number of features which are normally difficult to achieve using more conventional approaches. A major part of the paper describes the methodologies, particularly the artificial intelligence concepts, used to introduce these features into the OPTIMA system. The current abilities of the system are illustrated through example problems from two disparate domains.

2 Optimization in Computer-Aided Design

The traditional design process consists of a progressive series of four stages: a feasibility stage; a preliminary design stage; a detail design stage; and a revision stage. The principle of iteration is used at each stage of the process to improve the design. Iteration is also used between the design stages. The designer may proceed through the stages from feasibility to detailed design only to find that the design does not meet all the design requirements. This may necessitate a return to the feasibility stage. In practice, this results in considerable repetitive effort.

Since the introduction of computers much effort has been expended to improve their use in the design process. In the past, designers spent most of their time performing tedious, unchallenging tasks and much less time in developing creative solutions. Since then, numerous computer-based systems have been developed to assist the designers in those tedious and time consuming activities.

2.1 The Conventional CAD Process

The traditional design process outlined above has not been altered by the introduction of the computer. The four stages of design remain unchanged and the principle of iteration is still the method used to improve the design. The role of the computer is limited to tasks such as calculations, analysis, and drafting. In computer-aided design the design process is carried out through a computer model of the design. At the early stages decisions about the resulting design are taken in a heuristic way on the basis of incomplete knowledge about their consequences with respect to design goals. As a result, the design must be analysed and evaluated in the light of the design specification. If the goals are not met, the decisions must be appropriately corrected and the process repeated. The overall aim of the designer is to find a satisfactory solution within the limits of specified constraints. In a sense, design may be considered an optimization task even if the specification does not explicitly call for an 'optimum design', as designers will always try to improve the performance of their designs. One drawback of this approach is that usually it involves excessive computation before a satisfactory or near optimal solution is found.

2.2 The Optimum Design Process

Over the last two decades optimization has been used to improve the efficiency of the design process. 'Optimum design' means achieving the best solution to the design specification given the constraints. An optimum design can be obtained either as a result of iteration or by solving an optimization problem. The iterative approach typifies the method used by designers to improve their designs. The designer obtains information either graphically or numerically and on the basis of this information changes the design. The decision as to what to change and how to change reflects the experience-based insight of the designer. In this approach, the values of the variables are changed or made firm sequentially. On the other hand, in solving an optimization problem, the values of the variables that simultaneously satisfy the requirements and optimize a set of objectives are established. The design task is accomplished through an optimization model. Here it is assumed that the measure of merit function and the complete set of constraints can be expressed formally in terms of the design variables. In solving the optimization model, the search for the best design is carried out mathematically in an organised manner. Thus, it saves time in the design process.

A variety of computer systems have been developed based on optimization concepts and widely used to solve many different design problems (Radford *et al.*, 1985; Gero, 1985). Although computers have been indispensable assistants in the analytical process, the tasks such as problem formulation, algorithm selection and data preparation are still manual tasks.

3 Knowledge-Based Approach

In recent years one of the most promising developments in computer technology has been the work on knowledge-based systems (Hayes-Roth *et al.*, 1983). A knowledge-based system is a computer program which possessess a set of facts about a specific domain of human knowledge and, by manipulating these facts intelligently, is able to solve problems which ordinarily require human intelligence. The development of knowledge-based systems is an attempt to emulate the human process of problem solving. The power of such systems comes from the way their underlying knowledge is represented and manipulated so that the systems can make a 'knowledgeable' contribution to complex problems in a specific domain or field of interest. Many representation schemes have been suggested (Winston, 1984). In this research our interest is centered on frames, predicate logic and production systems or rules.

The conventional CAD systems provide assistance to perform special tasks during the design process. These tasks include one or all of the following: calculations, analysis and graphical display. Knowledge-based design systems aim to aid the designer during the entire design process, relying on comprehensive and complex domain knowledge at each stage. These systems should be able to process different types of information, specifically: graphical, numerical, mathematical, symbolic and textual information. In using such systems, the designer will be able to use graphical input, mathematical formulas, and textual and numerical information as communication media.

An optimization system utilizing a knowledge-based approach aims to take a more active role in design decision processes. In the context of optimum design, it is desirable that such system should contain the following features. It should:

1. accept and represent a designer's description of the problem in an effective and manipulatible form;

2. formulate the problem into a canonical form of an optimization model providing functional relationships for objectives and constraints;

3. recognize the types of design variables and the functional types of objectives and constraints;

4. select an appropriate optimization algorithm and carry out the solution procedure; and

5. provide a simple and semantically rich interface and flexible modeling features.

It is obvious from this list that such a system should operate on information at the same level as the designer. Thus, it must be able to represent in an effective way the designer's description of the problem and to recognize the information and provide functional relationships to objectives and constraints. This involves manipulation of algebraic expressions and recognition of their types. Furthermore, in order to identify the structure of the optimization models and to select appropriate algorithms, it requires knowledge about the structure of canonical optimization models and their solution methods. This form of knowledge needs to be encoded explicitly using knowledge representation techniques.

4 Problem Description and Representation

A typical design problem will have many components, each of which will be related to many other components. The relationships between components are represented and processed during the design process. In an optimization process the design constraints and the objectives specified by the designer must be represented and processed in symbolic forms. Thus, it is important to use suitable representation schemata that will handle these tasks efficiently. In this section we present the design problem description and the representation issues encountered in the development of the system.

4.1 Description of Design Problem

The design description consists of a collection of facts about objects and their properties interpreted as variables, and relationships between objects as sets of statements interpreted as the design constraints and objectives. It is presumed that the designer is able to establish the constraints that govern the design and can express the design goals which can be expressed mathematically. In this work a set of words and symbols have been predefined for use in the problem statement. In the following sections the reserved words and symbols are in bold type.

4.1.1 Variables The variables which form the basis of any description can be treated in terms of object-attribute-value triplets which effectively separate the three concepts associated with variables.

objects	any concept involved in the design domains;
attributes	any property of an object;
value	any value of an attribute

For example consider the following statement:

living_room length = 2 **times** kitchen width

Within this relationship, the objects and attributes involved are as follows:

Object	*Attribute*
living_room	length
kitchen	width

An object attribute can be given a value. Consider the statement:

bathroom length **is** 8.5 ft

In this statement, the object, attribute, and value is as follows:

Object	*Attribute*	*Value*
bathroom	length	8.5

4.1.2 Constraints The design constraints reflect the requirements that must be satisfied by the resulting design.The design constraints are stated by declaring a maximum or minimum value for an attribute of an object or by specifying equality or inequality relationships among object attributes. The following are some examples:

maximum living_room area **is** 300.0 sq ft
sum_of living_room width **and** kitchen width = house width

The first example declares that the maximum value of living_room area is 300.0 sqft. The second example establishes an equality relationship among the attributes living_room width, kitchen width and house width.

4.1.3 Objectives The design goals or objectives are set up by the designer and are optimized during the solution process. The design objectives are stated as maximize or minimize an object attribute or a function of object attributes. Some examples are shown below:

minimize house cost
maximize house area

The first objective is to minimize the 'cost' attribute of the object 'house' and the second objective is to maximize the 'area' attribute of the object 'house.'

4.2 Representation of Design Information

As discussed in the previous section, most of the design information may be stated using the object-attribute-value concept. A typical design problem may involve several objects with their attributes and attribute values. Although several distinct mechanisms may be used to represent this kind of information, the one which handles this task efficiently is the frame-based representation (Minsky, 1975). Procedural knowledge is required to carry out certain tasks in the modeling process. Frame representation allow procedures to be attached to a particular attribute of an object. This representation technique has been used previously in design problems (MacCallum *et al.*, 1985; Maher and Fenves, 1985). We discuss frames and some of their appropriate features in the following sections.

4.2.1 Frames and their features In simple terms, a frame can be viewed as a stereotypical representation of any object concept. It is typically represented as a data structure whose name is that of the concept. A frame can have any number of slots, each of which stands for an attribute of the object concept of interest; it can hold values of the attribute and procedures that can be invoked under certain conditions on the attribute value. A frame can have various types of links to other frames. Examples of commonly used links are a-kind-of, is-a, and part-of. These links allow unrestricted inheritance of attributes and attribute values.

Frame representation provides three key features which are important and useful. First, they allow explicit representation of objects, attributes, and default values. The notion of default value is very important in frame theory. The default value of an attribute of a concept can be normally used when there is no value explicitly given for that attribute. Second, procedures can be attached to the slots in the frames and can be executed automatically according to some specification. Examples for such specifications are if-needed, if-added, and if-removed. These procedures are automatically invoked when the slot's values are accessed or stored or deleted. Such automatic procedure invocations are called demons, or, sometimes, triggers. An important point is that demons allow for explicit representation of procedural knowledge and of the context of their use because demons are attached to parts of frames. Finally, frames can be related in a conceptual hierarchy; attributes, values, and demons can be inherited from higher up in a hierarchy. In other words, it is possible to define classes of data items that share attributes, procedures, and default values. Normally inheritance works through the a-kind-of and/or is-a slot.

For example, Fig. 1 shows a simple frame structure for a room. Whenever an instance of a room, say room-1, is created it will inherit all the attributes of the room frame. The following sections illustrate the representation of objects and their relationships, and design constraints and objectives in more detail.

4.2.2 Representing Variables A design problem consists of one or more variables treated as objects, attributes, and values. There are relationships among objects and among attributes of objects. For example consider the following example:

kitchen **is_a_kind_of** rectangle
kitchen width is 10 ft.

A frame which represents the object kitchen is shown below:

name:	kitchen	
a_kind_of:	rectangle	
width:	value :	10
	unit :	ft

```
name   :  room
slots  :  a_kind_of
                     value :    rectangle
          shape
                     default :  rectangle
          number_of_walls
                     default :  4
          area
                     if_needed :  calculate_area
          cost
                     if_exp_needed :  express_cost
```

Figure 1. *Example of a frame representation for a room.*

Using this concept, all the objects, and properties and relationships between them can be represented as a network of frames, each of which will have some form of links to other frames.

4.2.3 Representing Constraints The representation and processing of constraints form an integral part of the design process. In mathematical design modeling it should be possible to express all the design constraints in terms of the design variables. The constraints are expressed by declaring a maximum or minimum value for any object attribute or by specifying equality or inequality relationships among object attributes. Frames are used to represent the design constraints. These constraint frames are constructed by making an instance of the 'constraint' frame, shown below:

```
name :  constraint
slots : lhs :
        rhs :
        type :
```

The 'lhs' and 'rhs' slots represent the left hand side and right hand side expressions of a constraint and the 'type' slot represents the relation involved between those expressions. For example, consider a constraint used in steel beam design:

beam deflection <= 1 / 360 **times** beam span

This constraint is represented as follows:

name :	constraint−1	
slots :	lhs :	(beam deflection)
	rhs :	1 / 360 ∗ (beam span)
	type :	less_than_or_equal_to

4.2.4 Representing Objectives The optimal design process is concerned with producing designs which satisfy the design constraints and optimize one or more objectives. The design objectives are expressed as maximize or minimize some aspects of the design. The general form in which a design objective may be stated as follows:

maximize Z

or

minimize Z

where Z is an object attribute or an expression of one or more object attributes. The design objectives are represented by making an instance of the 'objective' frame shown below:

name:	objective
slots:	objective-function :
	optimality-criteria :

For example, consider the following statement:

minimize beam weight

This statement indicates that the objective is to minimize the 'weight' attribute of the object 'beam' and is represented as shown below:

name:	objective-1	
	objective-function :	(beam weight)
	optimality-criteria :	minimize

5 Problem Formulation and Recognition

In design optimization the major task involved is the formulation of the design problem for mathematical programming; in other words building the mathematical model of the design problem. In this section we briefly outline the basic tasks involved in constructing the mathematical model of the design problem and discuss their required features and their implementation for automating this process. We also describe the knowledge required to recognize various algebraic expression types and illustrate how it can be encoded into the computer system. Production systems are employed to carry out algebraic simplification and problem identification processes. We shall commence with a brief discussion of them.

5.1 Problem Solving Using Production Systems

Production systems were first proposed by Post (1943) as a general computational mechanism but their use today stems from the work of Newell and Simon (1972). Simply speaking a production system consists of three parts:

1. a rule base composed of a set of production rules;

2. a special data structure which is sometimes called the context; and

3. an interpreter, which controls the system's activity.

A production rule is a statement cast in the form 'If this condition holds, then this action is appropriate.' The if part of the production rule states a set of conditions in which the rule is applicable. The action or then part of the production rule states the appropriate conclusions to make when the conditions are satisfied. Production systems permit the representation of knowledge in a highly uniform and modular way. Knowledge in production rules is both accessible and relatively easy to modify. A further advantage of the production system formalism is the ease with which one can express certain kinds of knowledge. In particular, statements about what to do in predetermined situations are naturally encoded into production rules. Furthermore, it is these kinds of statements that are most frequently used by human experts to explain how they do their jobs (Gero, 1983).

There are two strategies employed in problem solving using production systems, namely, forward chaining and backward chaining. The forward chaining process starts with a collection of facts and tries all available rules over and over, adding new facts as it goes, until no rule is applicable. In backward chaining a problem solver starts with an unsubstantiated hypothesis and tries to prove it. The strategy involves finding rules that demonstrate the hypothesis and then verifying the facts that enable the rule to work.

These systems have two characteristic features which are particularly noteworthy. First, existing knowledge bases can be refined, and new knowledge added to increase their performance. Second, systems are able to explain their reasoning, making their logic practically transparent. Today, rule-based systems are used in many applications.

5.2 Design Optimization Formulation

Formulation of a design optimization problem consists in constructing a mathematical model that describes the behaviour of a physical system encompassing the problem area. This model must closely approximate the actual behaviour of the system for the solution obtained to be adequate and useful. At the stage of formulating the optimization model the designer has to decide which quantities are treated as variables and which are taken as fixed. The quantities whose values are fixed are called design parameters and the quantities for which values are chosen are called decision variables or design variables. Mathematical relations between the design variables and the parameters constitute a design optimization model. Formally speaking the basic tasks involved in constructing the design optimization model can be described as follows:

1. identification of the parameters and variables involved in the design problem;
2. provision of functional relationships in terms of the variables that state the objectives; and
3. provision of functional relationships in terms of the variables that represent the design constraints.

In the following sections we discuss the key issues involved in the above activities.

5.2.1 Identifying Design Parameters and Variables The design description contains knowledge of design variables (e.g. room length, room width, etc.) and of their relationships. As we discussed previously this knowledge is represented as a network in which knowledge about a variable is encapsulated in a frame. The information contained in such a frame mainly consists of a current value, units, and relationships. The current value represents the value given by the designer or the latest value calculated for a variable, the units slot contains the units in which the value has been measured, and the relationship slot provides the dependency of a variable upon some other variables. Each relationship, in turn, contains a list of dependent variables. In finding the design parameters the system starts with a set of variables for which fixed values have been provided and tries to find values for unknown variables using appropriate mathematical equalities. For example consider the mathematical equality given below:

beam overall_depth = **sum_of** beam web_depth **and**
2 times beam flange_thickness

This equality is represented as below:

name : equality-1
slots : lhs : (beam overall_depth)
rhs : (beam web_depth) + 2 * (beam flange_thickness)

The above frame representation is transformed into a mathematical equality as below:

(beam overall_depth) = (beam web_depth)
+ 2 * (beam flange_thickness)

It is then identified that there are three variables involved in this equality relationship, namely beam overall_depth, beam web_depth and beam flange_thickness. If the values for any two of these variables are known the value for the third variable is found by solving this equation. All available mathematical equalities are tried iteratively, updating the parameters as they are found until no further parameters can be found. At the end of this process the system will be able to know which variables are the design parameters and which variables are to be treated as design variables. The mathematical equalities which have not been used to determine the design parameters are treated as design constraints.

5.2.2 Formulating Constraints The design constraints simply describe dependancies among design variables and parameters and are represented as mathematical inequalities or equalities. As we illustrated in Section 4.2, each constraint is encoded as a frame with three slots, namely left-hand-side part, right-hand-side part and relationship type. In formulating a constraint into mathematical form, the system starts with the description of that constraint obtained from the appropriate frame representation. We will use an example to illustrate the basic process involved in transfoming the constraint description into canonical form. Consider the following constraint used in the beam design:

beam deflection <= 1/360 **times** beam span

The frame representing this constraint is shown below:

name : constraint-1
slots : lhs : (beam deflection)
rhs : 1/360 * (beam span)
type : less_than_or_equal_to

Initially the above frame representation is transformed into mathematical inequality or equality form accordingly. Then it is rearranged into one of the following forms.

lhs − rhs <= 0
lhs − rhs => 0
lhs − rhs = 0

In the above example the result is

$$(\text{beam deflection}) - 1/360 * (\text{beam span}) <= 0$$

At this stage the variables involved in the constraint are determined by scanning the left-hand-side expression of the constraint. It then introduces known values to those dependent variables. For those unknown dependent variables, the system uses the knowledge of dependence of one variable upon some other variables. A variable may have a functional relationship with some other variables. If such relationship exists for a variable, it is obtained directly from the frame representing that variable or by inheriting from another related frame. The functional relationship is then substituted for the variable in the constraint. This procedure is repeated for every newly introduced variable. The substitution process terminates when no further functional relationship is found to any of the variables involved. The variables which remain in the constraint are a subset of the design variables and are named with algebraic symbols (e.g. x_1, x_2, etc.). After replacing the variables with the appropriate algebraic symbols, the constraint is algebraically simplified to reduce it to a canonical form.

In our example the variables which form the constraint are beam deflection and beam span. Let us assume that the function describing the deflection of the beam is stated as follows:

$$\text{beam deflection} = 5 * w * L\text{\textasciicircum}4 \;/\; (384 * E * I)$$

where

L = beam span
w = beam u.d.l.
E = beam modulus of elasticity
I = beam second moment of area

Further we assume that the following information has been provided.

$L = 10$ m, $E = 200 * 10^6$ kPa,
$w = 600$ kN/m and the beam is an I-beam.

In this case the functional expression for I will be inherited from the frame representing the object 'I-beam,' i.e.

$$I = (B * (D\text{\textasciicircum}3 - d\text{\textasciicircum}3) + t * d\text{\textasciicircum}3) \;/\; 12$$

where

B = beam flange width
D = beam overall depth
d = beam web depth
t = beam web thickness

After the substitution process, the constraint is expressed in the following form:

$$5 * 600 * 10\text{\textasciicircum}4 \;/\; (384 * 200 * 10^{6} * (B * (D\text{\textasciicircum}3 - d\text{\textasciicircum}3) \\ + t * d\text{\textasciicircum}3) \;/\; 12) - 10 <= 0$$

At this stage algebraic symbols are introduced to the variables and the constraint becomes:

$$5 * 600 * 10\text{\textasciicircum}4 \;/\; (384 * 200 * 10\text{\textasciicircum}6 * (x_1 * (x_2\text{\textasciicircum}3 - x_3\text{\textasciicircum}3) \\ + x_4 * x_3\text{\textasciicircum}3) \;/\; 12) - 10 <= 0$$

where

x_1 = B = beam flange width
x_2 = D = beam overall depth
x_3 = d = beam web depth
x_4 = t = beam web thickness

The above form of the constraint is then algebraically simplified to reduce it to a canonical form as follows:

$$3200 * x_1 * x_2\text{\textasciicircum}3 - 3200 * x_1 * x_3\text{\textasciicircum}3 + 3200 * x_4 * x_3\text{\textasciicircum}3 => 1.5$$

5.2.3 Formulating Objectives In the optimization model the design objectives must be expressed as computable functions of the design variables. The design objectives are stated as maximize or minimize one or more aspects of the design. For example consider the following statement:

minimize beam volume

This statement indicates that the 'volume' attribute of the object 'beam' is to be minimized. Initially a functional relationship describing the volume of the beam is obtained. The function describing the volume of the beam is obtained from the appropriate frame.

minimize (beam span) * (beam sectional-area)

Given that the beam is an I-beam, it is obtained that

beam sectional-area = 2 * B * T + d * t

where

B = beam flange width
T = beam flange thickness
d = beam web depth
t = beam web thickness

As illustrated in the previous section the substitution process is applied to each variable. After the substitution process the objective function is expressed in terms of the design variables.

minimize $10 * (2 * B * T + d * t)$

Now algebraic symbols for the variables are introduced and the objective function is algebraically simplified to reduce it to a canonical form.

minimize $10 * (2 * x_1 * x_5 + x_3 * x_4)$

i.e.

minimize $20 * x_1 * x_5 + 10 * x_3 * x_4$

where

$x_1 = B =$ beam flange width
$x_5 = T =$ beam flange thickness
$x_3 = d =$ beam web depth
$x_4 = t =$ beam web thickness

5.2.4 Algebraic Manipulation The major mathematical operations involved in the processes discussed in the above sections are substitution, simplification, and equation solving. A system which allows this type of symbolic and numerical manipulation is MACSYMA (Martin and Fateman, 1971), a large interactive computer system designed to assist scientists and engineers in solving mathematical problems. However, in the work described here an algebraic manipulation package has been developed and implemented in Franz LISP (Wilensky, 1984). In this implementation an algebraic expression is represented as a list structure consisting of the operators and operands involved in the expression. The knowledge involved in the algebraic simplification process is encoded as a set of production rules. The form of these rules are similar to the rules normally used in design grammars (Coyne and Gero, 1985). The basic form of a rule is 'IF this pattern matches THEN execute this action.' That is, a rule consists of a pattern part and an action part. An example rule is shown below:

if (+P (restrict L_1 is_list) * (restrict L_2 is_list) +R)
then execute procedure-1

Where '+P' means that the variable P can match with zero or more number of items of any type, where as '(restrict L_1 is_list)' indicates that the variable L_1 can only match with an item in the form of a list. This rule indicates that if the pattern part matches with an expression apply the 'procedure-1' to simplify it. The procedures attached to the rules carry out the appropriate algebraic manipulation and are written as LISP functions. The pattern part of a rule is represented as a list which consists of variables and some key words. For instance the symbol '+' and the keyword 'restrict' are used to indicate the type of items to which those variables should match. Once a successful match is made against the pattern part of a rule, the variables involved in that rule become instantiated to appropraite values. Let us consider the following expression represented in the form of a list structure:

$$(X_1 + (X_2 + X_3) * (Y_1 + Y_2) - Y_3)$$

When applying the above rule an attempt is made to match the pattern part of the rule with this expression. As this match succeeds the variables in the pattern part become instantiated as follows:

$$\begin{aligned} P\ &= (X_1 +) \\ L_1 &= (X_2 + X_3) \\ L_2 &= (Y_1 + Y_2\ ) \\ R\ &= (-\ Y_3\ ) \end{aligned}$$

At this stage the function 'procedure-1' is invoked to simplify the expression. As a result of this the expression is transformed into a more reduced form as follows:

$$(X_1 + X_2 * Y_1 + X_2 * Y_2 + X_3 * Y_1 + X_3 * Y_2 - Y_3\ )$$

In simplifying an expression, all available rules are tried iteratively, reducing the expression as they are applied, until no rule is applicable.

5.2.5 The Canonical Forms of Optimization Models In this section we illustrate the canonical forms of the optimization models. A single objective design problem is formulated into the canonical form of:

$$\begin{array}{lll} \text{Maximize} & Z(X) & \\ \text{subject to} & g_i(X) \le G_i & i = 1, 2, \ldots, m \\ & x_j \ge 0 & j = 1, 2, \ldots, n \end{array}$$

where

$Z(X)$ is the objective function

$X = (x_1, x_2, \ldots, x_n)$ is an n-component vector consisting of design variables

$g_i(X) \le G_i$ are m constraint functions

In complex design optimization problems there often exist several noncommensurable criteria which must be considered. This situation is formulated as a multicriteria optimization problem (also called multiple objective or Pareto optimization) in which the designer's goal is to minimize and/or maximize not a single objective function but several functions simultaneously (Cohon, 1978). The general multicriteria optimization problem with n decision variables, m constraints, and p criteria is formulated into the canonical form of:

$$\begin{array}{lll} \text{Maximize} & Z(X) = [Z_1(X), Z_2(X), \ldots, Z_p(X)] & \\ \text{subject to} & g_i(X) \leq G_i & i = 1, 2, \ldots, m \\ & x_j \geq 0 & j = 1, 2, \ldots, n \end{array}$$

where

$Z(X)$ is the multicriteria objective function

$Z_k(X)$ is the k-th criterion of the p individual criteria

$X = (x_1, x_2, \ldots, x_n)$ is an n-component vector consisting of design variables

$g_i(X) \leq G_i$ are m constraint functions

5.3 Recognizing Algebraic Expression Types

The canonical form of the optimization model represents the objectives and the constraints of the design problem as mathematical expressions in terms of the design variables. In order to have the structure of the optimization model identified by a computer system, it needs to be able to recognize the variables and the algebraic relationships between the variables in the constraints and objective functions. In this work, a program has been developed in PROLOG (Clocksin and Mellish, 1981) that incorporates the necessary knowledge to recognize the types of various algebraic expressions. One of the significant features of PROLOG is its very powerful pattern matching facilities.

An algebraic expression, in its reduced form, is the sum of positive and/or negative terms, each of which is expressed in terms of the variables. The type of an algebraic expression is determined based on the types of the terms involved in that expression. Common types of algebraic terms are linear, quadratic, posynomial, and nonlinear. An algebraic expression is linear if all of its terms are of the linear form. Thus, in order to decide on the type of an algebraic expression the type of every term is to be analysed. This process can be simply encoded using the feature of recursion which is a popular and powerful technique in symbolic computation. In using PROLOG we take advantage of its automatic inference mechanism. Consider the following recursive rules:

```
expression(A + B, linear) :- term(B, linear), expression(A, linear).
expression(A − B, linear) :- term(B, linear), expression(A, linear).
```

where 'expression(E, T)' means that T is the type of expression E. The first rule simply states that an expression given in the form A + B, in which B represents its last term is linear if both B and the rest of the expression A are linear. The second rule deals with the case where the last term is a negative term. These two rules are only used when the expression matches with either A + B or A − B. Hence we need to define further rules to handle expressions with single term. The following rules are added for this purpose.

```
expression(A, linear) :- term(A, linear).
expression(−A, linear) :- term(A, linear).
```

These rules simply define that an expression which has a single term, is linear if its term is of the linear form. Now we need to define the necessary rules to determine whether a given term is linear. An algebraic term is linear if it is represented by a symbol or by a product of a symbol and a number. The following rules encode these conditions.

```
term(A, linear) :- symbol(A).
term(A * B, linear) :- number(A) , symbol(B, linear).
```

Using the above set of logical statements the system is able to identify any kind of linear expression. Similarly the necessary rules have been encoded to identify other types of expressions such as quadratic, posynomial and nonlinear.

6 The OPTIMA System

A prototype system, called OPTIMA, which incorporates the knowledge described in the previous sections, was developed and implemented as a general purpose design assistant for computer-aided design situations. The major issues considered in the development of this system are the following:

1. the system should provide simple and easy ways for designers to describe their problems;

2. the system should employ an efficient representation of the design problem;

3. the system should allow the designer to modify models, adding new design parameters or relationships, during the design process;

4. the system should have the ability to represent and use the designer's expertise with respect to optimum design processes;

5. the system should have the ability to formulate the problem mathematically, providing functional relationships for objectives and constraints;

6. the system should have the ability to recognize the type of functions which represent the objectives and the constraints; and

7. the system should have the ability to select an appropriate algorithm and carry out the solution procedure (Balachandran and Gero, 1987).

The OPTIMA system has been implemented on SUN Microsystems workstations. The domain specific knowledge is all represented as frames or production rules and encoded in LISP as are the inference engines. The pattern matching knowledge used to recognize variables and their relationships is encoded in PROLOG. The optimization algorithms are all encoded in C. Figure 2 shows the data flows in the OPTIMA system. The kernel of the system is the communication controller which allows the three main components of the system, namely problem formulator, problem recognizer, and problem solver to communicate with each other.

In the following sections two examples will be presented to illustrate the various aspects of the system. One example deals with optimum dimensioning of architectural floor plans and the other is concerned with optimum design of beams.

6.1 Example 1: The Floor Plan Problem

Here the formulation and solution to the floor plan problem originally formulated by Mitchell *et al.* (1976) will be illustrated. This problem concerns the optimal dimensioning of small rectangular floor plans for which a topology has been separately identified (Fig. 3). The maximum and minimum area requirements of various rooms are specified as design constraints.

The following is the listing of the problem description given to the system. The reserved words and symbols used by the system are shown in bold.

house length **=** **sum_of** living_room length **and** bathroom length
  and bedroom2 length
house width **=** **sum_of** living_room width **and** kitchen width
living_room width **=** **sum_of** bathroom width **and** hall width
bathroom length **=** hall length
bedroom1 width **=** kitchen width
bedroom2 length **=** bedroom3 length
house **and** living_room **and** kitchen **and** bathroom **and** bedroom1
  and bedroom2 **and** bedroom3 **are_kind_of** rectangle
sum_of living_room length **and** hall length **=** **sum_of** kitchen length
  and bedroom1 length

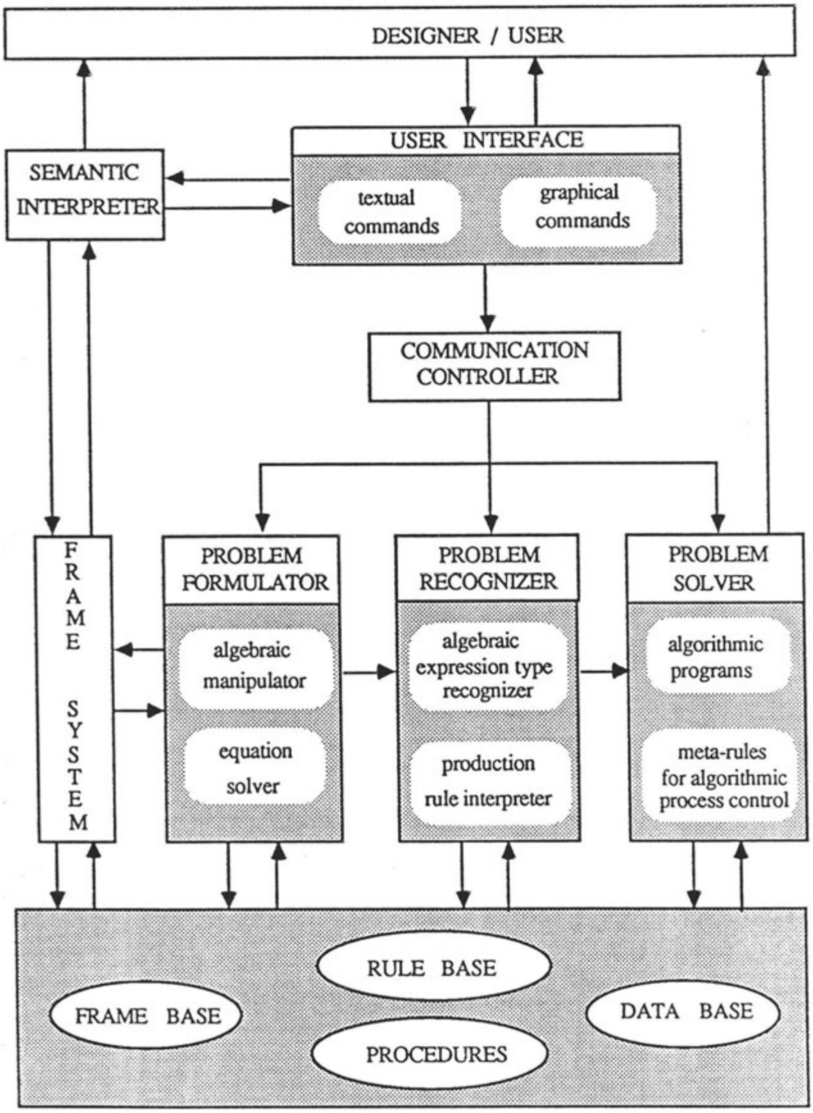


Figure 2. *The data flows in the OPTIMA system.*

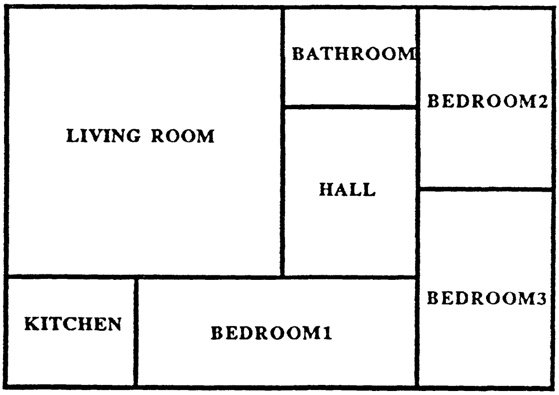
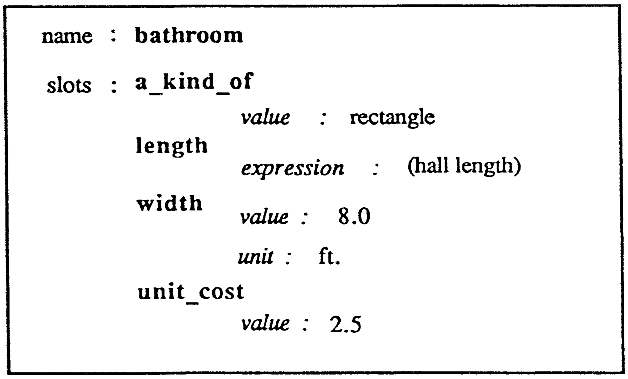


Figure 3. *Dimensionless representation of a house layout.*

```
sum_of living_room width and kitchen width = sum_of bedroom2 width
    and bedroom3 width
house cost = sum_of living_room cost and kitchen cost
    and bathroom cost and hall cost and bedroom1 cost
    and bedroom2 cost and bedroom3 cost
bathroom width is 8 ft.
hall width is 6 ft.
kitchen width is 10 ft.
bedroom2 width is 11 ft.
kitchen unit_cost is 2.0
bedroom2 unit_cost is 1.0
maximum living_room area is 300 sq.ft.
minimum living_room area is 150 sq.ft.
maximum kitchen area is 120 sq.ft.
minimum kitchen area is 50 sq.ft.
maximum bathroom area is 65 sq.ft.
minimum bathroom area is 45 sq.ft.
maximum hall area is 72 sq.ft.
minimum bedroom1 area is 100 sq.ft.
maximum bedroom1 area is 180 sq.ft.
minimum bedroom3 area is 100 sq.ft.
maximize house area
minimize house cost
```

Figures 4, 5 and 6 present frames showing how the objects, objectives, and constraints of this problem are represented in the system.



```
name  :  bathroom

slots :  a_kind_of
                     value    :  rectangle
         length
                     expression   :  (hall length)
         width
                     value :   8.0
                     unit :  ft.
         unit_cost
                     value :  2.5
```

Figure 4. *Frame representing room data.*

Figures 7, 8 and 9 show screen dumps of a session with the system during which the floor plan problem was solved with two conflicting criteria, namely maximize house area and minimize house cost. The problem was formulated as a canonical multicriteria optimization model by the system. Recognition of the types of variables, constraints, and objectives, etc., was performed by the logic programming system. Identification of the problem model and selection of an appropriate algorithm were carried out using the rules given in Balachandran and Gero (1987).

6.2 Example 2: The Beam Design Problem

This problem concerns the optimum design of a simply supported, single span, wide-flange beam presented in Fig. 10. The design goal is to minimize the weight of the beam.

The following is the listing of the problem description given to the system.

```
beam span is 7.5 m
beam uniformly_distributed_load is 60.0 kN /m
```

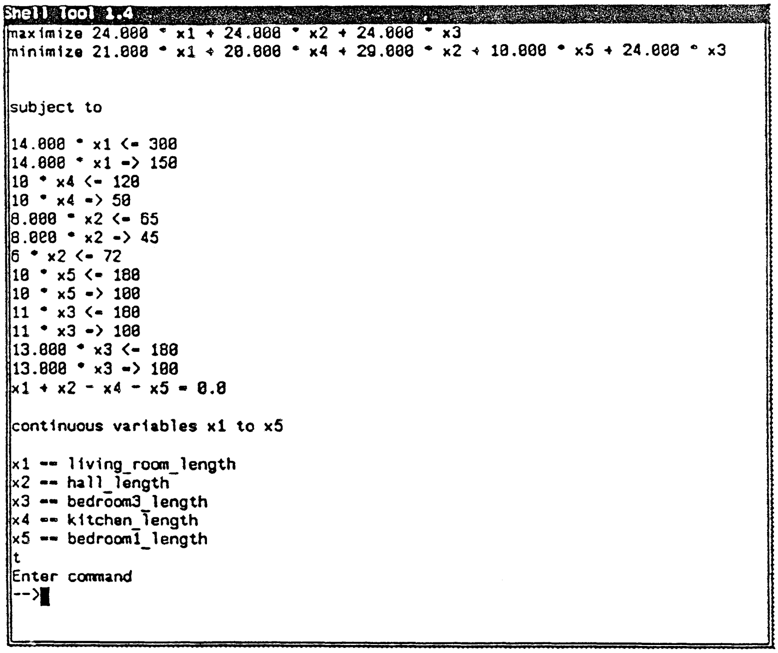
```
name  :   objective-1
slots :   objective-funtion
                      expression  :  (house area)
          optimality-criteria
                      value     :   maximize
```

Figure 5. *Frame representing a design objective.*

```
name  :   constraint-1
slots  :   lhs
              expression  :   (living_room  area)
           rhs
              value   :    300.0
              unit    :    sq. ft.
           type
              value   :   less_than_or_equal_to
```

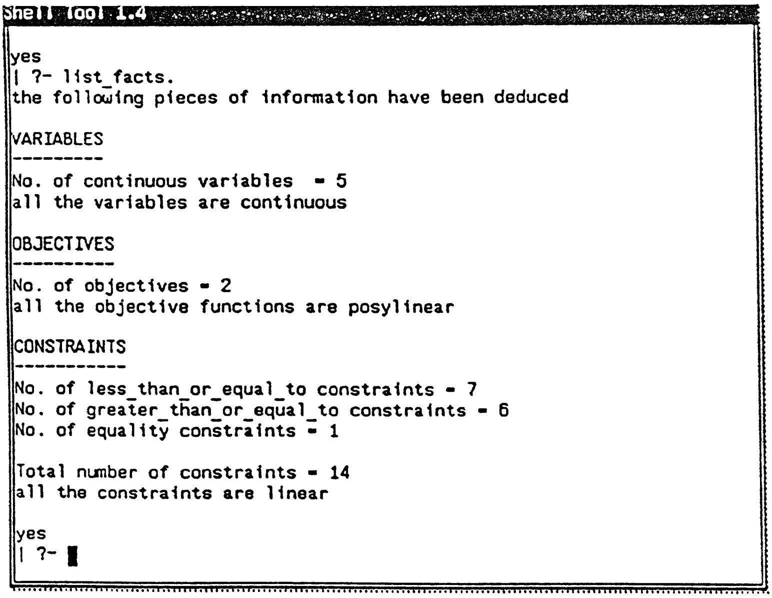
```
name  :   constraint-5
slots  :   lhs
              expression  :   (living_room length)
                                        +
                                   (hall length)
           rhs
              expression  :   (kitchen length)
                                        +
                                   (bedroom1 length)
           type
              value  :    equal_to
```

Figure 6. *Figure 6 Design constraint frames.*



```
Shell Tool 1.4
maximize 24.000 * x1 + 24.000 * x2 + 24.000 * x3
minimize 21.000 * x1 + 20.000 * x4 + 29.000 * x2 + 10.000 * x5 + 24.000 * x3


subject to

14.000 * x1 <- 300
14.000 * x1 -> 150
10 * x4 <- 120
10 * x4 -> 50
8.000 * x2 <- 65
8.000 * x2 -> 45
6 * x2 <- 72
10 * x5 <- 180
10 * x5 -> 100
11 * x3 <- 180
11 * x3 -> 100
13.000 * x3 <- 180
13.000 * x3 -> 100
x1 + x2 - x4 - x5 = 0.0

continuous variables x1 to x5

x1 -- living_room_length
x2 -- hall_length
x3 -- bedroom3_length
x4 -- kitchen_length
x5 -- bedroom1_length
t
Enter command
-->
```

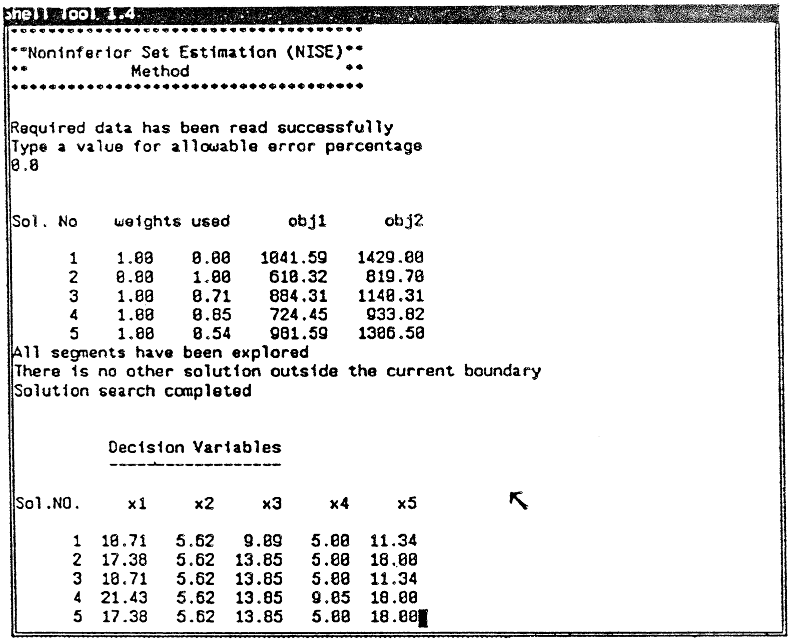
Figure 7. *The canonical optimization model constructed by the system for the floor plan problem.*



```
yes
| ?- list_facts.
the following pieces of information have been deduced

VARIABLES
---------
No. of continuous variables  = 5
all the variables are continuous

OBJECTIVES
----------
No. of objectives = 2
all the objective functions are posylinear

CONSTRAINTS
-----------
No. of less_than_or_equal_to constraints = 7
No. of greater_than_or_equal_to constraints = 6
No. of equality constraints = 1

Total number of constraints = 14
all the constraints are linear

yes
| ?-
```

Figure 8. *The information generated by the system when attempting to recognize the algebraic model of the floor plan problem.*

beam **is_a_kind_of** I_section
beam deflection = 5 * w * L^4 / 384 * E * I
L = beam span
w = beam uniformly_distributed_load
E = beam elastic_modulus
I = beam second_moment_of_area
beam bending_stress = w * L^2 * D / 16 * I
beam shear_stress = w * L / 2 * d * t
d = beam web_depth
t = beam web_thickness
beam **is_a_kind_of** steel
maximum beam bending_stress = 0.660 **times** beam yield_stress
maximum beam shear_stress = 0.370 **times** beam yield_stress
maximum beam deflection = 1/360 **times** beam span
sum_of beam web_depth **and** 2 **times** beam beam flange_thickness =



```
************************************
**Noninferior Set Estimation (NISE)**
**            Method               **
************************************


Required data has been read successfully
Type a value for allowable error percentage
0.0


Sol. No    weights used        obj1       obj2

      1     1.00     0.00   1041.59    1429.00
      2     0.00     1.00    610.32     819.70
      3     1.00     0.71    884.31    1140.31
      4     1.00     0.85    724.45     933.82
      5     1.00     0.54    901.59    1306.50
All segments have been explored
There is no other solution outside the current boundary
Solution search completed


          Decision Variables
          ------------------

Sol.NO.      x1       x2       x3       x4       x5

      1   10.71     5.62     9.09     5.00    11.34
      2   17.38     5.62    13.85     5.00    10.00
      3   10.71     5.62    13.85     5.00    11.34
      4   21.43     5.62    13.85     9.05    10.00
      5   17.38     5.62    13.85     5.00    10.00
```

Figure 9. *The set of Pareto optimal solutions generated by the noninferior set estimation method for the floor plan problem.*

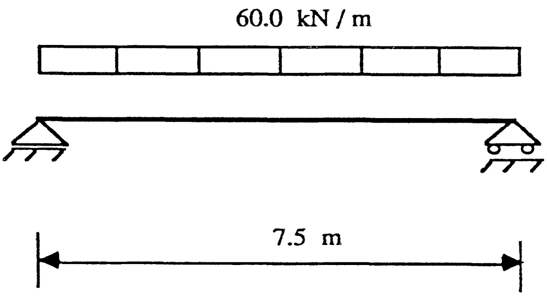
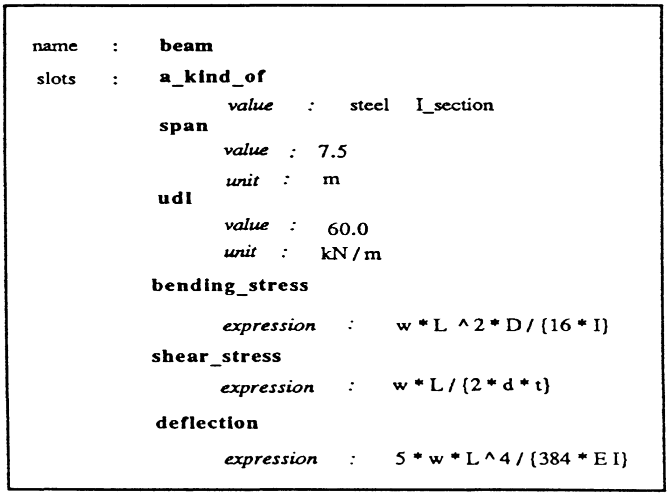


Figure 10. *A simply supported beam with uniformly distributed load.*

```
     beam overall_depth
quotient_of beam web_depth and beam web_thickness <= 180
quotient_of beam web_depth and beam web_thickness <=
     816/0.37 * Fy)^0.5
Fy = beam yield_stress
1 / 15 times beam span <= beam overall_depth <= 1/8 times beam span
1 / 5 times beam overall_depth <= beam flange_thickness <=
     1 / 3 times beam overall_depth
0.003 m <= beam flange_thickness <= 0.100 m
0.003 m <= beam web_thickness <= 0.100 m
minimize beam weight
```

Figure 11 shows how the description of the beam is represented symbolically in a frame prior to further computation.

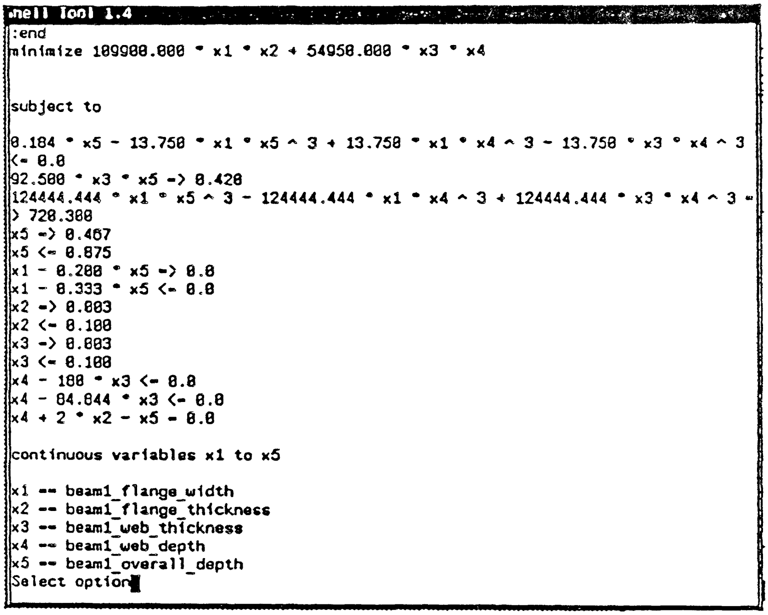


```
name   :   beam
slots  :   a_kind_of
                 value      :   steel   I_section
           span
                 value   :  7.5
                 unit    :  m
           udl
                 value   :  60.0
                 unit    :  kN / m
           bending_stress
                 expression   :   w * L ^2 * D / (16 * I)
           shear_stress
                 expression   :   w * L / (2 * d * t)
           deflection
                 expression   :   5 * w * L ^ 4 / (384 * E I)
```

Figure 11. *Frame showing description of the beam.*

7 Discussion

The central issue considered in this paper is the potential of introducing a knowledge-based systems approach within the optimal design decision processes. We have demonstrated the technical feasibility of developing a computer system that incorporates a variety of human expertise to assist in the optimum design process. The OPTIMA system which was developed based on these notions is considerably more



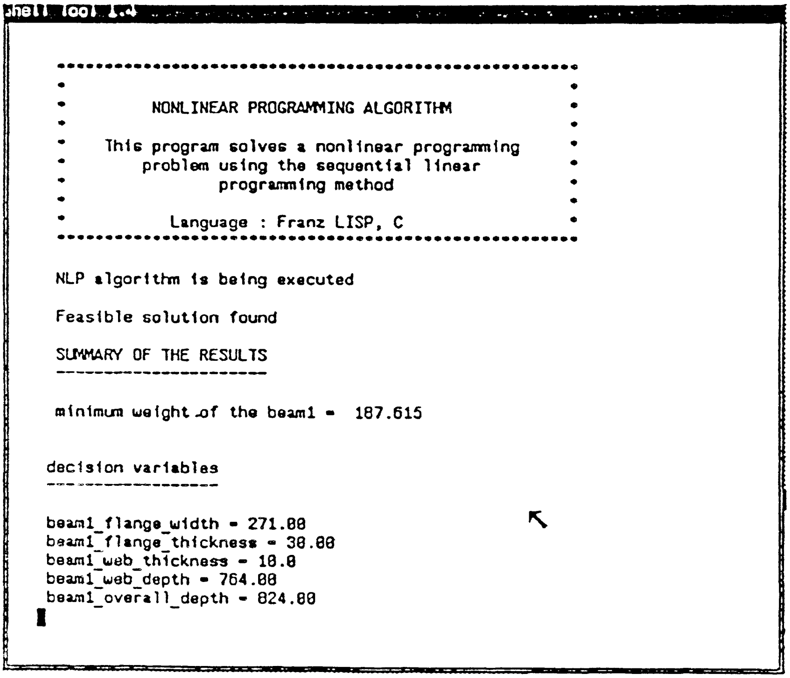
```
:end
minimize 109900.000 * x1 * x2 + 54950.000 * x3 * x4


subject to

0.184 * x5 - 13.750 * x1 * x5 ^ 3 + 13.750 * x1 * x4 ^ 3 - 13.750 * x3 * x4 ^ 3
<= 0.0
92.500 * x3 * x5 => 0.420
124444.444 * x1 * x5 ^ 3 - 124444.444 * x1 * x4 ^ 3 + 124444.444 * x3 * x4 ^ 3 =
> 720.300
x5 => 0.467
x5 <= 0.875
x1 - 0.200 * x5 => 0.0
x1 - 0.333 * x5 <= 0.0
x2 => 0.003
x2 <= 0.100
x3 => 0.003
x3 <= 0.100
x4 - 100 * x3 <= 0.0
x4 - 84.044 * x3 <= 0.0
x4 + 2 * x2 - x5 = 0.0

continuous variables x1 to x5

x1 -- beam1_flange_width
x2 -- beam1_flange_thickness
x3 -- beam1_web_thickness
x4 -- beam1_web_depth
x5 -- beam1_overall_depth
Select option
```

Figure 12. *The mathematical model constructed by the system for the beam design problem.*

versatile than existing optimization systems, and has a number of features which are difficult to achieve using conventional approaches. The system demonstrates the potential of knowledge-based systems in computer-aided design. The last section illustrated the OPTIMA system solving a floor plan dimensioning problem and a beam sizing problem. In summary it is worth emphasising some important characteristics of the system.

1. It is general purpose and can be applied to a large variety of problem domains.
2. The system provides a simple and semantically rich interface and flexible modeling features.
3. The information about a design problem is effectively represented and manipulated.
4. The system represents and uses a variety of human expertise with respect to optimium design process.

Knowledge-based systems can provide a effective method of automating much of the designer's work. The key is that such systems contain explicit knowledge



```
 ••••••••••••••••••••••••••••••••••••••••••••••••••••••
 •                                                    •
 •          NONLINEAR PROGRAMMING ALGORITHM           •
 •                                                    •
 •     This program solves a nonlinear programming    •
 •         problem using the sequential linear        •
 •                programming method                  •
 •                                                    •
 •            Language : Franz LISP, C                •
 ••••••••••••••••••••••••••••••••••••••••••••••••••••••

 NLP algorithm is being executed

 Feasible solution found

 SUMMARY OF THE RESULTS
 ----------------------

 minimum weight of the beam1 =  187.615


decision variables
------------------

beam1_flange_width = 271.00
beam1_flange_thickness = 30.00
beam1_web_thickness = 10.0
beam1_web_depth = 764.00
beam1_overall_depth = 824.00
```

Figure 13. *The final results of the optimum design with the decisions listed.*

and are able to manipulate that knowledge and reason with it. Although there are several knowledge representation techniques, each provides advantages in specific domains. In this work we have employed three different methodologies, namely, frames, predicate logic and production systems according to their key characteristics. The frame representation is more suitable for domains where complex structural descriptions are necessary to adequately describe the problem domain. Production systems capture in a manageable representation schema a certain type of problem-solving knowledge, particularly knowledge about what to do in a specific situation. Logic based systems are preferable in problem domains which can be readily axiomatized. The inadequacies of one representation technique can often be effectively handled by an alternate technique. Recently there has been a great deal of interest in developing a hybrid representation facilities by integrating two or more different methodologies (Fikes and Kehler, 1985). Such integrated systems can have the combined advantages of individual representation techniques.

Knowledge engineering provides tools and techniques which expand the role of the computer in design. This work has demonstrated that knowledge-based computer programs can capture and use designer's knowledge explicitly in a more useful way than is possible with traditional programming tools. Knowledge-based systems generally offer alternate approaches to design decision making (Gero *et al.*, 1985).

Optimization has a useful and valid place in a decision making paradigm of design, however, knowledge-based systems such as OPTIMA allow designers to handle their problem easily and provide better interaction compared to most of the conventional systems. The performance of this prototype is promising, but further research is neccessary to explore the full potential of this new technology for applications to design problems.

Acknowledgment This work is supported by a continuing grant from the Australian Research Grants Scheme and by a Sydney University Postgraduate Research Studentship.

References

Balachandran, M. and Gero, J. S., (1987), 'Use of knowledge in selection and control of optimization algorithms,' *Engineering Optimization*, **12**, 2, 163-173.

Clocksin, W. F. and Mellish, C. S., (1981), *Programming in Prolog*, Springer-Verlag, Berlin.

Cohon, J. L., (1978), *Multiobjective Programming and Planning*, Academic Press, New York.

Coyne, R. D. and Gero, J. S., (1985), 'Design knowledge and context,' *Environment and Planning*, B, 12, 419–442.

Fikes, R. and Kehler, T., (1985), 'The role of frame-based representation in reasoning,' *Comm. A.C.M.* **28**, 9, 904–920.

Gero, J. S., (1983), 'Knowledge engineering—future uses of computers in engineering,' *Computers and Engineering*, IEAust., pp. 159–162.

Gero, J. S., (1985), *Design Optimization*, Academic Press, New York.

Gero, J. S., Radford, A. D., Coyne, R. D. and Akiner, V. T., (1985), 'Knowledge-based computer-aided architectural design,' *Knowledge Engineering in Computer-Aided Design*, J. S. Gero (Ed.), North-Holland, Amsterdam, pp. 57–81.

Hayes-Roth, F., Waterman, D. A. and Lenat, D. B., (1983), *Building Expert Systems*, Addison-Wesley, Reading, Mass.

MacCallum, K. J., Duffy, A. and Green, S., (1985), 'An intelligent concept design assistant,' in *Preprints Design Theory for CAD*, University of Tokyo, pp. 233–249.

Maher, M. L. and Fenves, S. J., (1985), 'HI-RISE : An expert system for the preliminary structural design of high rise buildings,' *Knowledge Engineering in Computer-Aided Design*, J. S. Gero (Ed.), North-Holland, Amsterdam, pp. 125–135.

Martin, W. A. and Fateman, R. J., (1971), 'The MACSYMA system,' *Second Symposium on Symbolic and Algebraic Manipulation*, Los Angeles, pp. 59–75.

Minsky, M., (1975), 'A framework for representing knowledge,' *The Psychology of Computer Vision*, P. Winston (Ed.), McGraw-Hill, New York, pp. 217–277.

Mitchell, W. J., Steadman, J. P. and Liggett, R. S., (1976), 'Synthesis and optimization of small rectangular floor plans,' *Environment and Planning*, B, 3, 37–70.

Newell, A. and Simon, H. A., (1972), *Human Problem Solving*, Prentice-Hall, New Jersey.

Post, E., (1943), 'Formal reductions of the general combinatorial problem,' *American Journal of Mathematics* **65**, 197–268.

Radford, A. D., Gero, J. S., Rosenman, M. A. and Balachandran, M., (1985), 'Pareto optimization as a computer-aided design tool,' *Optimization in Computer-Aided Design*, J. S. Gero (Ed.), North-Holland, Amsterdam, pp. 47–69.

Wilensky, R., (1984), *LISPcraft*, W. W. Norton and Company, New York.

Winston, P. H., (1984), *Artificial Intelligence*, 2nd edn., Addison-Wesley, Reading, Mass.

A Knowledge-Based Expert System for Optimal Structural Design

Donald E. Grierson
Department of Civil Engineering
Solid Mechanics Division
University of Waterloo, Ontario
Canada

Abstract This lecture concerns the development of a knowledge-based expert system for the computer-automated least-weight design of structural steel frameworks subject to design code criteria and commonly used rules of design practise. The expert system is implemented for steel design standards and utilizes corresponding databases of commerically available standard steel sections. The numeric-based tasks of design are implemented in FORTRAN routines ; these include first and second-order structural analysis, sensitivity analysis, optimization and design verification. The knowledge-based tasks of design are implemented in rules and procedures encoded in an artificial intelligence language, OPS83. A knowledge base of rules controls the overall design synthesis process, which is organized into three stages; Preliminary, Solution and Critique. For fixed structure topology and known loads, the Preliminary stage determines material properties, section profile, fabrication group and an initial section size for each member of the structure. The Solution stage involves an iterative design process that controls the execution of FORTRAN routines until convergence to a reasonable least-weight design of the structure occurs. The Critique stage assesses the results of the Solution stage and suggests possible design improvements, if any, based on rules of good practise commonly employed by experienced designers; the Solution stage is then re-activated with the suggested changes implemented, or the results of the final design are printed if no improvements are considered.

The lecture commences with the FORTRAN-based software system presented by the writer in a previous lecture of the NATO-ASI, and first deals with the issues of concern in converting a FORTRAN program to a knowledge-based expert system program. Then, the issues involved in developing the expert system itself are addressed at the three stages of the design process noted in the foregoing: Preliminary, Solution and Critique. Finally, the expert system is applied for a number of design examples to illustrate its capability to design structural steel frameworks of the type encountered in professional practise.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 57–100.

1 Introduction

A routine activity in structural engineering offices is the design of structural steel building frameworks in conformance with the strength/stability and stiffness provisions of the governing steel design standard. To that end, a major task of the designer is to size the girder, column and bracing members of the framework using commerically available standard steel sections. This member-sizing design activity is the concern of the knowledge-based expert system presented herein.

Typically, the sizing of the members of a steel framework involves an iterative process wherein repeated computer analysis and design modification of the framework is conducted until the design standards have been met and some measure of economy has been achieved. In a previous lecture of the NATO-ASI, the writer has presented a FORTRAN software system that automates this process to find the least-weight design of structural steel frameworks [1]. The system is based upon optimization theory that enables the entire design of a steel framework for both strength and displacement requirements to be conducted in a single computer run. The design is carried out in complete conformance with the provisions of a specified steel design standard (currently for North America [2,3,4]), and the members are automatically selected from a corresponding database of commercially available standard sections (currently Canadian or American).

As noted in the previous lecture, the FORTRAN software system has a variety of features that relate directly to the provisions of the governing steel design standard. Default values are provided for Young's modulus, yield stress, etc. Bolted connections for truss members may be accounted for, as well as various gusset-plate thicknesses for back-to-back double angle sections. Members with symmetrical sections may be considered in strong-axis or weak-axis bending. The local buckling classification of each member section is automatically calculated. The effective length factor for each member is automatically calculated, for sidesway either prevented or permitted. Both out-of-plane and compression flange bracing are accounted for, and the unbraced compression flange length for each flexural member is automatically calculated. During a design run, a complete clause-by-clause verification is conducted for the member strength and structure stiffness provisions of the specified steel design standard. Specifically, standards-based provisions are verified for member slenderness (tension and compression), axial strength, bending strength, shear strength and combined axial and bending strength. Also verified are any user-specified requirements concerning allowable deformation of the structure (e.g., limited lateral sway).

The FORTRAN software system for the computer-automated sizing of least-weight girder, column and bracing members of structural steel frameworks [1] is widely used in professional practice (in North America) and embodies a broad base of computational and design knowledge. For example, commencing and then controlling the iterative design process so as to ensure convergence to a reasonable design solution involves considerable computational expertise that has been garnered over many years of research. In fact, maintaining and updating the system is primarily concerned with making adjustments and additions to this and designer-preference knowledge. This task is often quite difficult, however, precisely because all of the knowledge is coded in FORTRAN and is therefore embedded throughout the software system.

This lecture commences with the predecessor FORTRAN software system for structural steel design noted in the foregoing and develops a corresponding knowledge-based expert system. Specifically, the numeric-intensive algorithms for analysis and optimization are retained in FORTRAN, while the non-structured knowledge that drives the design process is collected together as rules in a separate knowledge base and implemented using the artificial intelligence (AI) programming language OPS83, [5]. The lecture first addresses the basic issues of concern in converting a FORTRAN program to a rule-based expert system program. Then, the issues involved in developing the expert system itself are addressed at three stages of the design process: (1) Preliminary stage to establish the basic data for the design; (2) Solution stage to establish the numerical results of the design; (3) Critique stage to evaluate the merits of the design. Finally, several example designs of structural steel frameworks are presented to illustrate the features of the expert system and how the knowledge-based technology effectively improves the synthesis strategy and the design outcome.

2 Interfacing FORTRAN and AI Languages

The basic strategy in developing the expert system involves retaining all of the algorithmic routines for the predecessor FORTRAN software system for structural steel design [1], while replacing the FORTRAN control structures of the existing system with corresponding routines written in the AI programming language OPS83, [5]. The retained FORTRAN routines have been well developed and extensively tested over many years and, in most cases, provide the most effective and efficient means to solve their respective tasks (e.g., first-order and second-order structural analysis, displacement and stress sensitivity analysis, continuous and discrete optimization). The rule-based OPS83 routines, which quantify the non-numeric aspects of design and drive the synthesis process by directly invoking the numeric FORTRAN routines, are very easily modified and updated to accommodate different design scenarios.

OPS83 is the primary control language and, therefore, all FORTRAN-based algorithms are treated as subroutines by OPS83. All information common to both environments is passed through a subroutine argument list. Since the number of entries in an OPS83 argument list is limited to twelve while, on the other hand, the number of variables and parameters used by both environments is in the hundreds, and since all global memory used by a FORTRAN routine is lost when control is returned to the OPS83 environment, two OPS83 and two FORTRAN mapping and recovery routines are utilized to manage the transfer of data between the two programming environments, as follows: (1) prior to accessing a FORTRAN algorithm from OPS83, an OPS83 mapping-routine is called to map all relevant data into (at most twelve) large vectors; (2) the FORTRAN algorithm is then called with these large vectors as its arguments; (3) once in the FORTRAN environment, a FORTRAN recovery-routine is then called to recover the data from the large vectors; (4) prior to exiting the FORTRAN environment, a FORTRAN mapping-routine is called to map the data back into the large vectors of the argument list; (5) upon returning to the OPS83 environment, an OPS83 recovery-routine is called to recover the data from the large vectors.

The foregoing mapping/recovery strategy allows for the effective transfer of data between the two programming environments and, during program development, also allows new FORTRAN and/or OPS83 variables and parameters to be readily introduced into the expert system program with little difficulty.

3 The Expert System

The architecture of the expert system is illustrated in Fig. 1, [6,7]. The designer enters basic data into the Context through the User Interface. The system itself posts other data in the Context during the course of conducting the design. The Knowledge Base contains rules based on expert knowledge that control the design process, including rules that reflect good design practise and designer preferences. The numeric FORTRAN routines and standard steel section databases are resources of the Knowledge Base. The operation of the expert system is managed by the Inference Engine, which employs the inference mechanism of the OPS83 language to fire rules in the Knowledge Base by matching their premises with data in the Context. The Explanation Facility provides the user with information as to why certain rules are fired during the design process.

While many rules are required to control the overall design process, only a limited number of these rules are applicable at any one time. This fact is exploited by dividing the Knowledge Base into three distinct sets of rules corresponding to three stages of the design process: (1) Preliminary stage that establishes an initial design, (2) Solution stage that establishes a corresponding least-weight design, and (3) Critique stage that evaluates the current design and suggests redesign with modifications. As shown in the following, the rules in each set are devised such that they can be fired only when the corresponding design stage is active. The described subdivision of the Knowledge Base, also indicated in Fig. 1, improves the computational efficiency of the expert system since it reduces the number of rules that need to be explicitly considered at any given stage of the solution process.

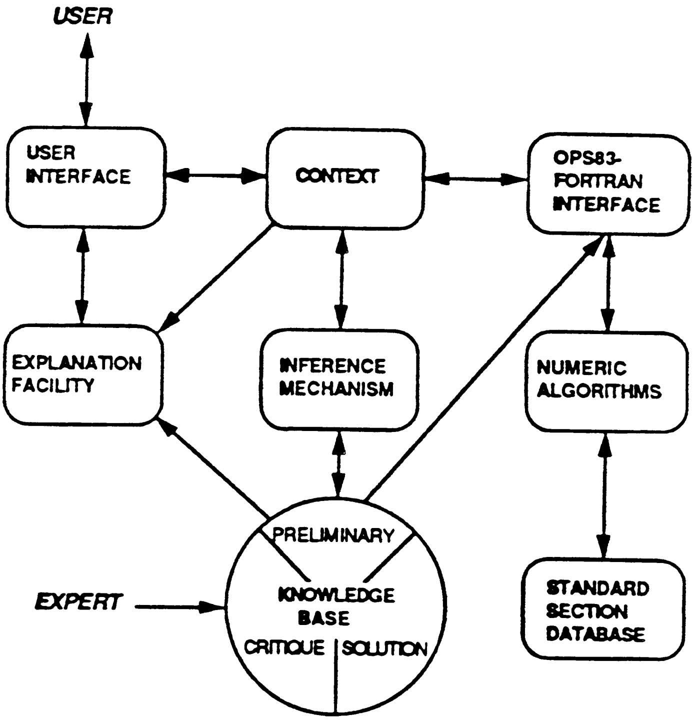


Figure 1. *Architecture of the expert system.*

An overview of the three stages of the synthesis process is given by Fig. 2.

3.1 Preliminary Stage

To begin the Preliminary stage of the design process, a number of basic parameters are input as data through the User Interface into the Context for the expert system (Fig. 1); specifically, the structure topology (bay widths, storey heights, connection and support types) and the design loading (dead, live, snow, wind, thermal, settlement). The Preliminary stage then employs a number of related rules in the Knowledge Base to establish an initial design for the structure.

Once the basic data for the design has been stored in the records of global memory and the elements 'stage = preliminary' and 'data input = done' have been posted in the Context, the activities of the Preliminary stage are established by the 'preliminary agenda rule' (written here in pseudo-code):

```
rule:  preliminary agenda
If     stage = preliminary
       data input = done
then   design cycle = 1
       member behaviour = required
       member profile = required
       member group = required
       initial design = required
       analysis = required
       member function = required
       verify = required
       preliminary termination = required
```

This rule is fired at the beginning of the Preliminary stage to define the tasks required to be completed to identify an initial design for the structure. The first action of the agenda rule is to set the 'design cycle' counter to unity (this counter is used as an index to store data for the various designs created by the synthesis process). Thereafter, the design tasks are performed sequentially in the order shown through the firing of corresponding individual rules. The sequential ordering of the rules is regulated by the OPS83 Inference Mechanism that directly fires a rule in the Knowledge Base whenever data that matches the rule premise is encountered in the Context (Fig. 1). Note from the rule descriptions in the following that the firing of some rules is conditional upon the completion of preceding rules, which further serves to explicitly regulate the sequential flow of action.

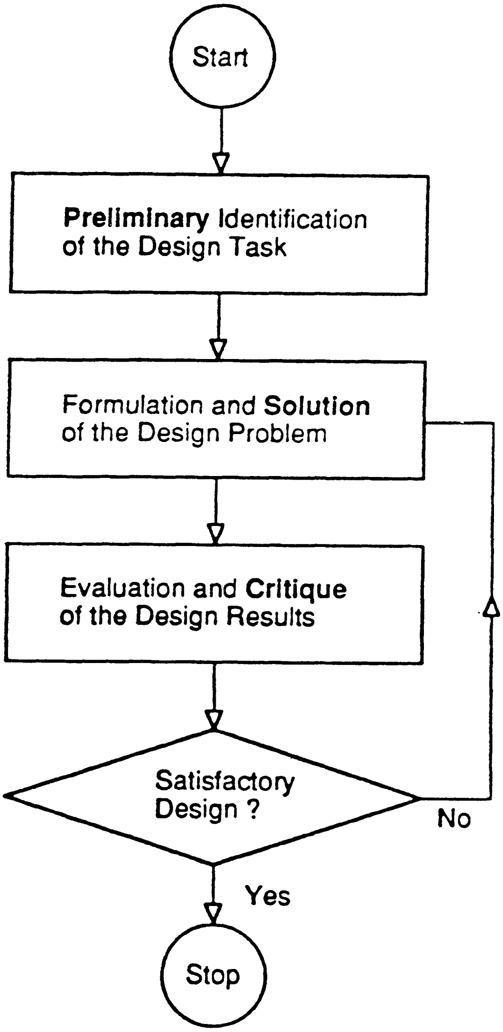

Start
Preliminary Identification of the Design Task
Formulation and Solution of the Design Problem
Evaluation and Critique of the Design Results
Satisfactory Design ?
No
Yes
Stop

Figure 2. *Overview of knowledge-based synthesis process.*

The ‘member behaviour rule’ is:

```
rule:  member behaviour
If     stage = preliminary
       member behaviour = required
then   for $i$ = 1 to number of members
            if     member $i$ end condition = pin-pin
                   member $i$ span load = zero
            then   member $i$ behaviour = axial
            else   member $i$ behaviour = flexural
       member behaviour = done
```

This rule examines the end conditions and span loading for each member to determine if its behaviour is axial or flexural. An axial member has pins at both ends and no load in span, while a flexural member has one or both ends fixed or is loaded in span.

The ‘member profile rule’ is:

```
rule:  member profile
If     stage = preliminary
       member behaviour = done
       member profile = required
then   for $i$ = 1 to number of members
            if       member $i$ behaviour = flexural
            then     min. length = max. span-to-depth × W-shape min. depth
                     if     member $i$ length > min. length
                     then   member $i$ profile = W-shape
                     else   query user
            else if  member $i$ behaviour = axial
            then     query user
       member profile = done
```

This rule establishes the cross-section profile for each member of the structure. If the member is flexural and satisfies a specified minimum length requirement, the section profile is taken to be a W-shape. Otherwise, the user is queried as to the choice of section profile (e.g., T-shape, hollow-box, etc.). This rule is based on the fact that the most common shape for flexural members of significant span is the wide flange or W-shape. The minimum span length considered to be significant is determined by multiplying the minimum available depth of W-shaped sections (default = 6 in, 150 mm) by the maximum allowable span-to-depth ratio (default = 20). This ratio is used by engineers as a quick rule-of-thumb check to ensure that adequate stiffness is provided by specified sections. If the length of the member exceeds the minimum significant length then the W-shape is suitable. Otherwise, some other profile might be more economical. This rule is readily extended to allow for the automatic assignment of a broader range of section profiles depending on the experience of the designer. Not shown here is the fact that the user always has the option to override any assignment made by the system and that the system will not override any member profile that the user might pre-specify in the input data.

The 'member group rule' is:

```
rule:  member group
If     stage = preliminary
       member profile = done
       member group = required
then   for i  = 1 to number of members
              if     member i group = undefined
              then   call member group procedure
              if     member i group = undefined
              then   query user
       for g  = 1 to number of groups
              if     group g properties = undefined
              then   call group properties procedure
              if     group g properties = undefined
              then   query user
       member group = done
```

This rule invokes two separate OPS83 routines that link members together into fabrication groups with common properties. If the input data has not assigned a member to a group, the 'member group procedure' is called. This procedure identifies members having the same behaviour type and section profile and then links them together into groups according to conventional fabrication practise (e.g., all flexural W-shape girders at each storey level of a regular framework are grouped together). If a member is irregular in that the system does not automatically assign it to a group, the user is queried for assignment.

The second 'group properties procedure' identifies the section profile common to all members of each group and then assigns to the group corresponding conventional material property values for steel (yield stress, ultimate stress, etc.). If a group is irregular in that the system does not automatically assign it material property values, the user is queried for assignment.

The 'initial design rule' is:

```
rule:  initial design
If     stage = preliminary
       member group = done
       initial design = required
then   call cross-section database routine
       for g  = 1 to number of groups
              if       group g behaviour = axial
              then     group g initial design = group g profile
                       with max. area from section database
              else if  group g behaviour = flexural
              then     group g initial design = group g profile
                       with max. I from section database
       initial design = done
```

This rule first invokes a FORTRAN routine to create the random access database of cross-section properties for each of the unique profile shapes identified for the structure by the 'member profile rule,' (Herein, the calls to FORTRAN routines are identified by *slanted* type so as to distinguish them from the OPS83 routines). The created data also includes the indices of the sections with the largest and smallest cross-section areas and moments of inertia for each shape category in the database, which then enable an initial size to be established for the section profile assigned to each group of members, as follows: if the member group behaviour is axial, the group is assigned the section having the largest cross-section area from the standard section database for the group profile; if the member group behaviour is flexural, the group is assigned the section having the largest cross-section moment of inertia from the section database for the group profile.

Once the initial sizes of the members have been established, the corresponding structure is analyzed to obtain numerical results that the expert system subsequently uses to establish the function of each member (beam/column/etc.), and to verify the initial design according to the provisions of the governing steel design standard. The 'preliminary analysis rule' is:

```
rule:  preliminary analysis
If     stage = preliminary
       analysis = required
       initial design = done
then   call properties routine
       call analysis routine
       analysis = done
```

This rule first invokes a FORTRAN routine that establishes the cross-section properties of each structural member in accordance with the design determined by the 'initial design rule,' The resulting properties are stored in corresponding FORTRAN vectors and eventually in OPS83 data records.

Once the member properties have been established, a FORTRAN routine is called to conduct analysis of the initial design of the structure using a linear elastic solver based on the Displacement Method of analysis. (For this first analysis only, a node re-numbering routine is called to minimize the band-width of the structural stiffness matrix so as to reduce computation and memory requirements, and the numerical integrity of the stiffness matrix is checked to ensure that a reasonable structure has been modelled).

First-order or second-order ($P - \Delta$) analysis is conducted, depending on the requirements of the design. Member stresses and nodal displacements are found for each design load case and stored in disk files. Most of the analysis results are not passed back to the OPS83 environment because much of this data need not be manipulated by that side of the expert system. Instead, before passing to the OPS83 side, the bulk of the data is reduced to a few scalars and vectors that contain summarized data such as maximum displacements at a few key nodes, or maximum force effects for each member from among all load cases. This is the type of information that a designer gathers from a structural analysis for later design purposes.

The 'member function rule' is:

```
rule:  member function
If     stage = preliminary or solution
       analysis = done
       member function = required
then   for $i$ = 1 to number of members
              if        $M_i$ > min. moment
                        $N_i$ > min. axial force
              then      Member $i$ function = beam-column
              else if   $M_i$ > min. moment
                        $N_i$ < min. axial force
              then      Member $i$ function = beam
              else if   $M_i$ < min. moment
                        $N_i$ > min. axial force
              then      Member $i$ function = column
              else      member $i$ function = unloaded
       member function = done
```

This rule employs the results of the analysis of the initial design to establish the functions of the members for the structure. Upon comparing minimum specified or default moment and axial force values with the maximum moment M_i and axial force N_i experienced by each member i, from among all load cases, the member function is classified as 'beam-column,' 'beam' or 'column' (or 'unloaded' if M_i and N_i are both less than their corresponding minimum values).

The 'preliminary verify rule' is:

```
rule:  preliminary verification
If     stage = preliminary
       member function = done
       verify = required
then   call verify routine
       verify = done
```

This rule invokes a FORTRAN routine that uses the analysis results for the initial design, and the functions designated for the members, to verify the member strength properties according to the provisions of the governing steel design standard. Stiffness properties are also verified by comparing nodal displacements for the structure with allowable upper-bound values specified by the user. The verification establishes the response ratio for each stress and displacement condition for the design (response ratio = ratio of actual response to maximum allowable response.) If the response ratio for any condition is found to be greater than unity, thereby signalling an infeasibility, the user is informed of this event prior to proceeding to the Solution stage (where design feasibility is generally restored after one or two design cycles).

Upon establishing an initial design for the structure with known strength and stiffness properties relative to the governing steel design standard, the Preliminary stage of the design process is terminated through the 'preliminary termination rule':

```
rule:  preliminary termination
If     stage = preliminary
       preliminary verify = done
       preliminary termination = required
then   stage = solution
       solution agenda = required
       optimizer = continuous
       preliminary termination = done
```

This rule causes the design process to proceed to the Solution stage of the expert system by modifying the Context element 'stage = preliminary' to 'stage = solution,' As well, the elements 'optimizer = continuous' and 'solution agenda = required' are added to the Context. Not shown here is the fact that all extraneous elements in the Context are removed at this point (e.g., 'verify = done,' etc.), which serves to prevent the Context from becoming cluttered with data that is no longer relevant to the synthesis process.

3.2 Solution Stage

Upon completion of the Preliminary stage, the Solution stage of the design process commences to determine a least-weight design of the structure. This stage involves the coordinated use of elastic structural analysis, first-order sensitivity analysis and a continuous/discrete optimization technique. For the initial 'trial' design of the structure from the Preliminary stage, structural and sensitivity analyses are conducted and the strength and displacement design conditions are formulated explicitly in terms of member sizing variables through the use of first-order Taylor's series. The structure weight function is formulated in terms of the sizing variables, and the optimization technique is applied to solve the optimization problem so as to achieve a lower weight design of the structure. The details of the synthesis process have been presented by the writer in a previous lecture of the NATO-ASI.

The initial design from the Preliminary stage is generally feasible but usually poorly proportioned. As a consequence, the Solution stage is devised to have two modes of operation. The first mode involves solving a continuous-variable weight optimization problem for a few design cycles until a reasonably proportioned structure is achieved. The second mode involves taking standard section sizes as discrete variables to the weight optimization. The continuous-variable optimization provides a computationally efficient means to achieve reasonable proportions for the structure before commencing discrete-variable optimization because there is no interaction with the cross-section database, the bounds of the strength constraints are taken to remain constant and only first-order analysis is employed. The latter two points recognize that the continuous-variable mode only produces an approximate design and thus the rigour of accounting for changing constraint bounds and conducting second-order analysis is not warranted. Based on the results of the continuous-variable mode, discrete sections are selected from the standard section database to initiate the discrete-variable mode of the Solution stage.

The activities of the Solution stage of the synthesis process are defined by the 'solution agenda rule':

```
rule:  solution agenda
If     stage = solution
       solution agenda = required
then   design cycle = design cycle + 1
       analysis = required
       member reselection = required
       analysis = required
       member function = required
       verify = required
       constraint deletion = required
       sensitivity analysis = required
       section subset = required
       optimization = required
       convergence check = required
```

This rule is fired at the beginning of each design cycle of the Solution stage to define the tasks required to be completed to achieve a lower-weight design. During the design cycle, the tasks are performed sequentially in the order shown through the firing of corresponding individual rules. Note from the description of these rules in the following that their sequential order is regulated by making the firing of each rule conditional upon the completion of the preceding rule.

The first action taken by the agenda rule is to advance the 'design cycle' counter by 1. Note that the second and fourth actions of the rule post the same 'analysis = required' element to the Context, and that these elements do not overwrite each other but exist as separate entities in the Context. The first structural analysis is performed prior to member reselection and ensures that cross-sections are selected based on the response of the current design. (The current design can result from actions in either the Preliminary or Critique stage, or may be the optimized design produced by the previous design cycle of the Solution stage). The result of the member selection procedure is somewhat approximate and, as such, a second analysis and subsequent verification of the structure are required to ascertain the appropriateness of the selected member sections. The two analysis rules are:

```
rule:  analysis before reselection
If     stage = solution
       member reselection = required
       analysis = required
then   call analysis routine
       analysis = done
```

```
rule:  analysis after reselection
If     stage = solution
       member reselection = done
       analysis = required
then   call analysis routine
       analysis = done
```

Note that the only difference between the two rules is the status of the 'member reselection' element. Both rules invoke a FORTRAN routine that conducts stress and displacement analysis of the current structure for each design load case (the same analysis routine as that called during the Preliminary stage). The analysis conducted is either first-order or second-order, depending on the requirements of the design, and is determined by the 'analysis type rule':

```
rule:  analysis type
If     analysis = required
then   if     optimizer = continuous
       then   analysis type = first order
       else   analysis type = specified analysis type
       if     specified analysis type = undefined
       then   specified analysis type = default analysis type
```

This rule is invoked each time the element 'analysis = required' is posted in the Context. (Note that this rule can be fired during any of the three design stages since it has no 'stage =' pre-condition). Currently, the rule ensures that if the user has not specified an analysis type then a default analysis type is adopted. The default analysis type is automatically determined from input data in the Context concerning the requirements of the specified design standard and whether sidesway is 'permitted' or 'prevented.' In addition, the analysis is automatically taken to be first-order if the Solution stage is in the continuous-variable mode. More sophisticated features can be added to this rule (e.g., even though the default analysis type is second-order, a comparison of first-order and second-order results for the current structure may show that $P - \Delta$ analysis is not warranted).

The member selection process commences by firing one of the following two reselection rules, depending on whether the Solution stage is in continuous or discrete-variable mode. The first 'member reselection rule' is:

```
rule:   member reselection-continuous
If      stage = solution
        optimizer = continuous
        analysis = required
        member reselection = required
then    member reselection = done
        analysis = done
```

This rule is fired for the continuous-variable mode simply to set the status of the Context element 'member reselection' from 'required' to 'done' (i.e., since member reselection is only required for the discrete-variable mode). Similarly, since no reselection is performed, a second structural analysis is not required and the status of the 'analysis' element is also changed to 'done' in the Context. The second 'member reselection rule' is:

```
rule:   member reselection-discrete
If      stage = solution
        optimizer = discrete
        member reselection = required
then    for g  = 1 to number of groups
               if        group g behaviour = axial
               then      group g stiffness property = area
               else if   group g behaviour = flexural
               then      group g stiffness property = I
        call selection routine
        member reselection = done
```

This rule first determines the cross-section stiffness property to be considered by the member reselection routine so as to appropriately account for the stiffness (displacement) conditions for the structure when making member strength selections. Namely, the *stiffness property* is set to *area* for axial members and to *moment of inertia I* for flexural members. (Note that the same stiffness property applies to all members of each group). A FORTRAN member reselection routine is then called to search the entire database of standard sections for a smaller section for each member group g satisfying sizing, strength and stiffness constraints for all members belonging to the group. To this end, the strength constraints are evaluated member-by-member based on the cross-section properties in the database and the requirements of the governing design standard. There are, however, no design standard formulae that permit the overall structural stiffness to be directly evaluated based on cross-section properties. For this reason, an approximate technique that utilizes displacement sensitivities is employed to estimate the overall effect of member cross-section properties (area and moment of inertia) on the stiffness of the structure. This technique enables minimum section size constraints to be imposed during the member-by-member selection process so as to ensure adequate overall structural stiffness.

Prior to verification of the current design, the 'member function rule' previously described for the Preliminary stage is re-fired for the Solution stage to ensure that the designated function of a member is consistent with the current distribution of forces in the structure. The 'solution verify rule' is then fired:

```
rule:  solution verify
If     stage = solution
       member function = done
       verify = required
then   call verify routine
       call design history update
       verify = done
```

This rule invokes a FORTRAN routine that uses the current analysis results to verify the member strength and structure stiffness properties according to the provisions of the governing steel design standard. (This is the same verify routine as that called during the Preliminary stage.) The verification establishes the response ratio for each stress and displacement condition for the design.

With a view to numerical efficiency, the stress and displacement conditions that have small response ratios are deleted from the active constraint set for the next weight optimization through the 'constraint deletion rule':

```
rule:  constraint deletion
If     stage = solution
       verify = done
       constraint deletion = required
then   deletion response = min. deletion response + (design cycle)/10
       if     deletion response > max. deletion response
       then   deletion response = max. deletion response
       call deletion routine
       if     no. of active constraints < min. no. of active constraints
       then   no. of active constraints = min. no. of active constraints
       constraint deletion = done
```

This rule invokes a FORTRAN routine that deletes displacement and stress conditions having response ratios less than a specified minimum value from the active constraint set. Prior to calling the deletion routine, the rule establishes the minimum response ratio value for the current design cycle based on the absolute minimum (default = 0.3) and maximum (default = 0.8) deletion response ratios provided as input data in the Context. (Note that the specified minimum response ratio progressively increases from a low value for the initial design cycle when the member sizes change significantly, to a high value for the final design cycle when convergence to the least-weight structure occurs). After the deletion routine has executed, the rule then ensures that a sufficient number of constraints yet remain to conduct the next weight optimization. (Note that deleted constraints are verified during subsequent design cycles and added to the active constraint set if their response ratios become greater than the current specified minimum value).

As described by the writer in a previous lecture of the NATO-ASI, the first-order Taylor's series approximations of the retained stress and displacement constraints are formulated using sensitivity analysis techniques. Within the expert system, this activity is carried out through the 'sensitivity analysis rule':

```
rule:  sensitivity analysis
If     stage = solution
       constraint deletion = done
       sensitivity analysis = required
then   call sensitivity routine
       sensitivity analysis = done
```

This rule invokes a FORTRAN routine that first conducts sensitivity analysis of the current structure to determine stress and displacement gradients, and then formulates first-order Taylor's series approximations of the corresponding constraint equations.

The upper and lower bounds on member cross-section sizes are supplied as data from the Preliminary stage and remain constant during the first mode of the Solution stage that involves solving the continuous-variable weight optimization problem. For the second mode that involves solving the discrete-variable weight optimization problem, subsets of candidate sections for the members are selected for each design cycle from the standard section database through the 'section subset rule':

```
rule:   section subset
If      stage = solution
        sensitivity analysis = done
        section subset = required
then    call subset routine
        section subset = done
```

This rule invokes a FORTRAN routine that selects a limited number of sections (default = 12) that are closest to the current section in the database and have adequate strength and stiffness properties. (Note that this may or may not result in the selection of sections having smaller weight than the current section).

The coefficients in the weight function for the structure are supplied as data from the Preliminary stage and remain constant for both modes of the Solution stage. The weight optimization for each design cycle is conducted through the 'optimization rule':

```
rule:   optimization
If      stage = solution
        section subset = done
        optimization = required
then    call optimization routine
        optimization = done
```

This rule invokes a FORTRAN routine [8] that solves the continuous-variable weight optimization problem for each design cycle of the first mode of the Solution stage, or the discrete-variable weight optimization problem for each cycle of the second mode, to achieve a lower-weight design of the structure.

The number of design cycles for the continuous-variable mode of the Solution stage is a fixed number that is supplied as input data in the Context. The transition to the discrete-variable mode is carried out by the 'continuous to discrete optimization rule':

```
rule:  continuous to discrete optimization
If     stage = solution
       optimizer = continuous
       design cycle = continuous_variable cycles + 1
then   call continuous to discrete routine
       optimizer = discrete
```

This rule is fired when the design cycle counter exceeds the allowable number of continuous-variable design cycles (default = 3). The first action of the rule is to change the Context element 'optimizer = continuous' to 'optimizer = discrete.' Then, a FORTRAN routine is called to convert the continuous-variable section sizes for the members to discrete sections from the database. (This simply involves taking the discrete section that has a size that is closest to, but larger than, the size of the corresponding continuous-variable section).

The number of design cycles for the second discrete-variable mode of the Solution stage depends on when the synthesis process converges to the least-weight structure, and is controlled through the 'convergence check rule':

```
rule:  convergence check
If     stage = solution
       optimization = done
       convergence check = required
then   call convergence routine
       if       convergence = false
       then     solution agenda = required
       else if  convergence = true
       then     stage = critique
                critique agenda = required
```

This rule invokes an OPS83 procedure that determines whether or not the solution process has converged to a least-weight design of the structure. If not, the 'solution agenda rule' is then fired again to commence another design cycle of the Solution stage. If convergence has occurred, the Solution stage of the design process is terminated and the Critique stage is activated.

3.3 Critique Stage

Upon completion of the Solution stage, the Critique stage of the design process commences to determine if the current least-weight design of the structure can or should be improved upon. The activities of the Critique stage are established by the 'critique agenda rule':

```
rule:  critique agenda
If     stage = critique
       critique agenda = required
then   critique termination = required
       improved fabrication = required
       improved section profile = required
       improved section depth = required
       improved unbraced length = required
       improved supports = required
       improved stiffness = required
```

This rule is fired at the beginning of the Critique stage to identify the areas to be examined for possible design improvement. The tasks are performed sequentially through the firing of corresponding individual rules, but in the reverse order shown. (A currently devised, the various improvement rules of the Critique stage are not conditional upon each other and, as such, the significance of their firing order is somewhat arbitrary). The reversed firing order of the rules is regulated by the 'younger-before-older' conflict-resolution strategy employed by the OPS83 Inference Mechanism. Namely, even though the pre-condition of a critique rule is satisfied as each agenda task is posted to the Context, no other rule can fire until the agenda rule has finished executing. As such, the last rule satisfied by the postings of the agenda tasks will be the 'youngest' rule and, hence, will be fired first by the Inference Mechanism. The 'critique termination rule' will be fired last since it is the first item of the agenda rule and, therefore, the 'oldest' element in the Context.

Recognizing that the improvements suggested by some rules may negate or conflict with the findings of other rules, no changes to the design are allowed until all the critique rules have fired. Then, the user is presented with a list of possible improvements which may be implemented in any order, or ignored. (For the current expert system, it is left to the discretion of the user to decide when an improvement negates or conflicts with any other suggested improvements). The user may invoke one or more of the suggested improvements, at which point the expert system will re-initiate the Solution stage (discrete mode) with the 'improved' design as the starting basis. Eventually, the 'critique agenda rule' will be fired again to re-evaluate the design with respect to the improvements considered thus far, and so on until no further improvements are made and the expert system is terminated.

The 'improved fabrication rule' is:

```
rule:  improved fabrication
If     stage = critique
       improved fabrication = required
then   call merge groups routine
       call split groups routine
       improved fabrication = done
```

This rule invokes two separate FORTRAN routines that evaluate specific fabrication details of the current design for the structure. The first routine compares the design sections for the different member groups for the structure and identifies sets of two or more groups that have nearly identical sections. For each set of member groups so identified, the user is later queried as to whether to merge the groups together into a single group so as to reduce the number of different sections required for the design. The second routine evaluates the maximum response ratio for the individual members within each member group and identifies situations where only a few members have high response ratios and, therefore, control the group design at the expense of those members with low response ratios. For each member group for which such a situation is identified, the user is later queried as to whether to divide the group into two or more smaller groups so as to more effectively utilize member capacities and thereby result in a lower weight design. (Evidently, other routines are readily added to this rule as additional fabrication details become of concern.)

The 'improved section profile rule' is:

```
rule:  improved section profile
If     stage = critique
       improved section profile = required
then   call improved profile routine
       improved section profile = done
```

This rule invokes an OPS83 procedure that determines if the design section for each member group is the biggest or smallest section available from the section profile database specified for the group. If the design section is found to be at a limiting size, the user is later queried as to whether to select a different section profile for the member group with less restrictive size limitations so as to result in a more effective distribution of the relative member sizes for the structure.

The 'improved section depth rule' is:

```
rule:  improved section depth
If     stage = critique
       improved section depth = required
then   call improved depth routine
       call relative depth routine
       improved section depth = done
```

This rule invokes two separate OPS83 procedures that evaluate the appropriateness of the member section depths for the current design of the structure. The first routine identifies any section depths that appear to be too large or too small for good design practise. In the former case, column depths are compared to a maximum allowable depth (default = 14 inches, 360 mm), and in the latter case, the span-to-depth ratio of each beam is compared to an allowable upper-bound value (default = 20). For each section depth so identified as being too large or too small, the user is later queried as to whether to impose a depth limitation for the section so as to satisfy conventional design practise, clearance requirements, aesthetics, etc. (The routine does not evaluate section depths if the user has pre-specified allowable depth limits). The second routine considers the member joints for the structure and identifies situations where there are significant differences in the relative section depths of connecting members (default = 4 inches, 100 mm). For each situation so identified, the user is later queried as to whether to impose a section depth limitation for one or more of the connecting members so as to ensure a more compatible joint connection.

The 'improved unbraced length rule' is:

```
rule:  improved unbraced length
If     stage = critique
       improved unbraced length = required
then   call lateral bracing routine
       call compression-flange bracing routine
       improved unbraced length = done
```

This rule invokes two separate OPS83 procedures that evaluate the unbraced lengths of certain members for the structure. The first routine identifies axial members for which the strength design is controlled by out-of-plane buckling. For each such member so identified, the user is later queried as to whether additional bracing is available (e.g., from a wall) such that a smaller out-of-plane unbraced length may be specified for the member. The second routine identifies loaded flexural members with wide-flange section for which the unbraced compression flange length is larger than a specified value (default = one-half the span length of the member). For each such member so identified, the user is later queried as to whether additional bracing is available for the compression flange (e.g., from a floor) such that a smaller unbraced compression-flange length may be specified for the member. (The reduced unbraced lengths suggested by both routines will allow for smaller section sizes for the same member load-carrying capacities).

The 'improved supports rule' is:

```
rule:  improved supports
If     stage = critique
       improved supports = required
then   call improved supports routine
       improved supports = done
```

This rule invokes an OPS83 procedure that examines the supports for the structure to determine if uplift forces are present (an undesirable event). For each support for which such forces are identified, the user is later advised of the situation and encouraged to review the lateral load resisting system so as to eliminate any uplifting action.

The 'improved stiffness rule' is:

```
rule:  improved stiffness
If     stage = critique
       improved stiffness = required
then   call improved lateral stiffness routine
       call improved beam stiffness routine
       improved stiffness = done
```

This rule invokes two OPS83 procedures that identify parts of the structure where increased stiffness is potentially required. The first routine checks the maximum lateral drift experienced at the top storey of the structure and, depending upon the load factors associated with the applied loads, estimates the lateral drift at the service load level. If the deflection so identified exceeds the allowable displacement limit (default = $h/500$, where h is the height of the structure at which the displacement is measured) the user is later queried as to whether to impose a limitation on the displacement such as to ensure a more appropriate stiffness response for the structure. (The procedure is not implemented if the user has already pre-specified a lateral displacement constraint for the top storey). The second routine checks for the presence of a constraint on the vertical deflection of at least one member of each beam group where span loads are present. If no such constraint is detected, the user is later advised that vertical deflection has not been constrained for the affected member group.

For the current expert system, the foregoing describes the rules that are fired during the Critique stage to arrive at suggestions to improve the design of the structure. (Evidently, other design improvement rules are readily implemented as different design scenarios and experiences are encountered). The Critique stage is terminated through the 'critique termination rule':

```
rule:   critique termination
If      stage = critique
        critique termination = required
then    call improvement implementation procedure
        critique termination = done
        query user for action
        if        action = solution
        then      stage = solution
                  solution agenda = required
        else if   action = output
        then      query user for final design
                  call output routine
                  QUIT
```

This rule first calls an OPS83 procedure that directly queries the user to selectively implement any or all of the improvements suggested by the Critique stage. Next, the user is queried as to whether to return to the Solution stage to account for any suggested design improvements (and to eventually arrive again at this 'critique termination rule'), or to directly select a final design from the synthesis history (the expert system will recommend the most recent least-weight design produced by the Solution stage) and then call a FORTRAN routine to output the results.

4 Design Examples

This section of the lecture presents a number of example applications that demonstrate the features and capabilities of the knowledge-based expert system for structural steel design. All results are achieved for the expert system implemented on a UNIX-based VAX11-780 (University of Waterloo) computer. The first example concerns the design of a ten-storey office tower and provides an overview of the expert system from the Preliminary stage through the Solution stage to the Critique stage. The second example concerns the design of a K-braced preheater tower for a cement plant and traces the evolution of the design of a single member through the Preliminary and Solution stages of the synthesis process. The third example concerns the design of a mill building with a trussed roof and demonstrates how a new rule can be introduced into the expert system when a design scenario occurs that is beyond the scope of the existing Knowledge Base.

4.1 Ten-Storey Office Tower

The planar one-bay, ten-storey frame in Fig. 3 is part of an office building tower. The frame is to be designed for the single load case shown in Fig. 3, where the indicated load values define the service-load level, [7,9]. The design is to be governed by the provisions of the American Institute of Steel Construction (AISC) Working Stress Design standard for steel structures [3]. Member sections are to be selected from the AISC database of standard steel sections. Initially, no limitations are placed on member section depths or on nodal displacements.

The basic data defining the structure topology and loading shown in Fig. 3 is input through the User Interface into the Context for the expert system (Fig. 1). The rules in the Knowledge Base pertaining to the Preliminary stage of the design process (Section 3.1) are then implemented to establish an initial design for the structure, as indicated in Table 1. Note from Table 1 that the column members are collected together into fabrication groups over every two stories of the frame, while all nine floor beams are collected together into a single group. The 'initial design rule' assigns a W36 × 300 section to all members for the frame, which corresponds to a total structure weight of 120,000 lbs. This design, which is feasible but excessively heavy, is indicated in Fig. 4 as the Preliminary Design.

Having the initial design of the frame from the Preliminary stage, the rules in the Knowledge Base pertaining to the Solution stage (Section 3.2) are then implemented to determine a corresponding least-weight design over a number of design cycles. (Recall that member sizes are taken as continuous variables to the weight optimization for the first three design cycles, and as discrete variables for all design cycles thereafter until weight convergence occurs). Table 2 indicates, for example, the rule outcome at the beginning (member reselection) and end

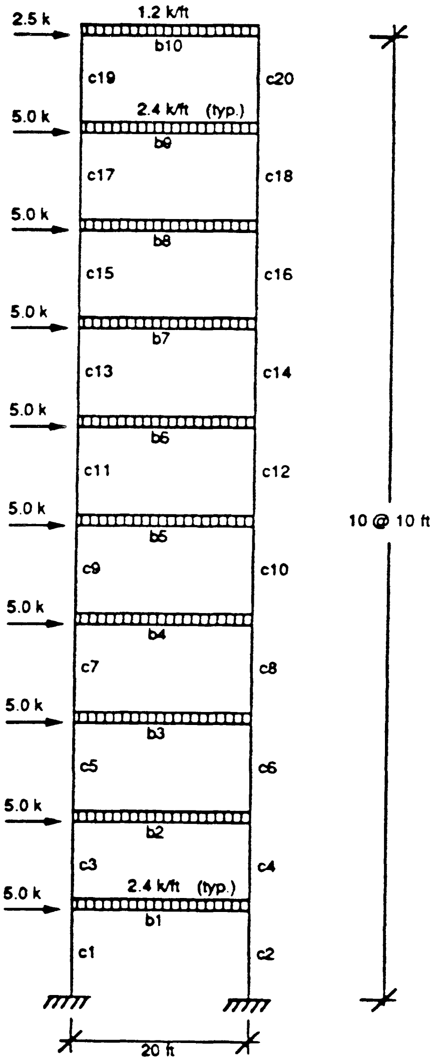


Figure 3. *One-bay, ten-storey frame.*

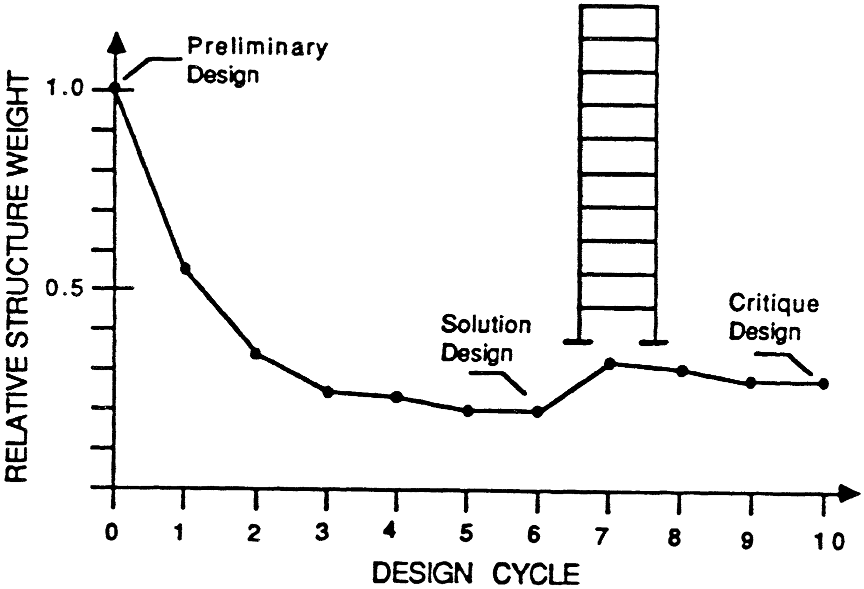


Figure 4. *Design history for ten-storey frame.*

Table 1. *Preliminary stage results for ten-storey frame.*

Rule	Outcome
Member Behaviour	Flexural (all members)
Member Profile	W-shape (all members)
Member Group	Group 1 (members c1–c4) Group 2 (members c5–c8) Group 3 (members c9–c12) Group 4 (members c13–c16) Group 5 (members c17–c20) Group 6 (members b1–b9) Group 7 (member b10) Young's Modulus = 29,000 ksi (all groups) Shear Modulus = 11,200 ksi (all groups) Yield Stress = 36 ksi (all groups) Ultimate Stress = 58 ksi (all groups)
Initial Design	W36 × 300 (all groups)
Member Function	Beam-column (all members)

Note: $Wd \times m =$ wide-flange section; depth d (in) and mass m (lb/ft).

(optimization) of the fourth design cycle (the first design cycle for which discrete-variable optimization is conducted).

Table 2. *Solution stage results for design cycle 4 for ten-storey frame.*

Member Group	Rule outcome	
	Member Reselection	Optimization
c1–c4	W30×108	W27×102
c5–c8	W21×73	W24×68
c9–c12	W18×60	W21×57
c13–c16	W16×45	W18×40
c17–c20	W16×31	W14×30
b1–b9	W24×68	W24×68
b10	W14×30	W12×30

The design outcome in Table 2 for the 'member reselection rule' corresponds to a total structure weight of 25,520 lbs., while that for the 'optimization rule' corresponds to a weight of 24,720 lbs. The Solution stage terminates after three discrete-variable design cycles (for a total of six cycles) with the least-weight design of the frame given in Table 3.

Table 3. *Solution stage results for ten-storey frame.*

Solution design			
Member Group	X-Section Designation	Length (ft)	Weight (lbs)
c1-c4	W30×108	40.	4321.
c5-c8	W21×73	40.	2391.
c9-c12	W21×57	40.	2277.
c13-c16	W16×45	40.	1813.
c17-c20	W16×31	40.	1243.
b1-b9	W24×62	180.	11160.
b10	W14×30	20.	603.
		Total :	24350.

The combined bending and axial compression stress condition for the roof beam (Eq. 1.6–2 in Ref. [3]) is the controlling condition for the design in Table 3, which is indicated in Fig. 4 as the Solution Design.

Having the least-weight design in Table 3 from the Solution stage, the rules in the Knowledge Base pertaining to the Critique stage (Section 3.3) are then implemented to determine if the current design of the frame can or should be improved upon, as indicated in Table 4.

In Table 4, the 'improved fabrication rule' suggests that the single group of nine members b1–b9 be subdivided into three independent member groups b1–b3, b4–b6 and b7–b9; this suggestion is prompted by the fact that, for the Solution Design, the stress conditions for the nine beam members b1 to b9 have maximum response ratios of 0.90, 0.96, 0.88, 0.80, 0.69, 0.60, 0.47, 0.37 and 0.27, respectively. The 'improved section depth rule' suggests that, consistent with good design practise, upper-bound limitations be placed on column section depths; this suggestion is prompted by the fact that, for the Solution Design in Table 3, column members c1–c4, c5–c12 and c13–c20 have large section depths of 30, 21 and 16 inches, respectively. The 'improved stiffness rule' suggests that the lateral drift at the roof level of the frame be limited to $h/500 = 2.4$ inches; this suggestion is prompted by the fact that the roof-level lateral drift of $h/490 = 2.43$ inches for the Solution Design is not acceptable in practise. (In fact, the $h/500$ limitation on lateral displacement becomes even more necessary to the design if the column section depths are limited as suggested.)

Upon adopting the suggested design improvements from the Critique stage, the design process returns to the Solution stage to account for the corresponding new

Table 4. *Critique stage results for ten-storey frame.*

Rule	Outcome
Improved Fabrication	Problem: member b2 dominates the design of member group b1–b9 Suggest: member group b1–b3 member group b4–b6 member group b7–b9
Improved Section Profile	No improvement suggested
Improved Section Depth	Problem: section depths are too large for column members c1–c20 Suggest: depth $\leq$ 14 in. for members c1–c4 depth $\leq$ 12 in. for members c5–c12 depth $\leq$ 10 in. for members c13–c20
Improved Unbraced Length	No improvement suggested
Improved Supports	No improvement suggested
Improved Stiffness	Problem: lateral displacement at roof level = h/490 Suggest: displacement $\leq$ h/500

data. The Solution stage (discrete mode) then commences again and converges after four more design cycles (for a total of ten cycles) to the least-weight design given in Table 5. The $h/500$ limitation on roof-level lateral drift is the controlling condition for this design, which is indicated in Fig. 4 as the Critique Design. Note from Tables 3 and 5 that the design improvements have resulted in a 18.6% increase in total structure weight from 24,350 lbs. to 28,870 lbs. The designer must decide whether this weight increase is justified for the design improvements achieved. Herein, the design given in Table 5 is deemed acceptable and the expert system is terminated.

4.2 K-Braced Preheater Tower

The planar K-braced framework in Fig. 5 is part of a preheater tower for a cement plant [7,10] and is to be designed in accordance with the provisions of the AISC Load and Resistance Factor Design (LRFD) standard for steel structures [4].

No displacement constraints are imposed for the design by the user, although the Critique stage of the expert system may later suggest such constraints so as to improve the stiffness characteristics of the frame (not considered herein). The struc-

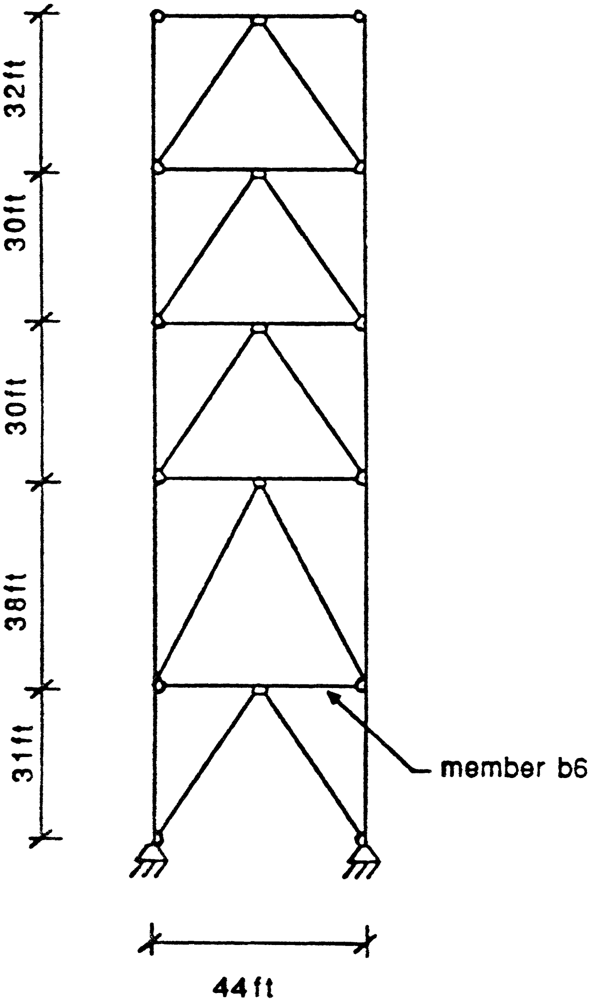


Figure 5. *K-braced frame.*

Table 5. *Critique stage results for ten-storey frame.*

Critique design			
Member Group	X-Section Designation	Length (ft)	Weight (lbs)
c1–c4	W14×109	40.	4362.
c5–c8	W12×106	40.	4253.
c9–c12	W12×96	40.	3844.
c13–c16	W10×60	40.	2399.
c17–c20	W10×30	40.	1205.
b1–b3	W27×84	60.	5071.
b4–b6	W24×62	60.	3722.
b7–b9	W24×55	60.	3313.
b10	W18×35	20.	7020.
		Total :	28870.

ture topology and five design load cases are initially input as basic data through the User Interface into the Context for the expert system (Fig. 1). The five load cases (not shown for the sake of brevity) represent various combinations of dead, live, wind and equivalent seismic loads applied to the joints and members of the framework: case 1 = dead + live; case 2 = 0.85 (dead + wind); case 3 = dead + 0.25 live + seismic; case 4 = 0.75 (dead + live + wind); case 5 = 0.75 (dead + live + seismic).

A number of default design parameters are automatically posted in the Context concerning material properties, deflection limitations, section depths, slenderness ratios, etc. For example, allowable slenderness ratios (KL/r) are specified to be 200 and 300 for members in compression and tension, respectively, in accordance with the LRFD design standard, [4]. In addition, for this design example, a number of parameters are pre-specified by the user concerning member profiles, section depths, member fabrication groups, and material properties. All member profiles are specified to be American W-shapes. To comply with conventional design practise, the depths of the column sections are limited to 14 inch (360 mm) and the depths of the K-bracing sections are limited to 12 inch (300 mm). No limitation is placed on the depths of beam sections. The specified fabrication groups include two column groups (the columns of the bottom two stories belong to one group and the columns of the top three stories belong to the other group), five beam groups (a separate group for each storey), and five K-bracing groups (a separate group for each storey). The default material properties normally assigned to American W-shapes (i.e. f_y = 36 ksi) are changed by the user to be those for f_y = 50 ksi steel.

With the view to illustrate both the operation of the expert system and the processing of the provisions of the LRFD steel standard, the remaining discussion for this example concerns the evolution of member 6 from the beginning of the Preliminary stage to the end of the Solution stage for the synthesis process. From Fig. 5, member 6 belongs to the first-storey beam fabrication group, named herein as group BEAM1.

Once the basic data has been input, the first action of the Preliminary stage for the expert system is to apply the 'member behaviour rule' to determine the behaviour of member 6. Since the member is pinned at the column but continuous over the point where the K-bracing members meet (Fig. 5), and since it is subject to span loading (not shown), the member behaviour is determined to be *flexural*. The 'member profile rule' and 'member group rule' are effectively bypassed since the user has already specified corresponding default parameters. The 'initial design rule' assigns member 6 to have a W14 × 730 section. (This is the section in the American W-shape database that has the largest moment of inertia; although such a shallow heavy section is a poor choice for a beam member it is nonetheless almost always feasible, which is a necessary condition to commence the weight optimization task of the synthesis process.) The 'analysis type rule' assigns second-order ($P-\Delta$) analysis as the basis for design (i.e., the default analysis type for the LRFD design standard). After analysis of the initial design has been conducted for all five load cases, the 'member function rule' determines that member 6 is a *beam-column* since it is subject to combined bending moment and axial force. To complete the Preliminary stage, the 'design verification rule' is fired to verify the properties of member 6 in accordance with the strength/stability provisions of the LRFD steel design standard, [4].

Table 6 lists the details of the verification of the initial design of member 6 for load case 1, for which the member is in combined *bending + tension*. From the American W-shape database for the expert system, the W14 × 730 section for member 6 is determined to be *compact*. Since member 6 is a beam-column, the clauses of the LRFD design standard relating to slenderness (B7), axial capacity (D1), shear capacity (F2.2), bending capacity (F1) and combined axial + bending capacity (Eq. H1–1b) are checked by calculating the corresponding 'response ratios' = 'prevailing value/allowable value.' For example, the slenderness ratio for member 6 having a W14 × 730 section is $KL/r = 55.65$ and the allowable slenderness ratio = 300 for a member in tension and, therefore, the response ratio = 55.65/300 = 0.186, as indicated in Table 6. The very small response ratios in Table 6 indicate that the initial design section for member 6 is significantly under-utilized. (In general, a fully utilized section exhibits a response ratio = 1.0 for one or more of its governing design code clauses).

The Solution stage of the expert system then commences to conduct the iterative synthesis process to achieve a least-weight design of the frame. For example, after three continuous-variable optimization design cycles the 'continuous to discrete optimization rule' assigns member 6 to have a W18 × 50 section. Based on

Table 6. *LRFD Verification results for initial design of member 6 of K-braced frame.*

```
-----------------------------------------------------------------
Member: 6,   Group: BEAM1,   W14X730,   Compact   , Bending + Tension
-----------------------------------------------------------------
Pu =  69.4 kips       Pr = 9680.       Ae =  215. in2
Vu =  77.8 kips       Vr = 1860.       Aw =  68.8 in2
Mu =  271. kip-ft     Mr = 6230.       Cb =  1.75       Lf =  21.8 ft
Clause B7          :                 (KL/r)/300 =    0.186
Clause D1          :                      Pu/Pr =    0.007
Clause F2.2        :                      Vu/Vr =    0.042
Clause F1          :                      Mu/Mr =    0.043
Clause eq.H1-1b    :             Pu/2Pr + Mu/Mr =    0.047
-----------------------------------------------------------------
(subscripts 'u' and 'r' refer to factored effects and calculated
resistance, respectively; P = axial force; V = shear force;
M = bending moment; Ae = effective area; Aw = area of the web;
Cb = bending coefficient; Lf = unbraced length of the flange)
-----------------------------------------------------------------
```

the analysis results for the current design of the frame, this is the least weight section in the American W-shape database that satisfies the LRFD strength/stability requirements for the member. The weight optimization for the fourth design cycle assigns member 6 to have a W18 × 46 section (i.e., the unit weight of the member decreased from 50 to 46 lb/ft.). Succeeding design cycles of the Solution stage produce still lighter-weight designs until weight convergence occurs, at which point member 6 is assigned to have a W16 × 36 section.

Table 7 lists the details of the verification of the final design of member 6 for load case 1 (the critical load case for the member from among the five load cases for the frame). Upon comparing the response ratios in Tables 6 and 7 it is seen that, contrary to that for its initial design section, the final design section for member 6 is almost fully utilized. The governing LRFD code clause in Table 7 concerns combined axial + bending behaviour, for which member 6 is 98.2% utilized.

The Critique stage of the expert system is not illustrated for this example.

Table 7. *LRFD Verification results for final design of member 6 of K-braced frame.*

```
-------------------------------------------------------------------
Member: 6,   Group: BEAM1,   W16X36,    Compact   , Bending + Tension
-------------------------------------------------------------------
Pu =  68.9 kips        Pr =  477.        Ae =  10.6 in2
Vu =  75.4 kips        Vr =  127.        Aw =  4.69 in2
Mu =  218. kip-ft      Mr =  240.        Cb =  1.75       Lb =  21.8 ft
Clause B7          :               (KL/r)/300 =     0.572
Clause D1          :                    Pu/Pr =     0.144
Clause F2.2        :                    Vu/Vr =     0.595
Clause F1          :                    Mu/Mr =     0.910
Clause eq.H1-1b    :            Pu/2Pr + Mu/Mr =     0.982
-------------------------------------------------------------------
```

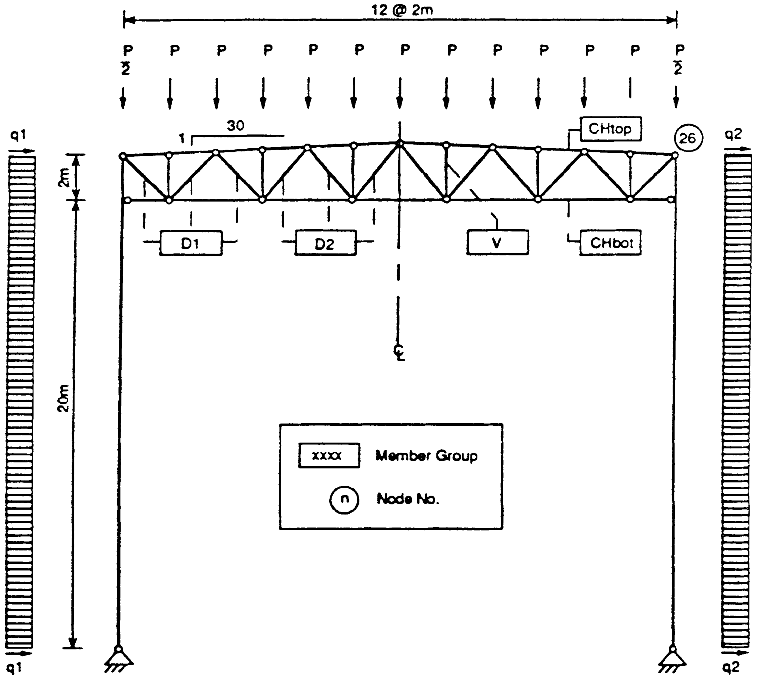
4.3 Mill Building Framework

The purpose of the following design example [7] is not to track the workings of the synthesis process in detail but, rather, to demonstrate that the expert system environment provides an excellent means to accommodate previously unforeseen design scenarios through the addition of new rules in the Knowledge Base. Specifically, the introduction of new rules is discussed for both the Solution and Critique stages of the design process.

The planar mill building framework in Fig. 6 is to be designed in accordance with user-specified displacement constraints and the strength/stability provisions of the Canadian Limit States Design (LSD) standard for steel structures [2]. Member sections are to be selected from the Canadian Institute of Steel Construction (CISC) database of standard sections.

The mill frame is subject to the three design load cases indicated in Fig. 6. The first load case consists of factored dead and live gravity loads, the second load case consists of service (unfactored) live gravity loads, and the third load case consists of combined dead, live, and wind loads at the service level. Vertical displacement at the midspan of the roof truss is limited to 80 mm ($L/300$) for load case 2. Horizontal displacement at the roof top is to be limited to 60 mm ($h/370$) for load case 3.

The user has specified that all members of the roof truss are to be square hollow structural (SHS) sections and that the column members are to be welded wide flange (WWF) sections. Six member fabrication groups have also been specified,



LOAD CASE	P	q1	q2
1	1.25D + 1.5L		
2	L		
3	1.25D + 0.70(1.5L)	0.70(1.5w1)	0.70(1.5w2)

Where: D = 15.0 kN, L = 37.5 kN, w1 = 6.3 kN/m, w2 = 4.0 kN/m

1 m = 3.28 feet, 1 kN = 0.225 kips, 1 kN/m = 68.5 plf

Figure 6. *Mill building framework.*

consisting of: top chord truss members (CHtop); bottom chord truss members (CHbot); vertical web truss members (V); diagonal members of the center three truss panels (D2); diagonal members of the outer two panels at each end of the roof truss (D1); column members extending the full height of the structure (COL). No depth limitations have been specified for any of the member groups. Out-of-plane bracing is provided at each structural joint of the roof truss (i.e., $K_y = 1$), and the WWF columns are additionally braced at third points (i.e., $K_y = 0.33$). The user has also specified that $P - \Delta$ effects are to be accounted for and, therefore, that second-order analysis is the underlying basis of the design.

The foregoing input data is first read into the Context of the expert system (Fig. 1), and then the Preliminary stage of the synthesis process begins. The initial design section determined for all truss members is a 305 × 305 × 13 section, which is the SHS section with the largest area in the CISC database. A WWF1800 × 632 section, having the largest moment of inertia in the WWF database, is assigned to the column members. (Section definitions are given in Table 10). A second-order analysis is conducted and the initial design is verified according to the user-specified displacement constraints and the strength provisions of the LSD design standard, [2]. The corresponding maximum constraint response ratio = 0.749, indicating that the initial design is feasible.

The Solution stage of the synthesis process then begins and, after three continuous variable design cycles, a discrete design is selected and verified. However, though the strength constraints are found to be satisfied (maximum response ratio = 0.945), the horizontal displacement constraint for load case 3 is found to be violated by 17%. This infeasibility is too large to permit the design process to continue to the optimization phase and, as such, a new and feasible design must first be found. However, no provision for this requirement has as yet been allowed for in the scheme of the Solution stage. Therefore, as discussed in the following, a new rule must be added to the Knowledge Base that recognizes the infeasibility condition and then restores design feasibility.

Recall from the 'member reselection rule' in the Solution stage (Section 3.2) that, prior to member reselection, a FORTRAN routine determines the lower bound area or moment of inertia value that a candidate section must possess in order to satisfy stiffness requirements during the strength-based member reselection process. This routine utilizes the results of the previous structural analysis to generate displacement sensitivity coefficients, which are then used to evaluate the contribution of each member group to the overall structural stiffness. The fact that the current design for the mill frame is 17% infeasible for a displacement constraint indicates that the displacement sensitivity coefficients should be updated, that new lower bounds on section areas and inertias should be determined and that another member reselection should take place. With the improved lower bounds, the member reselection process should produce a feasible or nearly feasible design after the next verification, although more than one update and reselection may be required.

The foregoing strategy to restore feasibility can be implemented by a new rule that places a subset of the Solution agenda tasks in the Context. This 'restore feasibility agenda rule' is:

```
rule:  restore feasibility agenda
If     stage = solution
       verify = done
       design = infeasible
then   member reselection = required
       analysis = required
       member function = required
       verify = required
```

This rule will only fire after the 'solution verify' rule determines that a design infeasibility exists and the 'design = infeasible' condition is posted in the Context. This new agenda rule sends four tasks to the Context that have the same effect as causing the original 'solution agenda rule' to backtrack and repeat its last four design tasks. The rule will continue to fire until feasibility is restored (or nearly restored; currently a 3% infeasibility is allowed) or until a fixed number of reselection attempts occurs (default = 3), at which point the Solution stage is allowed to continue to the optimization phase. (Note: even if some infeasibility still exists at this point it is generally significantly smaller than that which was originally detected and, as such, acceptable to the optimizer.)

Returning to the design of the mill frame with the 'restore feasibility agenda rule' now in the Knowledge Base, the expert system executes exactly as before until the 17% infeasibility is detected for the displacement constraint for load case 3. The new rule is then fired and the agenda of restorative design tasks is posted in the Context, causing the corresponding rules to fire and another design to be selected. The verification of this new design determines that the displacement infeasibility is reduced to 8% and, also, that the maximum response ratio for the strength constraints is decreased to 0.80 (from 0.945). The 'restore feasibility agenda rule' is fired again and this time the reselection procedure produces a feasible design, for which the maximum displacement response ratio is 0.986 and the maximum strength response ratio is 0.641. The foregoing restorative activity is reflected by the three references to design cycle 4 in Table 8. The expert system then continues normally to complete the design cycle.

The Solution stage converges to the least-weight structure over the next three design cycles, none of which experience any constraint infeasibility. The iteration history of the design process for the Solution stage is given in Table 8. The response activities of the displacement and strength constraints for the final design are given in Tables 9 and 10, respectively. Note that the design is strongly controlled by the displacement constraint for load case 3 (Table 9). In fact, it is precisely because of this displacement constraint that the response ratios for the strength constraints

Table 8. *Mill building design history.*

Design Cycle	Design Type	Relative Weight	Response___Ratio Strength	Displ.	Feasible Design?	Design Status
0	initial	1	0.360	0.749	yes	-
1	continuous	-	-	-	-	-
2	continuous	-	-	-	-	-
3	continuous	-	-	-	-	-
4*	discrete	0.7810	0.945	1.170	NO	reselect
4*	discrete	0.8544	0.799	1.084	NO	reselect
4	discrete	0.8919	0.641	0.986	yes	-
5	discrete	0.8905	0.641	0.990	yes	best
6	discrete	0.9117	0.642	0.979	yes	-
7	discrete	0.9072	0.537	0.966	yes	-
8	discrete	0.9072	0.537	0.966	yes	converged

are all well below unity (Table 10).

The Critique stage of the expert system begins upon completion of the Solution stage, and the firing of the corresponding rules produces the following findings and recommendations concerning improvements to the current design of the mill building framework:

```
Finding: 1  [improved fabrication rule]
  The width of the diagonal group ( D1 )
  is wider than the width of the chord group ( CHbot )

Recommendation: 1.1 - for SHS sections -
  Specify a minimum allowable depth limit for the chord group or
  specify a maximum allowable depth for the diagonal group.


Finding: 2  [improved profile rule]
  The following design group(s) are at their maximum available size:
  ( CHtop )
  ( COL )
```

Table 9. *Mill building displacement constraints for the final design.*

(units: X,Y displacement = mm; Rotation = radians)

Load Comb'n	Node Name	Direction X	 Y	 Rot	Actual Displ.	Permitted (+ or -)	Ratio of: Actual/Permitted	
2	7		*		-8.25	80.0	0.103	okay
3	26	*			59.4	60.0	0.990	okay

Table 10. *Mill building strength constraints of the final design.*

Group Name	Member Name	Section Shape	 Designation	Load Case	Code Clause	Response Ratio
CHtop	ch23	SHS	305X305X13	3	13.2	0.145
CHbot	ch12	SHS	203X203X8	3	13.3	0.641
V	v28	SHS	127X127X6	0	10.2.1	0.249
D1	d44	SHS	254X254X11	3	13.2	0.190
D2	d38	SHS	203X203X11	0	10.2.1	0.206
COL	col32	WWF	1800X632	3	13.4.1	0.369

SHS $d \times d \times t \equiv$ square hollow structural section;
depth d (mm), width d (mm) and thickness t (mm)

WWF $d \times m \equiv$ welded wide-flange section;
depth d (mm) and mass (m) (kg/m)

```
Recommendation: 2.1
  Re-specify a shape category that has larger available sections.
```

```
Recommendation: 2.2 - for axial members -
  If possible, reduce unbraced lengths.

Recommendation: 2.3 - for flexural members -
  If possible, reduce unbraced length of compression flange.
```

The ‘improved fabrication rule’ has determined that the section width of the web diagonal members (group D1) is wider than that of the bottom chord members (group CHbot). Recommendation 1.1 suggests to limit either the minimum section depth for the chord members or the maximum section depth for the diagonal members. In fact, the user may choose to impose either or both limits and have the expert system return to the Solution stage to regenerate a corresponding new final design (not shown here).

The ‘improved profile rule’ has determined that the final design sections for both the top chord group (CHtop) and the column group (COL) represent the largest available cross-sections from their respective databases. Recommendation 2.1 suggests that another shape category containing heavier sections might be more appropriate for these member groups. This is perhaps a good suggestion for the top chord members, for which a W shape could replace the SHS shape. However, this is not an appropriate suggestion for the column members since there are no larger shapes than WWF.

Recommendations 2.2 and 2.3 suggest that the unbraced lengths of the members of each fabrication group be reduced so as to increase member buckling strength. However, this suggestion is only valid if strength constraints or axial slenderness control the design of these groups. In fact, these strength-based recommendations are not meaningful for the mill frame because a deflection constraint is strongly controlling the design (see Tables 9 and 10). This suggests, then, that any strength-related recommendations of the Critical stage should be made conditional upon the relative activity of the strength constraints compared to the displacement constraints. A corresponding new rule could be added to the Knowledge Base that is activated when displacement constraints govern the design significantly more than strength constraints, and which informs the user of alternative actions that may be taken in this event. This ‘displacement governs rule’ would have the following general form (this rule has not yet been formally incorporated into the expert system):

```
rule:  displacement governs
If     stage = critique
       displacement governs = yes
then   strength-related recommendations = invalid
```

This rule would fire at the beginning of the Critique stage, after the final application of the ‘solution verify rule’ determines that a displacement constraint controls the design and the ‘displacement governs = yes’ condition is posted in the Context. The finding and recommendations of this rule for the present example could be as follows:

```
Finding: 3  [displacement governs rule]
   The following displacement constraint controls the design:
   ( X-disp.@ node 26 for load case 3 )

Recommendation: 3.1
   Ignore any strength-related recommendations from other Critique rules.

Recommendation: 3.2
   If possible, relax the displacement constraint bound.
```

Recommendation 3.1 is passive in that it simply informs the user that any strength-related recommendations from the other Critique rules are to be ignored as meaningless because the design is controlled in any event by stiffness conditions. On the other hand, recommendation 3.2 is pro-active in that it suggests to relax the governing displacement constraint bound for the design. In fact, providing it can be implemented, this recommendation makes good sense since it will lead to a lighter-weight design while, at most, causing the strength-related conditions to become more critical to the design. Sometimes the change in structure weight can be significant for only a moderate change in the displacement constraint bound. For the steel mill frame, for example, suppose the user relaxes the x-axis displacement bound at node 26 for load case 3 from 60 mm to 80 mm. Reapplication of the Solution stage for this modification results in a new final design that is 17.6% lighter than the previous design. For the new design, the displacement constraint for load case 3 still governs while the response ratios for all strength constraints have increased from their previous values (in fact, while some ratios increased more, the maximum response ratio for the strength constraints only nominally increased from 0.641 to 0.643). Evidently, then, recommendation 3.2 suggested by the proposed ‘displacement governs’ rule is certainly worthy of consideration for the mill frame design.

Acknowledgment The reported work has been sponsored by the Natural Sciences and Engineering Research Council of Canada under Grant A5306. Thanks are due to Dr. Gordon Cameron, who conducted the underlying research over the past few years while completing his graduate studies in Civil Engineering at the University of Waterloo.

References

[1] ___, (1987), 'SODA: Structural Optimization Design and Analysis,' Software co-authored by D. E. Grierson and G. E. Cameron, Waterloo Engineering Software, Waterloo, Ontario, Canada.

[2] Canadian Standards Association, (1984), *CAN3-S16.1-M84 Steel structures for buildings (limit states design).*

[3] American Institute of Steel Construction, Inc., (1978), *Specification for the design, fabrication and erection of structural steel for buildings.*

[4] American Institute of Steel Construction, Inc., (1986), *Load and resistance factor design specification for structural steel buildings.*

[5] Forgy, C. L., (1985), 'OPS83 User's manual and report,' Production Systems Technologies, Pittsburgh, Pa., U.S.A.

[6] Grierson, D. E. and Cameron, G. E., (1988), 'A knowledge-based expert system for computer automated structural design,' *Journal of Computers and Structures* **30**, 3, 741–745.

[7] Cameron, G. E., (1989), 'A knowledge based expert system for computer automated structural steel design,' Ph.D. Thesis, University of Waterloo, Waterloo, Ontario, Canada.

[8] Fleury, C., (1979), 'Structural weight optimization by dual methods of convex programming,' *Int. J. of Num. Meth. in Engng.* **14**, 1761–1783.

[9] Grierson, D. E. and Cameron, G. E., (1989), 'Developing an expert system for structural steel design: issues and items,' *Proceedings of AIENG'89, The Fourth International Conference on Artificial Intelligence in Engineering*, Cambridge, U.K., July 10–14.

[10] Cameron, G. E. and Grierson, D. E., (1989), 'An expert system for standards-based structural steel design,' *Proceedings of Sixth ASCE Conference on Computing in Civil Engineering*, Atlanta, Ga., U.S.A., September 11–13.

A Knowledge-Based Model for Structural Design

B.H.V. Topping and B. Kumar
Department of Civil Engineering
Heriot-Watt University, Riccarton,
Edinburgh, EH14 4AS, United Kingdom

Abstract This paper presents a model for structural design based on ideas from artificial intelligence, particularly knowledge-based technology. The model presented is different from conventional models in that there is a more elaborate treatment of the reasoning processes involved in the overall design process. The model is thought to be more amenable to automation. Implementation of part of the model is presented in reference [1].

1 Introduction

This paper presents a model for structural design which forms the framework for the development of a knowledge-based system for structural design. Components of the model are presented here. The model presented here is called INDEX, which stands for INDustrial Building Design EXpert. The reason for calling it INDustrial Building Design EXpert is that low rise industrial steel buildings are used as examples to illustrate the model. However, it is suggested that the model is general enough for all structural design.

In section 2 a description of the structural design process is presented. Section 3 discusses some aspects of the DESTINY model [2]. Section 4 describes the INDEX model. Section 5 is a comparison of the two models highlighting the differences between the two. Finally, section 6 contains the summary and conclusions for this paper.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 101–121.
© 1992 *Kluwer Academic Publishers.*

2 Structural Design

2.1 The Process

A civil engineering structure may be defined as an entity which will withstand imposed design loads and transmit them to the foundations. In doing so, the structure must fulfill certain engineering and other architectural constraints. The structural design process includes the proportioning and sizing of such a structure to ensure the appropriate levels of safety and serviceability specified in design documents such as Codes of Practice and Building Regulations. The whole design process may be divided into three distinct stages:

- **Preliminary design :** In this stage, the functional requirements and constraints are synthesised into a preliminary design concept. This involves the selection of a potential structural configuration satisfying layout and spatial constraints. This stage frequently includes an approximate analysis to evaluate the response of the alternative candidate structures selected for further consideration.

- **Detailed design :** This involves the detailed design of the candidate structure selected in the preliminary design and consists of the following three sub-stages:

  a. structural analysis ;

  b. proportioning and sizing the structural members ; and

  c. checking all the applicable design constraints.

  This stage typically consists of several iterations between *analysis, proportioning and sizing* to ensure that all applicable constraints are satisfied with economy of design. Most of these constraints are specified in the applicable design codes. There may also be a number of external constraints, such as restrictions on the height of a structure. A large and significant deviation in the properties of the components assumed at the analysis and proportioning stages might necessitate another analysis, proportion and sizing-check cycle. This is typical of most design problems. The iteration continues until a satisfactory design is determined. In some cases, there may be a return to the preliminary design stage resulting in a revision of the chosen structural concept.

- **Design documentation :** Detailing of the different components and preparation of the design documents.

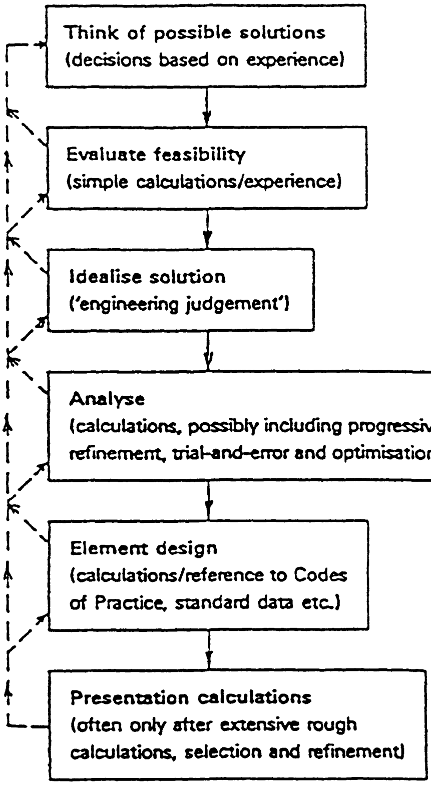

Think of possible solutions
(decisions based on experience)
Evaluate feasibility
(simple calculations/experience)
Idealise solution
('engineering judgement')
Analyse
(calculations, possibly including progressive refinement, trial-and-error and optimisation)
Element design
(calculations/reference to Codes of Practice, standard data etc.)
Presentation calculations
(often only after extensive rough calculations, selection and refinement)

Figure 1. *The different stages of structural design after Plank and Bell [3]*

2.2 Discussion

Figure 1, after Bell and Plank [3], shows the different stages of the structural design process and indicates the influencing factors (experience, heuristics etc.) at every stage. It is important to note the feedback that may become necessary at any stage, when the designer may be forced to go back to almost any earlier stage and reconsider his decisions. This aspect of design is the most difficult to incorporate in computer programs.

The above description of the structural design process is only a description of the different stages in structural design. However, the more important aspect of structural design is the inference mechanism involved. As discussed in the previous paragraph, the designer may be forced to go back and reconsider his earlier decisions and even change them in certain circumstances, as shown in Figure 1. In terms of Artificial Intelligence (AI), it would be said that there can be recurring changes in the *current set of beliefs* throughout the design process. This may arise owing to a sudden change of specifications or the emergence of new constraints, possibly in conflict with the existing ones. For a detailed discussion on the different aspects of *non-monotonic* reasoning, refer to [4, 5]. A more detailed treatment of the *non-monotonic* nature of the structural design process and ways of overcoming it is presented in [6].

3 The DESTINY Model

3.1 Architecture

The DESTINY model is based on a blackboard architecture [7]. The sole purpose behind selecting this architecture is to facilitate communication between the different experts involved in the structural design process, e.g., architects, space planners, service engineers and so on. Figure 2 is a schematic representation of the DESTINY model.

3.2 Blackboard

DESTINY's blackboard is divided into two parts. The first part is called the *Working level*, which contains entries relating to the execution of the various knowledge modules (KMs). The second part is split into eight levels: viz., *Top, Functional, Material, 3D, 2D, Location, Components and Property-Response.* These levels contain entries relating to the different stages of the design process and may be seen as a hierarchical decomposition of the building design process, i.e., they define the abstraction hierarchy of the design entities. Figure 3 shows the abstraction hierarchy on the blackboard of DESTINY [2].

3.3 Knowledge-Base

The knowledge-base of DESTINY is organised into a hierarchy of three levels. Each level comprises a number of knowledge modules each performing a particular task in the design process. The three levels of DESTINY's knowledge base are:

1. the strategy level;
2. the specialist level; and
3. the resource level.

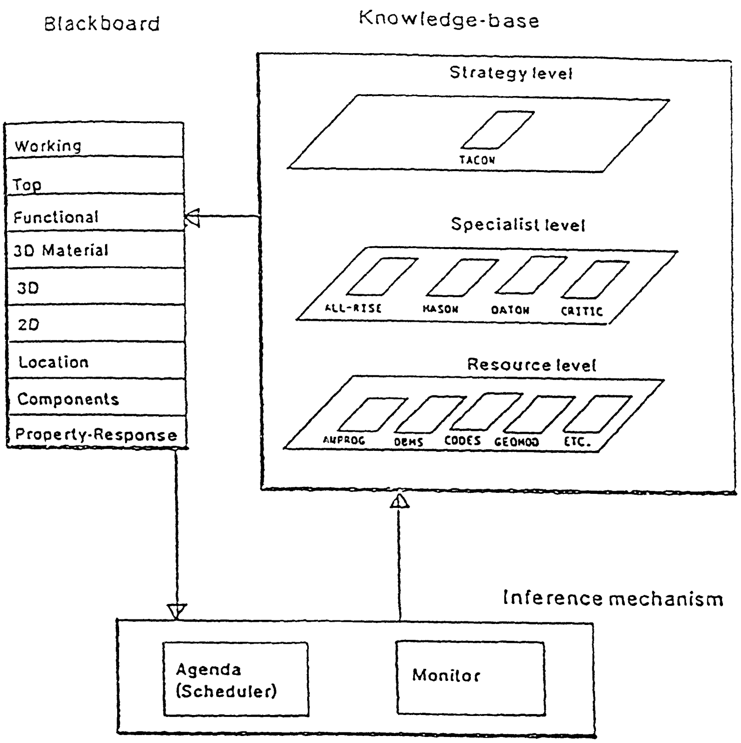


Figure 2. *Schematic Model of DESTINY [2]*

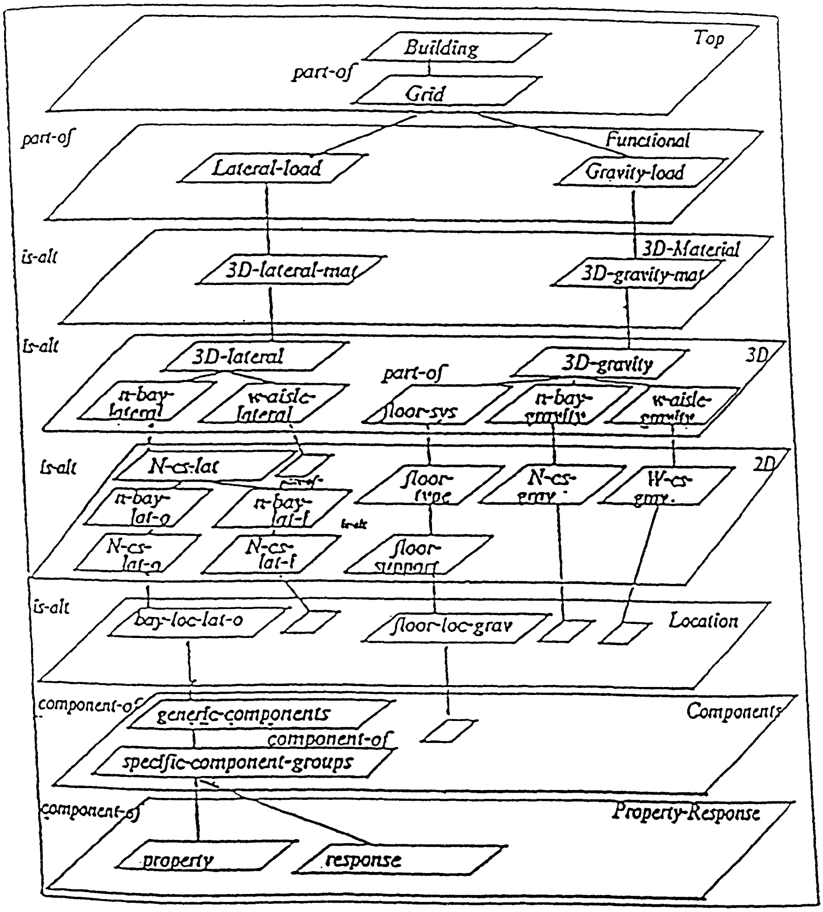


Figure 3. *Abstraction hierarchy levels on the blackboard of DESTINY [2]*

A brief description of these levels may be found in reference [2]. A relatively detailed description of the modules relevant to the discussions of this paper will be presented here. The module that is directly relevant to the extension of the DESTINY model proposed by this work is CRITIC. The purpose of this module is to check whether a design generated by the other modules (i.e., ALL-RISE, MASON and DATON) is satisfactory to perform the intended functions. This task of CRITIC is divided into two sub-tasks: viz., *Criticize and Evaluate*. The purpose of the *Criticize* sub-task is to assign one of the following four values to a design:

1. unsatisfactory;
2. modifiable;
3. fixable;
4. satisfactory.

There are four sets of production rules which determine one of these values for a particular design. The different cases which determine when a particular design is assigned one of these values are discussed below:

1. *Unsatisfactory* - this value is assigned to a design if the intended behaviour of the structure is not achieved.
2. *Modifiable* - this value is assigned to the design if there are significant differences between the assumed and the computed properties of the structure. With a *modifiable* design a re-analysis is recommended.
3. *Fixable* - this value is assigned to the design if there are minor differences between the assumed and the computed properties of the structure. A *fixable* design is one which does not have to undergo a re-analysis and requires just minor adjustment to some of its parameters.
4. *Satisfactory* - this value is assigned to the design if it satisfies all the specifications previously laid down.

Once a design is found to be *satisfactory* then a detailed evaluation is carried out by CRITIC as the *Evaluate* sub-task. For the descriptions of other knowledge modules reference should be made to reference [2].

3.4 Inference Mechanism

As with any other blackboard system, the inference mechanism of DESTINY also consists of an *agenda* and a *monitor*. The *agenda* contains a list of the sequence of *specialist level* knowledge modules to be executed from the elements of the following set:

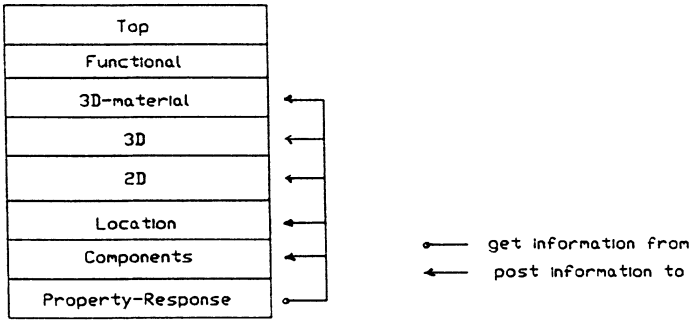


Figure 4. *The blackboard levels at which CRITIC posts and receives information adapted for reference [2]*

```
{ALL-RISE MASON DATON CRITIC}
```

The initial agenda called *Specialist Agenda* (SPA) set by TACON is:

```
{ALL-RISE DATON MASON DATON CRITIC}
```

The preliminary design is done by ALL-RISE and DATON. Once the agenda has been set up by the *strategy level* knowledge module, TACON, the *monitor* takes the first module from it and executes it. All the modules in the agenda are executed sequentially until it is empty. All the sub-tasks of a module are also executed sequentially until the module is completed. TACON is activated only after the execution of CRITIC. Conceptually, TACON may be activated at any stage when a more flexible mechanism is required.

3.5 Interactions between the knowledge modules

Figure 4 shows the levels on the blackboard to or from which the CRITIC module posts and retrieves information. Arrows indicate posting information while the circles indicate retrieval of information.

4 The INDEX Model

4.1 Architecture

The architecture of the INDEX model is the same as the DESTINY model, i.e., the blackboard architecture. The reasons for selecting this architecture are the same as those for DESTINY.

4.2 Blackboard

The abstraction hierarchy on the blackboard of INDEX is similar in nature to that on the blackboard of DESTINY since the abstraction hierarchy is essentially a translation of the stages in the structural design process which are the same in both cases. This is why a separate abstraction hierarchy is not given for INDEX.

4.3 Knowledge-Base

4.3.1 Brief description The knowledge base of INDEX consists of a number of knowledge modules as shown in figure 5. The knowledge modules are organised into a hierarchy of two levels, the specialist level and the resource level. The knowledge modules at the specialist level consist mainly of heuristics and other knowledge which are specialist-dependant. The knowledge modules at the resource level consist mainly of textbook knowledge. All the knowledge modules contain declarative as well as procedural knowledge. A brief description of the knowledge modules at the different levels is given below :

- **Specialist level** : This consists of knowledge modules primarily containing experience-based heuristics, but, some textbook knowledge is also stored at this level. This level consists of the following knowledge modules :

    - ALTSEL : This module is responsible for the ALTernative SELection of the feasible structural systems and decides on different design parameters for the required frame spacing, or the choice of a single or a multi-bay system etc.
    - STRANEX : This module carries out the modelling and analysis of the chosen structural system by ALTSEL.
    - DETDEX : This carries out the detailed design, i.e. detailed proportioning and sizing of the components of the chosen structure.
    - EVALUATOR : This module evaluates the different alternatives generated by the system.
    - OPTEX : This module consists of various heuristics and rules to be used for the optimisation of the structures.
    - DESCON : This module is responsible for solving problems arising out of a change of specifications or constraints described in section 2.2. DESCON acts as a DESign CONsultant to the other modules in such situations.

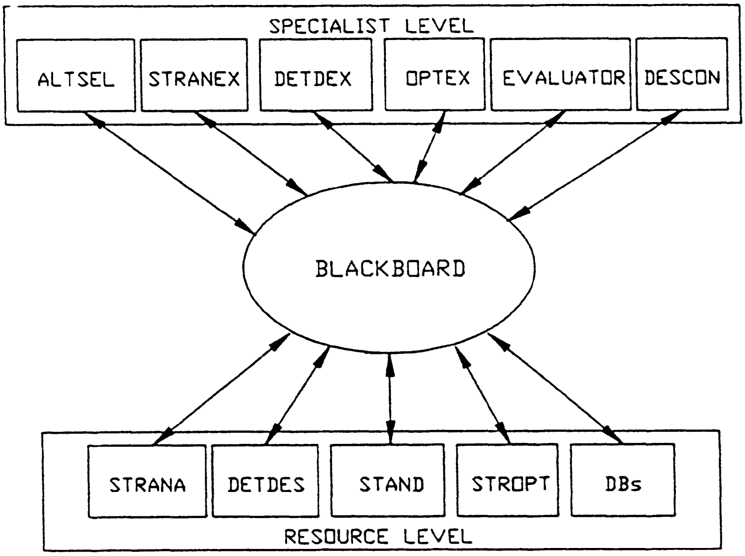


Figure 5. *Schematic model of INDEX*

- **Resource level :** This level generally consists of algorithmic programs such as structural analysis programs, standard codes, optimisation routines etc. The knowledge modules at this level consist of the following:

  - STRANA : This module includes the STRuctural ANAlysis programs.
  - DETDES : This module is responsible for the DETailed DESign of the structure, i.e. detailed sizing of the components of the structure.
  - STAND : This module includes the provisions of the applicable STANDards and is responsible for checking these standard constraints.
  - STOPT : This module consists of STructural OPTimisation routines.
  - DBs : These DataBases include the different dimensions and sectional properties of various structural sections, e.g. UBs, UCs etc.

4.3.2 Detailed description This section describes the Specialist level knowledge modules in some detail.

1. **ALTSEL**

   The main tasks for this module are as follows:

   - selection of the feasible alternative structural systems;
   - undertaking a preliminary analysis of the structural systems;
   - undertaking a preliminary sizing and proportioning of the components of the systems;
   - undertaking preliminary checks on the components;
   - posting relevant constraints for each of the alternative systems to the blackboard for later use by the other modules; and
   - undertaking a preliminary evaluation of the systems.

2. **STRANEX**

   The main tasks performed by the STRANEX module are as follows:

   - to model the structural systems generated by ALTSEL;
   - to select the structural analysis strategy, i.e., the appropriate type of analysis;
   - to prepare the input data for the analysis program; and
   - to return the output data from the analysis program.

   STRANEX can be seen to be performing a similar task as SACON [8]. The different sub-modules of STRANEX are as follows:

   - MODELLER - which models the structure for the analysis;
   - LOADEX - which decides the type and magnitude of loadings imposed on the structure;
   - PLANNER - which determines the appropriate analysis program to be used; and
   - INTERFACE - which prepares the data for the analysis program and receives the output data back from it.
   - The actual analysis is carried out by the analysis programs at the *resource level* . This knowledge module needs an interface with the analysis programs. An interface between PROLOG and FORTRAN77 was implemented for this purpose, details of which can be found in [9].

3. **DETDEX**

   The tasks performed by DETDEX are as follows:

   - to size the different components of the structure;
   - to select the applicable provisions of the Codes of Practice; and
   - to check the constraints prescribed by the Codes as well as other soft constraints;

   The functions of this module can be seen to be quite similar to that of SPEX [10].

4. **OPTEX**

   This module has the following tasks:

   - to formulate a model for optimisation of the structure;
   - to select the appropriate optimisation algorithm to be used;
   - to prepare the input data for the optimisation program; and
   - to receive the output from the optimisation program.

   The following are the sub-modules of this module which perform the above-mentioned tasks:

   - MODELLER - formulates an optimisation model;
   - PLANNER - determines the appropriate optimisation algorithm to be invoked;
   - INTERFACE - sends data to and receives it from the optimisation program.

This module also needs an interface with the optimisation programs. The same interface between PROLOG and FORTRAN77 discussed in reference [9] is used for the purpose.

5. **DESCON**

This module's function is to propose a solution to a design or partial design which is not satisfactory owing to a change of specification or a violation of some constraints. Thus, if a design is *unsatisfactory*, the following two possibilities exist:

- the design is either *modifiable* ; or
- the design is not modifiable and a *re-design* has to be undertaken.

In the case of a modifiable design, again the following two possibilities exist, depending upon the extent and nature of modification to be carried out:

- the modification will require a re-analysis of the structure; or
- the modification will not require a re-analysis of the structure.

Based on the above criteria, the task of this module is to decide if a design is one of the following:

- modifiable; or
- re-designable.

This module is similar to the CRITIC module of DESTINY. However, there are very significant differences in their scope and operation which will be discussed in section 5.

Once a design or partial design is categorised as discussed above, DESCON's tasks also include the following:

- to formulate the revised set of constraints in cases of a *modifiable* design;
- to suggest the exact nature of modifications to be carried out; and
- to post constraints to the appropriate modules.

The need for this additional module at the Specialist level will be discussed in section 5. This module comprises the most important difference between the DESTINY and INDEX models.

6. **EVALUATOR**

Once all the alternatives generated by ALTSEL have been designed *satisfactorily*, all of them are passed to this module. The task before this module is to evaluate all the designs based on different criteria and rank them accordingly. This module was not developed in this work.

4.4 Inference Mechanism

Unlike DESTINY, the inference mechanism of INDEX is handled in two ways. In routine situations, the sequence of execution of the knowledge modules is predefined, which is as follows:

```
(ALTSEL->STRANEX->DETEX->OPTEX->EVALUATOR)
```

Clearly, this sequence does not include the DESCON module. The reason is that DESCON may not be invoked in every case. Depending on the outcome of the other modules, DESCON may or may not be invoked. DESCON does not have to be directly invoked since the control mechanism rests mostly in the hands of DESCON itself. In such cases, DESCON sets up the sequence of execution of any other module. The different cases that may arise before DESCON is invoked will be discussed in section 5.2.

5 Comparison between the DESTINY and INDEX models

Although the INDEX model is based on the DESTINY model, there are significant differences between the two. The INDEX model can be seen as an extension of the DESTINY model as will become clear later.

Figure 5 indicates that the INDEX model has two levels, viz., the specialist and the resource level. For the sake of uniformity, the terminologies used are the same as for the DESTINY model (figure 2). However, the DESTINY model proposes an additional level called the Strategy level. INDEX does not recognise the need for this level. The reason is that in DESTINY, the execution of modules is *sequential* and does not require a separate set of rules to define. The sequence may be predefined, which is how it is undertaken in INDEX. However, if there is a requirement to change the sequence, there will be a need for a set of rules. This will become clear from the discussions in section 5.2.

Apart from this difference, the other major difference is that of an additional module at the specialist level, DESCON (In fact, there are two additional modules at the specialist level: viz., OPTEX and DESCON. However, the addition of OPTEX is not very important from the conceptual point of view of the design. It can only be considered to be an additional facility of the system). This difference is an important one as the functions of this module are considered to be vital for an integrated design system. The remaining modules at the specialist level can be seen to be quite similar in both cases.

5.1 Differences in the knowledge base

It was decided that an additional module was required to tackle some (if not all) problems detected by the CRITIC module of DESTINY. The detection of a problem (e.g., nonconformity to standard requirements) is done by the respective modules themselves in INDEX. The DESTINY model proposes passing back the control to either MASON or DATON. Unlike DESTINY, it is proposed to transfer the control from whichever module the problem is detected in to DESCON in every case. DESCON in turn, transfers the control to one of the other three design modules at the specialist level: viz., ALTSEL, STRANEX or DETDEX. The only exception is the *satisfactory* design in which case the design terminates at EVALUATOR. In some cases, DESCON considers situations which may not have clearly defined constraints in structural engineering terms. For example, at some stage, a purlin may have to be removed after it has been designed. The removal of a purlin is not an engineering constraint but may give rise to many of them. The input to DESCON in such cases relates only to the fact that the purlin has to be removed and not that the constraints relating to the stability of the rafters are violated. In these circumstances, DESCON's task is to infer the effects of any such changes to the existing design or partial design and formulate new constraints and propagate them to the appropriate modules. CRITIC can only detect the problem and suggest whether the structure is:

- unsatisfactory; or
- modifiable; or
- fixable; or
- satisfactory.

The DESTINY model does not indicate whether CRITIC is also responsible for suggesting to the appropriate module the precise modifications to be carried out to a design. It was proposed that very specialised knowledge may be required to accomplish such a task and, hence, the additional module, DESCON was developed. The basis on which CRITIC suggests whether a design is *modifiable* or *fixable* is whether or not a re-analysis of the structure is required. The only basis used by CRITIC to decide on such a re-analysis is the difference between the assumed and computed properties of the structure. In fact, such a decision could be quite a subjective one and may require other considerations to be taken into account. Furthermore, CRITIC cannot handle the emergence of new constraints or a change in specifications. CRITIC's scope is thus limited, and suffers from an important and serious drawback. DESCON of INDEX may be seen as an extension of CRITIC of DESTINY in that it also handles situations where some constraint suddenly emerges as a consequence of either:

(i) having been overlooked or ignored earlier; or

(ii) abrupt changes requested by the client; or

(iii) poor co-ordination between the structural and some other designer (e.g., services).

DESCON's task is to suggest a *qualitative solution*. Subsequently, the actual *quantitative solution* is carried out by whichever module to which DESCON passes control. DESCON is different from the other specialist modules in that its task is to suggest changes to an existing or proposed design rather than designing a structure from scratch given only the specifications. Its task is considered to be more difficult as it is required to explore in a more *intelligent* way the alternative solutions which will force least modifications to the existing or proposed design at the least expense. The problem is more critical in the case of an existing structure or an already fabricated design. The most knowledge-intensive part of such an exercise is to find out the design *entities* which play the most important role in a particular case. The details of the problem-solving strategies used by DESCON are discussed in detail in reference [11].

5.2 Differences in the interactions between the different modules

Figure 4 shows the levels on the blackboard of DESTINY to which CRITIC posts information. It is clear from this figure that in no case does it pass control back to the level where the feasible structural systems are selected. This is one improvement which the INDEX model proposes. In the case of a *re-designable* design, control is passed back to the ALTSEL module where a completely new alternative is selected form those generated earlier by ALTSEL.

Figures 6 and 7 are diagrammatic representations of the interactions between different knowledge modules of INDEX and DESTINY repectively. Yet another major difference between the two models is illustrated by these two figures. The interactions between the modules of INDEX will be explained in some detail before highlighting the differences from the interactions in the DESTINY model.

By considering figure 6, it is quite clear that the knowledge modules, ALTSEL, STRANEX, DETDEX, OPTEX and EVALUATOR are executed sequentially in a routine case. DESCON may or may not be invoked in particular cases. DESCON is only invoked when a problem is detected by any module. The transfer of control to DESCON can take place at any stage of design apart from the routine case when complete designs are passed to EVALUATOR to evaluate. Whenever a new constraint arises or there is a sudden change of specification, control is passed to DESCON. The transfer of control from DESCON to the other modules depends

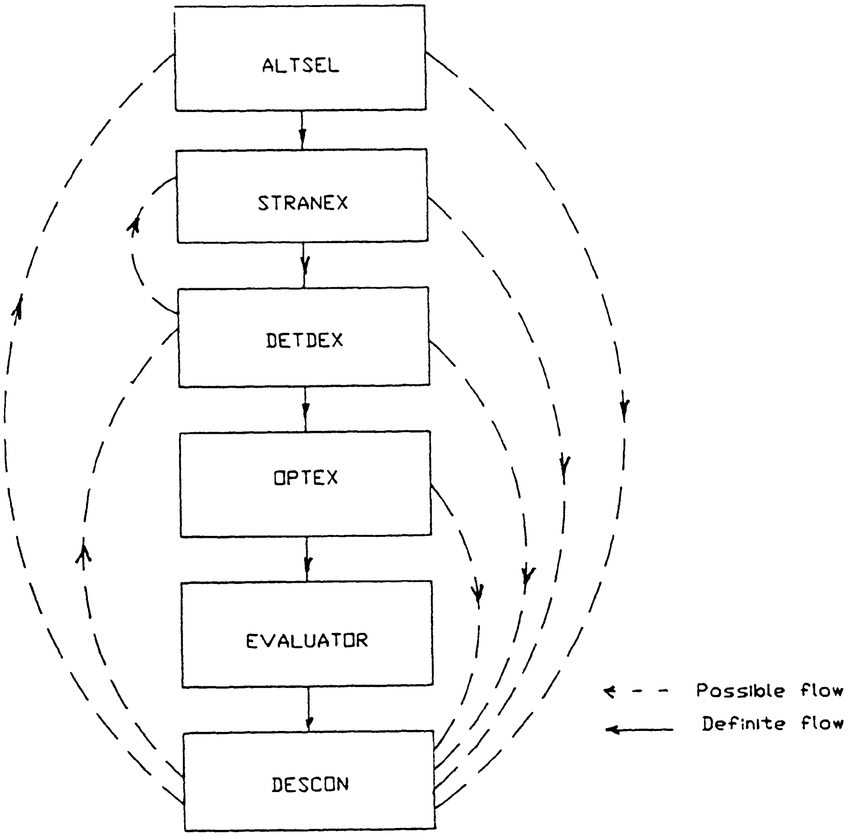


Figure 6. *Interaction between the KMs in the INDEX model*

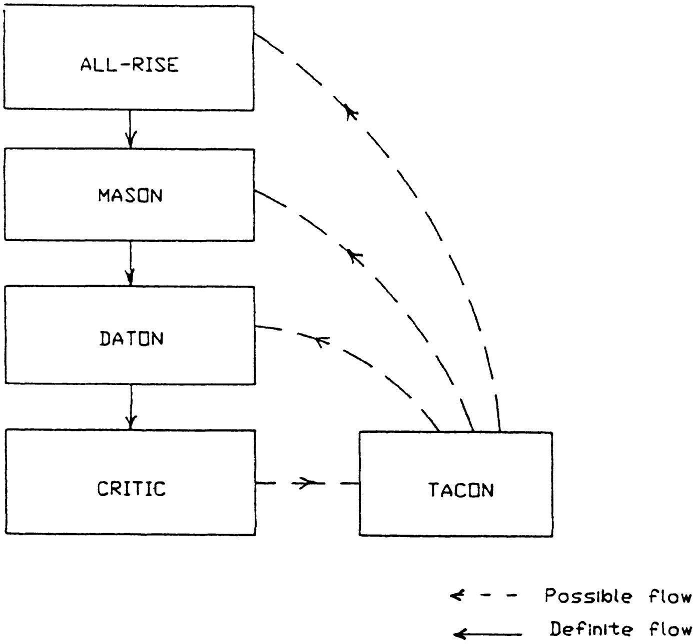


Figure 7. *Interaction between the KMs in the DESTINY model*

on the nature of the problem detected and the consequent decision taken by DESCON. The most straightforward case handled by DESCON is the *re-designable* design where it simply passes control to the ALTSEL module. The most complex case handled by DESCON is the *modifiable* design. Here, the precise nature of modifiability is the main problem for DESCON. Transfer of control is always to DETDEX. However, depending on whether or not a re-analysis is required, DETDEX has to pass control to either STRANEX or to EVALUATOR after carrying out standards checking. For example, if it is suggested by DESCON that the design is *modifiable* and that the exact nature of modification to be carried out is to increase the section size of one of the members of the structure, DETDEX will pass control to STRANEX to carry out a re-analysis. Even if control is passed back to STRANEX, the standards provisions still have to be checked before the design may be passed to EVALUATOR. This process continues until a *satisfactory* design has been found. Therefore every successful design terminates at EVALUATOR.

In DESTINY, the elements of the Specialist Agenda (SPA) are set by the Strategy level knowledge module, TACON. The TACON module is activated only after the execution of the CRITIC module. Although there is a mention [2] of the possibility of activating TACON after the execution of each specialist level module, the mechanism is not very clear. In the INDEX model, it is proposed that there must be a facility in the model to allow for any *non-monotonicity* whenever it arises. It is thought that such a situation may arise at any stage and even partial designs may have to be assessed by DESCON. DESTINY lacks this important feature.

In INDEX, part of DESCON's purpose can be seen to be quite similar to that of TACON's in DESTINY: i.e., invoking the specialist KMs. Invocation of DESCON is quite similar to that of TACON in that every time any module posts information to the blackboard indicates a problem. But, the difference lies in the scope and purpose of knowledge in DESCON.

As discussed earlier, DESCON may be invoked in the following situations:

- Whenever a change of specification or constraint occurs. This can happen in the following situations:

    1. at any stage in the design process, in which case even a *partial design* may be passed to it; or
    2. after the design has been completed, in which case the complete design will be passed to it.

- whenever a violation of constraints occurs.

The transfer of control to DESCON may be undertaken by any of the specialist KMs. In contrast, the CRITIC module is only activated after the other KMs of the SPA have been executed in the sequence prescribed by the SPA.

The scope and purpose of knowledge in DESCON has already been explained earlier. It is an extension of the CRITIC module of the DESTINY model.

6 Summary and Conclusions

A model for integrated structural design, INDEX, was presented. The model presented was an extension of an earlier model, DESTINY. Some limitations of the DESTINY model were pointed out and the INDEX model resulted from incorporating the missing features of the DESTINY model. It was concluded that the structural design process was a *non-monotonic* process and any system developed for structural design needed to have facilities to allow for this *non-monotonocity*. All the suggested extensions to the DESTINY model centre on this feature of design which is held to be fundamental.

References

[1] Kumar, B., Topping, B.H.V., "Knowledge-Based Preliminary Design of (low rise) Industrial Buildings", *in this volume, Optimization and Decision Support Systems in Civil Engineering, Kluwer Academic Publishers, 1991.*

[2] Sriram, D., "Knowledge-based Approaches to Structural Design", *Ph.D. Dissertation, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, U.S.A., 1986.*

[3] Bell, T., Plank, R.J., "Microcomputers in Civil Engineering", *Construction Press, London and New York, 1985.*

[4] McDermott, D., Doyle, J., "Non-monotonic Logic I", *Artificial Intelligence, Vol. 13, 1980.*

[5] Reiter, R., "On Reasoning by Default", *TINLAP-2, University of Illinois at Urbana-Champaign, 210-218, 1978.*

[6] Kumar, B., Topping, B.H.V., "Non-Monotonic Reasoning for Structural Design", *Civil Engineering Systems, v.7, n.4, 209-218, 1990.*

[7] Hayes-Roth, B., "A Blackboard Architecture for Control", *Artificial Intelligence, Vol. 26, 251-321, 1985.*

[8] Bennett, J.S. et. al., "SACON: A Knowledge-based Consultant for Structural Analysis", *Technical Report STAN-CS-78-699, Stanford University, Palo Alto, California, U.S.A., 1978.*

[9] Kumar, B., Chung, P.W.H., Topping, B.H.V., "Approaches to FORTRAN-PROLOG interfacing for an Expert System Environment" *in "The Application of Artificial Intelligence Techniques to Civil and Structural Engineering", Ed., Topping, B.H.V., Civil-Comp Press, Edinburgh, 15-20, 1987.*

[10] Garrett, J.H., Jr., Fenves, S.J., "A Knowledge-based Standards Processor for Structural Component Design," *Report No. R-86-157, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, U.S.A., 1986.*

[11] Kumar, B., Topping, B.H.V., "Knowledge-Based Processing for Structural Design", *Proceedings of the Institution of Civil Engineers, Part 1, v. 90, 421-446, 1991.*

Knowledge-Based Preliminary Design of Industrial Buildings

B. Kumar and B.H.V. Topping
Department of Civil Engineering
Heriot-Watt University, Riccarton,
Edinburgh, EH14 4AS, United Kingdom

Abstract This paper presents details concerning the implementation of one of the knowledge modules for the structural design model presented in reference [1]. A blackboard architecture was proposed for the overall model as well as its different modules. The module described here was called ALTSEL and was responsible for selecting (generating and evaluating) alternative structural systems for low-rise industrial buildings.

1 Introduction

The overall INDEX model has been described in [1, 2]. This paper describes the preliminary design module of INDEX called ALTSEL. ALTSEL represents a simple but effective rule-based prototype for the preliminary design of industrial buildings. The use of rule-based programming is illustrated by describing the development of ALTSEL. A description of knowledge elicitation techniques and some practical lessons learnt from using them will also be discussed. Some useful features of the Edinburgh Prolog Blackboard Shell which simplified the development of ALTSEL will be described. These shed light on the utility of blackboard architecture for a knowledge-based design system.

2 The AI tool used in this implemetation

The Edinburgh Prolog Blackboard Shell (EPBS)[3] was used in the development of DESCON. The rule syntax of this knowledge-based system shell has the following form:

```
if        Condition
then      Goal
```

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 123–147.
© 1992 *Kluwer Academic Publishers.*

```
          to          Effect
          est         Est.
```

2.1 Components of a rule in the EPBS

The following sections briefly describe each of the components of a rule.

2.1.1 Conditions of a rule The 'Conditions' of a rule are a combination of tests concerning the blackboard. Essentially they consists of testing the presence or absence of a certain type of entry on the blackboard.

2.1.2 Goal of a rule The 'Goal' is a Prolog procedure executed by each rule. This gives the user the flexibility of only partially specifying an entry when writing the rule and further specification comes from either the success of the condition or by calling the 'Goal'. In cases where no further specification is required other than that prescribed by the condition, the Prolog goal 'true' may be called which will succeed immediately.

2.1.3 Effect of a rule The effect of a rule may be one of the following:

- add[Index,Fact,Cf], which adds an entry Fact on the blackboard under the index Index with certainty factor Cf,
- or 'Action' which takes an action,
- or 'Delete' which deletes an entry from the blackboard,

where Index, Fact, Cf and Action are PROLOG terms.

2.1.4 'Est' of a rule 'Est' refers to the usefulness of a rule. By default, simple integers may be assigned to 'Est' in which case the 'usefulness' of the rules increases with the value of their 'Ests'. This default scheme may be over-ruled by assigning arbitrary Prolog terms to the Ests and these may be compared by a predicate defined by the developer. 'Est' provides a way of resolving conflicts regarding the firing of rules. In cases where more than one rule is eligible to be fired due to the success of their conditions, the rule with the lowest 'Est' is fired first, then the one with the next higher 'Est' and so on.

3 Some features and components of ALTSEL

The architecture of ALTSEL is a blackboard system. It consists of different knowledge-modules surrounding and communicating through the blackboard. The input to ALTSEL is the general layout and other spatial constraints of the building. Since generally the layout of the design is fixed by architectural design, the domain of the system is restricted to structural design.

3.1 Blackboard

The general description of the blackboard of INDEX, described in reference [1, 2], also apply ALTSEL. In fact, all the modules of INDEX can be seen to be separate knowledge-based prototypes. ALTSEL's blackboard is divided into different parts which contain entries posted to it by the different sub-modules of ALTSEL in the course of the solution process. The levels on the blackboard of ALTSEL may be seen as a hierarchial decomposition of the preliminary industrial building design process.

4 Knowledge Base Development

4.1 Knowledge Elicitation - Introduction

The construction of a knowledge-based expert system is an attempt to embody the knowledge of a particular expert within a computer program. The knowledge used in solving problems must be elicited from the expert to incorporate in the expert system. It is recognised that the elicitation of knowledge from experts is one of the major obstacles in the construction of expert systems. In many cases, the main reason for this is that experts find it hard to articulate and make explicit the knowledge they possess and use. An important part of a knowledge engineer's job is to help the expert to structure the domain knowledge and to identify and formalize the domain concepts. Although a number of knowledge elicitation methods do exist [4], the area is not well understood and few tools exist to mechanise the process.

In the following sections, a simple model of knowledge elicitation will first be presented followed by some specific techniques. Finally, in section 5, an account of the knowledge elicitation process for ALTSEL will be presented to highlight some of the practical issues.

4.1.1 A Framework For Knowledge Elicitation The framework is based on three generally accepted ideas:

- there are different types of knowledge;
- there are different knowledge elicitation methods for different types of knowledge; and
- the knowledge elicitation process may be divided into sub stages.

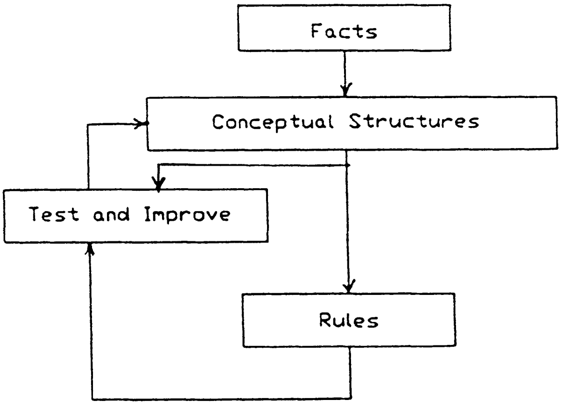


Figure 1. *Sequence of Knowledge Elicitation*

There is no doubt that there are different types of knowledge, even in a single domain of expertise. However, it is not clear how knowledge should be classified into different types. *"Finding a way to taxonomise knowledge on a principled basis is a difficult and ambitious task that has eluded philosophers for thousands of years"* [5]. For the practical purpose of building expert systems, knowledge may be conveniently divided into three types: *facts*, *conceptual structures* and *rules*. *Facts* are simply a glossary of terms and a list of domain entities. In an engineering domain, this type of knowledge may be a collection of engineering concepts and the names of the components of a particular structure or any other engineering artifact. The second type of knowledge, *conceptual structures*, describes the relationships between identified concepts and components. Finally, *rules* are the reasoning part of the domain knowledge. Facts and conceptual structures are reasonably static and are easier to elicit than rules. Figure 1 illustrates a simple but natural sequence of knowledge elicitation.

In each part of the cycle, a suitable elicitation technique should be used. Some studies have been carried out to match techniques with types of knowledge [4, 5]. In the next section, for each type of knowledge, a knowledge elicitation technique which has been identified as particularly suitable, is described.

4.1.2 Techniques There are two classes of techniques for knowledge elicitation as follows:

- **psychological techniques**, which involve some kind of interaction between the knowledge engineer (KE) and the domain expert (DE); and
- **machine induction**, in which the computer automatiaclly induces rules from examples.

For a domain such as structural design, *machine induction* seems inappropriate. Bloomfield [6] developed a set of criteria for selecting domains suitable for the elicitation of knowledge by *machine induction*. One such criterion is that *"any chosen domain must contain sufficient examples that it is possible to construct a training set which constitutes a comprehensive encapsulation of expertise in that domain"*. Structural design expertise cannot be completely encapsulated in examples. Hence, only *psychological techniques* are considered.

4.1.3 Interviews Direct interviewing is the technique most familiar to KEs and DEs. It is considered good practice to start the knowledge elicitation process using a technique with which the DE feels comfortable. An interview may range from an informal 'chat' to a highly structured discussion. Some interesting questions for an interview may be:

- if you had a good new graduate just starting to work for you what would you expect him to have learnt after six months ?
- you find a book concerning your application area which later turns out to be the book you wish you had started in the field. What chapter headings are in it ?

Using this technique, much information about the terminology and the main components of the domain may be generated in a relatively straightforward way. The problem is how to probe further so that ideas may be pursued to a greater depth. To ensure that an interview is productive, the KE should have a good questionnaire prepared beforehand to help him direct the discussion. In addition to open questions, he needs to have some clear and specific ones. The DE may also be asked to prepare and deliver an introductory lecture.

4.1.4 Concept Sorting As experts use specialist knowledge to solve problems they are likely to have a global perspective on how a domain is organised. Concept sorting is appropriate where there is a large set of concepts which need to be organised into a manageable form. The basic procedure is similar to the categorical knowledge elicitation technique described by Regan [7]:

1. collect a set of concepts in the domain. This may be obtained from the literature or from an introductory talk or from the DE;

2. write each concept on a small card;
3. ask the DE to sort the cards into groups;
4. ask the DE to label each group;
5. discuss with the DE each group to find out its characteristics;
6. ask the DE to specify the relationship between the groups and to organise them into a hierarchy.

4.1.5 Protocol analysis In this technique, the DE's behaviour is recorded (either video or audio) as they work through a problem or task, and the protocol is transcribed and analysed. In this way, the KE is not only given the answer to the problem but also the information about the problem solving process itself. In practice this technique is found to be very helpful. Though DEs may have difficulty in stating the general rules that they use, they can usually identify the specific rules which they are applying. However, it is easy for familiar ideas to be taken for granted, so they need to be kept aware of any tendencies towards omitting *trivial* details. For this technique to be effective a representative set of problems should be chosen, otherwise there could be serious errors of ommission.

There are three different ways of generating protocols:

- think-aloud protocols - the DE thinks aloud during the solving of a problem;
- retrospective verbalization - the DE solves a problem before reporting how it was solved;
- discussion protocols - a small number of DEs discuss with one another as they attempt to solve a problem.

Each of these variations has its own advantages and disadvantages. An important problem with think-aloud protocols is that the reporting may interfere with the DE's task performance. Related to this is any need to conform to real time constraints. For example, solving a mathematics problem allows the mathematician to stop and ponder. However, an operator dealing with an emergency may require immediate responses. These criteria may help when having to decide between think-aloud protocols and retrospective verbalization.

Expert system projects are often based on collaboration with a single DE. In fact most of the literature recommends this [8]. However, discussion protocols are helpful because they provide different perspectives on how a problem may be solved by clarifying alternatives and resolving conflicts. The problem here is that of managing the discussion. Avoiding the problem, the strategy that Mittal and Dym [9] adopted was to interview one DE at a time. Although this technique worked for them, it provides very little opportunity for the DEs to interact with one another and to discuss issues.

A potentially useful computer tool for collaborative problem-solving in face-to-face meetings is Colab, which has been created at Xerox Parc [10]. This project advocates the use of computers in meetings rather than a passive medium like chalkboards. The idea is that in the meeting room each person has a keyboard and mouse on his table and there is a very large screen to the front of the room. Each person may retrieve information from the computer and may easily write and draw on the screen by using the keyboard and mouse in front of him. In this mode of working a meeting may be dynamic and interactive, and at the same time all the text and sketches which have been generated in the meeting are stored in the computer. The abundance of information is conveniently accessible for analysis when needed.

4.1.6 Rapid Proto-typing The most obvious technique for testing and improving an expert system is rapid proto-typing. The DE is confronted with the behaviour of an unfinished version of the system which is modified in the light of his/her comments. Each iteration brings the behaviour of the system closer to completion although, since it is often carried out without a clearly defined notion of completion, it is perhaps better thought of as iteration towards adequate achievement.

4.1.7 Summary of the Techniques These are just some of the techniques that have been identified as useful. They should be viewed as complementary rather than mutually exclusive because different techniques may be used to capture different types of knowledge more effectively. Interviews are important for gaining an overall view of the domain; concept sorting is appropriate for structuring the domain; and protocol analysis is particularly helpful when collecting rules. The main point is that the KE needs to be aware that there are different techniques which may be applied. Their usefulness also depends very much on the individual KEs. Factors such as the KE's knowledge of the problem domain and how well he gets on with the DE are important.

From the description of different techniques, it should also be clear that feedback plays a very important role in knowledge elicitation. It is highly unlikely that a DE can impart all relevant knowledge at one meeting even if the domain is extremely simple. The question is then: *What form of feedback should be provided ?* An obvious but important comment is that what is fed back should be familiar to the DE so it may easily be understood and commented upon.

5 Knowledge Elicitation for ALTSEL

The following sections describe the experiences gained in the knowledge elicitation process undertaken for ALTSEL. The relevance of the different techniques described earlier will become evident in these sections.

5.1 Meeting the Experts

The KE contacted a consultancy company which specializes in designing industrial buildings. Four meetings took place, with each lasting approximately three hours. The following is a short commentary on what happened during each of these four meetings.

5.1.1 First Meeting At this meeting the KE met the DE, a design engineer with many years of experience. The DE knew that he had expertise and was sceptical that a computer could perform the same function. So, throughout this meeting, the KE tried to convince the DE by describing to him how expert systems work and showing him the listing and runs of a simple prototype design checker. The DE remained unconvinced. He had two basic doubts:

1. How could a computer reason except through obeying instructions?

2. Every design is different; how could a single set of rules apply to all designs?

The KE left the meeting frustrated and discouraged. Nonetheless, they agreed to have a second meeting two weeks later.

5.1.2 Second Meeting At this meeting, there were three DEs: the previous design engineer, another design engineer and an expert in computer aided design. The first part of the meeting was very much the same as the previous one with the KE trying to convince the DEs that expert system technology was workable.

However, this time the KE had a copy of a diagram with him which illustrated a abstraction hierarchy of the design of a building. The diagram (figure 2) is a simplified version of another diagram that the KE had found in the literature. The original abstraction diagram was developed by Sriram [11] for his work on a knowledge-based system for designing buildings. He showed this to the DEs who immediately identified that this reflected how they carried out design. In other words, the diagram helped the DEs conceptualise their own thinking processes and relate them to those of an expert system.

Some time later the KE was left with the second design engineer to work through a design problem that he had recently solved. The DE was quite happy to explain how he had made certain decisions when he was asked the question *"Why ?"*. The DE also pointed out some literature that practising engineers read.

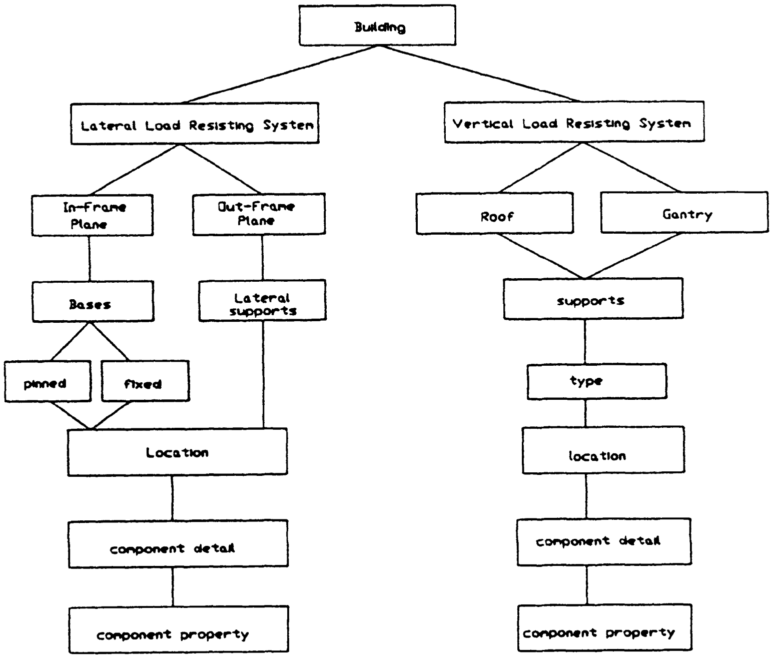
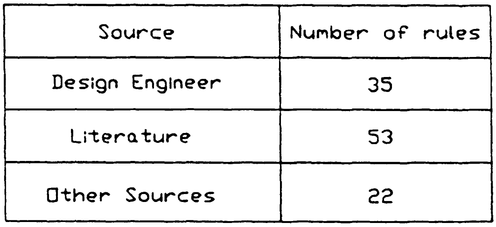


Figure 2. *A simplified abstraction hierarchy of the industrial building design process after Sriram [11]*



Source	Number of rules
Design Engineer	35
Literature	53
Other Sources	22

Figure 3. *Breakdown of the rules according to their sources*

From the informal protocol collected the KE was able to produce ten rules. Further, the KE was able to identify and glean more rules from the literature that he had read earlier. The KE then built a prototype which took a specification as input and produced alternative feasible structural systems as output, together with a recommendation of which of these alternatives was most favourable for further analysis and detailed design.

5.1.3 Third Meeting This meeting took place a month after the previous one. When the DE saw the runs and rules of the system, he was very surprised by the progress that had been made. He spent most of the time in this session commenting on the rules.

After this session the KE was able to refine his rule-set and try the system on other problems he had collected from literature.

5.1.4 Fourth Meeting At this meeting the DE introduced three new problems and described to the KE how he had solved them.

To date, the knowledge-base has over a hundred rules. The table in figure 3 gives a break down of the sources of the rules.

5.1.5 Discussion and Conclusion The most important lessons learnt from this project concerned the following:

- KE's familiarity with the domain; and
- KE's initial approach.

It is concluded that the factor that seemed to play the most important role in speeding up the knowledge elicitation process was the KE's familiarity with the domain. Computer scientists frequently claim to be equally effective knowledge engineers as one from the domain being considered. A very common opinion is that any person can be a good knowledge engineer after spending a little time in a DE's office. This was not the experience of this project. There were a number of occasions when the KE helped the DEs articulate their ideas. The only factor which seemed to help the KE in doing so was his familiarity with the domain. That sort of familiarity may not be acquired during a one week visit to an consultant's office.

Another useful feature which seemed to help the whole process was the abstraction hierarchy of the design process. Although a KE might not always be able to generate a relevant diagram by himself, he should be able to produce one with the assistance of the DE. The *concept sorting* procedure (described in section 4.1.4) is a good bottom-up technique to use. During the knowledge elicitation phase diagrams can form a useful part of the documentation for the system.

Protocol analysis, or more precisely, studying case histories, was found to be a very useful way of generating rules. However, it is interesting to note that only a third of the rules were elicited directly from the DE (see figure 3). Following through the information provided by the DE, looking for further or more detailed information yielded much.

The KE found that decomposing the problem, especially the preliminary design part, into sub-problems, at an early stage, was an extremely important step in formalising the domain knowledge. Once the problem was decomposed it not only helped the DE to recall and provide the relevant pieces of information but it also helped the KE to pick out relevant material from other sources. From the system construction point of view, it was also very helpful because the knowledge base could then be divided into smaller modules making them easier to maintain.

The DE was surprised and impressed by the result of proto-typing. It is certainly a very useful way of obtaining feedback from the DE. In this case it was disappointing that the DE could not see the prototype running but could only comment on the output of the program.

When using proto-typing as a technique to obtain feedback, the KE found it necessary to guard against letting the documentation slip. It is easy to get into the habit of making quick changes to the system without keeping a record of the changes made, thus making the system difficult to modify and maintain in the future.

At the time, it seemed quite obvious to approach the DEs by first convincing them about the potential of knowledge-based expert systems. As pointed out earlier, quite a number of hours were wasted in doing so. The lesson thus learnt was that an attempt should be made to collect information from the DEs without making overstated claims about the proposed system. It now seems important to stress the fact that the proposed system is only intended to assist the DEs and not to replace them. It seems important to reassure the designer of his supreme role in the design process in relation to any computer program. This point certainly appears to be a very important one.

5.1.6 Knowledge Representation - Introduction The knowledge representation formalism used in ALTSEL is the same as that for INDEX; namely *production rules*. The knowledge-base of ALTSEL is organised into different sub-modules as shown in figure 4. ALTSEL itself is at the *Specialist level* of INDEX and thus, almost all the knowledge contained in ALTSEL is heuristic and is obtained from either experts or the domain literature. The purpose of ALTSEL is as follows:

- selection of the feasible alternative structural systems;

- carrying out a preliminary analysis of the structural systems;

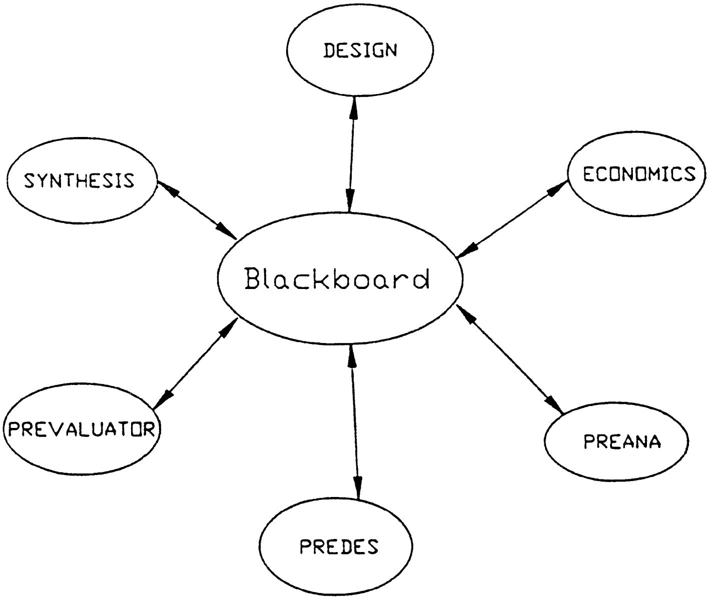


Figure 4. *The sub-modules of ALTSEL*

- carrying out a preliminary sizing and proportioning of the components of the systems;

- carrying out preliminary checks on the components;

- posting relevant constraints for each of the alternative system to the blackboard for a later use by the other modules; and

- carrying out a preliminary evaluation of the systems.

Section 5.1.8 describes the different knowledge modules of ALTSEL that perform these tasks.

5.1.7 Types of Constraints The identification and proper use of various constraints constitutes one of the most important aspect of any design. The different types of constraints considered by different knowledge modules of ALTSEL depend upon the task being performed. The constraints considered by the sub-modules of the ALTSEL module are mostly *external.* For a comprehensive description of different types of constraints in structural design, reference can be made to [11]. *External* constraints are the constraints which are not in the control of the designer. In other words, these constraints are *external* to the designer. These constraints are mostly governed by the requirements of the client.

In order to take the decisions listed above, various decisive factors must be determined first. On the basis of discussions with practising engineers and a study of the design literature used by them to assist in taking these decisions, it was concluded that the following parameters played the most decisive roles in the preliminary design of industrial buildings:

- span;

- loads;

- allowable pitch;

- intended industrial process to be carried out in the building; and

- any other client-related constraints.

Thus, all the rules in ALTSEL have one or more of the above-mentioned parameters as constraints to be satisfied. The constraints considered by the SYNTHESIS sub-module in deciding (about the) feasible lateral load resisting systems are shown in figure 5. Figure 6 shows the considerations used by the PREVALUATOR submodule in evaluating different feasible systems.

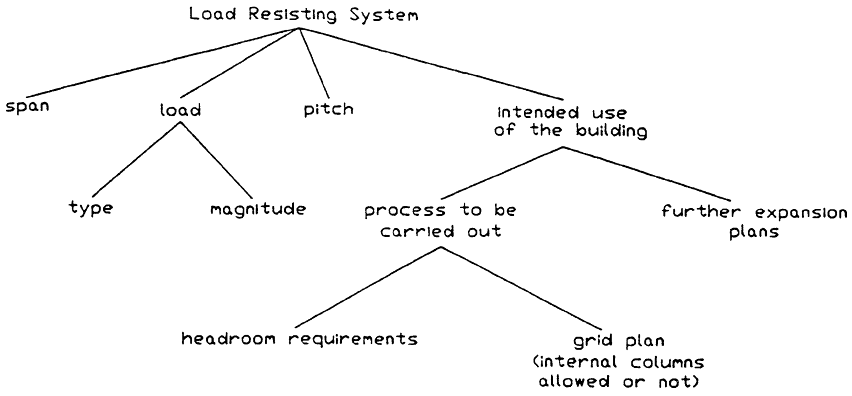

Load Resisting System
span
load
pitch
intended use of the building
type
magnitude
process to be carried out
further expansion plans
headroom requirements
grid plan (internal columns allowed or not)

Figure 5. *Constraints considered by SYNTHESIS*

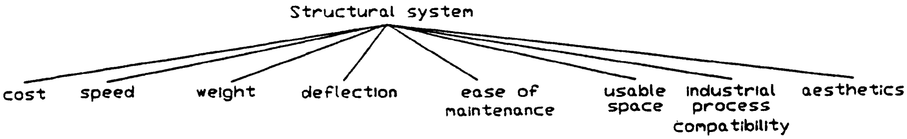


Figure 6. *Evaluation characteristics considered by PREVALUATOR*

5.1.8 Sub-modules of ALTSEL

SYNTHESIS - This module is responsible for selecting the a feasible structural systems for the building in question. A typical rule from this module is as follows:

```
if      [problem,span(X),true]
and     [problem,int_stanchion(no),true]
and     holds X > 60
then    true
to      add[lateral_load_sys,single_span_portal,true]
est     synthesis(1).
```

This rule is in the Edinburgh Prolog Blackboard Shell syntax and states that if the span of the building is more than sixty metres and that there are no internal columns allowed in the building then one alternative for feasible lateral load system is the single span portal frame. The different parts of the rule are explained in section 2. However, the value of *est* in this rule is different from the default provided by the shell. This will be explained later in the section 6.2.

In addition to selecting the feasible alternative structural systems, SYNTHESIS also decides on the frame spacing, appropriate systems for the roof, the sides and their claddings.

PREANA - This module is responsible for undertaking the preliminary analysis of the alternative systems generated by SYNTHESIS. The rules in this sub-module are mostly analysis formulae for different types of structural systems. Also included in this sub-module are rules to decide what type of analysis should be undertaken. For example, it carries out a plastic analysis for a portal frame whereas for a truss it can only undertake a routine elastic analysis. There follows a typical rule from this sub-module which applies to single span portal frames of fixed bases:

```
if    [synthesis,lateral_load_sys(single_span_portal),true]
and     [problem,bases(fixed),true]
and     [problem,span(L),true]
and     [problem,load(W),true]
and     [problem,eaves_ht(H1),true]
and     [problem,pitch(Y),true]
then    moment1(H2,((L/2)*tan(Y)),X,((H1/(H1+H2))*(W*L^2)/16))
to      add[preana,sspfb_pla_mom(X),true]
est     preana(1).
```

The successful execution of this rule will add an entry on the blackboard under the index *preana* . This entry will be the value of the plastic moment for the single span portal frame alternative. The actual *moment* is calculated by the PROLOG clause `moment1(H2,((L/2)*tan(Y)),X,((H1/(H1+H2))*(W*L^2)/16))` in the *consequent part* (i.e. the 'then' part) of the rule.

PREDES - is responsible for carrying out the sizing of the different members of the alternative structural systems generated by SYNTHESIS the preliminary analysis of which is already carried out by PREANA. The following is a rule from this module:

```
if    [preana,ssp_pla_mod(X),true]
and      [problem,ssp_rafters_sec_ext_cons(no),true]
and      [problem,ssp_stanchion_sec_ext_cons(no),true]
then     get_section(X,A,Y)
to       add[single_span_portal,ssp_feas_sec(A),true]
         and[single_span_portal,ssp_zp_provided(Y),true]
         and[single_span_portal,ssp_rafters_sec(ub),true]
         and[single_span_portal,ssp_stanchion_sec(ub),true]
est      predes(3).
```

This rule determines the feasible section from a database of Universal Beam and Column sections given the plastic modulus of the portal frame. It is also stated that there are no external constraints on the dimensions of either the stanchions or the rafters. Both the stanchions and the rafters are assumed to be of the same uniform section.

ECONOMICS - this sub-module is based fully on heuristic rules obtained from the results of a research project on the comparative costs of single-storey steel structures [12]. The firing of the rules in this sub-module depends on the presence of a particular type of lateral load system on the blackboard. The following rule illustrates this and also illustrates the type of knowledge contained in the sub-module:

```
if    [problem,span(X),true]
and      [synthesis,lateral_load_sys(roof_truss),true]
and      [problem,pitch(Y),true]
and      holds((13.3 =< X,X =< 26.7,Y > 0.3))
then     true
to       add [economics, lateral_load_sys_eco(roof_truss),true]
est      eco(1).
```

The rule simply states that if a roof truss is one of the feasible systems, that the span is between 13.3 and 26 metres and the pitch of the roof is greater than 0.3 radians then roof truss will be the most economical structural system.

DESIGN - this sub-module's purpose is not directly concerned with the preliminary design stage. However, it has a very important purpose to serve for the design process as a whole. The sub-module determines any relevant constraints which should be satisfied in the detailed design stage of any alternative structural system. The reason for keeping this sub-module in the ALTSEL module is that the knowledge contained in the sub-module is related to the feasible structural systems generated by SYNTHESIS. Also, all the constraints to be satisfied in the design of any structural system should be propagated to the other modules the moment a particular structural system is found to be feasible by SYNTHESIS. In some ways DESIGN may be regarded as a *meta-module*, i.e., a module which operates above all the other modules. The following is a typical rule from this module:

```
if    [synthesis,lateral_load_sys(single_span_portal),true]
and      [problem,span(Y),true]
and      holds(Y > 10)
then     output_message('The following things should be considered
         in the detailed design stage of single span portal
         alternative :-

1. pitch should be kept low because greater slope
   will give rise to greater spread at knees which
   may cause problems with cladding,

2. horizontal thrusts should be carefully examined
   and the foundation designed accordingly,

3. haunch should be provided at the eaves and the
   ridge should be deepened because the maximum
   bending moment will occur at the knees.')

to       add[design,design_cons(single_span_portal),true]
est      des(3).
```

The rule applies to the constraints for the single span portal frame alternative.

PREVALUATOR - This sub-module is responsible for carrying out a relative evaluation of the different feasible structural systems generated by SYNTHESIS. The different criteria considered by PREVALUATOR are given in figure 6. They are given different weights as suggested by various practising engineers and a final *value* is given to each alternative structure. The best structure is the one with lowest *value*. This sub-module could not be fully developed owing to problems in quantifying subjective considerations such as *aesthetics*.

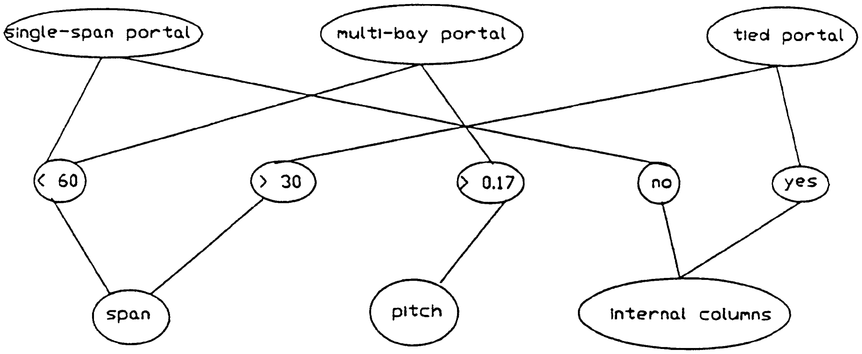


Figure 7. *Inference Network for lateral load resisting frames*

5.2 Problem-solving Methods Used

All the problem-solving strategies in Knowledge-Based Systems Technology adopt one of the following two approaches [13]:

- Formation approach or
- Derivation approach.

The *formation approach* as the name implies involves the formation of the most appropriate solution. This is achieved by putting together the different components of a complete solution stored in the knowledge-base at different levels. In contrast, the *derivation approach* involves selecting the most appropriate solution from a set of pre-defined solutions stored in the knowledge-base.

It is evident that the formation approach is probably the more general and intelligent way of solving a problem. However, for the domain we are working in, we found that the derivation approach provided an easier way of finding solutions. This was discovered after experimenting with the formation approach. So, ALTSEL utilises the derivation approach to solving the problem, in contrast to HI-RISE which uses a formation approach [13]. Figure 7 is an inference network for the selection of feasible lateral load systems. The knowledge-bases of ALTSEL consist of different feasible solutions for various situations. In doing so, it proceeds by handling different constraints [14], consisting of the following :

- constraint formulation,

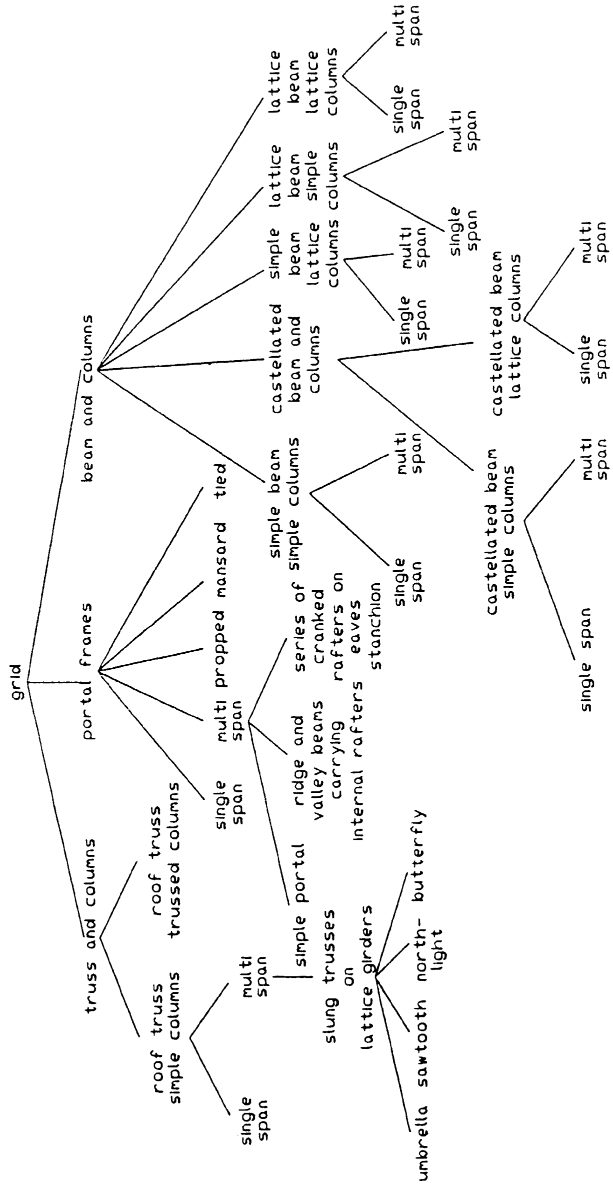

grid
truss and columns
roof truss simple columns
single span
multi span
roof truss trussed columns
portal frames
single span
multi span
simple portal
slung trusses on lattice girders
umbrella
sawtooth
north-light
butterfly
ridge and valley beams carrying internal rafters
series of cranked rafters on eaves stanchion
propped
mansard
tied
beam and columns
simple beam simple columns
single span
multi span
castellated beam and columns
castellated beam simple columns
single span
multi span
castellated beam lattice columns
single span
multi span
simple beam lattice columns
single span
multi span
lattice beam simple columns
single span
multi span
lattice beam lattice columns
single span
multi span

Figure 8. *Solution search space for feasible lateral loading systems*

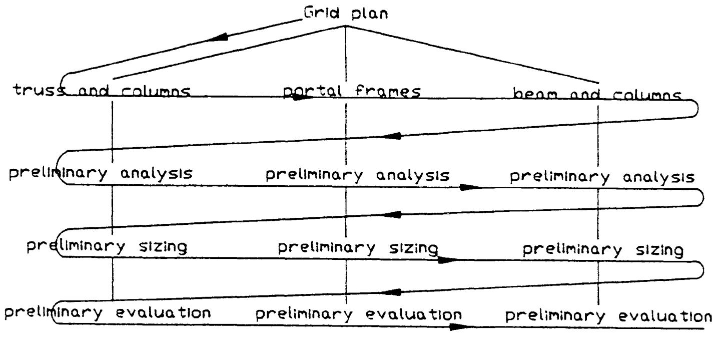


Figure 9. *Problem-solving sequence adopted by ALTSEL*

- constraint satisfaction and
- constraint posting.

The concept of constraint handling is accomplished in the system by first *satisfying the constraint* for a particular alternative; then by *looking for any constraint* associated with the alternative which will be used by other modules (i.e., *constraint formulation*); and finally by *posting* it to the appropriate module which uses it later (i.e, *constraint posting*). The *constraint satisfaction* operation is carried out by all the sub-modules. The *constraint formulation* is undertaken by the DESIGN sub-module; as discussed in section 5.1.8. The *constraint posting* is also accomplished by the DESIGN sub-module but the way it works is by actually exploiting an inherent feature of communication between different *knowledge sources* of the *blackboard architecture*. This is an example of an important use of the *blackboard*. The following rules (one from the SYNTHESIS, one from the DESIGN and one from the standards processing sub-module, STAPRO) will illustrate this feature.

```
if    [problem,span(X),true]
and  [problem,int_stanch(no),true]
and      holds(X =< 60)
then     output_message('A single span portal would be  feasible.')
to       add[synthesis,lateral_load_sys(single_span_portal),true]
est      synthesis(5).

if    [synthesis,lateral_load_sys(single_span_portal),true]
then true
to   add[design,single_span_portal(elastic_defl_check),true]
```

```
est   design(12).

if   [synthesis,lateral_load_sys(single_span_portal),true]
and  [design,single_span_portal(elastic_defl_check),true]
then check_clause(sec. 5.1.2.3)
to      add[conformance(portal_frame,5.1.2.3,deflection),
                                                   satisfied,true]
est     design_check(12).
```

where check_clause is a PROLOG procedure which checks the provisions of section 5.1.2.3 of BS5950.

In this example, the constraint has been posted to the blackboard with the index 'design' by the DESIGN sub-module. It may be accessed by any other module using this index. For example, the last rule from the standard provision checking module states that, if there is an entry on the blackboard which stipulates that the deflection should be checked by elastic methods, then the provisions of clause 5.1.2.3 of the standard must be satisfied. It is evident that giving appropriate *indexes* to different *entries*, allows for *partitioning* of the *blackboard*, which may be further used in *formulating* and *propagating* constraints to other modules.

5.3 Explanation facilities

Some explanation facilities have also been incorporated in ALTSEL. Two approaches have been investigated, one using an associated explanation for every conclusion produced; and the other using the front-end facilities of the shell to generate explanations.

The explanations one may obtain from the system are :

- the rule or set of rules which forced a particular conclusion;
- the current entries on the blackboard;
- the reasons for reaching a particular conclusion;
- the set of rules which were successful at the end of a session; and
- the details of any alternative feasible solutions generated by the system.

6 Implementation

6.1 General Description

The sequence of execution of the different knowledge-modules of INDEX is as follows :

```
(ALTSEL->STRANEX->DETDEX->OPTEX->EVALUATOR)
```

Each of these modules consists of different sub-modules. The sequence of execution of the sub-modules of ALTSEL is as follows :

```
(SYNTHESIS->PREANA->PREDES->ECONOMICS->DESIGN->PREVALUATOR)
```

The ALTSEL sub-modules are shown in figure 4. Based on the rules in these sub-modules, the system is able to select the feasible alternatives for the lateral load resisting systems for the industrial building in question.

The current version of the ALTSEL module incorporates over one hundred rules. Although some of the rules are based on discussions with working design engineers, most of them are taken from the published literature of steel section and frame manufacturing and fabricating organisations such as the Steel Construction Institute (formerly known as the Constructional Steel Research and Development Organisation).

The system is implemented on a Sun 3/50 workstation. The system has knowledge of the following types of steel frames :

- portal frames;
- roof trusses and columns; and
- beams and columns.

Apart from these, it also has rules for incorporating gantries for the design of gantry cranes if required. The solution search space for the feasible lateral load resisting systems is shown in figure 8. The search strategy adopted is the breadth-first search. The system generates all the solutions at one level before going on to the next level. Figure 9 illustrates the search procedure adopted by the ALTSEL module. One drawback with this approach has been the lack of transparency of the system. The user does not receive complete details of a particular alternative at a glance. To overcome this drawback, the user is given the facility of obtaining complete details of any alternative solution generated by ALTSEL at the end of the session using the `show_details_of` command.

6.2 Control Mechanism

The Edinburgh Prolog Blackboard shell (described in section 2) was used for the implementation so that the control mechanism was already built into the shell. It consists mainly of an agenda, dynamically built during the consultation process. The agenda sequences the firing of the rules inside a knowledge module.

As already mentioned in section 2, ‘est’ in the rules indicate the ‘usefulness’ of each rule and, thus, helps in building up the agenda. So, by giving appropriate ‘est’ values to the different rules, we may sequence the firing of these rules. The rule with the lowest 'est' value will be fired first, the rule with the next higher 'est' value after that and so on. This is the default methodology for conflict-resolution provided by the shell using numerical values for 'est'. However, the conflict-resolution strategy adopted in INDEX bypasses this approach and a new strategy was defined that suited the requirements of INDEX.

To accomplish this, the default method was to give simple numerical values to 'est' starting from the first rule of SYNTHESIS and incremented up to the last rule of PREVALUATOR. This was not an elegant way of approaching the problem for the simple reason that if a rule was added to any of the modules at a later stage, all the 'est' values have to be changed for all the rules following it. Another reason was that the whole idea of modularity would get lost by this approach and the set of rules, in effect, would become one module instead of being broken down into sub-modules.

The second approach was to give symbolic 'est' values to the rules of the different sub-modules. These symbolic names were specific to the rules of that sub-module only and define a different conflict-resolution strategy altogether. In this approach, the sequence of execution of the sub-modules had to be first defined and then the sequence of firing the rules inside each sub-module. The rules quoted earlier in this chapter have symbolic 'est' based on this approach. This approach avoids the problems of the default approach described above. To recap, the following is an example of a rule from the SYNTHESIS sub-module using this approach:

```
if      [problem,span(X),true]
and     holds(X =< 60)
then    output_message('Single span portal frame is a
                                              feasible alternative')
to      add[synthesis,lateral_load_sys(single_span_portal),true]
est     synthesis(5).
```

The 'est' value of this rule indicates that this rule is the fifth rule inside the SYNTHESIS sub-module. The numerical value in this 'est' decides the firing of the rule inside that sub-module. Invocation of the sub-modules is decided by the top level conflict-resolution. It is worthwhile pointing out that this operation of sequencing the firing of the rules as well as the knowledge modules according to the demands of the domain was made simpler by using the Edinburgh Prolog Blackboard Shell.

7 Summary

The concepts involved in the development of ALTSEL, the preliminary design module of INDEX, were outlined. The *knowledge elicitation* as well as *knowledge representation* aspects of the development of ALTSEL's knowledge base were described in detail. Some implementation issues were highlighted with examples of some representative production rules. A description of the *control mechanism* suitable for the domain of this project was also given underlining the ease of accomplishing this using an expert system shell.

8 Conclusions

The following conclusions were drawn from the development of ALTSEL:

- Artificial Intelligence tools and techniques provide a way of incorporating rules of thumb and heuristics into computer-aided design of structures.

- *Protocol Analysis* was found to be a useful knowledge elicitation technique in the domain of design.

- The *blackboard architecture* provided a flexible environment for the *propagation* of constraints between the different knowledge modules so vital for design.

- The development of a fully-working system requires many different types of knowledge. Mere heuristics are not sufficient to solve real-life problems in structural design. The system needs to have numerical as well as logical capabilities.

- Since a considerable number of decisions in the preliminary design stage are taken using heuristics, a system similar to the one described in this paper might perform satisfactorily in that domain. But, for the domain of detailed design, the system needs to have more capabilities, e.g., logical and mathematical inferencing from fundamental laws of structural engineering or general knowledge of arithmetic.

References

[1] Topping, B.H.V., Kumar, B., "A Knowledge-Based Model for Structural Design", *in this volume, Optimization and Decision Support Systems in Civil Engineering,* Kluwer Academic Publishers, 1992.

[2] Topping, B.H.V., Kumar, B., "INDEX: An Industrial Building Design Expert", *Journal of Civil Engineering Systems,* v.5, 65-76, 1988.

[3] Chan, N., Johnson, K., "Edinburgh Blackboard Shell User's Manual", *Artificial Intelligence Applications Institute,* University of Edinburgh, 1987.

[4] Welbank, M.A., “A Review of Knowledge Acquisition Techniques for Expert Systems”, *British Telecom Research, Martlesham Heath, 1983.*

[5] Gammack, J.G., Young, R.M., “Psychology Techniques for Eliciting Expert Knowledge”, In Research and Development in Expert Systems, Bramer, M.A. (Ed.)., *Cambridge University Press, 1985.*

[6] Bloomfield, B.P., “Capturing Expertise by Rule Induction”, *The Knowledge Engineering Review, Vol 1, No. 4, 1986.*

[7] Regan, J.E., “A Technique for Eliciting Categorical Knowledge for an Expert System”, *Paper submitted to AAAI-87, 1987.*

[8] Hayes-Roth, F., Waterman, D.A. and Lenat, D.B. (Eds.), “Building Expert Systems”, *Addison Wesley, 1983.*

[9] Mittal, S., Dym, C.L., “Knowledge acquisition from Multiple Experts”, *AI Magazine, 32-36, Summer 1985.*

[10] Stefik, M., Foster, G., Bobrow, D.G., Kahn, K.M., Lanning, S.,Suchman, L.A., “Beyond the Chalkboard: Using Computers to Support Collaboration and Problem Solving in Meetings”, *Paper submitted to CACM, 1986.*

[11] Sriram,D., “Knowledge-Based Approaches to Structural Design”, *Ph.D. Dissertation, Department of Civil Engineering, Carnegie-Mellon University, U.S.A., 1986.*

[12] Horridge, J.F., Morris, L.J., “Comparative Costs of Single storey Steel Framed Structures”, *The Structural Engineer, Vol. 64A, No. 7, 177-181, 1986.*

[13] Maher, M.L., ”Problem-Solving Using Expert System Techniques”, in Expert Systems in Civil Engineering, Eds., Kostem,C.L., Maher, M.L., *ASCE, New York, 7-17, 1986.*

[14] Stefik, M., “Planning with Constraints - MOLGEN Part 1”, *Artificial Intelligence, No. 16, 111-140, 1981.*

A Prototype Environment for Integrated Design and Construction Planning of Buildings

S. J. Fenves, U. Flemming,
C. T. Hendrickson, M. L. Maher, and
G. Schmitt
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh
United States of America

Abstract The dispersed nature of the construction industry raises communication issues that are exacerbated by the increased use of computer programs. Integration of the various disciplines and computer programs requires more than the transfer of geometric data. This paper presents an integrated computer environment in which knowledge-based systems communicate through a blackboard and a central, global database representing the design solution.

1 Introduction

The building construction industry is characterized by a dispersed organizational structure in which a number of diverse organizations participate in the planning, design and construction of each building project. Presently, the major media for communicating large volumes of information between the participants are drawings and written specifications. As increasing portions of the design-construction process become computer-based, the need for appropriate forms of electronic communication becomes increasingly apparent. Furthermore, as computer use shifts from purely numeric calculations towards symbolic and knowledge-based reasoning, there is additional need to communicate functional as well as geometric information.

The dispersed nature of the construction industry raises further communication issues. On the one hand, there is no pool of common knowledge: since architects and structural engineers only perform design, they do not possess first-hand knowledge of what constitutes a constructable design. On the other hand, there is no

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 149–165.
© 1992 *Kluwer Academic Publishers.*

direct way for constructors to provide feedback to designers on what changes in the design may improve constructibility and reduce cost. These issues indicate that communication between the dispersed organizations needs to be extended beyond the current exchange of data and that it is not clear what these extensions should be.

The prototype integrated environment described in this paper is intended to serve as a testbed for examining these communication issues. The integrated environment addresses the vertical integration of architectural design, structural design and construction planning of speculative high-rise office buildings. Attention is primarily focused on two aspects: (1) representation and communication of the project information as the project progresses; and (2) control and feedback in the overall process. The environment makes use of a number of Artificial Intelligence techniques. The processes are implemented as Knowledge Based Expert Systems. A Blackboard Architecture is used to coordinate communication between processes.

As the Integrated Building Design Environment (IBDE) is intended to serve as a testbed for exploring integration issues, it is an evolving system. In this paper we present IBDE in its initial, current, and future states, addressing the issues associated with its evolution. This paper begins with a description of the initial architecture of the integrated environment and its knowledge-based processes. The extension of the processes beyond their original 'forward pass' mode to provide and respond to criticism is discussed next. The evolution of the representation of project information from flat files to a global database with derived views is presented. The control issues are addressed and the current and future states of the implementation are described. The paper concludes with a summary of the major issues addressed.

2 Architecture and Processes

2.1 Architecture

The Integrated Building Design Environment (IBDE) integrates seven individual processes using a blackboard approach. The architecture of the initial system is illustrated in Figure 1. The status blackboard records the status of the processes. The controller is responsible for activating the processes and communicating project information to the processes. The datastore manager is responsible for retrieving, storing, and translating data for individual processes. The project datastore records the global representation of the project information. The common user interface is a graphical and textual display of the project information and the current status of the processes. The communication and control mechanisms are treated in detail in the succeeding sections.

The processes in IBDE include architectural planning, structural design, and

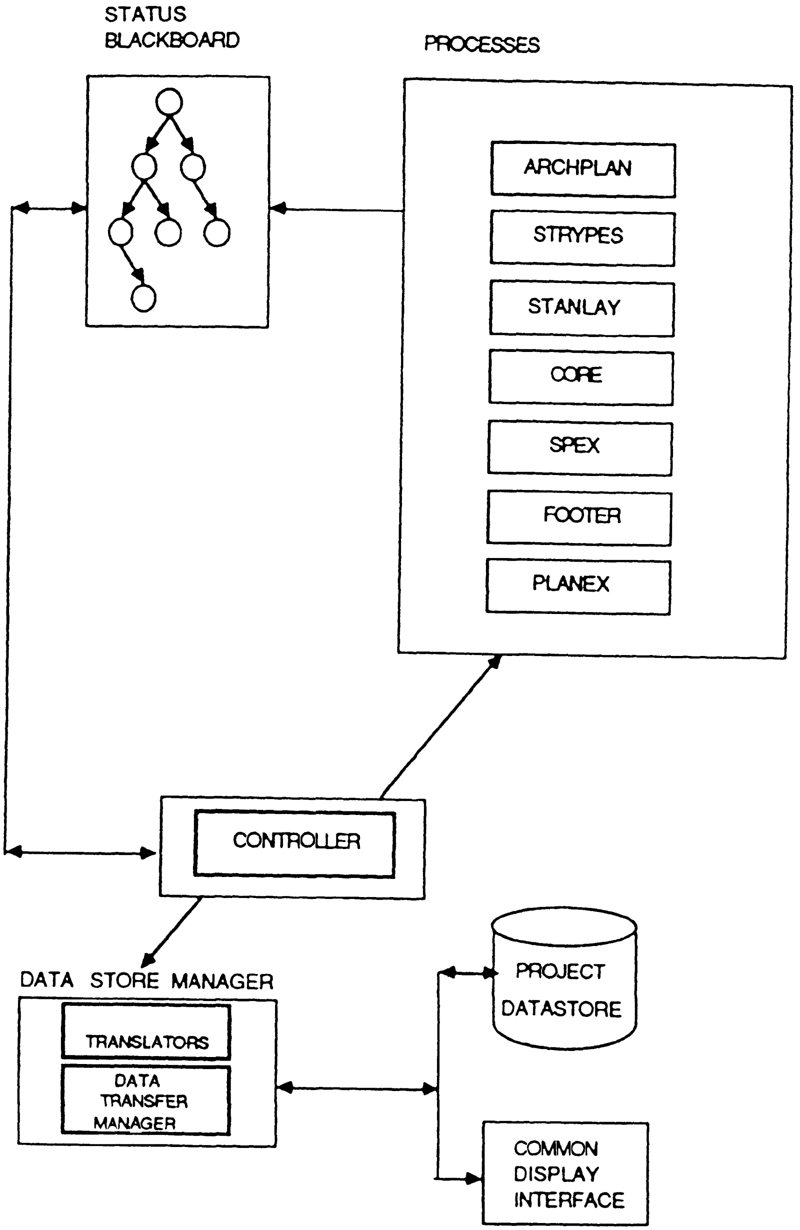


Figure 1. *Initial architecture of IBDE.*

construction planning. These processes are not typical of computer programs currently used in design and construction of buildings, but they accentuate some of the communication issues addressed in the project. Some of the processes existed before IBDE was developed, while others were developed in the context of IBDE. All processes are implemented as knowledge-based systems to permit rapid development and modification. The implementation of IBDE in a commercial environment would necessarily also include a number of conventional programs. The knowledge based processes are:

- ARCHPLAN: an interactive system for the development of the design concept;
- CORE: a space planner for the service core;
- STRYPES: a structural system configurer;
- STANLAY: a layout and approximate analysis system;
- SPEX: a structural component designer;
- FOOTER: a foundation designer; and
- CONSTRUCTION PLANEX: a construction planner, estimator and scheduler.

2.2 The Processes

Each of the processes comprising IBDE is briefly described below.

ARCHPLAN [7] is a knowledge-based ARCHitectural PLANning expert system for the interactive development of a design concept. Input to ARCHPLAN describes the given site, program, budget, and geometric constraints. The output provides three-dimensional information about the building's overall shape, the distribution of functions, and the circulation system. The program starts with a generic prototype of an office building, which is then refined by the user in interaction with the program and heuristic knowledge built into the program. This interaction takes place in three distinct, but interrelated decision modules. The *Site, cost, and massing module* (SCM) develops a massing model that will fit the given site and budget and a range of other parameters. Cost, site and massing options are treated as inter-dependent concerns. Conflicts are resolved based on the *Function module*. This module assists in determining the vertical and horizontal distribution of functions (office, retail, atrium, mechanical systems and parking) within the volume established by the previous module. The module proposes a three-dimensional layout scheme and presents it as solid or wire frame display. If conflicts occur with input data or earlier decisions, the program backtracks to the SCM module. The *circulation module* generates circulation proposals based on combinations of internal or external vertical circulation elements.

CORE generates layouts of the elements in the service core of the building (elevators, elevator lobbies, restrooms, emergency stairs, utility rooms etc.). The input to CORE includes the overall geometry of the building and the expected size and location of the service core. CORE's output includes the number of elevator banks, the number and speed of the cars in each bank, the floors served by each bank and the layout of cars, banks, and other elements. CORE is an adaptation of LOOS, [2] a general system for the generation of layouts that can be adapted to various domains. LOOS places particular emphasis on the generation of layout alternatives with interesting trade-offs.

STRYPES is a knowledge-based expert system that configures a structural system. It is based on the knowledge acquired through the development of HI-RISE [5]. STRYPES is implemented in EDESYN [6], an expert system shell for engineering design synthesis. The input to STRYPES includes: (1) the structural grid produced by ARCHPLAN, specifying potential locations for structural systems; (2) functional information about the building, such as intended occupancy and location and size of the service core; and (3) load information. The output of STRYPES specifies the types and materials for the lateral load system and vertical and horizontal gravity load systems. STRYPES considers frame or shear wall vertical systems and slab, steel deck, or prefabricated panel horizontal systems.

STANLAY, also developed using EDESYN, performs two major tasks for the preliminary structural design of the building. The first task is the layout of the structural systems specified by STRYPES, the second is an approximate analysis of the structural system. The input to STANLAY includes: (1) the structural grid; (2) the architectural function of the building; and (3) the structural systems selected by STRYPES. The output of STANLAY is the location of the lateral and gravity load systems and the load distribution and grouping of the structural components. The layout involves identifying several possible locations of the lateral load system and specifying the location of the gravity load system. The location of the lateral load systems requires the specification of 2D vertical subsystems, such as rigid frames or shear walls, and their location on the grid.

SPEX [3] performs the preliminary design of components for the structural system specified by STANLAY. In the IBDE implementation, SPEX receives as input the design parameters for each component group: (1) type of component (e.g., beam, column); (2) length; (3) material (steel or concrete); and (4) estimated loads. The SPEX interface supplies the material grade, the name of the design standard, the design focus, and an optimality criterion. The output of SPEX is the description of the optimal component. It implements a design strategy in which components are designed by applying three types of knowledge: knowledge contained in design standards; 'textbook' knowledge of structural, material and geometric relationships; and designer-dependent design expertise.

FOOTER is an expert system that performs a preliminary design of the foundation of a building; it is also implemented in EDESYN. The input to FOOTER includes: (1) soil conditions, such as the presence of obstruction, location of water table, depth of bedrock, and soil classification; and (2) imposed load conditions from the structure provided by STANLAY. The output of FOOTER is a description of a footing or pile for each column and/or shear wall. The foundation design problem is decomposed into: selection of foundation type; material type; casting type; excavation method; and parametric design of the foundation components.

CONSTRUCTION PLANEX [4] is a knowledge-based expert system to assist the construction planner. The input to PLANEX consists of: (1) specifications of the physical elements of the structure provided by other processes; (2) site information (such as soil type and elevations); and (3) resource availability (such as number of crews or equipment types). The output from PLANEX consists of a complete plan of construction activities including a provisional schedule and cost estimates. The system suggests technologies; generates necessary activities and precedences; estimates durations and required resources; and develops a construction schedule. The system will either generate a construction plan automatically or a planner can review and modify decisions during the planning process.

2.3 Evolution of Processes

The processes in IBDE are being extended to generate and react to criticism and feedback in the design process. The initial version of IBDE operates automatically in the 'forward pass' mode, where the controller activates the processes in a linear, sequential fashion. The addition of CORE introduced some parallel processing, where STRYPES and CORE could be running simultaneously. The addition of CORE also introduced potential conflicts because the service area set aside by ARCHPLAN could be determined to be inadequate by CORE, thus invalidating the design produced by STRYPES and STANLAY. Similar conflicts could arise at any point in the design and construction planning process. Furthermore, a 'downstream' process may provide feedback on ways that the design could be improved. The introduction of criticism and feedback in IBDE begins to address and identify potential conflicts.

The strategy of adding design critics in IBDE first takes advantage of the processes that already contribute to the project. Each process is being extended to include a process activator and a process critic, as illustrated in Figure 2.

The *process critic* posts one or more constraints on the message blackboard if the process was unable to produce a valid solution or if it can suggest possible improvements. The *process activator* serves two purposes:

1. Produce criticism by checking the input data for scope; if the input includes

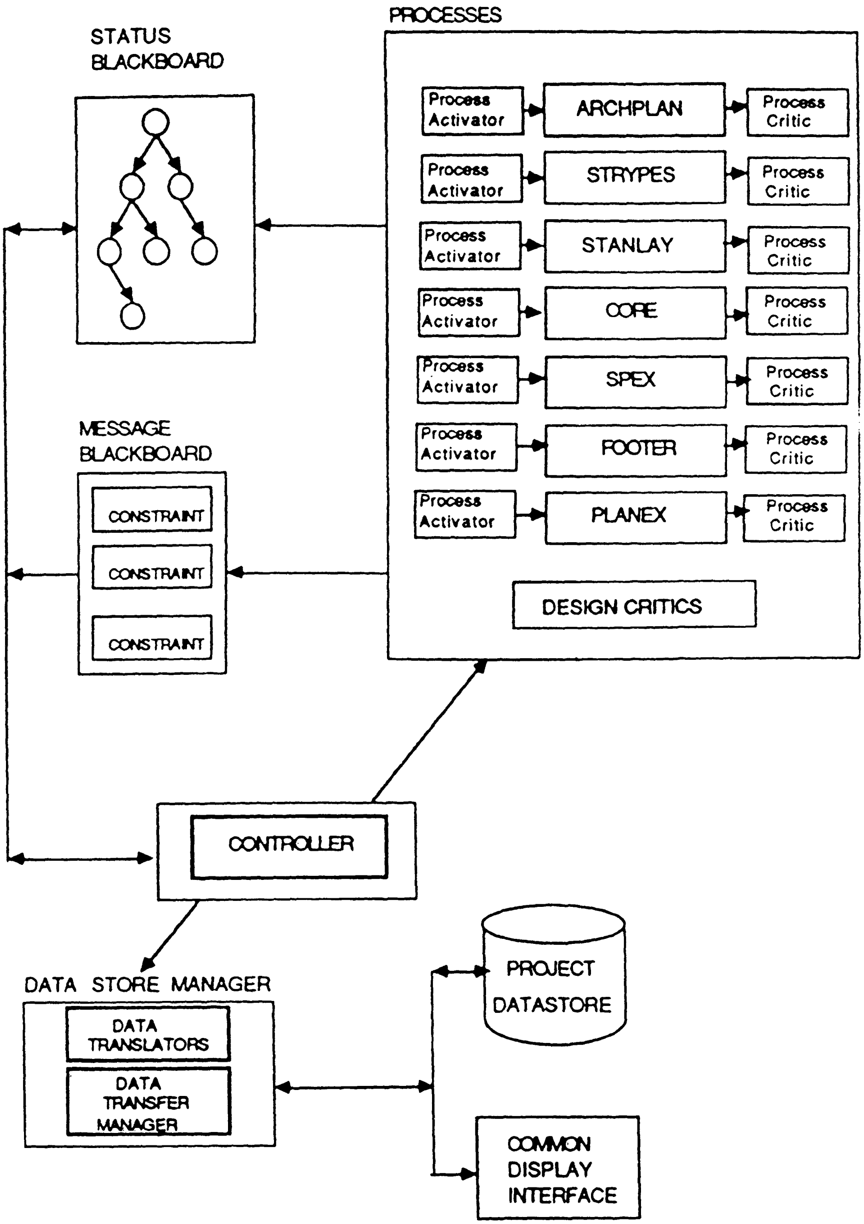


Figure 2. *Extended architecture of IBDE.*

design decisions that fall outside the capability of the process a constraint is posted on the blackboard and the process is not activated.

2. React to criticism by adding a constraint for consideration by the process; when the process is run again, the constraint ensures that the same conflict does not arise.

These two extensions allow criticism and feedback from the individual processes to be communicated to the controller in the form of constraints. Thus far these extensions appear to be easily incorporated into the original knowledge based processes.

Specialized design critics are also planned to be included in IBDE. These critics are intended to provide criticism on design decisions using knowledge external to the original knowledge base that produced the decisions. A critic currently under development is a structural analysis critic that will execute a formal analysis using the results of SPEX and check for consistency with STANLAY's approximate analysis.

3 Global Representation

The project datastore holds the global representation of the building and serves as the repository of data communicated between the IBDE processes.

3.1 Objectives

The design and subsequent evolution of the datastore was dictated by four considerations.

Provision of process views. The primary function of the datastore is to provide individual views or sub-schemas to the processes through which each process receives its needed input, without regard of where it was generated, and transmits its output, again without regard of where it may be subsequently used. These views need to be sufficiently flexible so that changes in the processes' data needs can be readily accommodated.

Flexibility of global schema implementation. In contrast to the process views, which must be highly responsive to the processes' functional needs, the global schema —known only to the datastore manager—can have an evolution of its own. As discussed below, we have successfully migrated from a tabular, flat file organization to a relational DBMS organization.

Explicit representation of important conceptual relations. In a static environment, the datastore need not contain more than the union of the processes' input and output requirements; all other data may reside inside the processes, following the principles of *information hiding* and *data encapsulation.* Expecting a dynamically evolving environment, it was decided to include in the datastore the important *conceptual relations* and *abstractions* to facilitate the subsequent addition of new processes and critics. Thus, for example, the objects representing *structural functions* and *frames* are presently used only by STRYPES and STANLAY; their successor processes (SPEX, FOOTER and PLANEX) deal only with individual building elements, such as beams or columns. Nevertheless, these high-level objects are included in the datastore for subsequent use.

Support for common display interface. The visualization of an object as complex as a building requires the display of a great variety of information. Rather than delegating this display to the individual processes, a single common display interface was desired, which could display all the information in the datastore in a variety of formats.

3.2 Overall Organization

The datastore is hierarchically organized as a tree of related objects. Objects may represent very high-level abstractions, such as the entire building, or very detailed information, such as individual building elements. The hierarchy primarily represents *part-of* relations, where each object is a part or component of a higher-level parent object. Provisions are also made for representing *is-alternative* relations, where an object is an alternate design solution of the parent object. Through this latter relation, redesign in response to critiques received is readily supported.

As discussed above, this overall organization is independent of the internal global schema implementation. Figure 3 shows the hierarchical organization among the major object classes in the current (relational DBMS) implementation. With this organization, each process can view the contents of the datastore relevant to it, but not of the segments relevant to the other processes. This organization has supported the concurrent development of the processes and provides complete data and process independence among the processes. The datastore provides at any time a complete snapshot of the current state of the building design and construction planning process.

3.3 Communication with Processes

Communication with the processes is the responsibility of the *datastore manager.*

The datastore manager works in concert with the controller and is responsible for supplying the input data to the processes and retrieving their output data. Prior to initiating a process by the controller, the datastore manager transfers the input data to the machine on which that process resides. When a process terminates, it leaves its output on its own machine; when its termination message is received, the controller causes the datastore manager to retrieve the data from

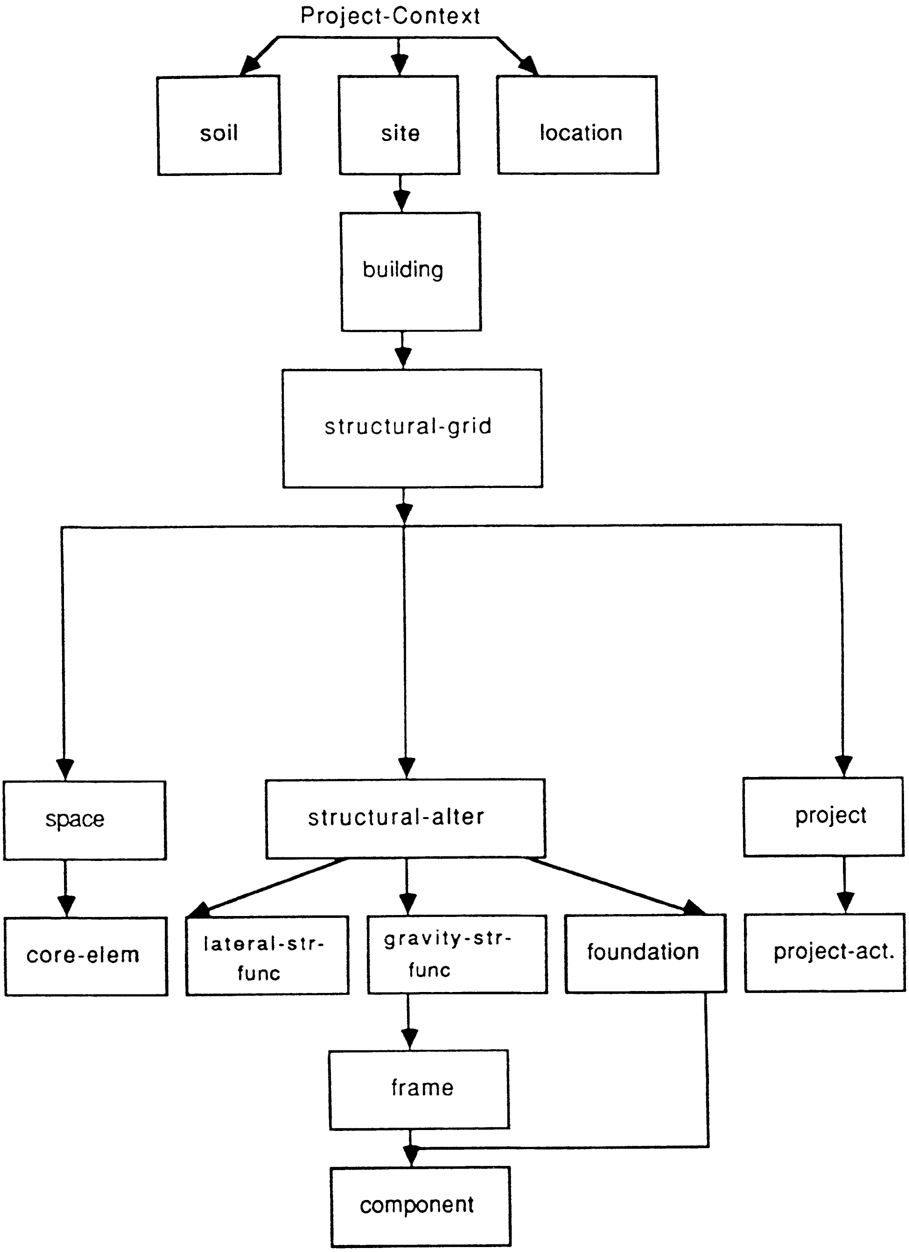


Figure 3. *Organization of project datastore.*

the processes' machine and merge it into the datastore.

The datastore manager is responsible for generating the views or subschemas as needed by the processes, including all format and structural conversions. The datastore manager is further responsible for maintaining an 'audit trail' in the sense of maintaining two descriptions for each object class: placed-by: the name of the process which provides the value(s) of its attributes(s); and (2) used-by: the names of the processes which use these values. Using this information, the controller can route any critique of a particular object to the process that was responsible for creating the object's attribute values.

3.4 Implementations

The local views of all of the processes consist of sets of objects with attributes in the respective implementation languages of the processes. Furthermore, in most processes, no explicit distinction is made between input and output attributes; the object contains all necessary attributes.

In the initial implementation, data is communicated between the processes by means of files. Each file contains all the instances of a particular object type (e.g., beams or columns). There is a one-to-one correspondence between the objects in the files and the individual process objects, although there are differences in format and attribute names. The datastore manager is responsible for format and name translation and for transferring the appropriate files to and from the processes.

As an illustration, Figure 4 shows the 'snapshot' of a BEAM object before PLANEX has supplied values for the last seven attributes.

In the current implementation, the datastore manager maintains the global representation in the form of a global database schema implemented as a set of relations. The schema closely resembles the conceptual hierarchical tree representation and relies extensively on two types of relations: (1) *'data' relations* which contain the attributes of the various objects, keyed by a single object-identifier attribute; and (2) *connection relations*, usually all-key, which represent the part-of relations of the hierarchical tree.

The datastore manager now performs two distinct operations for each input and output transaction between a process and the datastore. In the case of providing input to a process, the manager: (1) extracts an *input view* from the global schema through a sequence of relational operators, so that all information needed by the process is contained in the view; and (2) makes the view available to the process. Two options are available to the processes: (1) a flat file identical to that of the initial implementation; and (2) a file of database commands which, when executed by a copy of the DBMS manager on the process machine, creates a local copy of the view. The former option allows the processes to run without modification, whereas the latter option is made available to processes which may wish to access the data through their own DBMS queries.

ATTRIBUTE	VALUE	PLACED BY	USED BY
NAME	BEAM-378	STANLAY	SPEX PLANEX
CLASS	BEAM		
IS-A	DE-GROUP	STANLAY	SPEX PLANEX
NAME	BEAM-11	STANLAY	SPEX PLANEX
NAME-CODE	81	STANLAY	PLANEX
PART-OF	TYP-FRAME-1540	STANLAY	PLANEX
MPOS	5273.4375	STANLAY	PLANEX
MNEG	0	STANLAY	PLANEX
VMAX	0	STANLAY	PLANEX
MULTIPLIER	4	STANLAY	PLANEX
UNBRACED-LENGTH	25.0	STANLAY	PLANEX
X1	(...)	STANLAY	PLANEX
X2	(...)	STANLAY	PLANEX
Y1	(...)	STANLAY	PLANEX
Y2	(...)	STANLAY	PLANEX
Z1	3	STANLAY	PLANEX
Z2	19	STANLAY	PLANEX
CONSTRUCTION-TYPE	NIL	SPEX	PLANEX
DESIGNATION	W12 106	SPEX	PLANEX
WEIGHT	106	SPEX	PLANEX
CONCRETE-TYPE	NIL	SPEX	PLANEX
PSTEEL	NIL	SPEX	PLANEX
GRADE	36	SPEX	PLANEX
SHAPE	W	SPEX	PLANEX
XL-DIMENSION	25.0	SPEX	PLANEX
YL-DIMENSION	1.0183333	SPEX	PLANEX
ZL-DIMENSION	1.0741667	SPEX	PLANEX
XG-COORDINATE	NIL	PLANEX	USER
YG-COORDINATE	NIL	PLANEX	USER
ZG-COORDINATE	NIL	PLANEX	USER
MATERIAL-COST	NIL	PLANEX	USER
CREW-COST	NIL	PLANEX	USER
START-TIME	NIL	PLANEX	USER
FINISH-TIME	NIL	PLANEX	USER

Figure 4. *A BEAM instance object in the datastore.*

In receiving output from a process, the manager: (1) receives the *output view* of the process, again by either of the two options; and (2) merges the output view into the global schema through a sequence of relational operators.

The only major conceptual difference between the two implementations is the explicit separation of input and output views. The input view is provided in a 'read only' mode: it is not recovered from the process upon its completion. In this fashion, database integrity maintenance is shared between the processes and the datastore manager: (1) the processes insure that their output is consistent with respect to the input provided; while, (2) the datastore manager insures consistency among the outputs, i.e., consistency of the global representation.

3.5 Common Display Interface

The data residing in the datastore is inaccessible to users without a common user display interface. As the interface is intended for a variety of users with different backgrounds, it must conform to certain graphical standards and should exhibit a degree of intelligence. An interface of this type was developed for the IBDE project. It provides a uniform set of interface facilities for the following functions:

- *Graphical display of the status of all processes.* Each process is shown as either pending, active, or completed.

- *Graphical display of data at any level of the project datastore representation.* As soon as a process is completed, the content of the datastore can be displayed. The designer sees the geometric representation of these data as three-dimensional objects or as charts and symbols.

- *Textual and graphical display of object classes.* The user selects one of the datastore objects directly from a menu, and the geometric and textual information is displayed.

- *Graphical display of selected items.* The designer can specify constraints to view objects of a certain class or that fall within user-defined limits. All objects found conforming to the constraints are highlighted on the graphical display.

- *Graphical navigation to select specific objects.* Once selected graphically, the object is highlighted and the appropriate datastore object appears on screen in a pop-up window.

4 Control

The controller is responsible for activating the individual processes.

4.1 Objectives

The design and subsequent evolution of the controller was dictated by three major considerations.

Support of different strategies. From the inception of the project, it was intended that the controller be able to support a number of different activation strategies, ranging from simple, fixed scheduling through demand-driven scheduling and on to a variety of planner-based and blackboard-based opportunistic strategies.

Flexibility of implementation. As with the datastore, it was intended that the controller have an evolution of its own, without affecting the individual processes. The Design Systems Laboratory of Engineering Design Research Center at CMU, within which the IBDE project is conducted, has an active research program on generic design environments. The IBDE project is viewed by that Laboratory as a testbed for exploring the applicability of these environments in a specific application area. Therefore, flexibility in porting IBDE from one environment to another was a major consideration.

Minimal domain knowledge. In order to support the above two objectives, it is essential that the controller be as generic as possible, that is, that it require minimum knowledge about the domain of building design and construction planning. More specifically, it is desired that the controller 'know' as little about the individual processes as possible, but use the messages on the blackboards and a static description of each process to guide the controller.

4.2 Communication with Processes

The processes communicate with the controller by means of blackboard messages. There are two general classes of messages.

Status messages. Each process is in one of three states: *pending*, *active*, or *completed*. Whenever an active process terminates, it sends a new status message which the controller posts on the status blackboard. The message also signifies whether the process was successful in producing a feasible solution or not. As discussed in the preceding section, the controller also controls the data communication between processes.

Inter-process messages. Processes may also generate messages to other processes indirectly, particularly to critique some aspects of the design. These messages are stored on the message blackboard in the form of constraints. The controller uses these messages to modify the activation sequence of the process involved.

4.3 Implementations

The initial implementation provides a very limited control strategy, namely, an event-driven, sequential process activation. The controller maintains only the following static description about each process: (1) *preconditions* for its execution, namely, the process(es) that must have been successfully completed before the current process can be activated; and (2) *machine* on which the process runs.

When the preconditions of a process are satisfied, the controller causes that process to be activated.

The controller is implemented on top of the DPSK (Distributed Problem Solving Kernel) system developed at CMU [1]. DPSK provides an environment for distributed problem solving on multiple machines by programs written in several languages. DPSK provides utilities for sending messages and signals between processes running on different machines, generating and responding to events, and communicating between processes by means of a Shared Memory accessible to all the processes. DPSK was designed to facilitate the implementation of a variety of cooperative problem-solving architectures; the current IBDE implementation, with fixed precedence ordering between processes, is a relatively simple application of DPSK. However, because of the computational expense of communicating large volumes of data via DPSK's shared memory, that memory is used only for the status messages.

The implementation has been useful in bringing the first version of IBDE up to operational status. Changes in the processes, such as the addition of CORE and the replacement of the original HI-RISE process by STRYPES and STANLAY, were readily accommodated. The integrated system has been run on as many as 9 machines working simultaneously. The machines include HP-9000/320, Micro VAX, SUN 3 and SUN 4 systems. The controller and datastore manager reside on the SUN 4; the processes other than SPEX on HP's and SUN 3's; while for efficiency three copies of SPEX reside on three Micro VAXes and process, respectively, the bottom story columns (so as to supply input to FOOTER quickly), the remaining columns, and the other structural components.

We are in the process of designing the second implementation of the controller. It will handle the expanded processes discussed in Section 3 by acting on the design constraints posted on the message blackboard. A variety of activation strategies will be explored, including both a planner and an opportunistic scheduler. Two generic design environments developed by EDRC are being considered as the basis for implementation.

5 Conclusions

The IBDE project is a testbed for the exploration of integration and communication issues in the building industry. The processes are knowledge-based and can serve as surrogate experts (provided, of course, that they adequately capture and utilize the expertise relevant to their particular task). This enables us to run a variety of experiments when exploring a particular issue, so that the conclusions reached will have a strong empirical basis. Furthermore, the modular nature of the IBDE environment provides a testbed for the empirical evaluation and calibration of generic integrated design support environments. Experiments with these environments provides feedback to their developers and can eventually lead to extrapolations to other design disciplines.

The project deliberately did not start with an overall, normative model of *the* building design process. Rather, we prefer to arrive at generalizations about the design process based on the experience and insights gained from our experiments. The system itself is intended to serve as a vehicle for discovery and theory formation.

The IBDE project contrasts sharply with the integrated systems in use or under development in the building industry. The principal contrast is not knowledge based *vs.* algorithmic process components. Integrated systems in industry achieve a high level of *data* integration by tying CAD systems, analysis programs, etc. together through a shared database. However, such systems do not address the integration issues of building systems, of design and construction processes and of the thought processes of the disciplines involved. These are the issues addressed by the IBDE project, where the content and semantics of the communication is as important as the mechanism of data transfer.

It is premature to speculate what a 'production' version of the IBDE system may look like. The project must first discover a 'language' through which the project participants' intent may be communicated to others, and through which critiques can be fed back. Even after such a language is developed, it remains to be seen how the resulting intimate integration can be implemented in the present dispersed organizational structure of the industry.

References

[1] Cardozo, E., (1987), *DPSK: A Kernel for Distributed Problem Solving*, Technical Report EDRC-02-04-87, Carnegie Mellon University, Engineering Design Research Center.

[2] Flemming, U., Coyne, R., Glavin, T. and Rychener, M., (1988), 'A Generative Expert System for the Design of Building Layouts—Version 2' in J. Gero (Ed.), *Artificial Intelligence in Engineering Design*, Elsevier (Computational Mechanics Publications), New York, pp. 445–464.

[3] Garrett, Jr., J. H. and Fenves, S. J., (1986), *A Knowledge-Based Standards Processor for Structural Component Design*, Technical Report R-85-157, Carnegie Mellon University, Department of Civil Engineering.

[4] Hendrickson, C. T., Zozaya-Gorostiza, C. A., Rehak, D., Baracco-Miller, E. G. and Lim, P. S., (1987), 'An Expert System for Construction Planning,' *ASCE Journal of Computing in Civil Engineering* **113**, 5, 253-269.

[5] Maher, M. L. and Fenves, S. J., (1984), *HI-RISE: An Expert System For The Preliminary Structural Design OF High Rise Buildings*, Technical Report R-85-146, Carnegie Mellon University, Department of Civil Engineering.

[6] Maher, M. L., (1988), 'Engineering Design Synthesis: A Domain Independent Representation,' in *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, **1**, 3, 207–213.

[7] Schmitt, G., (1988), 'ARCHPLAN—An Architectural Planning Front End to Engineering Design Expert Systems,' in M. D. Rychener (Ed.), *Expert Systems for Engineering Design*, Academic Press, New York, pp. 257–278.

Water Resource Applications of Knowledge Based Systems: Hazardous Material Management, and Stream Quality Modeling

Mark H. Houck
School of Civil Engineering
Purdue University
West Lafayette, Indiana
United States of America

Abstract The development and application of knowledge based systems in water resources is growing rapidly. The purpose of this lecture is to review two different applications that illustrate the range of development of this type of tool. The first application is a knowledge based system to support decision making in the management of potentially hazardous or obnoxious dredged materials. The U.S. Army Corps of Engineers is charged with maintaining the navigability of the waterways of the country. Dredging is a principal tool used by the Corps for this purpose. However, dredging is problematic because the disposal of any dredged material that is determined to be hazardous is substantially more expensive, difficult, and time consuming than non-hazardous material disposal. The management strategy for dredged material disposal can be represented by a knowledge based system that is currently under development. The second application is in the area of stream quality modeling. Specifically, it is a knowledge based system to aid in the calibration and use of the extended Streeter-Phelps BOD-DO model for a river or stream.

1 Introduction

One of the emerging technologies that will have significant effects on the way we manage our water resources is knowledge based systems. They represent a new way of viewing problems. It is not a panacea but it is a new tool that can be used in conjunction with others to improve decision making substantially in some domains. Knowledge based systems can be embedded in other techniques, such as optimization and simulation models, or these models plus other procedures such as graphics, data base management systems, geographic information systems and other information processing procedures can be embedded in the knowledge based system.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 167–175.
© 1992 *Kluwer Academic Publishers.*

Although knowledge based systems technology has been developed only recently, there are already a significant number of applications in water resources. These applications span a wide range including the calibration and use of hydrologic and hydraulic models, water supply and waste water management, and hazardous waste management (Maher, 1987; Ortolano and Steineman, 1987). The purpose of this paper is to provide an introduction to two applications of knowledge based systems in the area of water resources. The first application is the management of potentially contaminated dredged material; the second is the calibration and use of a stream quality model.

2 Potentially Hazardous Dredged Material Management

This section contains an overview of the **Dredging And Disposal Ecological Evaluation System** (DADEES) computer program, prepared for the U.S. Army Corps of Engineers. The current version of DADEES is an experimental prototype of an application program designed to aid Corps of Engineers personnel in the decision making involved in the aquatic disposal of dredged material. As such a prototype, the program is constructed from a simplified representation of the aquatic disposal decision making problem taken primarily from Appendix A of Lee *et al.* (1985). DADEES is entirely menu-driven and relatively self explanatory to the knowledgeable user to whom the prototype is directed, and who it is assumed is somewhat familiar with the decision making framework described in Lee *et al.* (1985).

The purpose of DADEES is to lead the user through a three-tiered testing strategy for the determination of the aquatic disposal requirements for dredged material. DADEES is restricted to aquatic disposal; upland disposal alternatives are not considered in this prototype.

This testing system is invoked when an area for which dredging is being considered is believed to have some sort of contamination. To use the program, the user must have in hand bulk chemical analyses of the sediments to be dredged ('test sediments') and of the sediments in which aquatic disposal is being considered ('reference sediments'), for each contaminant of concern to the user.

2.1 Tier I

The first tier in the testing protocol is the bulk chemical analysis of the sediments. The list of contaminants for which bulk chemical data are available is identified by the user, and the concentrations of each contaminant for both the test sediment samples and the reference sediment samples are input in a spreadsheet format, in units selected by the user. Upon entry of these data, DADEES executes the primary set of rules in tier 1 to answer the question:

- *Do all test samples show concentrations below the average concentration of the reference samples, per contaminant?*

If the answer to the above question is *Yes*, then disposal of the dredged material in the area from which the reference samples were taken will create no long-term increases in the contamination levels of the disposal site, so DADEES arrives at the conclusion that *aquatic disposal is allowed with no restrictions* and the decision making analysis is completed. Otherwise, further testing is required.

At this stage of development, DADEES invokes a relatively detailed component, auxiliary to tier 1 testing, wherein potential maximum bioaccumulation values are estimated. These values are the Thermodynamically-defined Bioaccumulation Potentials (TBP's). The steps taken in the calculation of the TBP values in this module of the program are developed by McFarland and Clark (1986). Here it will suffice to note the actions taken by DADEES upon evaluation of the TBP's. Firstly, DADEES can evaluate TBP's only for those contaminants which are neutral organics. The TBP values for the neutral organic contaminants are presented to the user, who is asked to identify those values of concern. Then,

- *If the user identifies none of the TBP values as showing magnitudes of concern, and if all of the contaminants which failed primary tier 1 testing are neutral organics, then DADEES concludes that aquatic disposal is allowed without restrictions, subject to a Local Authority Decision (LAD) on further testing.*

If this is the case, the final conclusion is reached and the program may exit. Otherwise, either some of the contaminants showed TBP values of concern, or some of the contaminants failed the bulk chemical analysis and are not neutral organics and so were not considered in the evaluation of maximum bioaccumulation values, in which case DADEES proceeds to the next level of testing, tier 2.

2.2 Tier II

Tier 2 testing is the experimental evaluation of actual toxicity levels resulting from local species' exposure to the contaminated sediments. At this stage, DADEES requires the results of the bioassay (toxicity) experiments in the form of percentage toxicity per test and reference sediment sample. Also required is a percentage toxicity for a control sample. This value is used to calculate the LD50, which equals the toxicity of the control sample plus 50 percent.

Once DADEES obtains these data from the user, it enters the tier 2 rule base to determine whether a final conclusion may be drawn at this point or whether further testing is required. These rules may be summarized in the following two questions answered by DADEES. First,

- *Is the toxicity of any test sample greater than the LD50?*

If so, DADEES concludes that aquatic disposal is allowed only with restrictions and is finished. Otherwise, DADEES determines:

- *Are all test sample toxicities below the average of the toxicities of the reference samples and below the LD50?*

If so, DADEES arrives at the conclusion that aquatic disposal is allowed with no restrictions, with a Local Authority Decision (LAD) on further testing. On the other hand, if any test sample toxicity is above the average of the toxicities of the reference samples, DADEES enters the third tier of testing, the bioaccumulation tests.

2.3 Tier III

At this point, the user must provide results of experimental bioaccumulation assays, per test and reference sample, and per contaminant; DADEES then invokes the tier 3 rulebase to draw a conclusion. Currently, DADEES recognizes only single-species bioaccumulation data. This rulebase may be represented in the following three questions:

1. *Do FDA Action Levels exist for all contaminants?*

   If the answer to question 1 is *No*, DADEES concludes that a Local Authority Decision (LAD) is required, which may lead to either restricted or unrestricted aquatic disposal. Otherwise DADEES continues to question 2:

2. *Is the bioaccumulation concentration of any test sample above an existing FDA Action Level?*

   If this is the case, DADEES concludes that aquatic disposal requires restrictions. Otherwise, we continue to question 3:

3. *Is the bioaccumulation concentration of any test sample above the average of the bioaccumulation concentrations of the reference samples?*

   In the case where this is true, then all the test samples must have bioaccumulation concentrations below the FDA Action Levels (by question 2), yet some test concentrations are above the reference sample concentrations. Under these conditions, DADEES concludes that a Local Authority Decision (LAD) is required, which may lead to either restricted or unrestricted aquatic disposal. If the answer to question 3 is *No*, DADEES concludes that aquatic disposal is allowed with no restrictions.

At this point DADEES has reached a final conclusion, based upon the bioaccumulation data, and any required LAD's.

Work is continuing on the development and expansion of DADEES. Only a portion of the overall decision making process for managing potentially contaminated dredged materials has been incorporated in the program to date. DADEES is designed to run on a standard AT clone (286 machine), running DOS, with at least 640K bytes of memory, and a math co-processor. It was originally written in the KES expert system shell but is now rewritten in C.

3 Stream Quality Model Calibration and Use

Although simulation tools and techniques have long been successfully used in many applications, they have expanded and matured in recent years in concert with the onslaught of new and more powerful computer capabilities. Among the more promising enhancements to simulation methods has been the incorporation of artificial intelligence or expert systems technology. This integration is particularly beneficial when focused at simulation models that are complex and difficult to use or calibrate.

In many engineering areas, simulation models are not used to their full potential due to special expertise required by the user and/or the large investment of time and money involved. For such models, an expert system might be used to facilitate and perhaps automate the procedures necessary to run the model.

The focus of this section is on the development of an expert system to aid water resource engineers in the calibration and use of a stream quality simulation model. An expert system shell is used to automate and facilitate a successful stream quality modeling exercise. The shell environment enables fast and efficient prototyping as well as integration of all aspects inherent to traditional stream quality simulation and model calibration. A Streeter-Phelps calculation routine, graphics software, and a program to perform model calibration are all linked externally to a knowledge base which provides 'expert' advice and instructions.

A typical dissolved oxygen analysis of the stream consists of first gathering physical data (e.g., velocity, depth, discharge, temperature, etc.) along the length of river being modeled. Estimates of model parameters (e.g., deoxygenation and reaeration rates) are then made and the resulting model-predicted dissolved oxygen profile is calculated and visualized graphically. Finally, calibration of the model parameters may be performed to achieve conformity between the model-predicted profile and any available measured field data.

A complete dissolved oxygen analysis, incorporating several reaches of a stream, is a rather lengthy undertaking. It requires the collection of a large amount of data, the use of heuristic parameter estimation techniques that may result in conflicting parameter values, the use of expert judgment to resolve these conflicts, the use of a trial and error or formal optimization calibration process for refinement of the model, and the use of graphics to display the large quantities of measured and model-predicted data in meaningful ways.

Entire textbooks have been written on the subject of water quality modeling (e.g. Thomann and Mueller, 1987). Therefore, only a limited description of the stream dissolved oxygen analysis will be provided. The principal components acting to deplete stream dissolved oxygen include carbonaceous biochemical oxygen demand (CBOD), nitrogenous biochemical oxygen demand (NBOD), sediment oxygen demand (SOD), and aquatic plant respiration. Oxygen contributors are reaeration from the atmosphere and photosynthesis by aquatic plants.

Organic wastes, contained in point sources such as domestic and industrial sewage and non-point sources such a storm water runoff, when introduced into a stream system, directly affect the concentration of dissolved oxygen through the activity of microorganisms that derive a food source from the waste. As the amount of food source (wastes) increases, these microorganisms reproduce and more dissolved oxygen is required for metabolic energy. As the waste is consumed, the food source can no longer support the increased microorganism population and some die off. As the dissolved oxygen concentration in the stream decreases due to these microorganism activities, the rate of reaeration from the atmosphere increases. The result is that the stream reaeration rate eventually becomes greater than the stream deoxygenation rate and the stream is able to recover.

For this application, and thus demonstration of the use of an expert system tool to assist decision-making, photosynthetic effects have been ignored but carbonaceous BOD (CBOD), nitrogeneous BOD (NBOD), benthal or sediment oxygen demand (SOD), and reaeration are all considered. The differential equation describing the dissolved oxygen reactions is:

$$\frac{\partial D}{\partial t} = U\frac{\partial D}{\partial X} = -K_a D + K_d L_0 \, e^{-K(\frac{X}{U})} + K_n N_0 \, e^{-K(\frac{X}{U})} + \hat{S}_b \qquad (1)$$

where X is the distance downstream measured from the top of the reach (m); D is the dissolved oxygen concentration deficit measured from the saturation concentration (mg/L); U is the velocity of the stream which is assumed within a reach to be constant (m/d); L is the CBOD concentration at the top of the reach (mg/L); N_0 is the NBOD concentration at the top of the reach (mg/L); K_a is the reaeration rate coefficient (d^{-1}); K_d is the CBOD deoxygenation rate coefficient (d^{-1}); and K_n is the NBOD deoxygenation rate coefficient (d^{-1}). $\hat{S}_b$ is the effective SOD rate throughout the volume of water in the stream expressed in units of mg/L/d. It may be estimated by multiplying the usually reported SOD rate S_b (g/m^2/d) by the hydraulic radius (m^2) of the stream and then dividing by the cross sectional area (m^2) of the stream, or by dividing S_b by the depth (m) of the flow. Integration of Eq. 1 yields Eq. 2 which is actually used in the expert system.

$$D = D_0 \, e^{-K(\frac{X}{U})}$$

$$
\begin{aligned}
&+ \frac{K_d L_0}{K_a - K_d}\left(e^{-K_d\left(\frac{X}{U}\right)} - e^{-K_a\left(\frac{X}{U}\right)}\right) \\
&+ \frac{K_n N_0}{K_a - K_n}\left(e^{-K_d\left(\frac{X}{U}\right)} - e^{-K_a\left(\frac{X}{U}\right)}\right) \\
&+ \left(1 - e^{-K\left(\frac{X}{U}\right)}\right)\frac{\hat{S}_b}{K_a} \qquad (2)
\end{aligned}
$$

Other models of dissolved oxygen with more or fewer terms can be accommodated in the expert system.

At the outset of a run time session using the expert system, the total length of river to be modeled must be divided into a number of 'reaches.' For modeling purposes, each reach is assumed to have constant characteristics or model parameter values throughout its length. The system describes to the user typical occurrences along a river where a reach subdivision should be made: for example, point source discharges, slope changes, confluences, dams, waterfalls, etc. The calculations and/or calibrations will be performed for each reach sequentially.

Downstream boundary conditions from one reach will be transferred automatically to upstream boundary conditions for the next downstream reach and a mass balance performed in the case of inflow of some form. The system will keep track of results for individual reaches and can display continuous results for all reaches previously modeled at any time during the run time session.

Next, the expert system queries the user to determine whether the particular run time objective is for simulation or calibration. If it is simulation, the expert system merely acquires all necessary input data, performs the dissolved oxygen calculations and graphically presents the results to the user. This is performed for all the reaches sequentially. The calibration option is a bit more involved. If chosen, it implies measured field data are available and the user would like to calibrate the model to give a 'best fit' to the measured data.

For each reach, the value of the reoxygenation parameter, K_a, is estimated first using 'rules based' inferencing. These rules are based on empirical formulas and are intended only to give the user an approximate starting value of K_a, if one is not available. The user is informed of an empirical value of K_a as well as the corresponding literature reference. If more than one empirical formula applies for the given conditions, the user will be informed of each value. The system recommends values for K_a, but leaves the user the option of choosing any value.

Next, the value of the carbonacous deoxygenation coefficient, K_d is estimated. The user is first queried as to whether or not CBOD field measurements are available for the reach. If field data are available, the system is programmed to compute K_d based on those data using a sum of least square deviations procedure on log CBOD vs time downstream. Again, the user is left with the final decision as to the value of K_d. If no field data are available, the system queries the user for a value of K_d. The model possesses an analogous capability to determine the value of nitrogenous deoxygenation coefficient, K_n.

Next, the user is asked if there is any inflow into the stream at the beginning of the reach. The two most likely causes of such inflow would be from a point source discharge or another tributary stream. In some instances, it may include non-point source discharges. If there is inflow, the user is requested to input the total incoming discharge, the dissolved oxygen, CBOD, and NBOD concentrations as well as temperature, so that a mass balance calculation can be performed with the upstream conditions.

Now that all information pertinent to the reach has been ascertained, it is passed to an external calculation routine via a communication file. The user is informed that the calculation routine has been invoked and the appropriate adjustments have been made to K_a, K_d, K_n and S_b due to temperature of the stream. The calculations routine uses Eq 2 to determine the dissolved oxygen profile in the reach and a graphics package to display the profile for the user.

The calibration phase of the expert system is an extension of the simulation phase whereby the dissolved oxygen controlling parameters are perturbed iteratively until the calculated values are matched as closely as possible to the measured dissolved oxygen values. The calibration phase begins to deviate from the simulation phase after the first graphic of calculated dissolved oxygen vs length is brought to the screen. At this point, the user is asked to enter the measured dissolved oxygen field data for the reach. The user would have previously indicated that field data were available by choosing the calibration alternative for the run time option. Now that the system possesses calculated as well as measured data, it has the capability of presenting graphically the two together.

To calibrate the parameters for this reach, the expert system uses an external LISP program which will implement a pattern search to determine which direction in a four dimensional space will yield a lower expected error. The four dimensional space comprises the four oxygen controlling parameters K_a, K_d, K_n and S_b.

The search continues until one of three ending criteria is met: expected error is less than 1.00%, subsequent iterations do not improve expected error or ten iterations are performed. Other stopping criteria could easily be implemented if desired. Once calibration is complete, the user may choose to view a graphic of measured dissolved oxygen concentrations and model-predicted or calculated dissolved oxygen concentrations using the calibrated parameter values along the length of the reach. Once the model has been calibrated for this reach, the next downstream reach is considered.

Acknowledgment The work described in this paper is due in large measure to the efforts of Tim Ginn and Dave Wood.

References

Lee, C. R. *et al.*, (1985), *Decision making Framework for Management of Dredged Material: Application to Commencement Bay, Washington*, Miscellaneous Paper D-85-, U.S. Army Engineer Waterways Experiment Station, Vicksburg, Mississippi.

Maher, M. L., (1987), *Expert Systems for Civil Engineers: Technology and Application*, American Society of Civil Engineers, New York, NY.

McFarland, V. and Clarke, J., (*Draft*), 'A Simplified Approach for Evaluating Bioavailability of Neutral Organic Chemicals in Sediments,' Draft Technical Note, U.S. Army Corps of Engineers Waterways Experiment Station,

Ortolano, L. and Steinemann, A. C., (1987), 'New Expert Systems in Environmental Engineering,' *Journal of Computing in Civil Engineering* **1**, 4, 298–302, (October).

Thomann, Robert V. and Mueller, John A, (1987), *Principles of Surface Water Quality Modeling and Control*, Harper and Row, Inc., New York, NY.

The Development and Application of an Expert System for Drought Management

Richard N. Palmer
Department of Civil Engineering
University of Washington
Seattle, Washington
United States of America

Abstract A recurring problem in applying systems analysis tools in water resource management is the successful presentation of results to decision makers. Expert systems have been suggested as a decision tool to create models that are more easily incorporated into 'real' decision environments. The use of such models has been limited in the management of water resources, however, by the complex environment in which such decision are made and difficulty in developing simple rules of thumb for operation. In addition, modelers are often reluctant to abandon the information and insights offered by computer models.

This paper describes the development of a decision support system used to manage the Seattle, Washington water supply during droughts. The system includes an expert system, a linear programming model, database management tools, and computer graphics. The expert system incorporates operator experience and expertise using a rule base developed with interviews of water management personnel. The expert system is also used to integrate the other modeling techniques into a single system. The linear programming model determines system yield and optimal operating policies for past and predicted hydrologic regimes. Database management and graphic software allow the display of over two thousand operating policies generated by the linear program.

1 Introduction

This paper describes the development and application of a computer model used by the Seattle Water Department (SWD) for operations during drought. This model provides guidance for initiating voluntary and mandatory water-use restrictions. The model uses an expert system to integrate several programming techniques including linear programming, database management system, and computer graphics. The model is mathematically complex yet highly user-friendly. It incorporates both subjective and quantitative information. Heuristic knowledge obtained from managers of the water supply system provides the foundation for the expert system.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 177–187.

Although expert systems techniques are advocated for a wide variety of settings in Civil Engineering (Fenves, *et al.* 1984, Fenves 1986) few examples of successful applications exist in water resources. This is due, at least in part, to the complex decisions required for proper management. It also results from difficulties in converting these decisions into the simple 'rules of thumb' associated with expert systems. This paper directly addresses these problems. The approach suggested requires extensive numerical analysis before a drought event. However, it provides quantitative and qualitative information to aid decision makers in making rational and consistent operating policies.

This paper begins with a brief introduction to expert systems and their characteristics. Next, it describes the Seattle water supply system with a chronology of the 1987 drought event, the worst drought event on record. Procedures for drought management used before this event are also characterized. The remainder of the paper discusses the development and application of the integrated computer software used for drought management.

2 Seattle Water System

The Seattle Water Department (SWD) provides direct service to 541,000 Seattle residents and is a wholesaler to 549,000 residents of King County (SWD 1986). Water is taken from the Tolt and Cedar Rivers, both of which originate in the Western Cascade Mountains. Minimum instream flow requirements limit diversions from each river. These instream flow requirements exist to maintain water for fisheries, hydropower generators, and recreation usages.

Although averaging over thirty inches of rainfall annually, the area is susceptible to droughts. The SWD estimates its safe yield at 169 MGD with a 98% reliability (one shortage event in fifty years). Safe yield is the seasonally varying, maximum volume of water available while meeting operational constraints of the system. Municipal water demands currently average 170 million gallons per day (MGD) annually and are increasing at the rate of 2 MGD per year. Until new sources are developed, little excess capacity exists to meet unusually high demands or low supply situations.

For successful operation during the summer, reservoir storage levels must be near capacity after the spring snow melt. Average rainfall during July and August is 1.8 inches; thus, the system also depends on autumn precipitation to refill its reservoirs. Unusual climatic events, such as those that occurred in 1987, cause system storage to decline to levels that require water use restrictions.

To guarantee an orderly response to any water shortage, the SWD developed the Water Shortage Response Plan (WSRP). This plan addresses problems related to the 1-in-50 year drought event (SWD 1986). The objective of this plan is to maintain essential services while minimizing the net economic loss during a drought event. The WSRP envisions two types of shortages: a summer shortage and a fall shortage. Each type of shortage consists of a multi-stage conservation plan with progressively higher stages initiated as serious conditions develop. Summer shortages result from climatic and hydrologic conditions that cause system reservoir levels not to be refilled by late spring. The fall shortage scenario results from low fall precipitation that is insufficient to replenish system storage after summer peak use.

Conflicts between the City of Seattle, the Corps of Engineers, and Washington State Department of Fisheries, and the Washington State Department of Ecology are common during periods of low flow. These conflicts result from the different objectives the agencies have for water use in the Tolt and Cedar watersheds. Figure 2 illustrates the decision process used before 1987. Decisions were made in a formal, if ad-hoc manner, attempting to weigh conflicting objectives of the agencies. Only a limited amount of quantitative data were available to all agencies and often the precise impacts of decisions were not known.

3 The Drought of 1987

The 1987 drought began in the early summer and continued into the late fall. Total precipitation for the 1987 Water Year (October 1–September 30) was 80% of average. Above normal rainfall and snow occurred in the winter and early spring. However, strict adherence to flood control levels prevented storage of the water. Abnormally high temperatures during this period left the snow pack depleted and increased reliance on summer flows. Flows in the late spring were below normal and the reservoirs did not fill to maximum water supply capacity. By early June voluntary water use restrictions were considered and by late June, they were a reality.

Flows during June and July proved to be among the lowest on record. On August 3, mandatory water use restrictions were initiated including limitations on outdoor water use. These restrictions were the first required since the early 1960's. These restrictions resulted in decreased demands, but total usage remained some 20 MGD above the desired target. By late August, the drought became one of the worse, if not the worse, on record. The regional nature of the drought became obvious by September. Tacoma, Washington announced required purchases of water from other suppliers. Tacoma had relied on the fall rains and had not instituted any significant restrictions.

The decisions required in this situation are similar to those faced by the managers of other systems during an extended drought. When should the public be informed of impending problems? When should voluntary restrictions be initiated? What level of voluntary restrictions are required? When are voluntary restriction insufficient and mandatory restrictions required? When can operations return to normal?

4 Development of SID

Several programming approaches were used to develop an integrated model with the characteristics previously described. The model is denoted as SID (The Seattle Water Department Integrated Drought Management Expert System). A commercial expert system shell (Level 5 Research, 1986) is the primary interface with the user. The expert system activates other software programs providing specific functions not available with the expert system in isolation. In addition, the expert system serves to incorporate operator experience and institution constraints.

Two fundamental problems existed in the development of the integrated model. The primary problem concerned incorporating operator experience because little public knowledge existed describing operating policies. The second problem related to the development of quantitative information to aid decision makers. This information was to serve as guidance for operators and not to diminish their role as decision makers.

The flow solution approach adapted in SID was the integration of several types of modeling approaches. An expert system incorporates general operating rules directly using the rules developed by SWD managers. A linear programming model generates specific information concerning the probability of system yields and potential economic losses. In addition, the linear programming model generates specific operating policies for a sixteen week period. These policies include when to initiate water use restrictions and at what level. Database software stores these results and makes them accessible to the expert system. When necessary, the expert system also accesses graphic routines to display the results. After reviewing the rules and suggestions of the expert system, the user can modify specific policies to evaluate their impact on operation.

SID's primary output is a one week operating policy. SID is operated weekly to generate subsequent restriction policies. Because of rapidly changing climatic conditions, it is unwise to develop rigid long range policies. Each of SID's components is described below.

5 Description of Linear Programming Model

Although the expert system contains general rules of operation provided by the system managers, quantitative information can supplement operator experience. This provides operators with an estimation of the system drought potential based on its state at a given time. This information is generated using a position analysis (Hirsch 1977). In this procedure, past streamflow data serve as surrogates for potential future inflows. State variables, time and storage, are defined and the system operates for a prescribed period.

Hirsch performed his analysis with a simulation model, however, this research uses a linear programming model for this purpose. This model (denoted as LPW) operates on a weekly time period, incorporates physical and operational constraints and optimizes the system's operation. Two objectives are used: maximize system yield and minimize the economic loss associate with deficits from a specified (and time dependent) target. For this system, the planning period is four months and each year is an independent event. Streamflow data exists for nearly fifty years for primary sites of interest in the system, making this an especially attractive approach.

When estimating the system yield, the constraints include continuity at both reservoirs, instream flow requirements and continuity on the moraine aquifer in the Cedar system. The Cedar system is more complex because of this aquifer. The aquifer is recharged by seepage from Masonry Pool and returns water to the Cedar River further downstream. Both occur at unknown rates and which the model approximates (Palmer and Johnston 1984). Bounds exist for storage levels on all reservoirs, pipeline capacity and instream flow requirements for the Tolt system and Lake Washington elevation.

The second objective requires several additional constraints to meet water use implementation requirements. Upper bounds on each stage of WSRP restrictions represent the maximum reduction in water use that is possible for each stage. A piece-wise linear objective function approximates the economic losses associated with the implementation of water use restrictions. Additional constraints limit stage increases to one per week, preventing staging from skipping two or more levels in one week.

The model was executed using the historical record (1929–1975) for a variety of reservoir storage levels, starting dates, and system demands. System yield was calculated for each streamflow record for an initial storage of between 10 and 100% of capacity (by units of 10%) and for June through October. This requires 2,450 runs of the model. The results of the yield analysis identified configurations that result in economic losses for specific demands. The minimum economic loss, and its associated operating policy, was calculated for the appropriate configurations for base level demands of 170, 180, 190 MGD. This resulted in approximately 600 addition runs of the model.

Execution of each yield run requires approximately three minutes on an IBM/AT using XA, a linear programming software package (Sunset Software 1987). Execution time of the economic loss are less predictable, requiring between six and twelve minutes. Approximately 200 CPU hours were required for all linear programming runs. This task was simplified by writing software to automate this process.

6 DataBase Management

Database management software stores the results of the linear programs. Access to the previously generated results allows them to play a significant role in real-time operation. As previously stated, computation time prevents real-time generation of the results. The database management techniques also allow incorporation of the results into the expert system rule base.

Record code-information is used by graphics routines and expert system to access the correct record, starting week ,base level demand, initial storage, yield array, economic loss array, total number of drought years, drought years, and Water Shortage Response Plan staging data for drought years.

7 Graphics Routines

Software used to generate graphic routines activated by SID are written with the Turbo Pascal Graphics Toolbox (Borland International 1985). The graphics software displays the cumulative distribution functions (CDF) for system yield and economic losses, optimal WSRP staging sequence for the 10 worst droughts on record, and a weighted staging sequence. The expert system activates the graphics software for specific system configurations (initial week, initial storage levels, and base level demand) that have a potential for water shortage. Subsequent sections present examples of the displays.

8 Expert System Rule Base Development

The expert system functions to integrate the other programs used in SID. However, its primary purpose is to provide a mechanism to incorporate a rule base developed by SWD personnel. Drought management decisions rely on subjective evaluations and operational experience as well as quantitative variables. Decisions are made relative to the severity of a drought, the potential economic effect it may present and the proper management strategy to minimize it's impact. Although quantitative analysis can supply information to aid in these decisions, a final decision must also include the judgement of the manager who is responsible for the results. The goal of SID is to incorporate human expertise and insights into the modeling of water supply operation, allowing an accurate representation of the decision making environment.

Interviews with SWD personnel were conducted to determine the system variables most significant when making drought management decisions. Two SWD representatives participated in this effort: David Parkison, head of the engineering planning section and Rosemary Menard, chief information officer of the water conservation section. These individuals had direct responsibility for system operations and public information during drought events. A series of modifications and additions to the rule base were made through these meetings. In the interviews the most significant management information was defined as the likelihood and severity of potential system shortfalls. They also defined the most relevant results from the linear programming model as the indicators of potential system failure: time of year, storage levels, current demand, future inflows, future demands, estimations of yield for extreme conditions (such as the ten most severe years on record) and estimations of the future drought potential.

9 Expert System Rule Base for Drought Management

The rule base that developed contains two types of rules. Type I rules provide the user with general drought potential information for a system configuration but do not provide operation guidance. Type II rules incorporate the results from historic drought events and rules concerning system operation to recommend a specific action.

Type I rules require the user to provide the initial system configuration to the model. The user enters the initial week, initial system storage, and base level demand into the expert system. For the given configuration the system then rates the drought potential: Severe, Serious, Moderate, Minor, or None. This provides the user with general information on the severity of the current system configuration compared with historic events. An example of a Type I rule for July is:

Rule July 1
if The initial week is July 1
and Initial system storage = 70
and Base demand = 170
then Conservation restriction may be required
and Drought Potential is Minor

Discussion with SWD personnel lead to drought potential being defined as: None, for no droughts in the database for a given configurations; Minor, for 1 drought in the database; Moderate for 2 droughts in the database; Serious for 3 or 4 droughts in the database; and Severe of 5 or more droughts in the database.

Type II rules provide the user with guidance on system operations. The user enters a prediction of the expected inflows and demands for the upcoming month. The program then estimates the next month's system storage. From the database, the systems returns values for the number and economic loss of droughts associated with the current system configuration, the average yield of the 5 smallest years (the average 10% yield) and the number of droughts associated with next month's predicted configuration.

The system uses this information to determine the proper water use restriction level. SID uses the appropriate variables first to determine whether to lower the current drought level restriction. If this goal cannot be achieved, the system pursues rules associated with the goal of remaining at the current level of restriction. If none of these rules are satisfied, the system pursues rules associated with the goal of increasing the current level of restriction.

There are five basic heuristics gathered from interviews with system experts related to general operation. These are:

1. Lower stages (WSRP stages 1 and 2) are readily implemented and require examination of only a few system variables

2. Higher stages (WSRP stages 3, 4, and 5) are more difficult to implement and require the examination of many system variables

3. Stage 2 restrictions can be implemented directly from stage 0 if drought conditions are potentially serious

4. Stage 3 and stage 4 are interchangeable except for the season of implementation. Stage 3 is implemented during summer months (June, July, and August). Stage 4 is implemented during fall months (September and October)

5. Reduction in staging will not occur until system storage levels are sufficient to meet NORMAL fish flow requirements.

Different criteria are used for the lower and higher restriction level to determine whether drought restrictions are decreased. Both assure that normal fish flow requirements are met before stage reductions are made. Over 140 rules are incorporated into the rule base.

10 Use of Integrated Drought Management System

It is difficult to portray the use of SID in written text because of its reliance on user interactions and computer graphics. This difficulty is not unique to this model but has been a consistent problem in the literature. Unseen by the user, SID moves between the expert system shell, graphic software, database software, and other subroutines written in Fortran.

SID begins with an animated color screen showing reservoirs filling and emptying. ('Animated' implies movement on the screen.) Nine color graphic 'HELP Screens' follow describing the various functions of SID. These ten introductory screens are written in Turbo Pascal. An experienced user can activate SID and bypass all introductory information.

Next, SID activates the expert system. Three screens obtain user information describing the system. This information includes the week of interest, the current storage in the reservoirs, and the municipal water demands. Each screen is accompanied by HELP facilities that allows further explanations of the questions presented. In addition, the user can ask SID why the question is posed. If queried, SID presents the rule in its rule base that caused the question to be asked. At any point in the consultation, the user can request from SID a listing of all rules or any system variables.

Using this information, SID examines the database and classifies the drought potential as Severe, Serious, Moderate, Minor, or None using criteria described previously. If the drought potential is defined as none, the SID suggests that the user either begin with a new system condition or terminate its use.

If the drought potential is any level but 'None,' SID activates the graphics software. This software plots the cumulative distribution functions of system yield and potential economic loss. The plots contain statistical information concerning the average behavior of the system and the average of specified quartile events. The ten most severe events are listed in the order of their severity in the economic loss graph.

Next, the user identifies specific drought events for which more information is desired. The sixteen week optimal WSRP staging sequence for these droughts are presented as histograms, with as many as four presented on the screen at once. These histograms illustrate optimal management policies for past droughts that minimized economic impact. If desired, the user can develop a sixteen week policy for the current drought by weighting the staging associated with any of the ten previous droughts. The selection of the previous droughts to include depends upon their estimated probability of occurrence, the level of economic loss, or degree to which the user believes the current drought situation under evaluation is similar to any of the previous droughts.

The user then estimates the total system demand and total system inflow for the next month and the current level of water use restriction. Demand and supply can not be estimated with precision, however, supporting software (not incorporated into SID) has been developed to aid in this estimation. Using these estimates, SID evaluates the drought potential if these demands and supplies do occur during the next month. Drought statistics for the current month and the predicted subsequent month are then presented. The drought potential for both months are then used to determine the appropriate level of restriction for the current week.

The user can alter any input variable and examine the sensitivity at the conclusion of the consultation with SID. The rules activated in deciding the appropriate level of restrictions can be reviewed allowing the user to understand the exact logic used. If a rule appears inappropriate, it is possible to move directly into the rule base, alter the rule, re-compile the program, and begin a new consultation.

11 Conclusion

This paper describes the development and application of an expert system designed to aid in drought management. The model integrates the use of linear programming, database management, and computer graphics using an expert system. In addition, the expert system is used to incorporate the experience and insights of system managers into the range of possible operating policies.

Specific operational objectives for the model were identified at the outset of model development and the model was constructed to meet these goals. The need for real-time analysis and user friendliness dictated many of the techniques that were incorporated. Optimization techniques were needed to identify system yield and operation, but were inadequate in capturing the more subjective aspects of system operation and in presenting the results in a fashion meaningful to system managers.

The authors contend that the development and use of the expert system greatly aided the system managers during the 1987 drought. Because the event will not reoccur such a contention can not be proved nor disproved. However, the model did provide information that allowed managers to recognize the significance of the drought and its relationship to past events. In the process of developing the rule based, the managers formalized specific approaches to operation that allowed consistent operation for the droughts duration. The 1987 drought required the development of an improved rule base for operation that will aid future managers when other droughts occur.

Acknowledgment This research was funded by the Seattle Water Department. Numerous individuals at that agency contributed to this research including David Parkinson, Rosemary Menard, Julia Cohan, and Lionel Sun. Robert Tull, currently of Camp, Dresser, and McKee (San Francisco) made valuable contributions at the initiation of this project and much of its success is due to his efforts. William McCarthy also deserves special thanks for the generation and evaluation of linear programming data. The authors also wish to acknowledge their good fortune in that the completion of this project coincided with the most severe drought in the previous sixty-four years. This event brought a relevance to the work which might otherwise have been lacking.

References

Borland International, (1985), *Turbo Database Toolbox, User's Manual*, Borland International, Scotts Valley, CA.

Borland International, (1985), *Turbo Pascal Graphix Toolbox, User's Manual*, Borland International, Scotts Valley, CA.

Fenves, S. J., (1986), 'What is an Expert System,' in *Expert Systems in Civil Engineering*, Kostem, C. N. and Maher, M. L. (Eds.), American Society of Civil Engineers, New York, N.Y.

Fenves, S. J., Maher, M. L. and Sriram, D., (1984), 'Expert Systems: C. E. Potential,' *Civil Engineering Magazine* **54**, 10, 44–47.

Hirsch, R. M., (1979), 'Synthetic Hydrology and Water Supply Reliability,' *Water Resources Research*, **15**, 6, 1603–1615.

Level Five Research, (1985), *INSIGHT2+ User's Manual*, Level Five Research, Indialantic, FL.

Palmer, R. N. and Johnston, D. M., (1984), 'Completion Report for Optimization of Yield Analysis on Seattle Water Supply System,' Report to Seattle Water Department, Seattle, WA.

Palmer, R. N. and Tull, R., (1987), 'Expert System For Drought Management Planning,' *Journal of Computing in Civil Engineering, ASCE* **1**, 4, November, 284–297.

Palmer, R. N. and Holmes, K.J., (1988), 'Operational Guidance During Droughts: Expert Systems Approach,' *Journal of Water Resources Planning and Management, ASCE* **114**, 6, November, 647–666.

Seattle Water Department, (1986), *1985 Conservation Plan in the 1985 COMPLAN*, Vol. 6, Seattle Water Department, Seattle, WA.

Sunset Software Technology, (1987), *XA, A Professional Linear Programming System*, San Marino, CA.

The Potential Use of Decision-Support Systems for Integrated River Basin Management

D. G. Jamieson
Thames Water
Vastern Road, Reading
England

Abstract Proposals are made for the eventual real-time, near-optimal control of an entire river basin for a wide-range of different functions including river management, water supply and sewage disposal. A satisficing approach within a modular, hierachical control framework is advocated. In this way, scarse resources can be allocated across competing interests and the degree of conflict between non-compatible activities minimised.

1 Introduction

1.1 Background

Prior to the 1974 reorganisation of the water industry in England and Wales, river basin management depended on cooperation between independent organisations. In order to safeguard vested interests, there had to be prior agreement between the various authorities which usually took the form of abstraction licences, discharge consents, etc. While these arrangements worked reasonably well in normal circumstances, they could be somewhat of a hindrance in times of stress, especially if assistance to one aspect of river-basin management infringed an agreement on another, since that was likely to be with a different organisation. Moreover, since different organisations were only accountable for their own particular aspect, there was little incentive to assist others over and above their prior commitment, however sensible it might have been in the general interest.

Since the creation of water authorities, all aspects of river-basin management have been vested in the one organisation. At least in theory, this could lead to a more flexible operational policy based on objectives rather than agreements. However, a dynamic operating strategy which is continually updated in response to

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 189–203.
© 1992 *Kluwer Academic Publishers.*

changes in the system state and user interests would necessitate a more sophisticated monitoring and control scheme than one following pre-set rules.

More recently, the Government has outlined its intentions for privatising the water industry. In essence, the intention is to create a National Rivers Authority, thereby separating the regulatory and river management functions from water supply and sewage disposal. Whilst at first sight, this runs contrary to the concept of integrated river basin management, nevertheless a large element will of necessity, have to continue. Therefore, rather than having a prescribed operating agreement, the two organisations could adopt a common dynamic operating strategy.

1.2 Aims

The aims of this paper are as follows:

- to structure the general problem of operational river basin management by identifying compatible and non-compatible interests;
- to review control techniques currently available and postulate developments;
- to propose a framework for the eventual real-time management of an entire river basin system on an integrated basis.

1.3 Limitations

It is inevitable that an overview of this nature will be somewhat superficial and not show due reverence to the many practical difficulties. Moreover, in certain aspects the exposure of ideas for future operational control is somewhat premature since these are the subject on ongoing research. Nevertheless, it is hoped that the overall concept is not obscured by these shortcomings and that the approach adopted is applicable elsewhere.

2 River basin management

2.1 Interaction

For the purposes of this paper, river basin management is defined in its broadest terms and deemed to include all factors which relate to or impinge on the natural drainage of a river system. The 1973 Water Act recognises that most aspects of river basin management are inter-related and cannot sensibly be segregated, particularly at an operational level. Therefore, any attempt of rational control becomes an exercise in conflict management.

2.2 Component Parts

Within the overall water cycle, a multitude of separate facets can be identified but space considerations restrict this paper to the following common uses of rivers:

- land drainage—natural and man-made;
- water supply—direct or via storage reservoir;
- sewage disposal—raw sewage or treated effluent;
- navigation—natural or maintained depth;
- hydro-power generation—conventional or low-head;
- cooling water—oil or coal-fired power stations;
- fisheries—angling and fish-farming;
- recreation and amenity.

2.3 Compatibility of Interests

Obviously, a river basin managed for one specific aspect may well provide incidental benefits for some additional purposes but conflict with others. However, the constituents of river-basin management can roughly be divided into conservation activities and disposal activities. Conservation activities include water supply, navigation, hydro-power, recreation and amenity: disposal activities include land drainage, sewage disposal and cooling water. In general there would seem to be more common interests within each of these groups than between them. For example, the effect of storage tends to be an asset for conservation activities but a hindrance for disposal activities.

Such generalisations should not be stretched to the limit since some aspects of disposal activities are not an embarrassment to conservation activities and may well be a positive benefit as in the case of effluent disposal being used for downstream water supply. In fact, the more detailed any consideration of compatibility becomes, the more inappropriate the simple classification is seen to be, since the degree of fit depends on not only the particular interest but also the state of the system.

3 Objectives

3.1 Traditional View

Where river-basin management objectives have been formalised in the past, it has usually been a result of having to maintain statutory obligations or legal constraints such as minimum residual flow, navigational depth, effluent consent standards, etc. All considerations of the interactive nature of river-basin management are embraced within the constraints themselves. This has produced management targets which are static and therefore achievable with low-level technology.

However, even comprehensive pre-set operating rules are unlikely to be 'optimal' in any sense if no account is being taken of changes in the system state. Recognising the inadequacies of the methodologies currently employed, consideration has been given to ways of improving real-time control procedures albeit usually on a piecemeal basis.

3.2 Difficulties

Unfortunately, it is a fact of life that the different interests in river basin management have different criteria for assessing performance. Whilst minimising operating costs could be an appropriate objective for managing a pump-storage reservoir, it might not have much relevance to flood-alleviation where the objective is likely to be minimisation of flood damage. At least these two objectives have a similar basis, that of cost minimisation, whereas others such as fisheries or perhaps navigation might not even be measured in financial terms.

3.3 Satisficing Approach

Bearing in mind the difficulties of formulating an objective function, it may well be inappropriate to talk of 'optimal' operating procedures for an entire river basin. Instead, a satisficing approach could be adopted. Here, the state of the system is deemed satisfactory for specific interests provided it is within pre-defined boundaries for each activity (Fig. 1). If the system state lies within all those constraints, minimisation of operating costs, say, would be the general objective. If, however, a particular constraint is or about to be violated, then the objective specific to that activity would take precedence.

For instance, if flooding was imminent (the upper-bound constraint on land drainage about to be broken) then the flood mitigation objective of damage minimisation takes precedence over minimising operating costs (Fig. 2). If the depth of water in a river did not meet the navigational requirement (a lower-bound constraint on navigation), water supply might have to draw on storage or use an alternative source even if that was more expensive. Only if more than one set of constraints were likely to be violated simultaneously (say, insufficient depth for navigational requirements and no alternative source for water supply) would priorities have to be assigned.

Of course, there is nothing new in this concept: it is merely formalising what hitherto had been done intuitively. That being the case, there is a danger of simply copying what is currently done manually and computerising the same procedures.

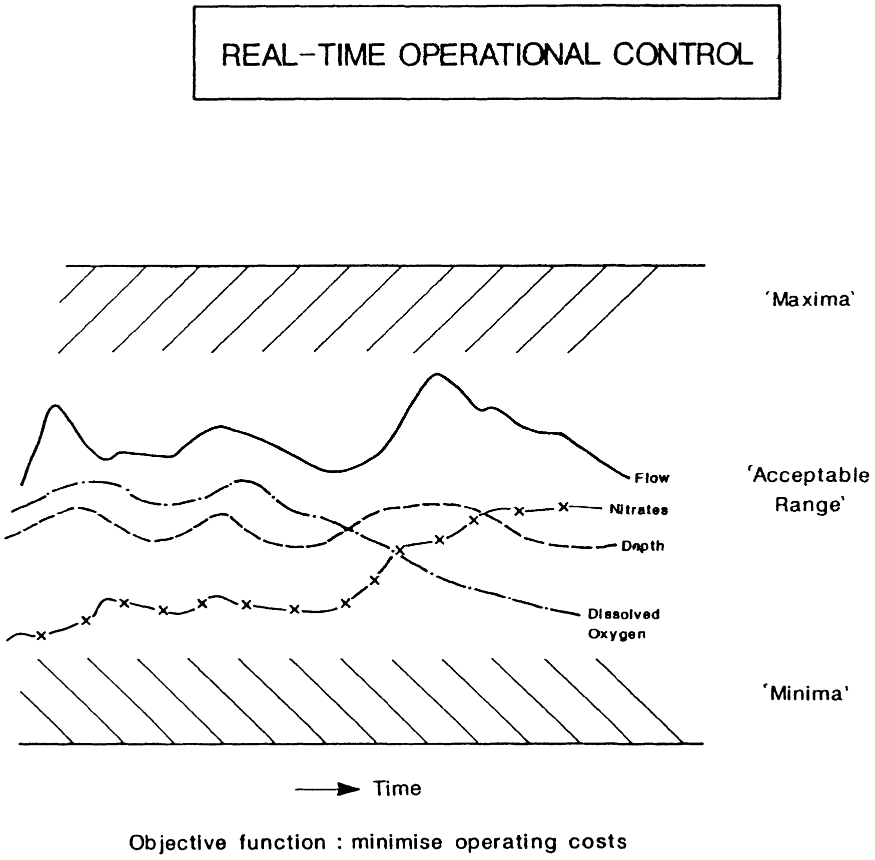

REAL-TIME OPERATIONAL CONTROL
'Maxima'
Flow
'Acceptable Range'
Nitrates
Depth
Dissolved Oxygen
'Minima'
Time
Objective function : minimise operating costs

Figure 1. *Satisficing approach: Normal conditions.*

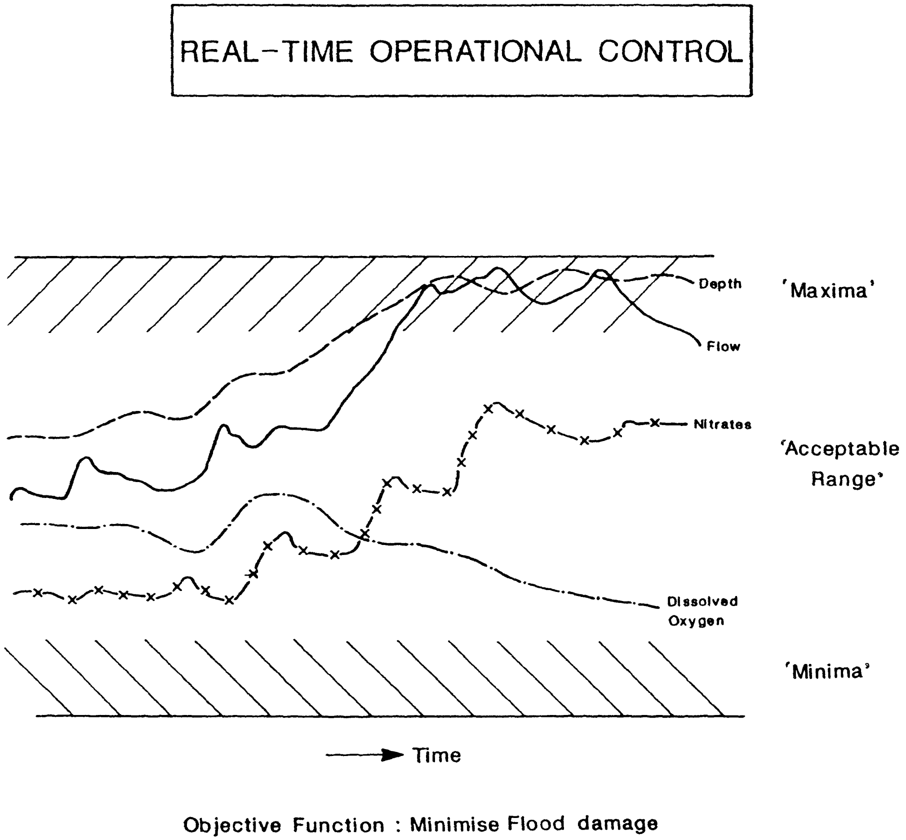


Figure 2. *Satisficing approach: Abnormal conditions.*

However, this would be extremely short-sighted since with the advantages in modern management techniques, those procedures can almost certainly be improved.

4 Operational procedures

4.1 Automation

A pre-requisite to a more flexible operating policy which dynamically responds to changes in the state of the system is quite obviously a definition of the present state. Since it is important to minimise the delay between a change of state occurring and the counteracting decision being implemented, SCADA is considered to be essential.

Depending on what aspect of river-basin management is being considered, the existing state of the system would be measure in terms of river levels, groundwater levels, reservoir levels, sluice-gate settings, quantities pumped, pipe network pressures, water-quality states, etc. The frequency of interrogation will be dependent upon the response characteristics of the specific parameter, e.g. groundwater levels require less frequent interrogation than, say, river flows. Similarly, the precision of measurement required will also be a function of the specific parameter and its response time.

In view of the complexity and the amount of information required even to define the state of one particular constituent of river-basin management, it is unrealistic to assume that any one person could assimilate, digest and utilise all the incoming data At best, the manager would keep a close watch on a number of key variables and ignore the rest.

For this and other reasons, it is expected that future control systems will have a higher degree of automation than at present. This should not be interpreted to mean replcing personnel by computers. On the contrary, the intention is to augment managers' capabilities through decision-support systems about which more will be said later. The role of the manager would be to oversee the control process and take operational decisions. The role of the computer would be to establish the existing state of the system, forecast the future state, assist with decision-making and carry out control instructions.

4.2 Real-time Forecasting

In control engineering terms, the proposal is to treat river-basin management as a feed-forward control system. That is to say, operational decisions are taken on the expected future state of the system rather than the known present state. Since perfect foresight cannot be assumed, there must be ability within the control procedures for self-corrective action as a means of compensating the decisions for the inevitable errors in the forecast.

What few examples there are of feed-forward control systems being used within the water supply industry, usually base their forecasting procedures on deterministic models. The simulation models used have traditionally been adaptations of explanatory models in which the algorithms purport to have physical reality in the sense that they attempt to mimic the real system.

Basically, there are two options for deterministic modelling. In the direct approach, the differential equations of flow/diffusion are themselves approximated (usually linearised) while in the systems approach, an equivalent operator is substituted as an approximation to the actual process. Whilst the systems approach has the advantage of robustness (no instability problems with ill-conditioned equations), the direct approach has the better definition. Besides being time-invariant in the sense that the parameters once estimated are assumed constant, both approaches, particularly the direct approach, are computationally tedious and not particularly suited to real-time use.

Various attempts have been made to improve the forecasting ability of such models. This has usually involved restructuring portions of the model to make it more complex. However, for operational control purposes, the requirement is for a reliable forecast of future events rather than a detailed understanding of the processes involved. For this reason, there has been growing interest in using techniques such as Kalman filtering as a means of improving forecasting procedures (Fig. 3). Amongst other things, these techniques include recursive parameter estimation so that discrepancies between the predicted and actual values are compensated as the forecasts are updated.

4.3 Optimal Control

Operational decisions in river-basin management are seldom simple and even more rarely optimal. At the present time, dynamic programming in one form or another is still perhaps the most commonly proposed decision mechanism for real-time use. However, in common with other optimal control techniques, the scale of the problem that can be considered is often curtailed by the limited computing facilities available.

The size of analtyical problem that can be realistically accommodated with the necessary level of detail is probably restricted to an individual subsystem such as a group of reservoirs or a series of river control weirs. To date, generalised control procedures have only been attempted for a limited number of subsystems and an entire river basin may comprise hundreds. Even if it were assumed that all subsystems were operated in an optimal fashion, obviously it does not necessarily follow that the entire system is optimally controlled.

Despite the fact that the research literature abounds with hypothetical examples of optimal control as applied to water resources, there is some doubt whether optimal control in the strict sense is justified, never mind possible, even for individ-

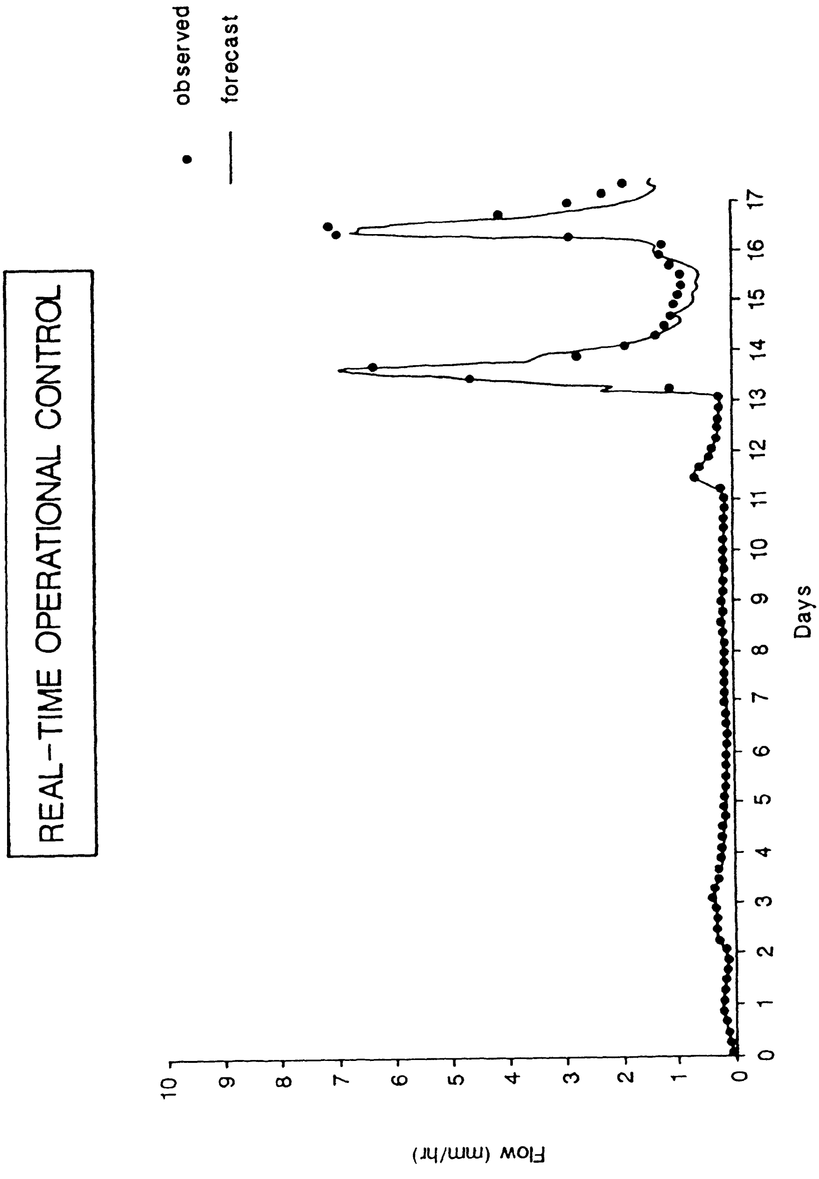


Figure 3. *Use of extended Kalman filter for forecasting (2-step ahead).*

ual subsystems. Looking to the future, perhaps emphasis will be placed on simpler control procedures which approximate the optimal decision to a degree where it makes little or no practical difference.

4.4 Hierarchical Control

Given that for the foreseeable future, anything approaching optimal control will be restricted to individual subsystems, there is an obvious problem in attempting to manage an entire river basin. However, if subsystem controls were capable of adapting to meet targets imposed by a more general control strategy, it is possible that this would provide an adequate approximation of optimal control for the entire system.

Three tiers of decision-making have been recognised namely strategic, tactical and local corresponding to the three levels of management, Headquarters, Divisions and Works. Whereas strategic decisions are confined to what is to be achieved, tactical control relates to how those targets are to be achieved. Thereafter, it is left to individual subsystems to implement the instructions received (Fig. 4). If for any reason, a Division or Works is unable to comply with its set-point, a feedback loop is activated and revised directives issued.

5 Implementation

5.1 Generalised Control Modules

Decomposing the system into inter-dependent subsystems clearly makes integrated river-basin management a more tractable proposition. Rather than repeatedly developing similar procedures for the same type of subsystem at different locations, the aim has been to formulate generalised computer packages which can be used throughout the region with minimal adaption. In the first instance, three such modules have been considered:

- **river management** (near-optimal control of abstractions, releases and impoundments);
- **water supply management** (near-optimal control of water treatment and distribution);
- **sewage disposal management** (near-optimal control of sewage treatment and effluent discharges).

The idea was that, initially, these procedures could be used for operating individual subsystems. If, however, they were to be developed within a common framework, as was the intention, this preserved the option of subsequently linking

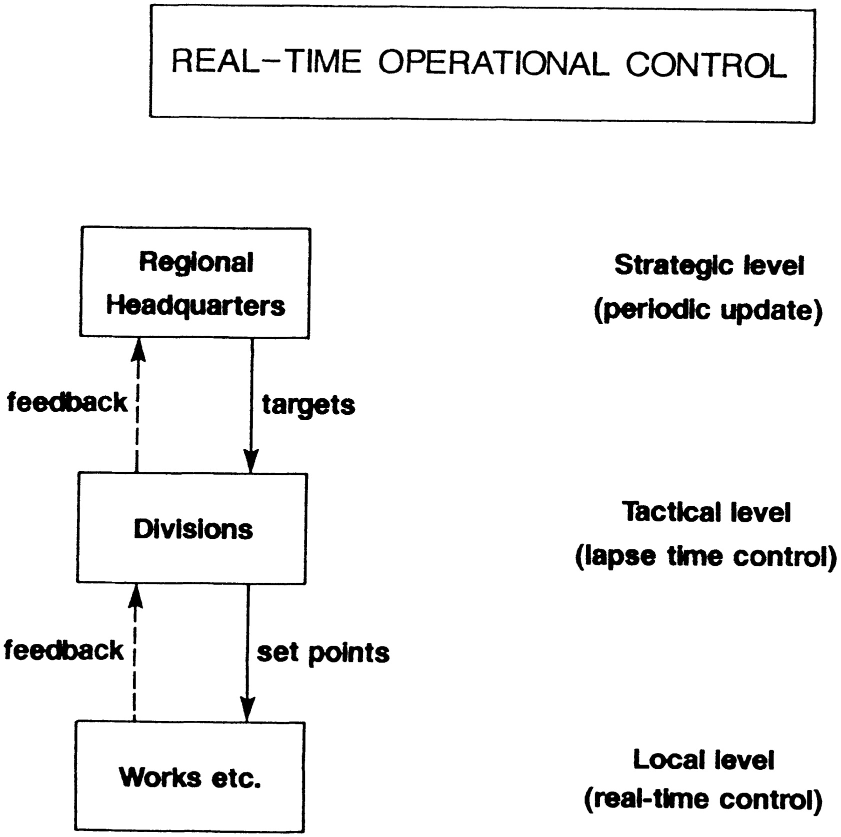

REAL-TIME OPERATIONAL CONTROL
Regional Headquarters
feedback
targets
Divisions
feedback
set points
Works etc.
Strategic level (periodic update)
Tactical level (lapse time control)
Local level (real-time control)

Figure 4. *Different levels of decision-making.*

appropriate modules to the strategic and tactical decision mechanisms, thereby introducing an element of integration.

5.2 User Interface

Rather than regard optimal control techniques and knowledge-based systems as rivals, the case is made for combining their different attributes to complement each other. By their very nature, optimal control techniques are not particularly user-friendly. Nor can uncertainty and other practicalities be easily incorporated into the algorithms. If, however, an expert system were used to interpret the results of the control algorithms, this would provide an enhanced level of understanding which could be readily assimilated by the manager.

5.3 Application

To illustrate the overall concept proposed with a hypothetical example, Fig. 5 depicts some typical elements of a multi-functional river basin. The system shown comprises a pair of pumped-storage reservoirs, an unconfined aquifer and three demand centres, each having water supply and sewage disposal responsibilities. Whereas two of the demand centres are reliant on a single source, the third can use the aquifer and river conjunctively. Besides water supply an sewage disposal, it is assumed the other activities include pollution control and flood warning to support the fisheries and navigation functions.

Figure 6 shows a schematic representation of the proposed control scheme corresponding to the hypothetical system given in Fig. 5. At a strategic level, the aim would be to set broad targets, allocating scarce resources between competing interests and minimising the degree of conflict between non-compatible interests. An attempt has been made to formulate this as a multi-functional, non-linear optimisation problem which is solved using a projected Lagrangian algorithm at a monthly time-step. At a tactical level, which can be multi-functional or single function depending upon the organisational structure, short-term control strategies are devised having regard to the targets set and the forecast state of the system. To that end, linear programming has been used to define the operating regime on an hour-by-hour basis over the next 24 hours. Thereafter, it is left to the local level to follow the set point by manipulating the control equipment. In that respect, dynamic programming has been used as a decision-aid.

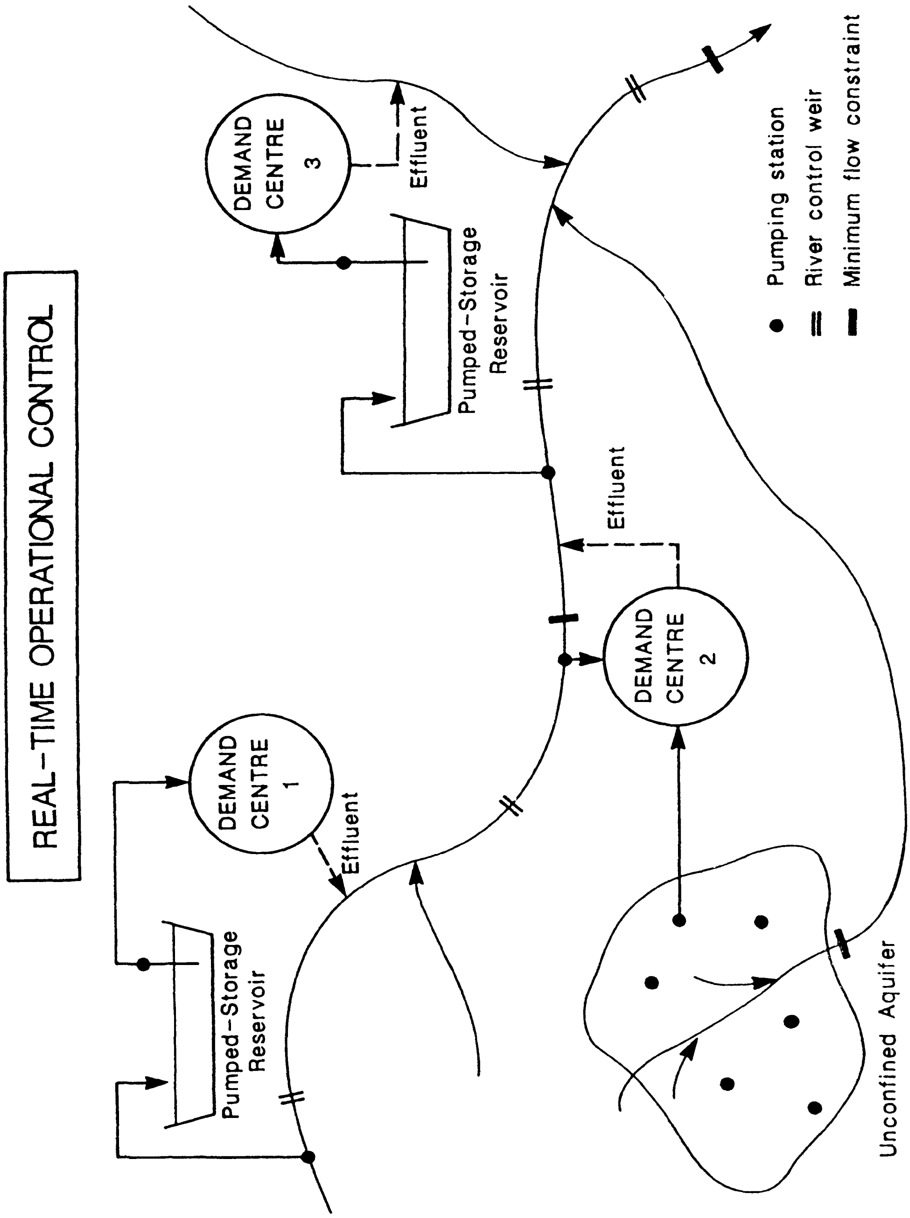


Figure 5. *Hypothetical multi-functional division.*

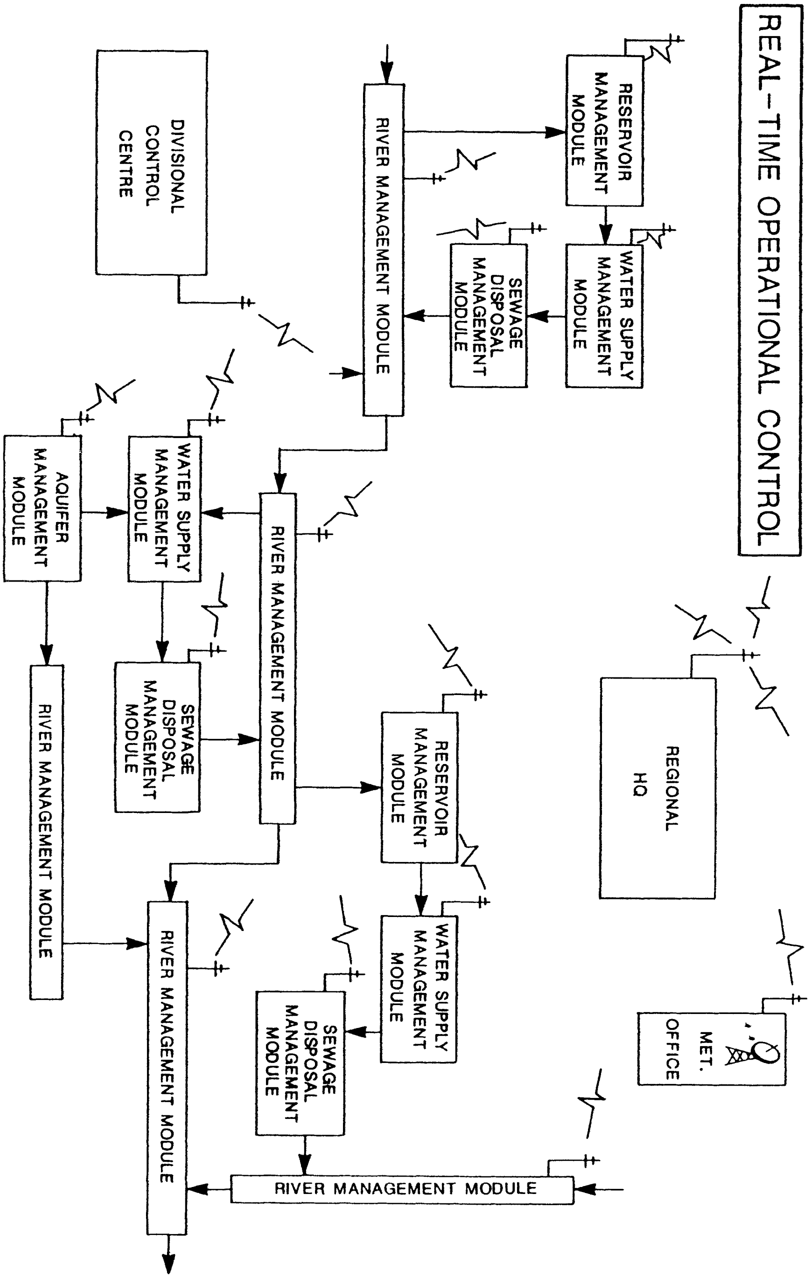

REAL–TIME OPERATIONAL CONTROL
REGIONAL HQ
MET. OFFICE
DIVISIONAL CONTROL CENTRE
RESERVOIR MANAGEMENT MODULE
WATER SUPPLY MANAGEMENT MODULE
SEWAGE DISPOSAL MANAGEMENT MODULE
RIVER MANAGEMENT MODULE
AQUIFER MANAGEMENT MODULE
WATER SUPPLY MANAGEMENT MODULE
SEWAGE DISPOSAL MANAGEMENT MODULE
RIVER MANAGEMENT MODULE
RIVER MANAGEMENT MODULE
RESERVOIR MANAGEMENT MODULE
WATER SUPPLY MANAGEMENT MODULE
SEWAGE DISPOSAL MANAGEMENT MODULE
RIVER MANAGEMENT MODULE
RIVER MANAGEMENT MODULE

Figure 6. *Corresponding divisional control system.*

6 Conclusion

6.1 Prospects

Up until recently, most control schemes within the water industry could at best be described in euphemistic terms as fragmented. There was little or no consistency of approach, let alone compatibility of hardware, software or even communication protocols. Telemetry was simply used for data acquisition rather than an integral part of a control system. All of the decision-making and much of the control was manual. However, that is changing rapidly. The need for improvements to efficient and effectiveness coupled with the availability of competitively-priced control equipment will inevitably lead to higher levels of automation.

6.2 Epilogue

These proposals may seem ambitious at first sight. However, one should anticipate that sophisticated control systems will be commonplace by the turn of the century. The point in question is not whether the industry will introduce improved control procedures, but how. The choice seems to be either drifting into automation in a piecemeal fashion or making a conscious effort to introduce an overall control strategy. This paper advocates the latter and outlines how it might be achieved.

INLET: Access to Water Resources Management Data Through a Natural Language Interface

Richard N. Palmer and Lynn R. Spence
Department of Civil Engineering
University of Washington
Seattle, Washington
United States of America

Abstract This paper describes the development of INLET, an Interactive Natural Language EnvironmenT, that allows convenient, rapid, and extensive access to water resource management data. This system is designed to require a minimum of training, making it immediately useful for water managers who do not have the time or interest to be trained in the use of traditional databases. INLET consists of a natural language processor that accepts commands posed to it in ordinary (conversational) English and a menu-driven query system. By using simple English commands, water resource managers and staff who are not familiar with either a formal database query system or computer programming can easily access hydrologic and management data and can use the analytical, statistical, and graphical capabilities provided by INLET.

This paper discusses the use of natural language interfaces and the role they play in improving the use of computer models. Next, the INLET system is described along with the water resources database. The paper concludes with a summary of INLET's use and potential improvements.

1 Water Resources Planning Models

One of the largest obstacles in using computer models as decision-making tools in the practice of water resource management is the lack of confidence and understanding managers have in the models. The increasing availability of mini and micro-computers and interactive software has made involvement of managers in all phases of model specification, development, and verification more feasible in recent years (Fedra and Loucks, 1985). A key feature of these decision-aiding systems is the direct involvement of policy analysts in an interactive policy-making process. An important step towards direct involvement of policy makers is the use of highly

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 205–215.
© 1992 *Kluwer Academic Publishers.*

user-friendly interfaces to models and data.

User-friendliness can be defined as characteristics of a computer or of software that allow them to be used without the knowledge of any classical programming languages. Menu-driven programs are an example of one type of user-friendliness. User-friendliness also applies to more general aspects of models such as semantic and syntactic consistency (which ensures that the model captures both the intended meaning and syntax) graceful (and instructive) recovery from failures, and a wide assortment of input-output devices. A user-friendly interface requires the underlying software to be easily understood, well structured, and compatible with the mental processes of the users (Loucks *et al.*, 1985; Hendler and Lewis, 1988).

Artificial intelligence (AI) tools are becoming increasingly popular for developing user-friendly interfaces compatible with the user's cognitive process. Two of the most common AI tools meeting this need are expert systems and natural language interfaces. The topic of this paper is the development of a natural language interface applied to a water resources database.

2 Natural Language Processing

Natural language processing can be defined as the ability of a computer to process language that humans use in ordinary discourse (such as English). A primary goal in natural language processing is to translate a potentially ambiguous input phrase into a precise form that can be directly interpreted by a computer system. This translation process, called parsing, is performed in many ways. Obermeier (1987) has classified the types of parsers that have evolved into five groups: grammar-based, semantic, pattern-matching, knowledge-based, and neural-network parsers. These groups are defined by the approach taken when parsing a natural language.

Parsers may analyze syntax or semantics or both. Syntax refers to the rules governing the order of the symbols. Semantics, on the other hand refers to the intended meaning of the expression. Computers can easily interpret syntax, but are poor at resolving semantics. Standard language has a prescribed, although sometimes variable, syntax defined by rules.

3 Grammar Parsers

Grammar-based parsers are concerned primarily with the syntax of the sentence, that is, the order in which the words appear and their grammatical definition. Grammar-based parsers use a set of rules that describe the types of sentences acceptable for that particular language. For example, two simple rules are:

$$S \rightarrow NP + VP$$
$$S \rightarrow VP$$

where S is the symbol for sentence, NP is the symbol for noun phrase, VP is the symbol for verb phrase, and → is the symbol for 'is defined as.' These rules, called rewrite rules or production rules, define a sentence to have a noun phrase and a verb phrase, or just a verb phrase by itself. These noun and verb phrases are themselves composed of smaller phrases. These phrases can, in turn, be decomposed until only the essential building blocks of grammar remain: individual words. Grammar-based parsers use all the words found in a sentence and also consider the phrasing in which the words appeared. They are good for generating natural language text and for determining sentence structure. For natural language systems used as interfaces to databases or expert systems, however, the semantics (or meaning) of the sentence, also must be considered in order to correctly perform the request.

4 Semantic Parsers

Semantic parsers attempt to find the meaning of the sentence, rather than just concentrate on syntax like the grammar-based approach. In semantic parsers, the rewrite rules are stated in terms of 'semantic classes' describing the meaning of the word, rather than word classes (ie. verb or noun) like the grammar-based parsers. For example a sentence could be represented by the following: <SENTENCE> ::= <PERSON> is eating <FOOD>. In this example, <PERSON> and <FOOD> are semantic classes for which words like 'Carla' and 'pasta' can be substituted to create a valid sentence. An advantage to semantic parsers is that the size of these semantic classes is generally much smaller than the size of an equivalent word class, resulting in a much more efficient parsing strategy. A disadvantage of using semantic grammar is that it is not easily transferrable from one domain to another.

5 Pattern-matching Parsers

Pattern-matching parsers include some of the earliest parsers developed. They look for a linguistic pattern in a sentence without using explicit grammatical rules. When processing a sentence, this type of parser attempts to match the input with a fixed number of patterns. If a match is found, the system performs a specified action. Pattern-matching parsers are popular as interfaces to databases because database commands can usually be decomposed into patterns and keywords.

Green *et al.* (1961) developed one of the first natural language interfaces to a database. His system, BASEBALL, provided access to information about all baseball games played in one season. Lane (1987) developed a natural language interface to DOS using Prolog. His system used a noise-disposal parser in which non-key words were ignored. The list of keywords were then matched with possible command patterns to determine the meaning of the input.

The primary advantages of a natural language interfaces (NLIs) over a menu system are (Hayes, 1986):

1. Natural language interfaces are typically more versatile, they can answer a wider range of questions.

2. Natural language interfaces can be more direct. To access complicated data with the menu system would require selecting many menus.

3. Natural language interfaces take less time for the user in many complex situations.

4. Natural language interfaces allow users to formulate questions in a manner that is consistent with the way they think about the problem.

Some disadvantages of natural language interfaces are that they may be very frustrating if help is not provided and the user is unfamiliar with the system domain.

6 INLET, A Natural Language Environment

INLET, the topic of this paper, uses an approach similar to Green *et al.* (1961) and Lane (1987) and is applied to a water resources database. The database contains streamflow data and optimal reservoir operating policies generated from a drought management model developed for the Seattle Water Department (Palmer and Holmes, 1988; Palmer and Tull, 1987). INLET uses a noise-disposal parser to find keywords, match them with a command pattern and then perform the command. A noise-disposal parser requires a strict sentence format (an ordering of the keywords), but it will accept a wide variety of sentences as long as the necessary keywords are present (Schildt, 1987).

The Seattle Water Department (SWD) provides direct service to 541,000 Seattle residents and is a wholesaler to 549,000 residents of King County (SWD, 1986). Water is taken from two major sources, the Tolt and Cedar Rivers, both of which originate in the Western Cascade Mountains. In recent years there has been a growing concern that the demand for water in this region has approached the safe yield of the supply system. This concern has, in turn, lead to increased interest in the proper management of these resources and the development of operating strategies for the system during droughts. To guarantee an orderly response to water shortage, the SWD developed the Water Shortage Response Plan (WSRP). This plan addresses problems related to the 1-in-50 year drought event (SWD, 1986). The objective of this plan is to maintain essential services while minimizing the net economic loss during a drought event.

INLET has two components: a natural language interface and a menu-driven interface. Both components provide direct access to the data and provide statistical and plotting capabilities. Both of these components are easily accessible from the other. Two types of data have been placed in INLET: hydrologic data and system management data. Monthly streamflow data for seven sites in the Seattle watershed provide historic hydrologic information. A database of system yield and management information is also included. System yield and potential economic losses have been calculated for a wide variety of potential conditions. These conditions include five monthly starting periods, nine storage levels, and 47 different hydrologic records (years 1929–1976). These two thousand scenarios were evaluated in a linear program. In addition, the optimal operating policy chosen from the WSRP for a sixteen week period was also calculated using the linear program for some 400 specific low-flow periods. All of this information is available with INLET.

INLET is written in Prolog. Prolog differs greatly from standard engineering computer languages such as Fortran or Pascal. Programming languages can be described as either procedural or descriptive. In procedural languages such as Fortran, program execution is sequential; it follows the order of the commands found in the source code. Prolog is a descriptive or 'data-driven' language. A Prolog program is essentially a database with a series of rules for analyzing the data. The control of execution is determined by the specific data in the database and the rules, not by a fixed algorithm. This allows the execution procedure to be dynamic; the procedure will vary depending on the data contained in the database.

Prolog supports recursive functions. Recursion is the process of a function calling itself or executing itself. Prolog also supports symbolic processing. Symbolic processing allows a problem to be solved by strategies and heuristics for manipulating symbols rather than using defined numeric algorithms.

7 INLET's Natural Language Processor

INLET's natural language processor reads an input sentence and translates it into a command the computer can execute. It is also responsible for answering the query in a full sentence, providing it is not a plot. INLET uses a noise-disposal parser to scan the input sentence, evaluate keywords and dispose the non-essential words (noise). The keywords are then matched to a pattern and the command is performed.

INLET recognizes words from the following keyword groups:

<command> <statistic> <site> <month 1>
<month 2> <year 1> <year 2>

The words in the brackets are the names of the word classes. The brackets indicate that the keyword inside is optional. The keywords can appear in the sentence in any order. Some example sentences that can be answered are:

What is the mean flow at Cedar 1 for June from 1960 to 1968?
What is the standard deviation?
List all the flows at the main stem of the Tolt for 1972.
Plot the site 15 flows for May all years.
At site 7, what is the lowest flow for all the years on record?
Sum the flows from June to September for 1969.
Given reservoir storage at 30%, what is the system yield starting with May 1?
What is the economic loss?

The command keywords that INLET recognizes are: what, plot, show, help, list, and storage. If no command keyword is found and context is unable to be determined, a default value of 'what' is used. The statistics available are: mean, standard deviation, skew, cdf, lowest, highest, all, sum economic loss and yield. Many of these keywords have synonyms which are also recognized by INLET. Not all the keywords need be present in order for the sentence to be processed. When some keywords are missing, the processor either uses the default values, or assumes that the question was asked in the same context as the previous question. In this later case, keywords from the previous sentence are used for the missing ones.

The Prolog code for the natural language processor consists of three main components: (1) clauses for parsing the sentence, (2) clauses defining the lexicon (dictionary) containing the words recognized by INLET and their associated synonyms, and (3) clauses used to determine the command for that particular grouping of keywords and the context of the sentence.

The first group of clauses contains all the clauses associated with reading the sentence and selecting the words recognized by the system. This is accomplished by reading each word, one by one, and evaluating the clauses contained in the lexicon. If the word is not one contained in INLET's dictionary, it is ignored and the parser moves to the next word. If it is recognized, the parser uses the lexicon to determine the word class to which it belongs. It then evaluates the keyword by examining the synonym clauses in the lexicon. For example, the word 'average' is a synonym for 'mean.' Both of these words are recognized by INLET and belong to the word class 'statistic,' but 'mean' is the actual keyword used by the system. If the parser encounters 'average,' it finds that it is a statistic and the associated keyword is 'mean.' 'Mean' is then passed to the clause that actually executes the command.

Some keywords are not single words, for example 'Site 1.' In this case, the actual word recognized by INLET is 'site.' When the system encounters 'site,' however, it expects to find a number after 'site.' INLET then reads the next word to see if it is a number. If so, this number is used to determine the keyword for that particular site.

The third group of clauses translate the list of keywords into the command. One of the rules used is:

If the *command* is **plot**,
and the *statistic* is **all**,
and the *site* is known,
and the *month1* is known,
and *month2* is **none**,
and *year1* and *year2* are both **none**,
then plot all the flows for *month1* at *site*.

In this example the variables are italicized and the values of the known keywords are in bold. There are many clauses in this group for processing all the types of commands that INLET performs.

There are also clauses that determine what to do when not all of the keywords are present. For example, INLET does not require the user to specify the site name each time. Once a site has been selected it remains active until another site is mentioned. If a site is not mentioned in the input sentence, INLET assumes that the previous site is the current one. INLET also determines the meaning of the sentence in context with the previous sentence.

To illustrate this process, suppose the following two commands were given:

'Plot the May flows at the North Fork of the Tolt.'
'Plot them at the South Fork.'

In the second sentence, the user most likely intended for the May flows at the South Fork of the Tolt to be plotted. The second sentence is within the context of the previous sentence. The only words recognized by INLET in the second sentence, however, are 'plot' and 'South Fork.' This is an ambiguous command for INLET because no time period or month is specified. In this case, INLET refers back to the keywords used in the previous sentence to find the right keywords to use, i.e. 'may.'

Context determination is made possible by storing the keywords from the previous sentence in Turbo Prolog's internal database. An internal database is a collection of clauses that can be added to, or retracted from while a program is running. INLET creates an internal database containing a 'context' clause. If insufficient keywords are present for INLET to determine the meaning of the sentence, the context clauses are evaluated to determine the values of the missing keyword(s). After all required keywords are assigned values, the previous context values are retracted and the new context is asserted into the database. For example, if a keyword for site name is not found in the current sentence, the program searches the context clause to find the keyword from the previous sentence and then asserts the site name to be equal to the previous one.

Context dependency is an essential feature of INLET that is often not available in simple natural language processors. This feature is extremely valuable for increasing the user friendliness of the program and decreasing the number of words required to express a command.

INLET also has a menu-driven system which can serve as a review of the options available for queries. This menu system is accessed from the natural language processor by entering <ESC>. The menu system offers the same statistical and plotting capabilities as the natural language processor. The optimal reservoir operation data, however, are not available through the menu system. These more complicated questions would require a large number of menus and are easily accessible from the natural language processor.

When INLET is activated, the user initially is provided a menu listing three options. Those options allow access to the natural language system, the menu system, or to DOS. Although the features available with either the menu system or the natural language system are similar, their use is very different. Menu-driven systems allow a user to choose one or more items from a list of items. The selection of the items typically is accomplished either with a mouse or by moving a cursor. When commands are hierarchical, (that is, several steps are required), numerous menus must be evaluated to complete a command.

For instance, suppose one wishes to plot all of the data at a particular streamflow gauging site using the menu system. The user first indicates the site location, next that a plot is desired, next, the time period for which the data should be included, and finally, the data type (all data, or just data for one month, etc.). In this process the user is required to select from four menus.

The natural language interface provides the user with a completely different approach to this process. The user types the command such as, 'Plot the June flows at Site 1 from 1950 to 1960.' All of the information needed is contained in the single command. Unlike the menu approach, the user gives the command to the computer just as he might naturally state the command.

8 Typical INLET Session

Consider a situation in which a water manager finds himself in a particular month, with low streamflows, and the reservoir system at thirty percent of its capacity. The manager is interested in a variety of information including:

1. How unusual are the current streamflows?

2. How likely are the low flows to continue?

3. What is the probability of flows being as low as the ones currently experienced?

4. Should water use restrictions be initiated?

5. When should restrictions be initiated and how stringent should the restrictions be?

It is possible that a wide range of other questions may occur to the manager. It is also likely that the order in which the manager wishes questions to be answered will vary from one session to another. The setting that is suggested here is a common one. A somewhat unusual event (low streamflows) has occurred and the manager wishes to place this situation into a context that will help him develop a reasonable response plan.

The purpose of INLET is to allow the manager to pose these questions in any order that he cares to and to analyze the situation to the extent necessary. INLET accomplishes this by providing the wide range of statistical analysis tools previously described, access to the operational database, and total flexibility in the extent and order in which the analysis is made.

After reviewing recent streamflows and the current storage levels of his system, the water manager's first step is to find the mean and standard deviation of the June flows. He decides to use the menu system first. From this menu, the user chooses the menu-driven system. Next, a menu appears containing the statistics, plotting and site selection choices. From this menu, he chooses site selection and the site menu appears. After choosing a site, the site menu automatically disappears and the user is returned to the second menu from which he selects 'Statistics.' The statistics menu appears providing statistical options: mean, standard deviation and skew. After choosing mean, a menu appears with the time period options: month, year, or specify time period. He chooses each option in turn to find the mean of all the June streamflows on record, the mean of 1972, and the June mean from 1950 to 1960. The answers to these selections are displayed in a window covering the top half of the screen. To find the standard deviation, the user would enter <ESC> until he is back at the statistics window. From there he would choose 'Standard Deviation' and repeat a similar process to that of specifying the time periods when he determined the various means.

Having evaluated the basic statistics, the user now wants to view two plots, a time series plot of the June flows and the cumulative distribution function of those flows. He returns to the second menu. From that menu, he chooses 'Plotting.' A system of menus appears from which he selects the time period to plot. He specifies that he wants a time series plot of monthly streamflow data for all years on record for the month of June. From this figure the manager recognizes immediately the high variability of the June streamflows.

Next, the manager wants to view a cumulative distribution function (CDF) of the June streamflows. He presses the escape key until the menu with 'Cumulative Distribution Function' appears. After choosing this option, a similar set of menus appears to that of the time series plot, and he specifies the month and desired time period. From the CDF, the user can compare the current streamflow to a ranking of all the streamflows on record.

By pressing any key, the user returns to the second menu. He decides to plot all the streamflows for the year 1970. He chooses the 'Plotting' option from the second menu and year from the next menu that appears. He then specifies '1970' and hits return.

The user now decides that he wants to use the natural language system and ask some similar questions for comparison with the menu-driven system. He enters the escape key until he is at the main menu where he chooses the first option, 'Natural Language Interface.' In the query window the user enters his questions about the mean and standard deviation for the June flows:

```
What is the mean flow for June at site 1?
What about site 5?
Between 1950 and 1960?
What is the standard deviation for this period?
What is the lowest flow in June?
Plot all the streamflows on the North Fork Tolt between 1950 and 1970.
Plot the cdf.
```

It should be noted that the questions are posed in simple English statements. The responses by INLET provide a complete reply and answer to the questions. The last two requests generate plots. The user returns to the natural language system by entering any key.

After carefully reviewing the flow data, the user decides he now wants to analyze some reservoir operation policies. From his previous analysis of streamflow data, he determines that the current flow conditions are similar to those experienced in four different historic sequences, 1930, 1938, 1954, and 1959. To investigate the optimal operating pattern associated with those time periods, the user wishes to review the appropriate data. To do so, the user enters 'Please show the operating policies for June, 1930, with initial storage of 30%.' This request generates a bar chart showing the optimal restriction levels for that year. After reviewing the restriction policies for 1930, the user can review those of 1939, 1954, and 1959.

9 Results and Conclusions

The evolution toward more user-friendly software will allow managers to participate effectively in model development and use. This is important because it increases communication between model developers and managers. With increased communication, models can be developed that will more directly serve their users.

Prolog has been demonstrated to be an ideal language to develop user-friendly software. It makes natural language processing possible in an efficient and fast manner. This suggests that more engineers should become familiar with non-procedural programming languages.

INLET allows novice computer users access to complex data and provides very useful statistical and plotting capabilities. This has significant implications on the use of such models by water resource managers. These managers are now able to explore a large number of operational policies and their impacts on system reliability with ease.

References

Fedra, K. and Loucks, D. P., (1985), 'Interactive Computer Technology for Planning and Policy Modeling,' *Water Resources Research* **21**, 2, 114–122 (February).

Green, B., Wolf, A., Chomsky, C. and Laughery K., (1961), 'BASEBALL: An Automatic Question Answerer,' *Proceedings of the Western Joint Computer Conference 19*, pp. 219–224.

Hendler, J. and Lewis, C., (1988), 'Introduction: Designing Interfaces for Expert Systems,' in *Expert Systems: Designing the User Interface*, J. Hendler (Ed.), Ablex Publishing, Norwood, NJ, pp. 1–13.

Lane, A., (1987), 'DOS in English,' *BYTE*, December, 1987, pp. 261–263.

Loucks, D. P., Kindler, J. and Fedra, K., (1985), 'Interactive Water Resources Modeling and Model Use: An Overview,' *Water Resources Research* **21**, 2, 95–102, (February).

Obermeier, K. K., (1987), 'Natural-Language Processing,' *BYTE*, December, pp. 225–233.

Palmer, R. N. and Holmes, K. J., (1988), 'Operational Guidance During Droughts: Expert System Approach,' *Journal of Water Resources Planning and Management* **114**, 6, 647–666, (November).

Palmer, R. N. and Tull, R. M., (1987), 'Expert System for Drought Management Planning,' *Journal of Computing in Civil Engineering* **1**, 4, 284–297, (October).

Schildt, H., (1987), *Advanced Turbo Prolog*, McGraw-Hill Inc., Berkeley, CA.

Seattle Water Department, (1986), *Seattle Comprehensive Regional Water Plan, 1985 COMPLAN*, Vol. 6, Seattle Water Department, Seattle, WA.

Learning from Optimal Solutions to Design Problems [1]

John S. Gero, Conrad A. Mackenzie and
Sally McLaughlin
The University of Sydney
NSW Australia

Abstract Designs can be described by the morphism between two descriptor sets: decisions and performances. Characterised designs are those in which both decisions and their consequent performances are articulated. Pareto optimization is discussed as a means of structuring performances and decisions. The induction algorithm ID3 is presented as a means of abstracting general relationships from characterised designs. An example from the domain of building design is presented.

1 Introduction

Design is characterised by decisions which generate solutions that are best in some sense. If it is impossible to achieve the best in all design objectives, then the solution should exhibit a satisfactory compromise. This introduces the notion of design optimality and its use as a means of structuring design information so that one can learn about design decision making. This paper is concerned with extracting knowledge about decision/performance relationships from hypothetical or exisiting designs by structuring design data through optimization. It is assumed that the choice between design decisions must be made on the merits of their consequences. The problems which are addressed are usually complex in the dimensional sense (which makes them difficult to represent and solve) because of the many constituent criteria and constraints that have influence over the solution. It must be noted that for problems that involve conflicting objectives, a single optimal solution is unlikely to exist. However, a set of solutions can be identified that are all optimal

[1] This lecture draws directly from the following papers: Mackenzie, C. A. and Gero, J. S., (1987). 'Learning design rules from decisions and performances,' *Artificial Intelligence in Engineering* **2**, 1, 2–10, and McLaughlin, S. and Gero, J. S., (1987), 'Acquiring expert knowledge from characterised designs,' *AIEDAM* **1**, 2, 73–87.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 217–249.
© 1992 *Kluwer Academic Publishers.*

in some sense, and external knowledge must be used to choose between the optimal solutions.

Central to the argument of this paper is the need for a strong model of a structuring hypothesis. This can be achieved by structuring the observations through a known process. Indeed, the field of machine learning has been divided into four categorical tasks (Langley and Carbonell, 1984), all of which structure observations in a particular fashion. In all cases, the properties of the imposed structure are known beforehand, and the systems are designed to recognise and exploit these properties.

A system is presented that extracts decision/performance knowledge from multicriteria problems in the context of an ultimate goal state: a solution that exhibits desirable characteristics in every criteria. The structuring of the data to facilitate knowledge extraction is discussed. The properties of the imposed structure which carry design knowledge are investigated and the knowledge is made explicit in rule form. The methodology is demonstrated with two examples from the domain of building design. Inherent difficulties with the derived knowledge are discussed and the articulation of the derived knowledge as an investigative design tool are explored in the last sections.

2 Methodology

A design attempts to satisfy a goal state through a solution which exhibits the most desirable characteristics achievable in all criteria by which the goal state is judged. If these criteria are quantifiable then they have meaning when maximized or minimized. Multicriteria optimization is one means of identifying a set of design solutions among which a best solution for any group of evaluable goals must lie. However, if the objectives of the design solution conflict, it may not be possible to achieve the best performances in all criteria simultaneously. Conflicting objectives force a tradeoff in the optimal solution between criteria performances. Thus, it may not be possible to identify a single best solution but a set of solutions that are all optimal in some sense.

Since the performances are consequents of the decisions, it is possible to define a decision space and a performance space between which a mapping exists. The 2-dimensional case is illustrated in Fig. 1. A point in the decision space is characterised by a vector of design variable values; a point in the criteria space is characterised by a vector of quantifiable criteria performances. A design decision is the selection of values for the design variables. The mapping from the decision space to the criteria space characterises the performance consequences of design decisions and is not necessarily a one to one mapping.

The most desirable decisions are those that exhibit the best criteria performances. If the best performances in the criteria space are known then it is possible

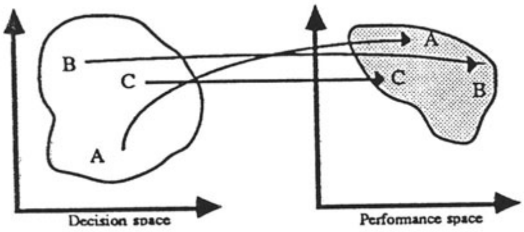


Figure 1. *The decision and performance spaces for two variables. The mappings from the decision space to the criteria space indicate the performance consequences of the decisions.*

to follow the mapping from the criteria space back to the decision space to identify the best design decisions. This structures the domain of interest into a general schema; a decision space, a performance space and a mapping that connects the two spaces.

2.1 Pareto Optimization

A set of measurable criteria is Pareto optimal if no other feasible solution exists which yields an improvement in one criterion without causing a decrease in at least one other criterion (Cohon, 1978). Fig. 2 shows the Pareto optimal set for the performance space of Fig. 1. Pareto optimization has been employed extensively in design related fields (Balachandran and Gero, 1986; Gero, 1985a, 1985b; Gero and Balachandran, 1986; Radford, Hung and Gero, 1984; Radford *et al.*, 1985).

In the following illustrations, the Pareto optimal set is denoted by a heavy line. The axes of the 2-dimensional space have been aligned so the criteria performances improve with distance from the origin. Pareto optimization is effective in a space of any dimensions; the examples below represent the 2-dimensional case.

Pareto optimization establishes a structure in the performance space that exhibits known characteristics. By connecting optimal points in the performance space to points in the decision space via the mappings it is possible to extract knowledge about decision/performance relationships. This process is hypothesis driven; the hypothesis is the imposed structure and deductive learning occurs when specializing the hypothesis to match the domain in question. The specialization process involves matching known characteristics in abstract Pareto optimal sets to characteristics observed in the domain specific Pareto optimal set.

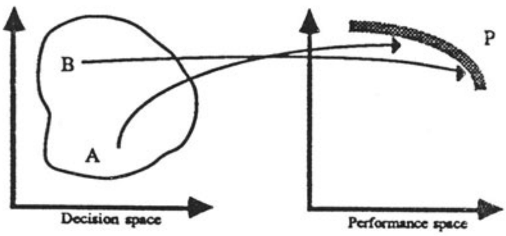


Figure 2. *Mappings from the decision space to the Pareto optimal set in the criteria space. The Pareto optimal set is marked 'P.'*

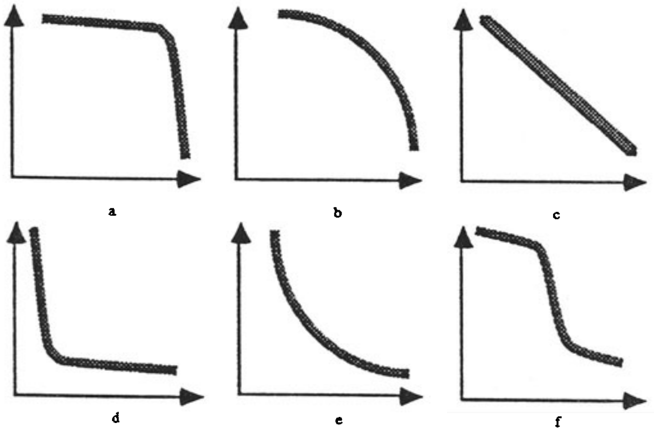


Figure 3. *Pareto optimal sets which exhibit the basic knowledge carrying geometrical features.*

3 Knowledge in Pareto Optimal Sets

Each Pareto optimal set in Fig. 3 exhibits a distinctive geometrical feature. For example, the single dominant feature in Fig. 3a is a curve that exhibits a pronounced convexity, while Fig. 3f has two features, a concave and convex curvature. For simplicity, multiple features are analysed independently of their context.

3.1 Convex Pareto Optimal Sets

If the Pareto optimal set exhibits a pronounced convex curvature (Fig. 3a), it is possible to obtain good tradeoffs between all criteria where all criteria have been optimized. Such points exist at the elbow of the curve. Small perturbations near the elbow in either criteria will not cause much change in the other criterion's performance. The decisions which correspond to the elbow points are the most desirable. Within that set of decisions, all solution performances are high, which introduces the concept of performance stability.

If it is possible to make several different decisions which all yield similar performances in all criteria, then the performance is said to be stable for those decisions. This can be stated in production rule form:

If the Pareto optimal set exhibits a pronounced convex curvature

then the performances in all criteria at the elbow will be very good choices

and most decisions at the elbow represent good solutions

and the performance is stable.

A Pareto optimal set which has a small convexity (Fig. 3b) will exhibit less stability since the tradeoffs between criteria are more pronounced.

3.2 Concave Pareto Optimal Sets

If the Pareto optimal set is concave (Fig. 3d, 3e), then it is harder to find a good solution. In the extreme case the set exhibits a pronounced concave curvature. A small shift in performance along either axis will result in a large decrease in one performance accompanied by a large increase in the other performance. This causes performance instability, its magnitude being dependent on the degree of concavity of the Pareto optimal set.

In such a situation, the solution most likely to be chosen will be at the extreme ends of the Pareto optimal set. In other words, poor performance at the point of equal compromise between criteria will probably result in abandoning the consideration of one criterion in favour of optimizing another. This is restated in the rule below.

If the Pareto optimal set exhibits a pronounced concave curvature

then good performances in any criteria lie at the extreme ends of the Pareto optimal set

and decisions at the extreme ends of the Pareto optimal set are fair solutions.

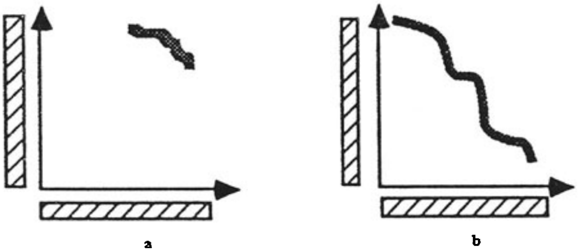


Figure 4. *'Close' and 'Extended' Pareto optimal sets. The set in 5a is close with respect to the sensible ranges, while the set in 5b is extended. The rectangles parallel to the axes represent the sensible ranges of the criterion on that axis.*

3.3 Planar Pareto Optimal Sets

A planar Pareto optimal set (Fig. 3c) indicates that all performance tradeoffs between criteria are linear so there is constant performance stability over the whole set. No solution can be said to be better than any other with respect to the overall performance.

If the Pareto optimal set is planar
then the performance tradeoffs between all criteria are linear
and further information is needed before a solution can be found.

3.4 Close and Extended Pareto Optimal Sets

The sensible range for a criterion is the range over which the criterion is determined to be meaningful. Such background knowledge is useful in identifying the set of solutions that are feasible for a specific problem. Comparison of a criterion value with its sensible range will indicate if the performance is acceptable or not. If a sensible range is given for each criterion, a value can be assigned to the range covered by the Pareto optimal set.

If the Pareto optimal set covers only a small range of the sensible values, then the Pareto optimal set is defined to be close (Fig. 4a). A close Pareto optimal set exhibits performances which are all similar. An extended Pareto optimal set covers most or all of the sensible values (Fig. 4b). A close Pareto optimal set indicates that the performances of criteria will be largely unaffected by decision selection.

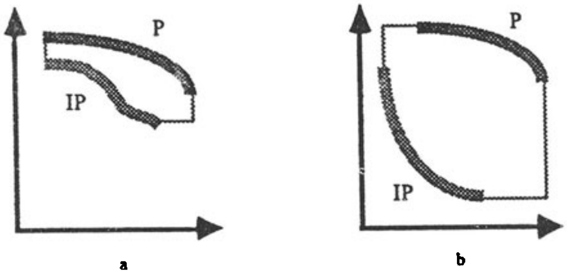


Figure 5. *Two Pareto optimal sets and their inverses.*

If the Pareto optimal set is very close
then all performances are similar
and any decision results in a good solution.

If a subset of the criteria covers only a small section of their sensible ranges, then all the performances in these criteria will be similar. The set of decisions which map onto these points in the criteria space will all exhibit similar performances. Therefore, it does not matter which of these decisions is chosen. Thus, the solution is independent of the set of decisions resulting in the close criteria. This information allows a dimensional simplification of the problem.

If a set of criterion performances is very close
then these performances are unaffected by the decision choices in the decision space
and the criterion can be removed from further consideration.

One problem with these rules is that they carry no knowledge of the sensitivity of performances. To do so, a lower bound on the performances is required. This is discussed below.

3.5 Inverse Pareto Optimal Sets

The inverse Pareto optimal set is formed by reversing the sense of optimization of each criterion and provides a lower bound on the set of performances for each criterion. The magnitude of the space between the Pareto optimal set and its inverse articulates knowledge about performance sensitivity. This space is bounded in Fig. 5 by P, IP and the dotted lines.

The bounded space represents the set of all possible performances. The magnitude of the space can be measured in relation to the sensible ranges of each criteria.

If the space is small, as in Fig. 5a, then the range of performances is also small with low sensitivity. If the bounded space is large, the range of performances and the sensitivity of the criteria may be large.

If the bounded space is very small

then the range of performances is narrow

and the performances of the criteria are insensitive to decisions in the decision space.

If the bounded space is very large

then the range of performances is large

and the performances of the criteria may be very sensitive to decisions in the decision space.

If any 1-dimensional projection of the bounded space is close with respect to the sensible range then that criterion can be removed as described in the rules below.

If the projection of the bounded space onto a criterion axis is close

then the performance sensitivity of that criterion is small

and the problem can be reduced by ignoring that criterion.

If The projection of the bounded space onto a criterion axis is extended

then the performance sensitivity of that criterion may be large

and the criteria corresponding to that axis is significant.

A relative ordering of criteria sensitivity to decisions is determined by the ratio of the ranges of the criteria with respect to their sensible ranges. This enables the identification of the least sensitive criteria that can be given less consideration if a compromise is necessary.

If the coverage of a sensible range by the projection of criterion A is small compared to the coverage of another sensible range by the projection of criterion B

then criterion A is less sensitive to decisions in the decision space than B

and A can be given less consideration.

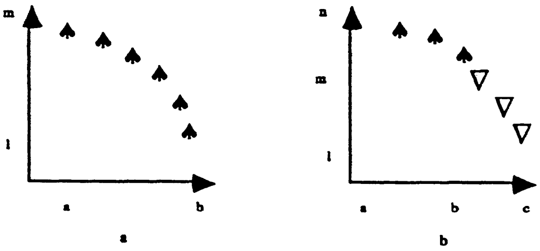


Figure 6. *Decision clustering in the Pareto optimal set. Note that the ♠ and ∇ symbols represent the common design decision value shared by the points in the criteria space.*

3.6 Decision Clusters in Pareto Optimal Sets

Decision clusters occur in Pareto optimal sets when the performance points locally correspond to the same decision. This can be discovered by following the mappings from the performance space back to the decision space. A decision cluster may span the entire Pareto optimal set or cover only a very small part of it. The sensitivity of a decision can be found by observing the position of the matching decision cluster in relation to the shape of the Pareto optimal set. If a decision cluster is found at the elbow of a convex set, then that decision is deduced as stable. The relative importances of each decision are found by observing the relative sizes and density of each decision cluster, while the performance ranges over which the decision is important are determined by the performance ranges that are covered by the decision cluster. The latter enables the extraction of explicit quantitative knowledge about decision/performance relationships. In Fig. 6a, the decision is important in the range [a, b], [l, m]; in Fig. 6b is important over the range [n, m], [a, b] and is important over [b, c], [l, m].

Explicit quantitative knowledge concerning decision cluster importance can be generalized into the form of the rule below.

If the Pareto optimal set has recognisable clusters of decision variables A, B, C ... over ranges AR, BR, CR ...

then decision variable A is important in range AR

and decision variable B is important in range BR

and decision variable C is important in range CR

and ...

4 Induction Algorithm: ID3

The induction algorithm used is ID3 (Iterative Dichotomizer 3.2) written by Quinlan (1983). Given a set of examples and counterexamples of a concept, ID3 provides the machinery for developing a rule which discriminates between these examples and counter examples. Each exemplary object must be described in terms of a fixed set of attributes, each of which has its own set of possible attribute values. The appearance of a person may, for example, be described in terms of the attributes 'height,' 'hair,' 'eyes,' an individual, by the vector of attribute values [short, blond, blue] .

ID3 develops a decision tree which discriminates between two classes of objects, the examples and counter examples of the concept to be learnt, by applying the following recursive procedure.

1. Commence with a given collection C of objects.

2. If C is empty then associate it arbitrarily with either class.

3. If all objects in C belong to the same class, then the decision tree is a leaf bearing that class name.

4. Otherwise C contains objects which belong to both classes. An attribute is selected and C is partitioned into disjoint sets C1, C2, ..., Cn where Ci contains those members of C that have the ith value of the selected attribute. This rule-forming strategy is then applied to each of the subcollections Ci.

5. The attributes/attribute values chosen to describe the objects in C may be insufficient. C may contain objects which belong to both classes but there may be no attributes remaining on which C can be partitioned. In this case the decision tree is a leaf bearing the name 'search.'

Quinlan (1983) illustrates this process with the problem of discriminating between the positive and negative examples in the collection listed in Table 1.

The decision tree for this problem is illustrated in Fig. 7.

Attribute selection should be such that the final decision tree is in some sense minimal. In the method employed in these experiments attribute selection is based on an information theoretic approach aimed at minimizing the expected number of tests required to classify an object (Quinlan, 1983).

The decision trees developed in these experiments have been translated into conjunctive rules by specifying each conjunction of attribute values that defines a path in the decision tree that leads to a positive node. Consider the decision tree illustrated in Fig. 7. This decision tree would be converted into the rules given in Table 2.

The number of positive and negative examples associated with each node in the decision tree conveys information about the importance of the conjunctive rule

Table 1. *Sample set of examples required as input for ID3, where '+' indicates an example and '−' indicates a counterexample.*

C[height, hair, eyes] =	
short, blonde, blue: +	tall, dark, blue: −
tall, blonde, brown: −	tall, blonde, blue: +
tall, red, blue: +	tall, dark, brown: −
short, dark, blue: −	short, blonde, brown: −

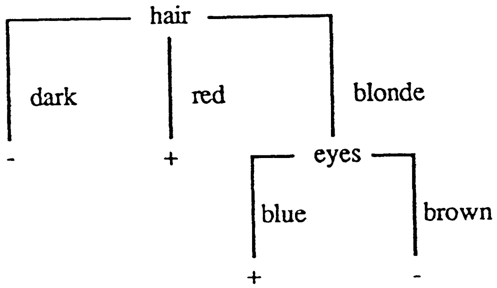


Figure 7. *Decision tree developed from the examples listed in Table 1.*

Table 2. *Conjunctive rules extracted from the decision tree in Fig. 3.*

Rule	Evidence
Rule 1: blonde hair/blue eyes	2 positive
Rule 2: red hair	1 positive

which describes the path leading to that node. The rules that best represent the concept to be learnt are those with which the most positive examples and least negative examples are associated. In this case Rule 1 would seem to be a better representation of the concept to be learnt than Rule 2.

5 Learning about the Physical Implications of Environmental Performance Criteria in Building Design

5.1 Description

The generation-simulation-optimization-induction paradigm described earlier provides a potentially useful basis for exploring the physical implications of the environmental performance requirements of a space as a vehicle for understanding the utility of induction processes in acquiring expert knowledge from designs. The reasons for this are the following.

1. Environmental design requirements typically conflict. For example, winter heating and lighting requirements may best be achieved within a space by including large areas of glass in the external walls. This may, however, result in summer temperatures which are unacceptably high.

2. The qualitative nature of potential tradeoff decisions is understood, or can be investigated, by the designer. Meaningful decision and criteria spaces can be defined.

3. Quantitative specifications of the environmental performance requirements of a space are readily available.

4. The behaviour of a design solution in terms of environmental performance criteria can be quantified. A considerable body of theory aimed at predicting the environmental performance of design proposals exists.

Radford and Gero (1980) explored the physical implications of optimizing the performance of a design proposal in terms of the conflicting performance criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature. A north facing office in Hobart, Australia was simulated to generate decision and performance data. The design decision variables considered were: wall type, sunshade size and the size, position and construction of the windows. The daylight factor, mean summer temperature and mean winter temperature of each proposal were calculated. This data was analysed using tradeoff diagrams derived from the Pareto optimal sets. The decision space has been simplified to that defined by the design variables: glass type, wall type, window size and sunshade size. This data is used in this experiment and is presented in the Appendix.

An important limitation of the data available is that it consists only of the decision and performance data of solutions which are Pareto optimal in terms of the three criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature. There are no examples of design decision combinations that describe solutions which are inferior in terms of these three criteria. Two thirds of the data set available was used to develop representations of the concepts being explored. This set of 49 sample designs was augmented by generating combinations of design decisions which are not included in this set and which are inferior in performance.

Three induction problems were formulated. In the first the concept of design solutions which are Pareto optimal in terms of all three criteria was explored. The positive example set for this problem consisted of all the instances in the unaugmented training set. The negative example set consisted of the generated inferior set. The second concept explored was that of solutions which are Pareto optimal in terms of the two criteria maximize daylight factor and minimize mean summer temperature. The positive training examples of this concept were those solutions in the unaugmented training set which are Pareto optimal in terms of these two criteria. The negative training examples were the remaining instances in the augmented training set. The concept of solutions which are Pareto optimal in terms of the performance criteria minimize mean summer temperature and maximize mean winter temperature was explored in a similar manner.

A representation of the concept of solutions which are Pareto optimal in terms of the performance criteria maximize daylight factor and maximize mean winter temperature was not developed as the training set for this concept contains only a single positive example. The representation that would result would recognise those examples characterised by the attribute values of this single positive example and only those examples as positive examples of the concept learnt.

5.2 Results

The induction algorithm ID3 was used with the data in the Appendix as described to produce a set of decision trees each of which has been interpreted as a set of conjunctive rules. These are shown in Tables 3, 4 and 5 for the three cases investigated.

5.3 Evaluation of Induced Generalisations

The generalisations developed are evaluated in terms of the criteria proposed by Michalski.

Table 3. *Conjunctive rules extracted from the decision tree developed to describe the concept of solutions which are Pareto optimal in terms of the three criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter tempearature.*

Rule		Evidence
Rule 1:	300mm cavity brick wall, unplastered / sunshade small / window medium	16 positive
Rule 2:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade small / 2*3mm clear float glass double glazing / window medium	5 positive
Rule 3:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade small / 2*3mm clear float glass double glazing / window small	4 positive
Rule 4:	300mm cavity brick wall, unplastered / sunshade small / window large / 2*3mm clear float glass double glazing	3 positive
Rule 5:	300mm cavity brick wall, unplastered / sunshade medium / 3mm clear float glass / window medium	3 positive
Rule 6:	300mm cavity brick wall, unplastered / sunshade medium / 6mm heat absorbing glass /window small	3 positive
Rule 7:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade small / 3mm clear float glass / window large	3 positive
Rule 8:	300mm cavity brick wall, unplastered / sunshade none / 2*3mm clear float glass double glazing / window large	2 positive
Rule 9:	300mm cavity brick wall, unplastered / sunshade medium / 6mm heat absorbing glass / window medium	2 positive
Rule 10:	300mm cavity brick wall, unplastered / sunshade medium / 3mm clear float glass / window small	2 positive
Rule 11:	300mm cavity brick wall, unplastered / sunshade small / 6mm heat absorbing glass / window small	1 positive
Rule 12:	300mm cavity brick wall, unplastered / sunshade small / window large / 3mm clear float glass	1 positive
Rule 13:	110mm brick, 90mm air space, 50mm insulation, 12mm plasterboard / sunshade large / 6mm heat absorbing glass / window medium	1 positive
Rule 14:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade large / 6mm heat absorbing glass / window small	1 positive
Rule 15:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade none / 2*3mm clear float glass double glazing / window large	1 positive
Rule 16:	20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / sunshade medium / 6mm heat absorbing glass / window small	1 positive

Table 4. *Conjunctive rules extracted from the decision tree describing the concept of solutions which are Pareto optimal in terms of the performance criteria: maximize daylight factor and minimize mean summer temperature.*

Rule		Evidence
Rule 1:	300mm cavity brick wall, unplastered / sunshade small / window medium / 3mm clear float glass	3 positive
Rule 2:	300mm cavity brick wall, unplastered / sunshade medium / 3mm clear float glass / window medium	3 positive
Rule 3:	300mm cavity brick wall, unplastered / sunshade medium / 6mm heat absorbing glass / window small	3 positive
Rule 4:	300mm cavity brick wall, unplastered / sunshade small / window medium / 6mm heat absorbing glass	2 positive
Rule 5:	300mm cavity brick wall, unplastered / sunshade medium / 6mm heat absorbing glass / window medium	2 positive
Rule 6:	300mm cavity brick wall, unplastered / sunshade large / 6mm heat absorbing glass / window small	1 positive
Rule 7:	300mm cavity brick wall, unplastered / sunshade none / 2*3mm clear float glass double glazing / window large	1 positive
Rule 8:	300mm cavity brick wall, unplastered / sunshade small / window large / 3mm clear float glass	1 positive
Rule 9:	300mm cavity brick wall, unplastered / sunshade small / window large / 2*3mm clear float glass double glazing	1 positive
Rule 10:	300mm cavity brick wall, unplastered / sunshade medium / 3mm clear float glass / window small	1 positive

Table 5. *Conjunctive rules extracted from the decision tree describing the concept of solutions which are Pareto optimal in terms of the performance criteria: minimize mean summer temperature and maximize mean winter temperature.*

Rule		Evidence
Rule 1:	2*3mm clear float glass double glazing / sunshade small / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / window small	4 positive
Rule 2:	2*3mm clear float glass double glazing / sunshade small / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / window medium	4 positive
Rule 3:	2*3mm clear float glass double glazing / sunshade none / window large / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard	1 positive
Rule 4:	6mm heat absorbing glass / sunshade large / window small / 300mm cavity brick wall, unplastered	1 positive
Rule 5:	6mm heat absorbing glass / sunshade large / window small / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard	1 positive

Validity The design decision combinations identified as being most characterisitic of solutions which are Pareto optimal in terms of the performance criteria mimimize mean summer temperature and maximize mean winter temperature, namely, weatherboard walls, double glazed small or medium windows with small sunshades, correspond to those features found in the predominant type of residential construction in Hobart. The design decision combinations identified as being most characteristic of solutions which are Pareto optimal in terms of all three criteria, namely, heavy brick construction, medium sized windows and small sunshades, are 'reminiscent of British buildings and early government buildings in Hobart' (Mackenzie and Gero, 1987). This makes some intuitive sense, the primary concern in designing residential buildings would be to achieve optimal thermal conditions whereas in early government buildings all three criteria would have been important.

It is indicated in Table 6 that there is a high degree of fit between the representations developed in this experiment and that defined by the full set of 73 design solutions available (Appendix).

Explanatory Power The rules developed using ID3 were used to classify the 24 unseen instances in the original training set. The results are listed in Table 7.

The representations developed exhibit reasonable predictive behaviour when used to classify the 24 unseen examples but these are all examples of solutions that are Pareto optimal in terms of the three criteria: minimize mean summer temperature, maximize mean winter temperature and maximize daylight factor. If examples of solutions that are inferior in terms of these criteria were available and were considered to be negative examples of the concept to be learnt then it would be expected that the representations developed would misclassify many of these inferior examples as Pareto optimal examples. The fact that non-Pareto optimal solutions may be close to Pareto optimal solutions in both the decision and performance space would suggest that the approach adopted here of using the inverse set of design decision combinations to those which correspond to Pareto optimal solutions may be necessary in order to develop representations of the concept of Pareto optimal solutions for any criteria, Fig. 8. This approach results in the development of a maximally specific representation of the concept being learnt but in this situation such a representation is appropriate for the following reasons.

1. The decision and performance data are automatically generated, it is likely that the entire Pareto optimal set would be generated. The training set would thus completely specify the concept, generalizations aimed at including unseen examples of the concept would not be necessary.

2. The designer is looking for prescriptive information. Those decisions which are significant in terms of the performance criteria being modelled should be fully specified.

Table 6. *Percentage of the 73 sample designs available that are misclassified by the decision trees representing the concept of a) solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature (dsw), b) solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor and minimize mean summer temperature (ds), and c) solutions that are Pareto optimal in terms of the performance criteria: minimize mean summer temperature and maximize mean winter temperature (sw). Predictions are also made about the misclassification errors that would have occurred if a decision tree had been developed to describe the concept of solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor and maximize mean winter temperature. The 'percentage errors' value is the percentage of the 73 sample designs that were misclassified. The 'percentage of false positives' value is similar to the 'percentage errors' value but only negative examples of the concept to be learnt that were misclassified are counted.*

	dsw	ds	sw	dw
Percentage Errors	11%	7%	10%	1%
Percentage of False Positives	Negative examples not available	3%	3%	0%

Table 7. *Percentage of the 24 unseen examples misclassified by the decision trees developed in each of the induction problems.*

	dsw	ds	sw	dw
Percentage Errors	33%	21%	17%	4%
Percentage of False Positives	Negative examples not available	8%	8%	0%

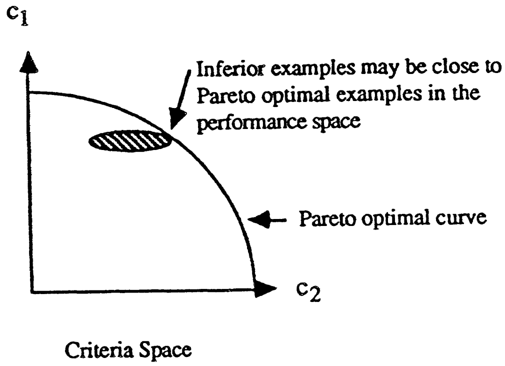

c1
Inferior examples may be close to Pareto optimal examples in the performance space
Pareto optimal curve
c2
Criteria Space

Figure 8. *Schematic diagram indicating that non-Pareto optimal solutions may exhibit high levels of performance in terms of the criteria modelled. It may not be appropriate to regard the full set of inferior solutions as negative examples of the concept of Pareto optimality in terms of these criteria.*

Effectiveness The rules developed make design options explicit and establish a hierarchy within these options based on the predominance of the examples that they cover in the training set. These rules do not include information about the regions of the performance space covered by solutions characterized by particular design variables (sensitivity of performance) or about the nature of the tradeoff decisions involved. This information could be incorporated by augmenting the representation with information as to the range of values in each dimension of performance of the solutions in the training set (and the performance of any non-Pareto optimal solutions available) associated with each conjunctive rule extracted from the decision tree. In addition the location of these solutions on the two dimensional graphs defined by each pairwise combination of the dimensions of performance considered in formulating the problem could be indicated. Such a strategy would allow good compression of information about the sensitivity of design options.

Executing the Pareto optimization induction process for each subset (with more than one criteria) of the performance criteria generates information about the stability of the design options identified when the full set of criteria were considered in relation to these subsets of criteria. For example the design decisions 300mm unplastered cavity brick wall, small sunshade and medium sized window [Table 3 Rule1] identified as being important when the full set of performance criteria was considered are also important when the criteria maximize daylight factor and minimize mean summer temperature are considered alone [Table 4 Rules 1 and 4]. The performance ranges associated with this option are optimal in terms of these two criteria, they serve as a basis for comparison with the performance ranges associated with other design options and thus as a means of exploring the implications of reducing the importance of these criteria in relation to the third criteria, that of maximizing mean winter temperature. In addition, the decision combinations identified when just the two criteria were considered [Table 4 Rules 1 and 4] are more specific than that identified when all three criteria were considered. Information as to the glass type to be preferred if this option were to be pursued and the criteria maximize daylight factor and minimize mean summer temperature were more important than that of maximizing mean winter temperature is specified.

Comprehensibility The implications of a particular set of performance criteria in terms of the design features that should be adopted in developing a proposal to satisfy these criteria are better expressed in representations such as those developed in this experiment than they are as either sets of characterized design such as those that were the data for these experiments or the conventional form of representation of the Pareto optimal set for a three criteria problem: a three dimensional graph. Furthermore as the number of performance criteria increases graphical representations become increasingly difficult to construct and to understand.

5.4 Comparison with a Heuristic Based Rule Extraction System

The representations developed should not be judged against an imaginary perfection but against the representations that would otherwise be available. PARE (Mackenzie and Gero, 1987) is a heuristic based system which analyses the shape of a Pareto optimal curve for a given multicriteria design problem and extracts consistencies in the decisions that characterise important features of this curve. The PARE system models the performance of a human in visual analysis of Pareto optimal curves (visual analysis of such curves is difficult where the dimensions of performance to be considered are greater than two). Mackenzie and Gero (1987) list the ten most interesting rules extracted by the PARE system, Table 8.

Validity There is some correspondence between the rules developed in this experiment and the rules developed using the PARE system. The design decisions identified here as being predominant in characterising solutions which are Pareto optimal in terms of all three criteria considered [Table 3 Rule 1] are the same as those identified using the PARE system as being the most important characteristics of solutions with performances that represent the best tradeoff between these three criteria [Table 8 Rule 8]. The design decisions identified in this experiment as being those which best characterize solutions that are Pareto optimal in terms of the performance criteria minimize mean summer temperature and maximize mean

Table 8. *Heuristics extracted from the data in the Appendix using the PARE system.*

Rule 1	if summer_temp and winter_temp are important then 2*3mm clear float glass double glazing is extremely important.
Rule 2	if daylight_factor and summer_temp and winter_temp are important then large_ sun shade is not very important.
Rule 3	if daylight_factor and summer_temp and winter_temp are important then 6mm heat absorbing glass / 110mm brick, 90mm air space, 50mm insulation, 12mm plasterboard is not at all important.
Rule 4	if summer_temp and winter_temp are important then 2*3mm clear float glass double glazing / small_ sun shade is extremely important.
Rule 5	if daylight_factor and summer_temp and winter_temp are important then 3mm clear float glass / small window is not at all important.
Rule 6	if daylight_factor and summer_temp and winter_temp are important then small_sun shade / medium window is very, very important.
Rule 7	if daylight_factor and summer_temp are important then 6mm heat absorbing glass / 300mm cavity brick wall, unplastered / medium window is very, very important.
Rule 8	if daylight_factor and summer_temp and winter_temp are important then 300mm cavity brick wall, unplastered / small_sun shade / medium window is very, very important.
Rule 9	if summer_temp and winter_temp are important then 2*3mm clear float glass double glazing / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / small_sun shade / small window is very, very important.
Rule 10	if summer_temp and winter_temp are important then 2*3mm clear float glass double glazing / 20mm weatherboard, 25mm airspace, 75mm insulation, 12mm plasterboard / small_sun shade / medium window is very, very important

winter temperature [Table 5 Rules 1 and 2] are the same as those identified by the PARE system as being characterisic of solutions with performances that represent the best tradeoff between these two criteria [Table 8 Rules 9 and 10]. As noted in Section 5.3 the features identified in both induction problems correspond to those characterising particular construction types that have evolved in the region.

The relative performance of the representations developed using each of the systems as representations of the full set of 73 Pareto optimal solutions is indicated in Tables 9, 10 and 11. The fact that the performance of the ID3 developed representation for the three criteria problem is much better than that developed using PARE suggests that the concept to be learnt is disjunctive. This will be discussed further in the next section.

Explanatory Power The rules developed using ID3 were used to classify the 24 unseen instances in the original training set. The results, together with those of the corresponding rules devoped using the PARE system are listed in Tables 12, 13 and 14.

While the PARE system is designed to extract consistencies in the solutions which result in the best tradeoff between the conflicting performance criteria rather than consistencies in the solutions that make up the Pareto optimal set as a whole, the rules developed using ID3 describe the attribute values of a dominant subset of similar solutions (solutions which are close in the decision space) of the Pareto optimal set.

ID3 can develop representations of disjunctive concepts. Bundy *et al.* (1985) state: 'Classification will usually produce a disjunctive rule, even when focusing would produce a conjunctive rule on the same data.' The level of explanation provided by the representation developed using ID3 for the three criteria problem in relation to that developed using the PARE system indicates that the concept to be learnt is a disjunctive one. Bundy *et al.* (1985) state that classification algorithms such as ID3 are preferable to the other main methods of 'learning from examples,' focusing and the candidate elimination algorithm, when the concept to be learnt is disjunctive. They note that this ability to learn disjunctive concepts is obtained at the expense of the simplicity of the representation developed, the decision trees developed by ID3 are often non-optimal.

The number of positive examples in the set of unseen examples misclassified by the representation of the three criteria Pareto optimal set developed using ID3 is, however, high (1 in 3). The strategy employed in this experiment of generating a negative example set which consists of the complement of the positive examples in the training set, prevents the ID3 algorithm from generalizing about this training set so as to include unseen examples of the concept. The PARE system is oriented towards the development of general conjunctive representations of a concept. It appears that in the case of the two criteria problems the predictive power of this type of representation is greater than that of the specific disjunctive representa-

Table 9. *Percentage of the 73 sample designs available that were misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature developed using ID3 and PARE.*

	ID3	PARE Rule 8	PARE Rule 6
Percentage Errors	11 %	71 %	60%
Percentage of false positives	Negative examples not available.		

Table 10. *Percentage of the 73 sample designs available misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor and minimize mean summer temperature.*

	ID3	PARE Rule 7
Percentage Errors	7 %	22 %
Percentage of false positives	3 %	1 %

Table 11. *Percentage of the 73 sample designs available that are misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: minimize mean summer temperature and maximize mean winter temperature.*

	ID3	PARE Rule 9 & 10	PARE Rule 4	PARE Rule 1
Percentage Errors	10 %	12 %	19 %	21 %
Percentage of false positives	3 %	4 %	15 %	18 %

Table 12. *Percentage of the 24 unseen sample designs available that were misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature developed using ID3 and PARE.*

	Rules developed using ID3	PARE Rule 8	PARE Rule 6
Percentage Errors	33 %	79 %	67 %
Percentage of false positives	Negative examples not available.		

Table 13. *Percentage of the 24 unseen sample designs available that were misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: maximize daylight factor and minimize mean summer temperature.*

	Rules developed using ID3	PARE Rule 7
Percentage Errors	21 %	21 %
Percentage of false positives	8 %	0 %

Table 14. *Percentage of the 24 unseen sample designs available that were misclassified by the representations of solutions that are Pareto optimal in terms of the performance criteria: minimize mean summer temperature and maximize mean winter temperature.*

	Rules developed using ID3	PARE Rule 9 & 10	PARE Rule 4	PARE Rule 1
Percentage Errors	17 %	13 %	13 %	13 %
Percentage of false positives	8 %	4 %	13 %	13 %

tions developed in these experiments. As the number of performance dimensions increases, however, it is likely that the concept to be learnt will be more complex, and therefore better described by a disjunctive representation.

Effectiveness The representations developed in these experiments are oriented towards representation of the entire Pareto optimal set. Unlike the rules developed by the PARE system the representations developed are not based on implicit tradeoff decisions (the PARE system analyses the shape of the Pareto optimal curve, giving preference to solutions which occur in certain regions of this curve). The representations give priority to the decisions which best differentiate between design decision combinations that describe solutions that are in the Pareto optimal set from those which do not describe solutions that are in this set.

In determining which of the feasible points in the performance space represent the most desirable level of performance in terms of conflicting perfomance criteria a designer would take into account both the nature of the tradeoff decisions involved and information about the absolute optimal ranges of performance values.

The PARE system, which employs information about the nature of the tradeoff decisions involved, identified the decisions:

w4 (300mm cavity brick wall, unplastered)/
small sunshade/
medium window
[Table 8 Rule 8]

as being important characteristics of solutions with performances that represent the best tradeoff between the three criteria: maximize daylight factor, minimize mean summer temperature and maximize mean winter temperature (these decisions were also identified in these experiments as being important characteristics of solutions in the Pareto optimal set as a whole [Table 3 Rule 1]). The range of performance of the solutions in the training set described by these decisions in the three dimensions daylight factor, mean summer temperature and mean winter temperature is given in Table 15.

Table 15. *Range of performance of solutions in the training set described by the design decisions specified in Rule 1, Table 3.*

daylight factor	0.51–2.38
mean summer temperature	23.8–27.0
mean winter temperature	12.3–18.7

The Australian Government publication 'Thermal Comfort At Work' specifies desirable temperature ranges for working environments:

> For humidity of 30%–60% light work/sedentry activities can be performed comfortably in the indoor dry bulb range of 18°C–30°C. Optimum comfort occurs in the range of 21°C–26°C.

By augmenting the conjunctive rules extracted from the decision tree describing solutions which are Pareto optimal in terms of the three criteria with information as to the range of performance of the solutions described by the design decisions specified in each rule, it can be seen that the range of performance of solutions described by the variables specified in say Rule 7 Table 3 is closer to the optimal temperature ranges of both summer and winter temperature and is higher in daylight factor than that of the solutions described by the variables identified using PARE, Table 16.

Table 16. *Range of performance of the solutions in the training set described by the design decisions specified in Rule 7, Table 3.*

daylight factor	2.26–2.53
mean summer temperature	27.0–27.4
mean winter temperature	18.3–18.9

Only the ten most interesting rules developed using the PARE system have been listed. As the system explores consistencies over broader regions of the Pareto optimal curve decision performance relationships identifying the decisions which characterize solutions closer to the optimum temperature ranges specified above will no doubt be identified. In addition the PARE system incorporates heuristics for establishing associations between particular design decision variables and regions of the performance space. The problem here is a combinatorial one. The number of rules generated before one which covers a range of performance values close enough to the optimum range is found may be excessive.

Comprehensibility The number of positive and negative examples associated with each node in the decision tree conveys information about the importance of the conjunctive rule which describes the path leading to that node. Heuristics could possibly be developed that would allow this information to be expressed as an index of importance such as that used by Mackenzie and Gero (1987) where the importance of the rules is specified using the set of linguistic descriptors. Similar indexes could be developed to describe the sensitivity of design options, the appropriateness of the range of performance covered by the examples used to generate each option in relation to the optimal ranges of each criteria modelled and the stability of each option.

6 Conclusion

The knowledge inferred can be directly translated into IF-THEN rules and employed in the knowledge-base of an expert system which supports CAD. For example, in a CAD system which aims to assist designers of office buildings in Hobart at the preliminary stage of design a set of decisions would be recommended. If the requirements included maximizing daylight factor, minimizing mean summer temperature and maximizing mean winter temperature then the initial recommendation would be to use an unplastered 300 mm cavity brick wall, small sunshades and a medium size window [Table 3, Rule 1]. If the designer rejected this recommendation on the basis that he wanted a large window then an expert system could readily produce the following recommendation: unplastered 300 mm cavity brick wall, small sunshade, large window and 2*3 mm clear float glass double glazing.

Knowledge acquisition is fundamental to the success of knowledge-based systems. Like many other professionals designers are often unable to articulate much of the knowledge which they use implicitly. However, designs can often be characterized by a morphism between design decisions and design performances, and induction used in this characterization to acquire knowledge. The knowledge acquired is of the abductive kind which supports design decision making. It can be put into a variety of forms and used in an expert system context in the form of experiential and phenomenological knowledge with a solid foundation. More significantly, such knowledge provides a direct mapping from performances to decisions which is precisely the direction that designers need; rather than mappings from decisions to performance which is what deep models produce. This knowledge may also be used to provide initial solutions to optimization algorithms to improve their efficiency.

References

Balachandran, M. and Gero, J. S., (1986), 'Dimensioning of architectural floor plans under conflicting objectives,' *Environment and Planing B*, **14**, 29-37.

Bundy, A., Silver, B. and Plumer, D., (1985), 'An analytical comparison of some rule-learning programs,' *Artificial Intelligence* **27**, 2, 137–181.

Cohon, J. L., (1978), *Multiobjective Programming and Planning*, Academic Press, New York.

Coyne, R. D., Rosenman, M. A., Radford, A. D. and Gero, J. S., (1987), 'Innovation and creativity in knowledge-based CAD,' in Gero, J. S. (Ed.), *Expert Systems in Computer-Aided Design*, North-Holland, Amsterdam, pp. 435-465.

Coyne, R. D. and Gero, J. S., (1986), 'Semantics and the organisation of knowledge in design,' *Design Computing* **1**, 1, 68–69.

Gero, J. S. and Radford, A. D., (1984), 'The place of multicriteria optimisation in design,' in Langdon, R. and Purcell, P. (Eds.), *Design Theory and Practice*, The Design Council, London, pp. 81–85.

Gero, J. S., (Ed.) (1985a), *Optimization in Computer-Aided Design*, North-Holland, Amsterdam.

Gero, J. S., (Ed.) (1985b), *Design Optimization*, Academic Press, New York.

Gero, J. S. and Balachandran, M., (1986), 'Knowledge and design decision processes,' in D. Sriram and R. Adey (Eds.), *Applications of Artificial Intelligence to Engineering Problems*, Springer-Verlag, Berlin, pp. 343–352.

Langley, P. and Carbonell, J. G., (1984), 'Approaches to machine learning,' *J. Amer. Soc. Inf. Sci.* **35**, 5, pp. 306–316.

Mackenzie, C. A. and Gero, J. S., (1987), 'Learning design rules from decisions and performances,' *Artificial Intelligence in Engineering* **2**, 1, pp. 2–10.

Michalski, R. S., (1986), 'Understanding the nature of learning: issues and directions,' in Michalski, R. S., Carbonell, J. G. and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, 2, Kaufmann, Los Altos, pp. 3–25.

Pareto, V., (1971), *Manual of Political Economy*, Kelly, New York.

Quinlan, J. R., (1983), 'Learning efficient classification procedures and their application to chess endgames,' in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, pp. 463–482.

Radford, A. D. and Gero, J. S., (1980), 'Tradeoff diagrams for the integrated design of the physical environment in buildings,' *Building and Environment* **15**, 3–15.

Radford, A. D., Gero, J. S., Rosenman, M. A. and Balachandran, M., (1985), in J. S. Gero (Ed.), 'Pareto optimization as a computer-aided design tool,' *Optimization in Computer-Aided Design*, North-Holland, Amsterdam.

Radford, A. D., Hung, P. and Gero, J. S., (1984), 'New rules of thumb from computer-aided structural design: acquiring knowledge for expert systems,' in J. Wexler (Ed.), *CAD 84*, Butterworths, Guildford, pp. 558–566.

APPENDIX

Design decision and performance data generated by simulating a north facing office in Hobart, Australia

g1: 3mm clear float glass

g2: 2 * 3mm clear float glass double glazing

g3: 6mm heat absorbing glass

w4: 300mm cavity brick wall, unplastered

w5: 110mm brick/90mm air space/50mm insulation/12mm plasterboard

w6: 20mm weatherboard/25mm airspace/75mm insulation/12mm plasterboard

smwin: small window

medwin: medium window

largewin: large window

none: no sunshade

small: small sunshade

medium: medium sunshade

large: large sunshade

	+ Maximise	Sensible range	
Criterion	− Minimise	min	max
daylight_factor	+1	0.07	2.91
summer_temp	−1	23.1	29.6
winter_temp	+1	9.6	23.1

Design variables

g1 g2 g3 w4 w5 w6 none small medium large smwin medwin largew

Points in the criteria space

	Criteria		
Decisions	Daylight factor	Summer temp	Winter temp
g3/w4/large/smwin	0.07	23.1	9.60
g3/w4/medium/smwin	0.14	23.2	9.70
g3/w4/large/smwin	0.16	23.2	10.0
g3/w4/medium/smwin	0.28	23.3	10.0
g3/w4/medium/smwin	0.46	23.5	10.4
g3/w4/medium/medwin	0.54	23.6	10.6
g3/w4/medium/medwin	0.63	23.7	10.7
g3/w6/large/smwin	0.07	23.2	10.7
g3/w4/medium/medwin	0.63	23.7	10.7
g3/w6/medium/smwin	0.28	23.5	11.0
g3/w4/medium/medwin	0.83	24.0	11.4
g3/w4/large/medwin	0.92	24.2	11.5
g3/w4/small/medwin	0.51	23.8	12.3
g3/w5/large/medwin	0.67	24.0	11.8
g3/w5/small/medwin	0.51	23.8	12.3
g3/w4/small/medwin	1.13	24.6	12.3
g1/w4/medium/smwin	0.93	24.3	12.8
g3/w4/small/medwin	1.40	25.0	12.8
g3/w4/small/medwin	1.49	25.1	12.8
g1/w4/medium/smwin	1.07	24.5	13.3
g2/w6/small/smwin	0.18	23.5	13.2
g1/w4/medium/medwin	1.19	24.7	13.5
g3/w4/medium/medwin	1.20	24.9	12.6
g1/w4/medium/medwin	0.93	24.3	12.8
g1/w4/small/medwin	0.79	24.3	14.3

Points in the criteria space (cont.)

	Criteria		
Decisions	Daylight factor	Summer temp	Winter temp
g2/w6/small/smwin	0.27	23.7	14.2
g2/w4/small/medwin	0.62	24.2	14.6
g1/w4/small/medwin	0.94	24.6	14.8
g1/w4/medium/medwin	1.41	25.1	14.8
g1/w4/small/medwin	1.22	25.1	15.1
g1/w4/small/medwin	1.22	25.1	15.1
g2/w6/small/smwin	0.37	23.9	15.1
g1/w4/medium/medwin	1.57	25.3	15.1
g1/w4/medium/medwin	1.62	25.5	15.1
g2/w4/small/medwin	0.77	24.4	15.5
g1/w4/medium/largew	1.82	25.7	15.6
g1/w4/small/medwin	1.28	25.3	15.6
g2/w4/small/medwin	0.83	24.6	15.9
g1/w4/small/medwin	1.45	25.5	16.0
g2/w6/small/smwin	0.46	24.0	16.0
g1/w4/small/medwin	1.58	25.7	16.3
g1/w4/small/medwin	1.99	26.3	16.6
g1/w4/small/medwin	1.68	25.9	16.6
g2/w6/small/smwin	0.54	24.2	16.7
g2/w4/small/smwin	1.13	25.2	16.7
g1/w4/small/medwin	2.11	26.5	17.1
g1/w4/small/medwin	2.26	26.8	17.4
g2/w4/small/medwin	1.28	25.5	17.5
g2/w4/small/largew	2.38	27.0	17.8

Points in the criteria space (cont.)

	Criteria		
Decisions	Daylight factor	Summer temp	Winter temp
g2/w6/small/medwin	0.70	24.5	17.8
g1/w4/small/medwin	2.38	27.0	17.8
g2/w6/small/medwin	1.13	25.5	19.7
g1/w4/small/largew	2.53	27.2	18.0
g1/w4/none/largew	2.91	29.0	18.2
g1/w6/small/largew	2.26	27.0	18.3
g2/w4/small/medwin	1.39	25.7	18.2
g2/w6/small/medwin	1.07	25.3	18.9
g2/w6/small/medwin	0.83	24.8	18.3
g2/w4/small/medwin	1.49	25.9	18.7
g1/w6/small/largew	2.53	27.4	18.9
g1/w6/small/largew	2.38	27.2	18.9
g2/w6/small/medwin	0.98	25.2	18.9
g1/w6/none/largew	2.91	29.2	19.1
g1/w6/small/largew	2.53	27.4	18.9
g2/w6/small/medwin	1.13	25.5	19.7
g2/w6/small/medwin	1.28	25.8	20.2
g2/w4/none/largew	2.29	28.5	20.3
g2/w4/small/largew	2.10	27.1	20.6
g2/w6/small/largew	1.49	26.2	20.9
g2/w4/small/largew	2.23	27.4	21.2
g2/w4/none/largew	2.57	29.2	21.5
g2/w6/small/largew	2.23	27.6	22.8
g2/w6/none/largew	2.57	29.6	23.1

Intelligent Systems for Pre and Post Processors for Large Structural Analysis Computer Packages

A. T. Humphrey, R. A. Corlett*,
J. N. Trumpeter*
GEC-Marconi Research Centre
West Hanningfield Road
Great Baddow
Chelmsford, Britain
* Former members of GEC-Marconi Research Centre.

Abstract The Nastran advisor was devised as an experimental system aimed at examining the feasibility of using a knowledge based approach to aid a novice or intermittent user of NASTRAN in the selection of the required program subsets to solve particular problem classes.

1 Introduction

Nastran is currently available to GEC Product Units via Company Data Centres. Expertise is vested in a number of specialist areas within research and development groups. The program is used during the design and development of a wide range of advanced mechanical systems including under water weapons, radar, telecommunications, avionics and space communication system forms. Many of the structural systems are novel and require the use of sophisticated modelling, analysis and experimental facilities during the development programme. NASTRAN is of particular value in that it incorporates a broad spectrum of capabilities directly suited to the analysis of these classes of system. The provision of a knowledge based Consultant would go some way in providing novice and intermittent program users with a procedure for accessing the program without reference to a NASTRAN specialist.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 251–262.

2 The Role of a Knowledge Based Consultant

This paper describes the first stages of development of an automated knowledge based Consultant (NASCON) created to advise non experts in the selection and development of analysis and modelling strategies specifically for NASTRAN. Knowledge based programs may be distinguished from conventional programs in that they possess a corpus of domain specific information which relates to real world situations.

The class of expert system under development consists of two main segments (Fig. 1) namely a knowledge base and an inference mechanism. The former contains information encapsulated in the form of situation/action rules relating to a particular context. The inference mechanism provides a means for searching through the knowledge base to establish the relevance of the information contained to the context currently under examination by the computer Consultant. A decision tree provides a convenient construct for visualising the process although for the system under discussion here the tree is assembled dynamically as the Consultation proceeds. How domain knowledge is characterised and classified is somewhat arbitrary although there are recognisable conventions and domain specific vocabularies appropriate to branches of engineering science and technologies and also of course appropriate to NASTRAN.

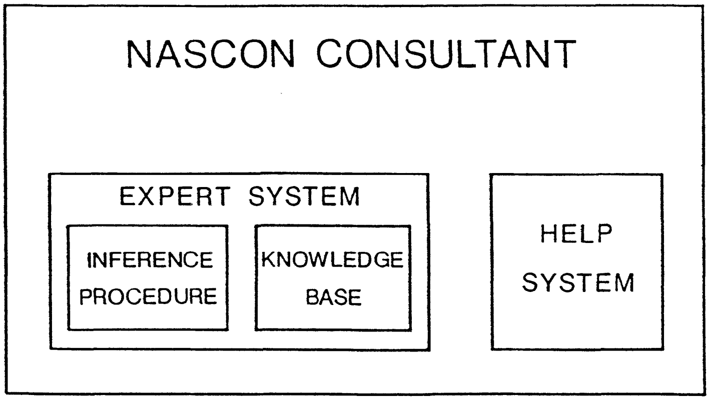


Figure 1. *NASCON components.*

Knowledge engineering science is comparatively recent in origin and there are few large scale knowledge based systems currently in use. One of the earliest practical examples is MYCIN [1]. This was originally developed to assist medical practioners in the diagnosis and treatment of infectious diseases. The earliest reference to an application in structural mechanics appears to be SACON [2] devised to facilitate the utilisation of the MARC finite element code. SACON developers took the E(MYCIN) inference mechanism and substituted structural engineering knowledge for the medical knowledge so converting the program from the domain of infectious diseases to the domain of structural mechanics.

Components of the design and analysis process are outlined in Fig. 2. Categories of domain knowledge pertinent in the present context include for example:

- Product System Knowledge
- Technologies (states and processes)
- Finite Element Methodologies
- Finite Element Modelling Strategies
- NASTRAN Methodologies

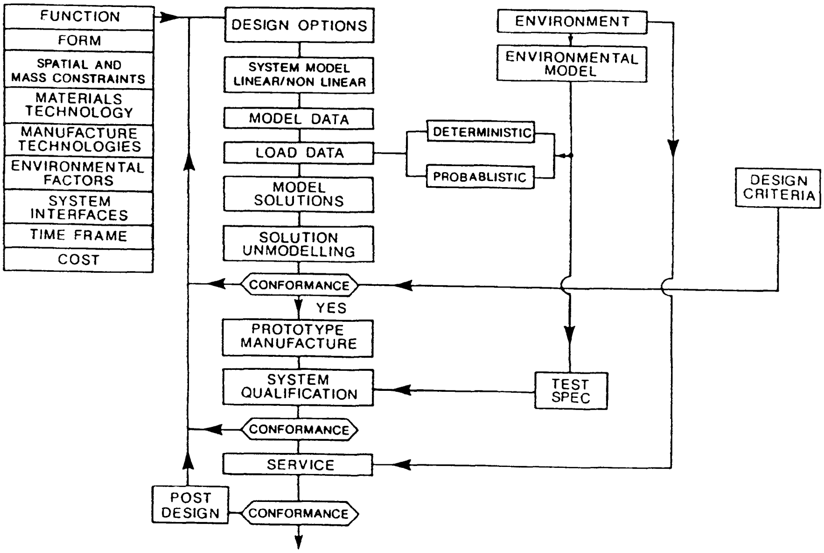


Figure 2. *Design and analysis.*

Technological domains are only inexactly describes in terms of natural language. Difficulties occur in categorising and encapsulating this class of knowledge unambiguously. Characterisation of domain knowledge provides a necessary preliminary, however, to establishing both the form and content of the knowledge base and the design of the expert system interface through which the user communicates with the Consultant. NASTRAN possesses a well defined vocabulary for characteristing program methodologies and data formats.

Based on the analyst's perception and characterisation of the problem domain, information on NASTRAN methodologies and modelling strategies may be abstracted systematically from the knowledge base and information modules as the Consultation proceeds. Information elicited includes selection of the appropriate Rigid Format or Solution Sequence together with the associated series of data card formats aligned to Case Control and Bulk Data. This information will be supplemented by information on detailed finite element modelling strategies addressing usefully cost/accuracy trade-offs in the next phase of development.

A Computer Consultant can, like its human counterpart, modify its line of questioning adaptively, although not with the same degree of flexibility. If the user possesses insufficient knowledge to answer a question, the Computer Consultant can emulate the human expert via the provision of 'Information'. A simple user information provision mode is seen as a useful adjunt to the Consultant.

Information defining specialised modelling templates can for instance be constructed for classes of stylised engineering forms—masts and towers for example, to aid the user directly in devising acceptable models. The development of knowledge based rules governing the generation of models for novel applications will undoubtedly present some difficulty, especially as far as establishing the appropriate level of model refinement is concerned.

3 Nascon Consultant

3.1 Introduction

The NASCON system exists in an environment called POPLOG. This environment is a commercial product written by Sussex University. It contains high level languages (POP11, PROLOG and LISP) an editor and an incremental compiler. NASCON uses the languages PROLOG and POP11. It was initially intended to write the code totally in PROLOG but this was found to be impossible and an increasing number of calls to the POP11 language are now being made, mainly to utilise some of the many POP11 built-in functions.

The Consultant NASCON is divided into a number of separate files. Most of the files are information files but a still substantial number of files make up the core NASCON system. The user does not need to know what these files are in order to use the system since they are loaded automatically by 'nasconload' the main loader file for the system.

NASCON contains two major components, an expert system and a help file system. The expert system consists basically of a production rule system and an inference mechanism which allows both forward and backward chaining. The help file system contains procedures for accessing various types of information sources. These sources include 'active' files and the POPLOG 'database'. There is also a partially completed tool for constructing the knowledge base.

3.2 The Knowledge Base

The inference mechanism and the knowledge base are closely coupled as far as the expert system is concerned but uncoupled as far as the domain is concerned. This means that any domain specific knowledge base must use specific PROLOG predicate names, e.g. know, goal, etc. which are domain independent but can contain any domain specific knowledge as part of the predicates. The production rules used by the inference mechanism (ACE) are written in the form:-

IF <*condition*> THEN <*conclusion*>

where the condition can be of the form:-

<*fact*> AND <*condition*> or

<*fact*>

'IF', 'THEN' and 'AND' are keywords which are presented in PROLOG by operators. 'IF' is a prefix operator and 'THEN' and 'AND' are infix operators.

The conclusion can take the form:-

<*fact*>

A fact can take the form:-

<*atom*> or

NOT(<*atom*>)

Everything that the system learns from the user or other sources is stored in facts of the form:-

KNOW(<*atom*>,<*value*>).

At present the only values allowed are true and false but it is intended to permit any value. It might even be possible to replace the <*atom*> by a condition such as 'N>6', and retain the value as true or false. The goals are given to the system by facts of the form:-

GOAL(<*atom*>,<*goal title*>)

The system then attempts to provide <*atom*> as true.

The final element in the knowledge base is the askable predicate. Askable is of the form:-

ASKABLE(CONDITION,QUESTION,ASSISTANCE FILE).

The 'CONDITION' is a condition from a production rule.

The 'QUESTION' is a question which is put to the user which at present elicits a yes or no answer which means that the condition is present or not present.

The 'ASSISTANCE FILE' is the name of a file of information which is displayed if the user asks for help instead of answering the question. In the final version of NASCON every question will have an associated assistance file. It is possible that some part of the assistance file will be used in an explanation subsystem. The 'ASSISTANCE FILE' will consist of an explanation of why the question is being asked, what the terms in the question mean and may possibly point to, or even load, other relevant files.

In the following all references to assistance files have been removed for clarity.

The following are some of the questions the Consultant askes the User:-

```
     ASKABLE('SUPER ELEMENTS'.
          'DO YOU WISH THE MODEL TO BE CONSTRUCTED  IN  TERMS  OF  SUPER
          ELEMENTS?'
     ASKABLE('CYCLIC SYMMETRY@,
          'DOES THE MODEL EXHIBIT CYCLIC SYMMETRY?'
          THE FOLLOWING QUESTION SUBSET IDENTIFIES CLASSES OF
          MECHANICAL NONLINEARITY WITHIN THE MODEL*/
     ASKABLE('LINEAR ELASTIC PROPERTIES',
          'ARE  ALL  THE  MATERIALS  ASSUMED  TO  HAVE  LINEAR   ELASTIC
          PROPERTIES?'
     ASKABLE('TEMPERATURE DEPENDENT MATERIAL'.
          'ARE THE MATERIAL PROPERTIES TEMPERATURE DEPENDENT?'
     ASKABLE('STRUCTURAL DISCONTINUITIES',
          'DOES THE MODEL CONTAIN ANY STRUCTURAL DISCONTINUITIES WHICH
          COULD GIVE RISE TO GAPPING UNDER LOAD'.
     ASKABLE(*GEOMETRIC NONLINEARITY',
          'WILL THE  MODEL  EXHIBIT  ANY  GEOMETRIC  NONLINEARITY  UNDER
          LOAD?,
*/        THE FOLLOWING QUESTION SUBSET IDENTIFIES CLASSES OF MODEL
     DAMPING*/
     ASKABLE('HYSTERETIC DAMPING'
     'DO YOU WISH TO INCLUDE VISCOUS DAMPING ELEMENT?'
*/        THE FOLLOWING QUESTION SUBSET IDENTIFIES PROBLEM SUBCLASSES
          RELATED TO MODEL BEHAVIOUR*/
```

```
ASKABLE('MODEL BUCKLING',
     'DO YOU WISH TO EXAMINE MODEL BUCKLING BEHAVIOUR?'
ASKABLE('MODEL VIBRATION',
     'DO YOU WISH TO EXAMINE MODEL VIBRATIONAL BEHAVIOUR?'
```

A Selection of domain rules include for example:-

```
IF
'SUPER ELEMENTS' AND
NOT('CYCLIC SYMMETRY') AND
'LINEAR ELASTIC PROPERTIES' AND
NOT('TEMPERATURE DEPENDENT MATERIAL') AND
NOT('STRUCTURAL DISCONTINUITIES') AND
NOT('GEOMETRIC NONLINEARITY') AND
'RANDOM EXCITATION FORMS' THEN
'SELECTION OF SOLUTION SEQUENCE 71'.
IF
'SUPER ELEMENTS' AND
NOT('CYCLIC SYMMETRY') AND
'LINEAR ELASTIC PROPERTIES' AND
NOT('TEMPERATURE DEPENDENT MATERIAL') AND
NOT('STRUCTURAL DISCONTINUITIES') AND
NOT('GEOMETRIC NONLINEARITY') AND
'MODAL COORDINATES' AND
'TIME DEPENDENT EXCITATION FORMS' THEN
'SELECTION OF SOLUTION SEQUENCE 72'.
IF
'SUPER ELEMENTS' AND
NOT('CYCLIC SYMMETRY') AND
'LINEAR ELASTIC PROPERTIES' AND
NOT('TEMPERATURE DEPENDENT MATERIAL') AND
NOT('STRUCTURAL DISCONTINUITIES') AND
NOT('GEOMETRIC NONLINEARITY') AND
'DIRECT COORDINATES' AND
'FREQUENCY DEPENDENT EXCITATION FORMS' THEN
'SELECTION OF SOLUTION SEQUENCE 68'
```

Associated goals comprise:-

```
*    GOAL('SELECTION OF SOL SEQ 71','SUPER ELEMENT MODAL FREQUENCY
     RESPONSE')
*    GOAL('SELECTION OF SOL SEQ 72','SUPER ELEMENT MODAL TRANSIENT
     RESPONSE')
*    GOAL('SELECTION OF SOL SEQ 68','SUPER ELEMENT DIRECT FREQUENCY
     RESPONSE')
```

3.3 The Inference Mechanism

The expert system is based upon Corlett's expert system ACE. ACE is a simple production rule interpreter, written in PROLOG. The interpreter can carry out both forward and backward chaining on a set of production rules.

3.3.1 The Backward Chainer The backward chainer works by choosing a goal (as an example, making a hypothesis about which rigid format or solution sequence is required) and then attempting to verify the facts which make the goal (hypothesis) true. As an example of this consider the backward chainer, choosing the goal:-

```
'selection of solution sequence 81 or solution sequence 82'.
```

The system will look for a rule which has this goal as its conclusion.
The rules are of the form:-

If fact1 and fact2 and fact3 ... then conclusion.

The conclusion is true if all the facts are true. So the system must prove each fact in turn.

In the case of our example the system will find the rule:-

```
IF    'SUPER ELEMENTS' AND                    ....fact 1
      'CYCLIC SYMMETRY, AND                   ....fact 2
      'LINEAR ELASTIC PROPERTIES' AND         ....fact 3
      NOT('TEMPERATURE DEPENDENT MATERIAL')   ....fact 4
      AND
      NOT('STRUCTURAL DISCONTINUITIES') AND   ....fact 5
      NOT('GEOMETRIC NONLINEARITY') AND       ....fact 6
      'STATIC EXCITATION FORMS' THEN          ....fact 7
      'SELECTION OF SOLUTION SEQUENCE 81 OR   ....conclusion
      SOLUTION 82
```

It now has to prove that 'SUPER ELEMENTS' is true. There are several ways in which it can do this:-

1. It may already know that it is true because it finds a cause

   ```
   KNOW('SUPER ELEMENTS',TRUE).
   ```

2. If it does not know that it is true or already been shown to be false, it may have a clause

```
ASKABLE('SUPER ELEMENTS','DO YOU WISH TO HAVE THE
MODEL CONSTRUCTED IN TERMS OF SUPER ELEMENTS?')
```

This means that it can ask the user the associated question.

3. It may find the fact as a conclusion of another rule, in which case it tries to prove the facts of the other rule, so that it can deduce that the fact of the first rule is true.

If it cannot do any of these then the fact must be false or not provable until other rules have been examined.

3.3.2 The Forward Chainer The forward chainer finds each rule in turn for which it does not know the conclusion. It then attempts to prove the conditions of the production rule and if it succeeds it remembers the result. If the conclusion was a goal, it asks if the user has finished and if the user has not finished or the conclusion was not a goal, then it makes a recursive call to the forward chainer. When it runs out of rules it informs the user and terminates.

3.4 Help System

At any time the user may input any string and, if it does not take the form of an answer to a question, it is passed to the help system for analysis. The help system consists of a large number of information files and the following sub-systems:-

1. The Assistance File System
2. The Menu System
3. The Dictionary
4. Explanation System

These sub-systems selectively display files relevant to what the user is currently doing and what he needs to know. We describe each of these sub-systems in turn.

3.4.1 Assistance File System The assistance file system is entered by typing 'h' at any time. The system displays a file appropriate to the point the user is at in the consultation. This file is the assistance file, and will consist of information relating to why the question is being asked, what the terms in the question mean, and possibly pointers to other files. In some cases it may only consist of a pointer to a file already in the menu system.

3.4.2 The Menu System The NASCON menu system is entered by typing 'm' at any time. The user is then presented with a numbered menu, from which he must make a selection. A page of the relevant information file will be displayed and the user will be given an option to display the next page or descend to a more detailed information level by entering a string relevant to the current displayed information, e.g. The user might type 'elements' is he wants more information on elements. There is also a special 'card mode' which will display a NASTRAN card format if the consultation context demands.

The items in the menu are not all at the same hierarchical level. For example, the user may select option NASTRAN bulk data summary from where he can proceed to lower levels such as the card descriptors. Alternatively the user may access the card descriptor information files directly. As the user's knowledge of the system grows, this facility will enable the user to get to the information he requires quickly without having to descend through menu levels.

Typical MENU entries include:-

```
USE OF THE CONSULTANT
RUNNING NASTRAN ON THE IBM
NASTRAN EXECUTIVE CONTROL
NASTRAN CASE CONTROL
     SUMMARY
     CARD DESCRIPTORS
     CARD FORMAT
NASTRAN BULK DATA
     SUMMARY
     CARD DESCRIPTORS
     CARD FORMAT
NASTRAN ERROR MESSAGES
NASTRAN DEMONSTRATION DATA
```

3.4.3 The Dictionary At times the user may only want a short definition of a term in question or may want a brief information concerning a data card. The dictionary provides this information. It is also the default information source, when information cannot be found in other sources. There is a problem when another information source with the same name as the dictionary entry is encountered. To enter the dictionary the user enters the required word or string.

3.4.4 The Explanation System There is a very primative explanation system available, which is invoked by the input of the command 'why'. During the running of the system the name of the condition being investigated is placed upon a 'trail'. When the investigation of the condition is complete the name of the condition is removed from the trail. When the command 'why' is entered, the trail is displayed, indicating the main gaol and the sub-goals. As previously stated, the system is very primitive at the moment but will be expanded when more of the rule base is available.

3.5 The Expert Interface

3.5.1 Introduction An expert interface is being created so that the expert may give his knowledge to the system. This is a simple interface and cannot be regarded as a true knowledge acquisition system. The interface is far from complete and at the moment only contains a sub-interface for obtaining dictionary definitions. This is a separate system from NASCON and can be run at any time by the expert.

The interface consists (or will consist) of the three main sections:-

1. The dictionary update interface
2. The rule base update interface
3. The information update interface.

3.5.2 The Dictionary Update Interface There are two main parts to this interface:-

1. The undefined strings interface
2. The expert's dictionary interace.

The Undefined Strings Interface When a user inputs a string that is not recognised by any part of the NASCON system it is recorded by the dictionary sub-system for subsequent analysis by an expert. The undefined strings interface allows an expert to look at the undefined strings and add definitions of them into the dictionary.

The Expert's Dictionary Interface This interface allows the expert to enter his own entries into the dictionary.

3.5.3 The Rule Base Update Interface This has yet to be written but it will permit the expert to enter new rules. It may contain various types of error checking including cyclic rule definition error checking. It will ask the expert if the rule conditions are askable or not, or whether the conditions are goals or not. It may deal with the structuring of the rules.

3.5.4 The Information Update Interface This has yet to be written but it will permit the expert to enter new information and assistance files. It will deal with the construction of card information files. It will also deal with the associated title files and information definition files.

4 Conclusions

The provision of an expert system interface for NASTRAN would seem a worthwhile objective. Use can be made of knowledge engineering methodologies in order to achieve this. Whilst a start has been made, it is evident that characterising the process of analysis is complex and the provision of a fully working system will need considerable effort if it is to be other than a simple demonstrator. Emphasis, to date, has been placed on the development of the program structure. The generation of the knowledge base to embrace all of the NASTRAN facilities and the model formulation is in the early stage of development.

Acknowledgment The authors wish to thank the Technical Directorate of GEC Marconi Research for permission to publish this paper.

References

[1] Shortliffe, E. H., (1976), 'MYCIN of Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection,' Computer Based Medical Consultations, America Elsevier.

[2] Bennet, J., Creary, L., Englemore, R. and Melosh, R., (1978), 'SACON A Knowledge Based Consultant for Structural Analysis,' Computer Science Department, Stanford University, September.

A Knowledge Based System for the Diagnosis of the Causes of Cracking in Buildings

I. M. May and W. Tizani
Department of Civil Engineering
University of Bradford
Bradford
United Kingdom

Abstract This paper describes a knowledge based system for the diagnosis of the causes of cracking in buildings. Given a building which has suffered cracking, the system diagnoses the possible causes of this cracking and ranks them in order of likelihood of occurance according to the most likely ones.

The system comprises an inference engine, a situation model and a knowledge base. The 'production rules' form of knowledge representation is used. This was found to be suitable for the particular problem of the diagnosis of cracking in buildings.

The knowledge base comprises rules and meta-rules. Each rule has as its goal a cause of cracking. The meta-rules are used to select, prior to the detailed investigation, those rules which are likely to be applicable for the particular problem.

The system uses a probability technique developed to deal with uncertainty in the problem of the diagnosis of causes of cracking.

1 Introduction

The occurrence of cracking in buildings is one of the major defects in masonry structures. The causes of it are numerous and are often difficult to determine. It is not unknown for two engineers to disagree on the cause of the cracking partly because it is unlikely that many engineers will have experience of all types of cracking. In addition, cracking may result from a combination of causes. The difficulty in identifying the causes of cracking and the cost and effort involved in carrying out the identification may be expensive. However, when the cause of the damage has not been correctly identified this can lead to either unnecessary remedial works at what could be a substantial cost or the delaying of necessary remedial work which might lead to a higher cost later on.

There is currently a tremendous interest in the use of knowledge based systems

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 263–283.

in all branches of engineering. In civil engineering there have been numerous reports on their possible applications [1,2,3]. One major area in which such systems can find immediate application is diagnostic problems. One such application is the diagnosis of the causes of cracking in buildings.

This paper describes a system for the diagnosis of the causes of cracking in buildings. The system uses knowledge about the problem acquired from literature, for example references [4], [5], and [6], and experts. It provides assistance to engineers in assessing existing buildings which have signs of cracking within them. The system is intended not to replace, but to assist, engineers, surveyors, etc., when they are inspecting buildings which show signs of distress.

The system, which is menu driven, is written in PROLOG [7] and mounted on an IBM AT compatible with 640kb RAM and a hard disk. The system comprises three parts namely the knowledge base, the inference engine and the situation model. The knowledge base contains the knowledge in terms of rules about cracking in buildings. The inference engine is used to obtain information from the user which it can then combine with information from the knowledge base in order to arrive at a diagnosis for a particular problem. The current state of the problem is stored in the situation model which is updated as the consultation of the knowledge base and the user, for a particular problem, proceeds.

Several approaches for knowledge representation are described in the artificial intelligence literature [8,9,10]. In the system described the knowledge is represented using production rules because this was determined to be the best way for this particular set of knowledge.

The production rules are of the form,

$$\text{if } (C_1 * C_2 * \cdots * C_n) \text{ then (CONCLUSION).}$$

where C_1 to C_n are conditions and $*$ represents ‘and’ or ‘or’ and CONCLUSION is the conclusion of the production rule. The conditions can be negated, e.g. not C_m. Each of the conditions also contains a ‘certainty directive’ which is used to assess the certainty of a given conclusion.

The PROLOG characteristic of negation by failure, i.e. if a condition in a production rule has failed the conclusion of this production rule will fail, has not been used in the inference mechanism. Instead, if a condition has failed, the rule will not fail but the certainty of the conclusion of this production rule will be modified depending on the ‘certainty directive’ of the failed condition. The rest of the conditions in the rule will then be processed.

One problem with large knowledge bases is that if care is not taken a large amount of time can be wasted by investigating unlikely routes through the knowledge base. In order to overcome this problem a ‘rule-selector’ procedure has been introduced. This procedure selects only those rules in the knowledge base which contain the likely causes. This is done by using ‘meta-rules’ which are rules about rules. Each of the rules has a meta-rule which determines whether that rule is likely to be applicable for the case under investigation.

In any particular session, the inference engine is used to ask questions of the user, retrieve all the necessary data and knowledge from the knowledge base and select the likely causes of the faults being diagnosed. Those causes which have been selected will be investigated one at a time. At the end of the consultation, for each cause selected, a certainty value, which indicates its likelihood as a cause of the cracking, is given. All the likely causes are stored as facts in the situation model, in addition to all the data collected during this session. An explanation of HOW the system has arrived at each cause is available in order that the user can check that the logic for the particular example is correct. Explanation of WHY the system is asking any particular question is also available.

A WHAT-IF facility has been developed which allows the user to investigate the effects of changes of the input data without re-running the entire consultation. This is particularly useful in situations in which a piece of data is not known or only a vague estimate is known and the user wishes to check the sensitivity of the solutions to the data without re-running the entire consultation. This is more fully explained later.

The system deals with uncertainty using an approach which has been developed to suit the problem of diagnosis of the causes of cracking in buildings. The technique uses certainty factors in the range -5 to $+5$ where a factor of -5 means definitely not, a factor of 0 means not known, a factor of $+5$ means definitely yes. Intermediate values denote varying degrees of certainty. A fuller explanation is given later.

Fig. 1 shows the various components in the system in schematic form, a detailed description of various aspects of the system are given later in this paper in the following sections:

2. Production rules.
3. The inference engine.
4. The situation model.
5. The knowledge base.
6. Example run.

2 Production Rules

As mentioned above, for the problem under investigation it was decided that production rules would be best for the representation of the knowledge. The particular form of production rules used in this system are called subrules and they are of the

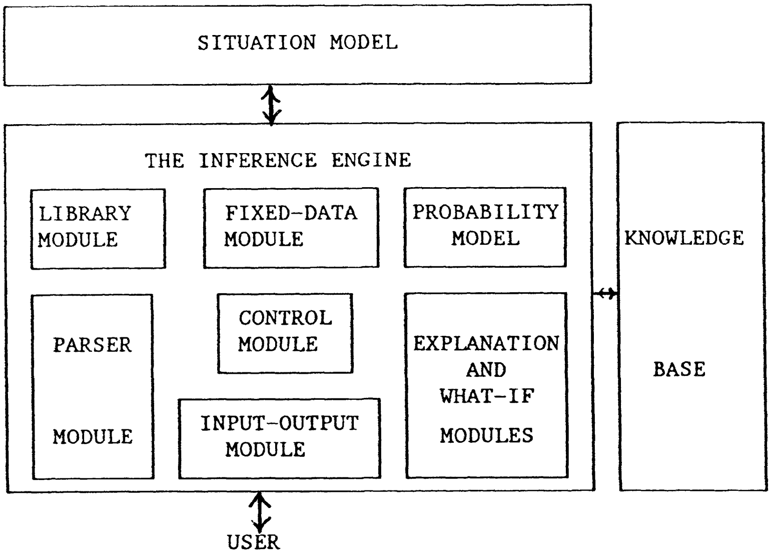


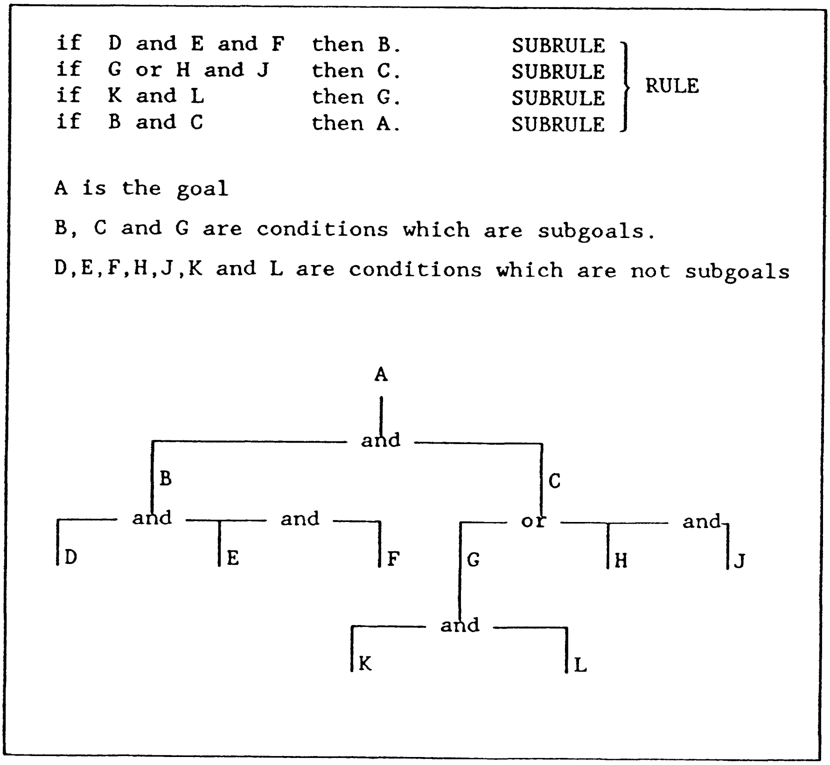
Figure 1. *The system components.*

form 'if PREMISE then CONCLUSION,' where the PREMISE is the part which contains the conditions. For example:

```
if
'cracks appear at a return'   and
'length of return' is less-than '1000 mm'    and
not 'adequate movement joints at returns'
then
'not accommodated thermal movements cause flexure at a
short return.'
                                        subrule (1)
```

where, 'not accommodated thermal movements cause flexure at a short return' is the CONCLUSION and 'cracks appear at a return' and 'length of return' is less-than '1000 mm' and not 'adequate movement joints at returns' is the PREMISE which, in this case, contains three conditions.

A set of subrules which achieves a goal is known as a 'rule.' These subrules are organised into a hierarchy according to the level of their conclusions. At the top of the hierarchy is the subrule which has as its conclusion the goal of the rule. Beneath this level are the subrules which have as conclusions the subgoals which need to be established prior to the goal being established. A subgoal is a condition in the PREMISE of another subrule higher up in the hierarchy of the rule. The hierarchy grows downwards through an unlimited number of levels using subrules, see Fig. 2.



```
if  D and E and F  then B.    SUBRULE ┐
if  G or H and J   then C.    SUBRULE │ RULE
if  K and L        then G.    SUBRULE │
if  B and C        then A.    SUBRULE ┘
```

A is the goal

B, C and G are conditions which are subgoals.

D,E,F,H,J,K and L are conditions which are not subgoals

Figure 2. *An example of a rule hierarchy.*

A subrule within any rule may also be a subrule within another rule, for example see Fig. 3.

In the knowledge base, the goals are the causes of cracking. Each one of these causes is the conclusion of a subrule at the top level of the hierarchy of the rule used to check this cause of cracking. Each of the pieces of evidence which affects the likelihood of this cause are included in the hierarchy of the rule.

An example of a rule which investigates whether the cause of cracking is due to flexure at a short return because of the lack of provision of a satisfactory movement joint, is given is Fig. 4. This follows the formula shown in Fig. 2 and described above.

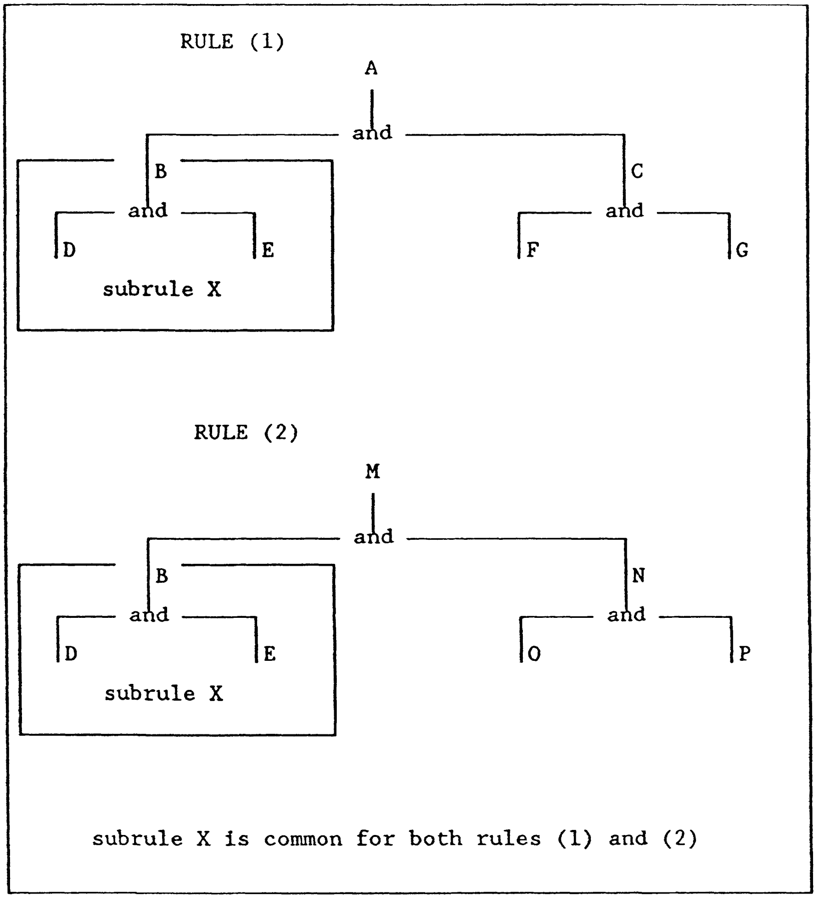
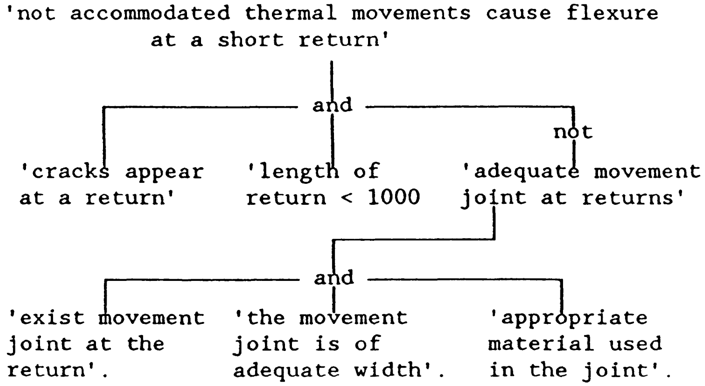


Figure 3. *Common subrules for more than one rule.*

3 The Inference Engine

The inference engine consists of number of modules each of which deals with a particular aspect of the problem using the data input by the user and the knowledge base in the way described below. The inference engine modules are, the 'control,' the 'parser,' the 'explanation,' the 'what-if,' the 'certainty,' the 'input-output,' the 'library' and the 'fixed-data' modules, Figure 1.

```
 if
 'cracks appear at a return'   and
 'length of return' is less-than '1000 mm' and
  not 'adequate movement joints at returns'
then
 'not accommodated thermal movements cause flexure
 at a short return'.
                                       subrule (1)


if
 'exist movement joint at the return'    and
 'the movement joint is of  adequate width'   and
 'appropriate material used in the movement joint'
then
 'adequate movement joint at returns'. subrule (2)



           subrule (1)      }  RULE
           subrule (2)


 'not accommodated thermal movements caused flexure
 at a short return'    is the goal
 'adequate movement joint at returns'  is a subgoal
 The remaining conditions are not subgoals



 The hierarchy is:-
```



'not accommodated thermal movements cause flexure at a short return'

and

not

'cracks appear at a return'

'length of return < 1000

'adequate movement joint at returns'

and

'exist movement joint at the return'.

'the movement joint is of adequate width'.

'appropriate material used in the joint'.

Figure 4. *Example of a rule.*

3.1 Problem Solving Strategy

The strategy used is to ask a series of preliminary questions in order to select only those rules applicable to the case to be analysed from the knowledge base. The selected rules are then processed one at a time. The major aspects of the problem solving strategy are discussed below.

3.1.1 The elimination of non-applicable rules Knowledge bases are normally necessarily large and only some of the rules included are applicable for a particular problem, hence the elimination of non-applicable rules as early as possible in any session is beneficial in terms of the efficiency of the system. As mentioned in Section 1, the selection of applicable rules is achieved by meta-rules. The meta-rules are determined by the writer of the knowledge base who determines a number of parameters which categorise the rules. These parameters will have values which are obtained from the answers to the preliminary questions. The meta-rule for each rule contains the prescribed values of the parameters. When the values of these parameters have been ascertained from input data the meta-rule for each rule is checked to determine whether the rule is to be selected. For example, the rules could be categorised based on the type of structure, the age of the structure, the location of cracking and the cracking pattern. At the start of a session, the values of these parameters are requested from the user. The meta-rules are then consulted, and using these values, the likely causes are selected and will then be further investigated. Subsequently only these rules will be processed. The removal of these unnecessary rules also makes the system appear to the user to behave like an expert by not investigating causes which can not apply to a case under investigation.

3.1.2 Multi Solution Output The rules selected by the system, as described above, are then processed. During the processing some of these rules may indicate the possible causes of the distress. The solution or solutions are output with a degree of certainty for each solution. Thus the causes can be ranked in order of probability of being the likely cause.

3.1.3 Rejecting Selected Rules Rules which have been selected as applicable rules for a particular session may be rejected at a later stage if, after further investigation, they are found to be 'definitely unlikely' to have caused the fault under diagnosis. The rejected rules are divided into two categories.

The first category applies to those rules for which the likelihood of them not being causes of the fault is obvious and for which no explanation for rejecting them is necessary. For example, given the problem of cracking in a free standing wall for which the rule which deals with cracking in free standing walls at short returns has been determined as being possibly applicable, if, however, it is determined that the cracks did not occur at a return, then this rule will be processed no further. No explanation will be given as to why this rule is 'definitely unlikely' to be the cause of cracking in the free standing wall as it is considered obvious.

The second category applies to those rules for which their likelihood of not being the cause of the fault is not as obvious as in the above case. For these rules an explanation for rejecting them is available. For example, if in the example above the cracks appeared at a short return, but the provision of movement joints at the return is adequate, then there must be another cause of cracking other than the inclusion of the short return. The rejection of the rule is announced and an explanation for the reason why this rule is unlikely to give the cause is available for the user.

The inference engine is used to determine if a rule is 'definitely unlikely' at an early stage of the processing of the rule by using 'dependence directives' attached to certain of the conditions in the rule. If a condition which contains a 'dependence directive' has not been satisfied the rule is considered 'definitely unlikely.'

Two dependence directives are used, 'essential' and 'conditional.' If a condition has an 'essential' directive and during a consultation the condition has not been satisfied, the rule is then 'definitely unlikely.' The reason for failure is obvious and the rejected rule falls in to the first category described above. However, if the 'conditional' directive has been used and the rule is rejected, and hence is 'definitely unlikely,' the reason may not be obvious therefore an explanation is given as described above.

3.2 The CONTROL Module

The control module is responsible for the selection of applicable rules and it is from this module that access to other modules is controlled. The control module looks first for the preliminary questions set by the knowledge base writer, the basic-parameters, and then requests input of their values through the input-output module. After determining these values, the control module uses the meta-rules and searches the entire knowledge base for all rules which satisfy the values. The applicable rules are then stored in a list, in the situation model, and are used for the particular session. The control module then invokes the parser module to process each of the rules, one at a time. After processing each rule, the parser module will return it with its certainty factor, or report that it is 'definitely unlikely' to be the cause of the fault. If the rule was 'definitely unlikely' with obvious reason, as category one above, then the control module will automatically invoke the parser module to process the next rule in the list. Otherwise, the results are stored in the situation model, and the input-output module is invoked to output the results for the particular rule. The option of processing the next rule is then given. If the next rule is to be processed, the cycle will be repeated from the point where the parser module is invoked to process a rule. If no further rules are required to be processed, the session will end. A summary of all the results achieved in the session is available. Finally an option to start another consultation is available.

3.3 The PARSER Module

The parser module is responsible for processing the rules when they are supplied by the control module. The control module determines the goal which is to be investigated. The goal is then matched with the conclusions of the subrules in the knowledge base to find the corresponding subrule. This subrule, which will be the subrule at the top of the rule hierarchy, is then retrieved. The subrule is then processed by first dividing it into PREMISE and CONCLUSION parts. Each condition in the PREMISE part is processed and a certainty factor for it is obtained, this is repeated for each condition. The processing of a condition can be by asking input from the user, or by deduction, or, if the condition is a subgoal, by processing the premise part of a further subrule which has the subgoal as its conclusion. In the latter case the certainty factor for the condition is equal to that determined for the subrule. Certainty factors are described in the next section. The certainty factor for each condition of the subrule is then used to determine the certainty factor for the conclusion of the subrule. Similarly, the certainty factor for each condition of the subrule at the top of the rule hierarchy is used to update the overall certainty factor of the goal. All rules which has been discarded at any stage are given a certainty factor of -5. During the processing of a rule, the logical relationships of all the evidence used in order to arrive at the conclusions for each goal are stored in the situation model, in addition to all input data, to enable the user to obtain reasons for the solutions.

3.4 The CERTAINTY Model

Knowledge based systems should be able to deal with uncertainty in a satisfactory manner so as to simulate the decision making skill of a person expert in the domain for which the system is intended. However, the nature of knowledge varies enormously between domains. Therefore the probability approach, or technique, which is to be used for a problem should be chosen carefully in order to model the uncertainty in the given problem most appropriately. Certainty factors, fuzzy logic and Bayesian theories are examples of methods which have been used for knowledge based systems applications [11,12,13]. A sound understanding of the problem in hand is essential for the choice of method to be made in order to ensure that the chosen method is reliable for the particular problem.

The knowledge for the diagnosis of the causes of cracking in buildings is incomplete and sometimes the cause of cracking can not be uniquely determined. Therefore the system uses a probability approach which is suited to this type of knowledge and which can be used to rank the possible causes.

The technique used is an approach developed particularly for the diagnosis of the causes of cracking. A certainty factor is used to indicate the degree of certainty of a piece of evidence. The certainty factor is on the scale of −5 to +5, where a factor of −5 means definitely not, a factor of 0 means not known, a factor of +5 means definitely yes. Intermediate values denote varying degrees of certainty. Every piece of evidence in a production rule is given a certainty factor in the range −5 to 5 depending on the users responses to questions about it. The certainty factor of the conclusion of a subrule is based on the certainty factors and the 'certainty directives' given to each piece of evidence leading to this conclusion. The 'certainty directives' can be considered to be a weighting given to a particular condition which is independent of the certainty factor determined by the input. The updated certainty factor is also in the same range, −5 to +5. The certainty directives and the method of calculating the updated certainty factor is given below.

3.4.1 Certainty Directives The method used for the calculation of the combined certainty factor of conditions has been derived from observing the effects of conditions on the conclusion of a subrule. It was realised that the various conditions will affect the conclusion in different ways. Therefore 'certainty directives' were introduced. Each condition of a subrule has a certainty directive which affects the value of the updated certainty factor for the conclusion of the subrule.

There are six certainty directives namely, 'necessary,' 'ext-supplementary,' 'supplementary,' 'supportive,' 'contradictive,' and 'parameter.' Other directives could be easily added to the certainty model if needed.

They are attached to conditions in the form:

```
condition which-is CERTAINTY DIRECTIVE
```

For example, in subrule (2) given in Fig. 4, the certainty directives are added as following:

```
if
   conditional
   'exist movement joint at the return' which-is necessary
and
   'the movement joint is of adequate width' which-is
   ext-supplementary
and
   'appropriate material used in the movement joint'
   which-is supportive
then
   'adequate movement joint at return.'
```

The necessary directive

When the directive 'necessary' is attached to a condition in the premise of the subrule, the updated certainty factor of the conclusion of this subrule is calculated as follows. If the condition is satisfied with certainty A and the conclusion has a prior certainty B, then the updated certainty factor of the conclusion, C, is given by:

$$C = \text{minimum } (A, B)$$

The certainty directives 'ext-supplementary,' 'supplementary,' 'supportive' and 'contradictive' are listed in order of strength, e.g., when attached to a condition the first will support the conclusion more strongly than the second. For the values of A, B and C as defined above, the certainty directives affect the certainty of the conclusion as follows:

The ext-supplementary directive

The ext-supplementary directive is used for those conditions which support or contradict the conclusion the strongest of the four above directives:

$$C = B + 2 \cdot A/5, \quad \text{and}$$

$$C \leq 5, \quad \text{and}$$

$$C \geq -5.$$

The supplementary directive

The supplementary directive is used for conditions which support or contradict the certainty of the conclusion less strongly than the 'ext-supplementary' directive. C is calculated as follows:

$$C = B + A/5, \quad \text{and}$$

$$C \leq 5, \quad \text{and}$$

$$C \geq -5.$$

The supportive directive

The supportive directive used for those conditions which only support positively the conclusion. C is calculated as follows:

If $A < 0$, then $C = B$.

If $A \geq 0$, then:

$$C = B + A/5, \quad \text{and}$$

$$C \leq 5, \quad \text{and}$$

$$C \geq -5.$$

The contradictive directive

The contradictive directive is the opposite of the supportive directive. C is calculated as follows:

If $A > 0$, then $C = B$.

If $A \geq 0$, then:

$C = B - A/5$, and

$C \leq 5$, and

$C \geq -5$.

The parameter directive

The parameter directive is used for those conditions which do not affect the certainty of the conclusion and for which,

$$C = B.$$

If the order in which the conditions in a rule appear is altered this may affect the probability and so when rules are written care must be taken particularly with the choice of the 'necessary' directive.

3.5 The EXPLANATION Module

The explanation module can be accessed by the user at any time during a consultation. The explanation module generates responses when the user types in HOW, WHY or HELP. These responses are retrieved partly from the knowledge base and partly from the situation model, HELP texts are retrieved which give assistance with the questions, WHY gives the reasons why certain questions are asked, and what they are trying to achieve and HOW explains how a goal has been arrived at. This is achieved by using the logical relationships of the evidence used to arrive at the goal which have been stored in the situation model.

3.6 The WHAT-IF Module

The WHAT-IF module allows the user to change previously entered input and to test the effects of these changes on the diagnosis of the cause of cracking. This module can be accessed whenever the likelihood of a cause of cracking has been diagnosed by the system. Once the module is accessed the system is put in the 'WHAT-IF' mode and a menu which includes all the parameters for the diagnosis is given. The user can then inspect the current value of any of these parameters and change any of the values. If changes are made, the user can then re-run this particular case. The system will then re-process the case and return the cause and certainty based on the new values of the parameters. During the re-run additional data may be requested. As with normal processing the user is able to find HOW the diagnosis has been arrived at. The changes can be made permanent by the user, or the original values restored. At this stage, the user can exit from the WHAT-IF mode or can repeat the process by inspecting, changing parameters and then re-running as many times as required.

This facility has been found to be extremely useful. Of particular note are the following aspects:

- The ability to investigate the effects of changes in the input data on the likelihood of various causes of cracking without having to postpone this investigation until all the causes, with the initial data, have been processed.

- The ability to change input data without having to input all the data again.

- The ability to investigate the effects on the diagnosis if the value of a parameter which is vague or if it is not known, through the user trying alternative values to check the parameter's effect on the final outcome of the diagnosis. This can be of particular use in cases where it may be difficult or expensive to determine accurately the value of the parameter. The analysis can be used to determine if an accurate value is required by checking the sensitivity of the diagnosis to values of the parameters.

- The facility also proved to be very useful when developing the knowledge base. When a rule was added to the knowledge base, the facility was used to test and validate all the paths in the rule quickly and efficiently.

- Finally, the WHAT-IF facility could be used for educational or training purposes.

3.7 The INPUT-OUTPUT Module

The input-output module is the 'user interface.' All input is requested through the module which generates the menus, and then stores input in the situation model. All output is also displayed through this module.

3.8 The LIBRARY Module

The library module contains functions which are used frequently in the system. These functions can be called as required by the various modules.

3.9 The FIXED-DATA Module

The fixed-data module contains standard questions and messages. The standard questions are those asked by the inference engine which are not contained in the knowledge base, e.g., the question 'Would you like to see HOW?' which is asked whenever a result is output. The standard messages are used when errors are detected, e.g., when an unknown input is used the message 'Answer unknown, please try again' is displayed, additionally examples of the expected input are given.

4 The Situation Model

A situation model is created by the inference engine for each session. During a particular session all data and knowledge for that session is held as facts in the situation model. These facts include all input data and all the goals which have been investigated together with the supporting evidence and the logical relationship of the evidence. This information is consulted whenever the user requires explanations of WHY a particular question is being asked or of HOW a particular goal has been arrived at.

5 The Knowledge Base

The knowledge base contains the subrules, information for the selection of applicable rules, and definition for all parameters used.

Every cause of cracking is the goal of a rule, Figure 5, and each rule consists of one or more subrules. Subrules are written in the knowledge base as independent sentences. The order in which the subrules are organized in the knowledge base is irrelevant, Figure 2, i.e., they are written in a declarative form. A subrule can be called from any premise part of any subrule, and hence a subrule can be common to more than one rule when the same subgoal needs to be established, Figure 3.

As mentioned above, the knowledge base contains, in addition to the subrules and the parameters needed for the selection of the likely goals, definitions for all the parameters in the knowledge base. These definitions include text which is used by the explanation module in order to generate help and explanations. Also, they include facts about the parameters defining their type and the values or range of values which can be accepted for input.

6 Example Run

This section gives details of a typical run together with explanations.

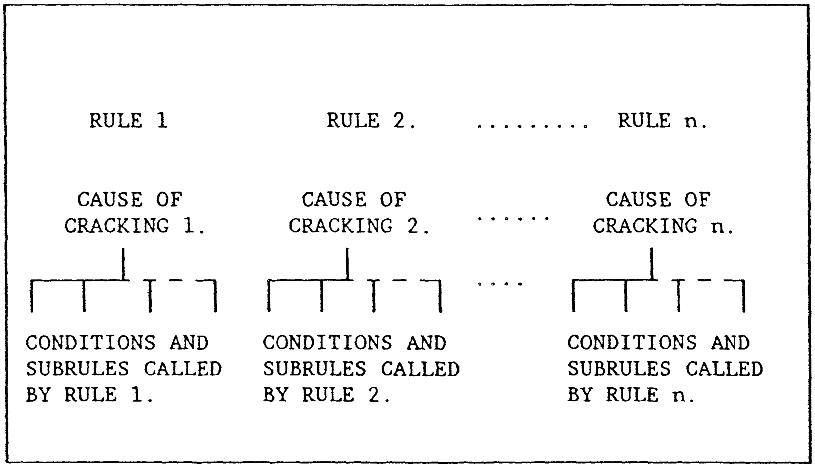

RULE 1
RULE 2.
....
RULE n.
CAUSE OF CRACKING 1.
CAUSE OF CRACKING 2.
......
CAUSE OF CRACKING n.
....
CONDITIONS AND SUBRULES CALLED BY RULE 1.
CONDITIONS AND SUBRULES CALLED BY RULE 2.
CONDITIONS AND SUBRULES CALLED BY RULE n.

Figure 5. *Schematic representation of the rules in the knowledge base.*

The problem considered is a parapet with vertical cracking. When the system is loaded, a series of preliminary questions are asked. In the description below

-   indicates system questions
*   indicates user answers and
**   indicates comments.

```
        -Select the type of structure.
                *  brickwork

** This and subsequent non-numeric answers would be selected from a
   menu.

        -Enter the age of the structure in months.
                *  25

        -Select the distressed member.
                * parapet

        -Select the material used.
                * claybrick
```

** Based on these values a number of rules will be selected, for example:

- Thermal and moisture movement in parapets.
- Flexural cracking at a short return.
- etc.

** The user can control the order in which the rules are to be investigated. Alternatively, the selected rules will be investigated one at a time in the order above. The system displays the goal it is investigating at any time, for example:

```
-INVESTIGATING: Thermal and moisture movement in parapets

        -Are there any vertical movement joints in the parapets?.
                *  yes

        -Enter the spacing of movement joints in metres.
                *  why
  WHY
   I am asking this question in order to know whether
      the spacing of the movement joints is adequate
   This is necessary to establish whether
       adequate movement joints have been used, from:
          movement joint(s) exist and
          movement joint(s) have adequate spacings and
          movement joint(s) have adequate width and
          appropriate joint material(s) have been used
   This is necessary to support or contradict the current
   investigation that the cause of cracking is due to:
      unaccommodated thermal and moisture movement in parapets

        -Enter the spacing of movement joints in metres.
                *  7

        -Enter the width of the movement joint in mm.
                *  10

        -Select the material used as filler in the joint.
                *  help
  HELP
    The material used as filler in the movement joint are those
    which are used to fill the gap between two panels separated
    by a movement joint.  For example:
    polyethelene, polyurethane and foam rubber, can be used
   for expansion joints.  Cork, fibreboard and hemp can be used
   for contraction joints but should never be used for
   expansion joints.       (Ref: BS 5628:part 3)

       -Select the material used as filler in the joint.
               *  polyethelene
 CONCLUSION
  -  It is  LIKELY with certainty factor 3 that
     the cause of cracking is due to:
  unaccommodated thermal and moisture movement in parapets.

-  Would you like to see how?
               *  yes
 HOW
 -  Unaccommodated thermal and moisture movement in parapets
      is  LIKELY was derived by:
  1.   cracking occurs in parapet     and
  2.   cracked material is claybrick     and
  3.   not adequate movement joint is VERY LIKELY
          was derived by:
    3.1   exist movement joints is DEFINITE  and
    3.2   spacing of movement joint is 7  <8 m   and
    3.3   width of movement joint is 10 NOT  2*spacing and
    3.4   adequate joint material is DEFINTE
               was derived by:
        3.4.1 movement joint is expansion joint   and
        3.4.1 joint filler is polyethelene.
```

** If WHAT-IF is selected, and the value of the parameter 'exist movement joint' is changed from 'yes' to 'no.' After re-running the diagnosis the conclusion will be:

```
 CONCLUSION
  -  It is MOST LIKELY with certainty factor 4.5 that
     the cause of cracking is due to:
  unaccommodated thermal and moisture movement in parapets.

-  Would you like to see how?
               *  yes
 HOW
 -  Unaccommodated thermal and moisture movement in parapets
      is  MOST LIKELY was derived by:
  1.   cracking occurs in parapet     and
  2.   cracked material is claybrick     and
  3.   not adequate movement joint is DEFINITE
```

```
          was derived by:
     3.1  exist movement joints is DEFINITELY IMPOSSIBLE.

** The user is now given the options of retaining either the original
   values of the parameters or the new values input in the 'WHAT-IF'
   mode.  If the user had retained the original values on exit from the
   'WHAT-IF' mode the diagnosis continues as below.  Retaining the new
   values the diagnosis would also continue by investigating the next
   cause which could, of course, now be different from the original
   possible causes.

        -Would you like to investigate another likely cause?
                *  yes

-INVESTIGATING: flexural cracking at a short return.
   .........
   ......

** The next likely cause is then investigated and the result is given.
   Investigation of the likely causes will continue until either the
   user has answered NO to the last question or all the selected likely
   causes have been investigated.

** Finally a summary of all the likely causes which have been
   investigated is given, for example:

  SUMMARY
  1- Unaccommodated thermal and moisture movement in parapets
         LIKELY with certainty factor 3.
  2- Parapet is not separated from the vertical brickwork in a
     multi-story building, (vertical expansion of the brickwork
     can cause cracking in parapets)
         LIKELY with certainty factor 2.5.
  3- Flexural cracking in parapets, caused by a short return
         POSSIBLE with certainty factor 0.3.
  4-... etc.

** A report can be printed which will include all the results with the
   details of HOW they have been arrived at.
```

7 Summary and Conclusions

A knowledge based system for the diagnosis of cracking in buildings has been described which produces the most likely causes of cracking in a building ranked in order of likelihood. The main aspects of this system are:

1. It comprises an inference engine, a situation model and a knowledge base. The inference engine may well prove to be capable of handling other similar types of diagnostic problem.

2. It uses the 'production rules' form of knowledge representation which has been found to be suited for this particular diagnostic-type problems.

3. The system uses a probability technique which has been developed to deal with the uncertainties met in the diagnosis of causes of cracking. Using this approach it is possible to rank the causes in order of their likelihood of occurance.

4. The knowledge base includes rules, subrules and meta-rules. Each rule has a goal which is a cause of cracking and is made of one or more subrules. Meta-rules are used to select those rules which are most likely to give the cause of cracking for the particular problem under analysis.

5. A WHAT-IF facility is available which enables the user to investigate easily the effects of changes of the input in order to assess the sensitivity of the given cause of cracking.

6. The system can be used to assess the cause of cracking for a number of cases. The knowledge base can be easily extended to include new causes.

Acknowledgment The authors would like to thank all those who have assisted in the development of the knowledge base in particular Stuart Bell of George Armitage & Sons PLC.

References

[1] Adeli, H. (Ed.), (1988), *Expert Systems in Construction and Structural Engineering*, Chapman and Hall.

[2] Kostem, C. N. and Maher, M. L. (Eds.), (1986), *Expert Systems in Civil Engineering*, American Society of Civil Engineers, New York.

[3] Topping, B. H. V. (Ed.), (1987), *The Application of Artificial Intelligence Techniques to Civil and Structural Engineering*, Civil-Comp Press, Edinburgh.

[4] CIRIA, (1986), *Movement and cracking in long masonry walls*, CIRIA Special Publication 44.

[5] Churchill, W. M., (1982), 'Surveys of movement in calcium silicate brickwork,' *Ceram. Res. Ass.*, March.

[6] BRE, (1981), *Assessment of damage in low-rise buildings*, BRE digest 251, (July).

[7] Expert System International Ltd, (1986), *PROLOG-2 Language Reference Manual*, Expert System International Ltd, Oxford, U.K.

[8] Brachman, R. J. and Lvesque, H. J., (1985), *Readings in Knowledge Representation*, Morgan Kaufman, Los Altos, California.

[9] Hayes-Roth, F., Waterman, D. A. and Lenat, D. (Eds.), (1983), 'An Overview of Expert Systems,' in *Building Expert Systems*, Addison-Wesley, Reading, Massachussetts, U.S.A.

[10] O'Shea, T. and Eisenstadt, M., (1984), *Artificial Intelligence-Tools, Techniques, and Applications*, Harper and Row, New York.

[11] Shortliffe, E., (1976), *Computer based medical consultation: MYCIN*, American Elsevier.

[12] Forsyth, R., (1984), 'Fuzzy reasoning systems,' in *Expert systems principles and case studies*, R. Forsyth (Ed.), Chapman and Hall.

[13] Baldwin, J. F., (1981), 'Fuzzy logic and fuzzy reasoning,' in *Fuzzy reasoning and its applications*, E. M. Mamdano and B. R. Gaines (Eds.), Academic Press, pp. 133–148.

Recent Advances in AI Based Synthesis of Structural Systems

P. Hajela and N. Sangameshwaran
Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida, Gainesville
United States of America

Abstract The present paper describes a general framework for an AI based approach for developing near-optimal designs of load bearing truss and frame structures. The proposed framework is an adaptation of a recently proposed model for generic problem solving system. The approach is based upon a systematic decomposition of the problem space, and an application of design heuristics to incrementally generate an optimal structural topology to account for the applied loads. The principal shortcomings of a heuristic decomposition approach are underscored, and more rational quasi-procedural decomposition methods are proposed.

1 Introduction

The past two decades have witnessed the emergence of optimization methods as viable tools in the structural design process. An extensive body of literature is available that pertains to the use of formal mathematical optimization methods in structural synthesis. A significant majority of these publications are devoted to the subject of member resizing to attain a minimum weight design, with constraints on structural integrity and response. References [1–4] are typical of such efforts. For most such applications, the structural geometry is specified before the optimization sequence is initiated, and this initialization determines, to some extent, the effectiveness of the optimized design. Recent efforts have attempted to focus on the subject of configuration or topology optimization [5–7], albeit from the standpoint of starting with a given design, and varying the geometry shape variables to attain an optimum.

The approach adopted in the present work is one in which the structural geometry is created from a specification of load conditions and available support points in the design space. This is a realistic scenario in a design process. Numerous stud-

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 285–306.
© 1992 *Kluwer Academic Publishers.*

ies have been conducted to gage the thought process of human designers. Studies based on protocol analysis [8–9] seem to suggest that human designers appear to be very skilled at identifying one promising configuration, which is then refined in a set of sequential steps. Other studies lend substance to the belief that creative design is the result of systematic application of simplistic logical processes. The philosophical basis of the present work is to regard the topology definition problem as one for which no unique solution exists. If, however, the topology assignment is done in a sequence of top-down refinement steps, with each step emphasizing the optimality of the partial configuration, the final topology is expected to be near-optimal. In order to implement such an approach, careful consideration must be given to the decomposition of the problem, and the order in which the various design criterion are met in the topology definition problem.

The preliminary structural layout problem described above is most amenable to methods of heuristic design. Conventional mathematical programming techniques have been applied in such problems, but with limited success. The addition and deletion of structural members creates a design space that may contain various degrees of nonconvexities and disjointness; hence the difficulties in using nonlinear programming strategies. Such preliminary layout problems have received some attention but with no emphasis on the optimality of the final design. Shah [10] reports exploratory efforts in the development of expert systems for preliminary form design of structural machine parts. The approach adopted was one in which previously catalogued case studies were used to build the structural form, starting at the primitive level and incrementally adding degrees of complexity. Nevill and his co-workers [11] and Brown [12] also recognize the importance of the preliminary synthesis problem. The research activity in this group has resulted in the development of an automated preliminary design system MOSAIC, currently implemented for 2-D mechanical and structural systems.

One of the major objectives of the present work was to emphasize the need for some degree of formalism in the approach adopted for conceptual design. Most of the reported efforts pertaining to this discipline have been somewhat disjointed, each devoting significant development effort in building a framework that was considered unique by its developer, and necessary for the application at hand. Other than the fact that such systems have common components in the form of a knowledge base and an inference facility, there is very little adherence to a set of common development guidelines. Such a lack of standardization is at least partially attributable to the fact that there is no general agreement on what is regarded as a good model for engineering design practice. Recent efforts of Tong [13] have focussed at attempting to establish some formalism in knowledge-based design.

A brief description of a framework for generative model abstraction, emphasizing the usefulness of algorithmic processing, and deriving from the general approach proposed by Tong, is presented here for completeness. The framework is discussed in greater detail in Refs. [14–15]. An implementation of the process in the domain

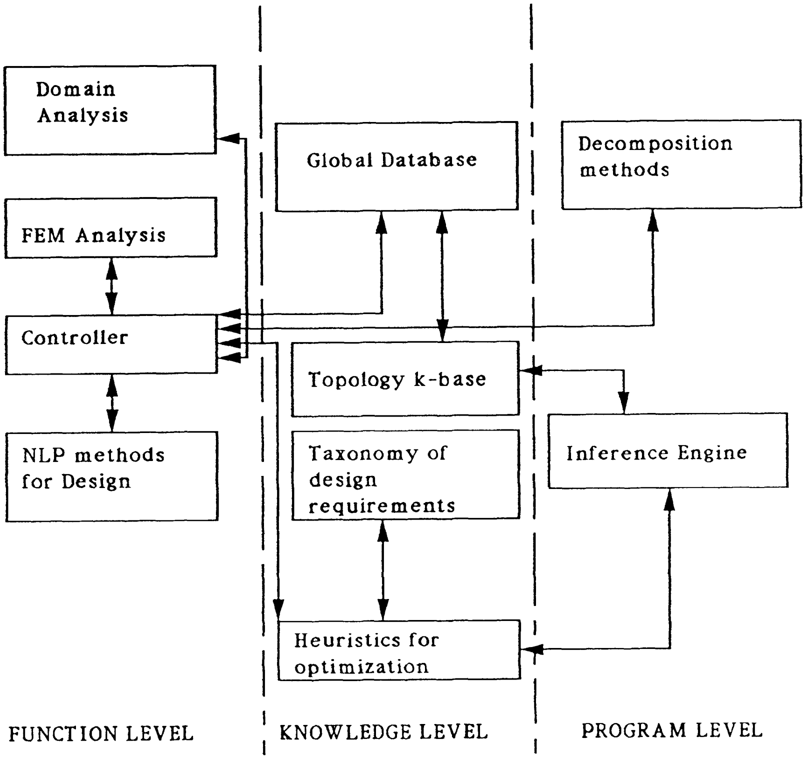


Figure 1. *Organization of design system.*

of optimum structural synthesis is another focus of the present paper.

2 Generic Problem Solving Systems

The process of automated conceptual design proceeds with a synergistic application of design heuristics and domain analysis programs, resulting in a tightly coupled quasi-procedural approach. The organization of a problem solving system to achieve this task determines the overall efficiency of the process. Separation of various components of such a system is important, as it retains a level of modularity in the system which allows for component updates. At the very outset, it is important to recognize three distinct levels at which the automated design process can be organized—the knowledge, function, and program levels. A typical relational arrangement of these levels of organization are shown in Fig. 1.

3 Knowledge Level

A detailed description of the design domain is relegated to this level. This includes all pertinent specifications which define the acceptability of a solution. All necessary and applicable tools for analysis are also made available at this level. Furthermore, the types of design applications envisaged for such systems would very seldom generate designs that bear no resemblance to their predecessors. A possibility exists, therefore, to develop at this level a general taxonomy of problem types, of the most typically encountered design constraints, and of possible solution strategies. The domains that must be considered in this exercise include geometric modeling, structural analysis, and optimization methodology.

In creating a taxonomy of designs based on problem specifications, one is essentially identifying abstract features that result from an imposition of such requirements. Structural design requirements may be classified on the basis of strength, stiffness, elastic stability, degree of redundancy, types of support conditions, dynamic behavior, and a requirement of least weight or least complexity in manufacturability. Clearly, each of these requirements has an influence on the design that distinguishes it from designs dominated by other requirements. As an example, a structure that is governed by structural stability requirements will be dominated by elements that can withstand compressive loads, and further, such elements will typically have appropriate aspect ratio and stiffness properties. However, in as far as possible, it is advantageous to relate one abstract feature with one problem requirement. Failing this, the classification must clearly indicate the relative contribution of a requirement to a salient characteristic.

Clearly, the information available at the knowledge level determines the class of problems for which solutions can be obtained. The coupled algorithmic and heuristic process can be computationally demanding in some situations. It is in such cases that a taxonomy of designs based on problem requirements is particularly useful. It is conceivable to abstract partial designs in a primitive form, where such primitives adequately model the displacement field of more refined models, but may have considerably fewer structural elements. These primitive forms may be retained in the analysis to a point where it becomes necessary to introduce greater refinement in order to meet a specific class of design requirements. One can think of a macroelement that has specific stiffness characteristics to satisfy displacement requirements. Such a macroelement can then be defined in terms of component elements with the same aggregate stiffness, and which account for strength requirements.

4 Function Level

Perhaps the most significant feature of this level of organization is a controller which directs the flow of the conceptual design process. The controller is essentially an embodiment of the problem solving strategies that may be invoked for the problem at hand. The design specifications handed down from the knowledge level are attempted to be satisfied at the function level. Although designs can be generated by considering all requirements simultaneously, this methodology is not considered appropriate for the task at hand. Instead, a more natural process of refinement in steps is adopted, wherein the problem is decomposed into smaller, more tractable, and preferably, single goal problems. The underlying principle in such a refinement is that the solution space is more likely to be unique in the presence of a higher degree of specification detail.

In a problem where the design philosophy is one of sequential refinement to satisfy an ordered set of goals, there is a need for evaluating the proposed partial concepts for acceptance. Such testing operators are made available at the function level, and simply interact with the available tools at the knowledge level to recover the pertinent information. The failure of a proposed concept to meet the acceptability test is generally followed by alternative design moves. Such moves are facilitated by the initial decomposition of the problem by design specifications, and which allows for construction of tree-like deduction paths.

Finally, the controller must have the option of modifying the design rules, particularly if it assists in realizing the design specifications. The acceptability tests can themselves be relaxed to admit designs. Yet another option available at this stage is to extend the design by adding features which enable it to pass the acceptability requirements. It is important to understand that for any given task, there may be several transfers of control to the program and knowledge level as deemed appropriate by the controller at the function level. This transfer of control is repetitively invoked for each of the component tasks.

5 Program Level

At this level, no problem solving knowledge is explicitly available. However, the implementation of all design steps on the basis of information received from other levels, is carried out at this level. In addition, tasks of data management, programming procedures, and production rules are assigned to this level. The database management capabilities of such systems are of particular importance, as significant amount of data is generated and must be managed for a design system to work efficiently. This is even more crucial as large amounts of algorithmically generated numerical data must be stored and post-processed to use meaningfully in the iterative process.

Two levels of data management were implemented in the current system. At the core level is a global data base that records information for long term usage. Problem and subproblem related databases are extracted from the main system and are local to the knowledge level. Such an approach provides a convenient blackboard for constraint posting and propagation as the design is taken through a process of incremental refinement.

The inference facility is perhaps the most significant feature at this level. In the structural design problem that is implemented in the present work, a rule-based, C Language Integrated Production Systems (CLIPS) [16] is used. This utility can be invoked from within a Fortran program, making available a convenient link between algorithmic and heuristic processing of information.

6 Optimal Structural Synthesis

The procedure described in the preceding sections was implemented in an automated design system for the generation of near-optimal, two dimensional structural configurations. The problem required the development of both the structural form and member cross section dimensioning to carry a prescribed set of concentrated and distributed loads and moments, with the least structural weight, and constraints on allowable displacements, fundamental frequency, and structural strength. In keeping with the proposed framework of developing generic problem solving systems, the various tools necessary for the task were assembled into each of the three levels of organization.

The knowledge base comprises an important segment at the knowledge level, and for the problem under consideration, the design heuristics can be classified into two broad categories. These categories correspond to the design heuristics for topology generation and those for member resizing. The former deals with the actual definition of the structural topology under given load conditions and for specified supports. The latter is a heuristic implementation of constrained nonlinear optimization to determine optimal member cross sections, once the topology is defined.

7 Knowledge Base for Topology Generation

The definition of the design domain is placed into the knowledge level. In particular, this includes specification of the magnitude, type, and orientation of the loads with respect to the supports in some reference axes system. The type and location of support points are also part of such a database. This extended database is utilized to provide all pertinent data to the knowledge base, and on the basis of which various heuristic actions are invoked to generate the structural form.

In the present implementation, the controller at the function level initiates the process by reading in as input, the total number of loads and the magnitude and location of each load in order of increasing distance from the origin in the x coordinate direction. Corresponding to each of these loads, the support numbers and support types are read in order of increasing distances from the loads.

The synthesis itself is conducted in a sequential manner, with the design heuristics attempting to stabilize one load location at a time. The x and y components of the load are read. If the applied load happens to be a moment, one design decision becomes immediately obvious. The element that transfers this load to a support must allow for rotational degrees of freedom at its node. Hence, a beam element is selected to transfer this load. To stabilize this load and to maintain a minimum weight, the nearest support is examined. If this support is of clamp type, then a beam element of nominal cross sectional dimensions is chosen. In case, the support is not a clamp, then the next nearest support is considered. If this support is a clamp, then a beam element is chosen between this support and the load. If not, the load is connected to both the supports using two beam elements. A further level of refinement is possible by checking to see if the third nearest support is a clamp, and if so assess the weight penalty of connecting to this support against the use of two beam elements to the two nearest supports.

This strategy is also used in the event that at a given point, both applied forces and moments are present. For the situation where the loads are only applied forces with nonzero x and/or y components, the orientations of the loads stored in the database are used. The angle between the force resultant and the hypothetical element connecting the point of load application and support, is computed. If this angle is less than 20 degrees, then the load is considered to have a large axial component, and an axial rod element is preferred.

The other possible scenario is one in which a large transverse component of the load exists in addition to an axial load. The orientation of the load resultant with respect to the next nearest support is considered. In the event that the load can be largely considered axial for such an element, an axial rod element is introduced between the load and this support. In the event that this condition is also violated, the first two supports are considered. If the nearest support is a clamp, then a beam element is chosen between the load and support. If not, then a beam element is chosen between the load and the second support, provided it is a clamp. Otherwise, two beam elements are chosen between the load and the two supports. Multiple loads acting at a point can simply be resolved into the x and y components, and handled as above. Likewise, distributed loads could be present as part of the design problem. In such situations, equivalent end loads and moments are computed by assuming the distributed load to act on a beam segment, and these loads and moments are connected to supports as explained in the foregoing discussion.

Once a load is stabilized by the process described above, the point of application of the load becomes available as a support point. This is physically an elastic support but may be modeled as a rigid support point. The controller uses the above heuristics until all the loads in the input have been accounted for and are stabilized. Alternate topologies are generated and considered for analysis when the initial topology fails to satisfy any of the structure behavioral constraints, and for these, a specific order is assumed. The displacement constraints are first met, followed by frequency, and finally the stress constraints. The process is quasi-procedural, wherein heuristics suggest a design which is then verified by an algorithmic process.

The analysis of the structure is performed using a general purpose finite element program EAL [17]. The actual connectivity between the algorithmic and heuristic process is described in a later section. Displacement constraints are first checked for feasibility, the failure of which invokes a set of heuristics designed to obtain constraint satisfaction. These heuristics are based on incremental changes in the structure designed to incur the least weight penalty. An alternate structure is generally proposed by addition of a member at the node location for which the displacement constraint was violated. The member is connected to nearest support point to which it was not previously connected. This process is repeated for a satisfaction of the frequency and stress constraints by invoking a similar set of heuristics.

In the event that after all possible connections the current constraint is still violated, the controller uses a backtracking strategy to return to the previous constraint and look for the next best configuration acceptable for that constraint. This design is then passed to the lower level for refinement and satisfaction of the failed constraint. A feasible structure may or may not result from the exercise. In the event of an infeasible design, one may either discontinue the exercise or attempt constraint satisfaction by varying member cross sectional properties. The latter would be attempted in any event as a logical step for weight minimization. In the present work, the option exists of either using a strictly procedural, nonlinear programming based optimization algorithm [18], or a heuristic implementation of such a scheme.

8 Heuristic Optimization

Once an initial topology has been generated, the next task in the optimal design process is the resizing of the cross sections of the structural members. In the present work, an attempt was made to reduce extensive numerical evaluations by incorporating some heuristics in the optimization procedure. The objective of this work was to simulate the search process of a constrained minimization scheme, but to reduce the computational effort by making gross heuristic assumptions regarding the suitability of a chosen search direction.

The sensitivities of the objective and constraint functions with respect to the design variables were first evaluated. Likewise, the initial values of the normalized constraints and the objective function were obtained as an assessment of the starting design. These were evaluated on the basis of a finite element analysis. The section areas and moment of inertias were the design variables for the problem.

The design variables were ordered on the basis of the objective function sensitivities, with the first corresponding to the one that gave the best improvement in the objective function. The controller first attempted to satisfy any violated constraints, by changing the design variable that resulted in the least weight penalty. The actual step length was determined on the basis of a piecewise linearization of the violated constraint about the initial point.

Once an initially feasible design was established, the objective function and constraint sensitivities were recomputed and ordered as before. While some of the constraints may be critical, others could be satisfied by large buffer margins, holding out hope for further decrease in the objective function. The design variable with the maximum possible improvement in the objective function was then chosen. Constraint gradients were checked to see if a change in this design variable could be accommodated without violating a constraint. If the piecewise linearization indicates the possibility of an infinite move in this direction, then a 25% change in the design variable was allowed. Otherwise, the constraint closest to critical for a move in this direction was checked to see if at least a 10% improvement was possible before violation. If affirmative, then the allowable change was affected. Failure of this check resulted in examination of the next best available move direction. Note that if an unbounded move is suggested for a design variable in more than one iteration, it indicates a member that can be removed from the structure, provided it does not render the structure unstable. Deletion of members is a difficult step in strictly procedural optimization.

9 Representation of Design Heuristics

All the design heurstics were represented in the form of rules, written in a format required by the inference environment of CLIPS. Both an interactive mode of execution and one which is similar to an embedded expert system format, are permitted in this environment. Initially, all illustrative examples were run interactively. This provides the opportunity to view the execution process, of editing pertinent information, and the ability to maintain a record of the status of the system. A typical usage involves creating a rules and facts file, followed by the execution of CLIPS which in turn loads the rules and facts file.

The other mode of execution is the use of CLIPS as an embedded expert system. In this mode, the CLIPS is executed from a FORTRAN program, preceded by the loading of the rules file. A basic feature in the CLIPS environment is that rules and data are two separate entities. The inference engine uses a forward chaining algorithm to apply the knowledge (rules) to the data.

In a basic execution cycle, the knowledge base is examined to see if conditions of any rule are satisfied. This is done by simple pattern matching of the data with the conditions. If the fields of the data which are asserted as variables, match the conditions of a rule, then that rule is activated for firing. In case of more than one rule match, all the matched rules are activated and pushed onto a stack. The most recently activated rule has the lowest priority and is put at the bottom of the stack. The top rule is selected and executed. As a result, new rules may be activated and some previously activated rules may be deactivated. This cycle is recursive until no further pattern matches are possible.

A general rule syntax in CLIPS is given as follows:

```
(defrule <name> ["comment"]
     (<first pattern>)
     [(    ..    )
           ..              ;LHS

           ..
      (<nth pattern>)]
=>
     (<first action>)
     [(    ..    )         ;RHS
      (    ..    )
      (<nth action>)])
```

The LHS consists of one or more patterns which form the condition(s) to be satisfied, before the rule can be fired. An implicit 'and' is present in the LHS if more than one pattern is present. The RHS of the rule is a list of one or more action(s) to be performed as a result of the firing of the rule. An explanation of a typical rule for topology generation is presented in Appendix A. The rule base also includes a few meta-rules. These are also treated as rules in the inference environment, each of which is a superset of two or more rules. The advantage of using meta rules is the efficiency of concatenating task specific rules. Furthermore, a better control over the evaluation of rules is possible.

10 Integration of CLIPS with Algorithmic Procedures

As described in an earlier section, the input to the design process includes a specification of loads, type and location of supports, and specification by coordinates of any forbidden zones in the design domain. Using this input, a FORTRAN module extracted all relevant data such as lengths, any possible non-permissible members, angles subtended etc., to obtain a quantitative enumeration of the design domain. One of the major tasks for this module included the creation of a facts file. This file contained the description about each load, as shown below.

```
(load#, load location, x component, y component, moment,
support#, type of support, support location, distance, angle,
(next nearest support information).....  )
```

A CLIPS batch file was created which uses the facts file, and the rules file to generate a topology. The topology generated was stored in a data file in a two dimensional array. The array format is shown below.

member	load#	support/load#	type
1	1	3	b
2	1	2	–
.	.	.	.
.	.	.	.

Once the topology was generated so as to stabilize all loads, the completed structure was available for finite element analysis to check satisfaction of other constraints and possible modification of topology as described in an earlier section. In each of these incremental modifications to obtain constraint satisfaction, the controller takes the output of CLIPS topology generation, and creates an input runstream for the EAL finite element analysis program. Upon execution of this program, the necessary constraint and objective function information is read and returned in the proper format for it to be read as a CLIPS input file. It should be very clear from the foregoing discussion that there is a significant transfer of information between various levels of organization in such an integrated design system.

11 Implementation in Structural Synthesis

The specific organization of design tools in the three level approach embraced in the present work is shown in a schematic sketch in Fig. 1. At the function level, the topology analysis module is responsible for developing a table of pertinent features for the loads and supports. Two other modules at the function level were programs for finite element analysis, and a nonlinear programming-based optimization program. The latter is available to perform procedural optimization on a given geometry in lieu of the heuristic optimization. Also present at this level is a controller which directs the flow of the synthesis process. In the present work, the controller assumes a heuristic decomposition of the problem, and addresses the various design requirements in that order. A more rational approach of using dynamic programming concepts for this purpose is presently under study. As stated earlier, no problem solving knowledge is introduced at the program level. However, the inference facility which uses information from the knowledge and function level to suggest new design moves, is available at this stage.

A global database is central to the knowledge level organization. Both domain and design specifications are resident in this database as are results obtained from any enumeration of the design space during the process of sequential refinement. Local databases are created from this global database for use in both algorithmic and heuristic processing in the knowledge and program levels. The creation of these local databases is closely linked to passing of control from one level to another.

The approach presented in the preceding sections was applied to the generation of optimal topologies for two dimensional structural systems. Two examples are included in this paper. The first deals with structure synthesis for the loads and supports shown in Fig. 2. The design domain contains both concentrated loads and moments, as well as forbidden zones in which no element may be introduced. Figs. 3a–c demonstrate the incremental refinement as the structural geometry is completed and then optimized for minimum weight. The final cross sections of the beam and truss members are shown in Fig. 3d. The process of heuristic refinement of the cross sections reduced the structural weight by 19.44%. A second design domain is shown in Fig. 4, where both concentrated and distributed loads are defined. The final geometry and cross sectional dimensions are shown in Fig. 5. In this problem, the weight before and after the heuristic refinement of cross sections was 18.9 lbs and 14.3 lbs, respectively. A step-by-step execution of the design procedure is summarized in Appendix B.

12 Closing Remarks

The paper presents an artificial intelligence based approach for the optimum synthesis of structural systems. The thrust of the present effort is in the establishment of a general framework for such automated design systems, and its adaptation in the structural design domain. The problem approached in the present work is one of optimal topology generation, followed by heuristic refinement of cross sectional dimensions for minimum weight. The approach used is best described as quasi-procedural, in which significant algorithmic processing is used to assist in enumeration of the design space, and to invoke pertinent design heuristics. A heuristic element is necessary to overcome problems related to disjointness in the design space. The extensions to this work are related to developing more rational decomposition strategies and to extend the synthesis to account for uncertainty in load and support definitions. At a more fundamental level, there is a need for extensive research in obtaining abstract representations of partial designs for overall computational efficiency. The latter is especially applicable in more realistic structural applications.

References

[1] Johnson, E. H., (1976), 'Optimization of Structures Undergoing Harmonic or Sochastic Excitation,' SUDAAR No. 501, Stanford University.

[2] Sensburg, O., Fullhas, K. and Schimdinger, G., (1988), 'Interdisciplinary Design of Aircraft Structures for Minimum Weight,' AIAA Paper No. 88-2302, (April).

[3] Hajela, P., (1986), 'Geometric Programming Strategies for Large-Scale Structural Synthesis,' *AIAA Journal* **24**, 7.

[4] Hajela, P., (1987), 'The Gauss Constrained Method for Optimum Structural Synthesis,' *Computers and Structures* **27**, 6.

[5] Hajela, P. and Jih, J., (1988), 'Boundary Element Methods in Optimal Shape Design—An Integrated Approach,' *IUTAM Symposium on Structural Optimization*, Melbourne, Australia, February, Structural Optimization, G. I. N. Rozvany and B. L. Karihaloo (Eds.), Kluwer Academic Publishers.

[6] Braibant, V. and Fleury, C., (1985), 'An Approximation-Concept Approach To Shape Optimal Design,' *Computer Methods in Applied Mechanics And Engineering* **53**, 119–148.

[7] Choi, J. H. and Kwak, B. M., (1988), 'Boundary Integral Equation Method for Shape Optimization of Elastic Structures,' *International Journal for Numerical Methods in Engineering* **26**, 1579–1595.

[8] Ericsson, K. A. and Simon, H. A., (1984), *Protocol Analysis: Verbal Reports as Data*, MIT Press.

[9] Stauffer, L. A., Ullman, D. G. and Dietterich, T. G., (1987), 'Protocol Analysis of Mechanical Engineering Design,' *Proceedings of the International Conference on Engineering Design*, Vol. 2, Boston, MA.

[10] Shah, J. J. and Pandit, L., (1986), 'DEZINER—An Expert System for Conceptual Form Design of Structural Parts,' *Proceedings of the ASME Computers in Engineering Conference*, Vol. 1, July.

[11] Nevill, G. E., Jr., Jackson, L. A, and Clinton, J. H., (1988), 'Automated Hierarchical Planning for Structural Design,' *Proceedings of the ASME Computers in Engineering Conference*, August.

[12] Brown, J. P., (1988), 'Managing Interacting Design Subproblems in Incremental Preliminary Design,' M.S. Thesis, Department of Aerospace Engineering, Mechanics and Engineering Sciences, University of Florida, August.

[13] Tong, C., (1987), 'Toward an Engineering Science of Knowledge-Based Design,' *Artificial Intelligence in Engineering* **2**, 3, 133–166.

[14] Hajela, P., (1988), New Directions for AI Methods in Optimum Design, *Proceedings of the NASA/Air Force Symposium on Multidisciplinary Analysis and Optimization*, NASA CP-3031, September.

[15] Hajela, P. and Sangameshwaran, N., (1990), 'A Coupled Algorithmic and Heuristic Approach for Optimal Structural Topology Generation,' to be published in *Computers and Structures*.

[16] *CLIPS Reference Manual*, (1988), JSC-22948, NASA Johnson Space Center, April.

[17] Whetstone, D., (1977), *SPAR—Reference Manual*, NASA CR-145098-1, February.

[18] Vanderplaats, G. N., (1984), 'An Efficient Feasible Direction Algorithm for Design Synthesis ,' *AIAA Journal* **22**, 11, 1633–1640, (October).

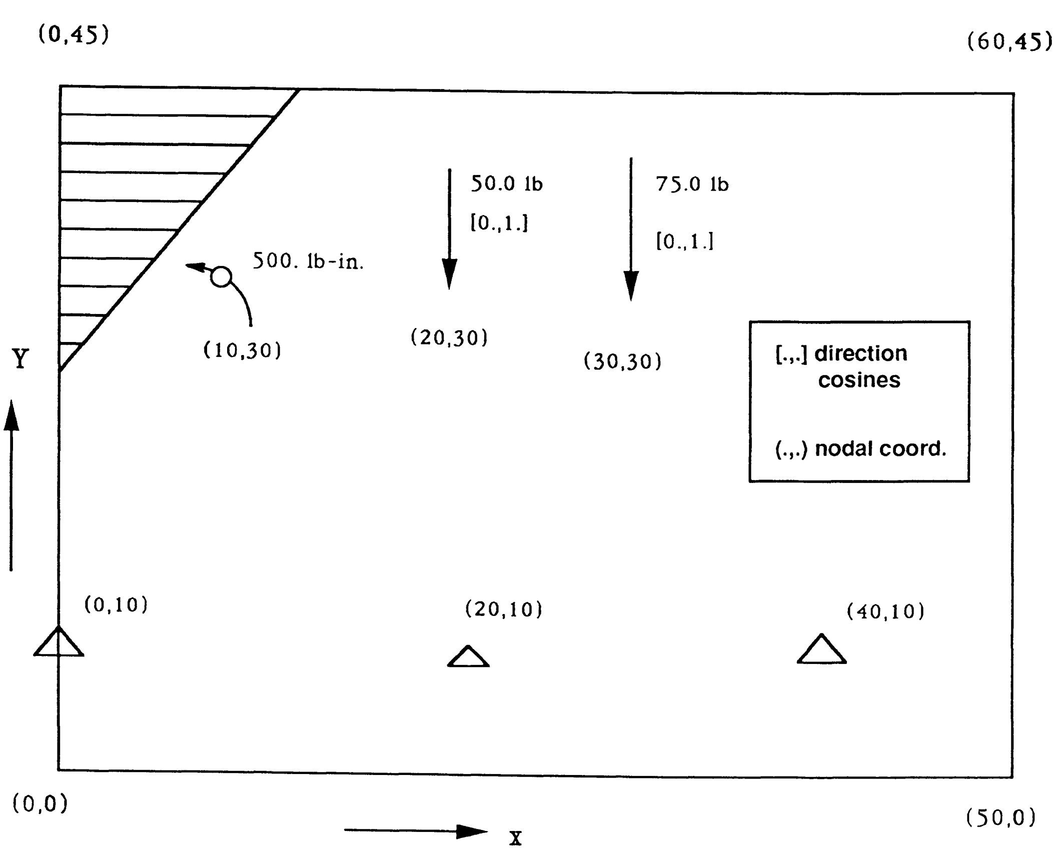
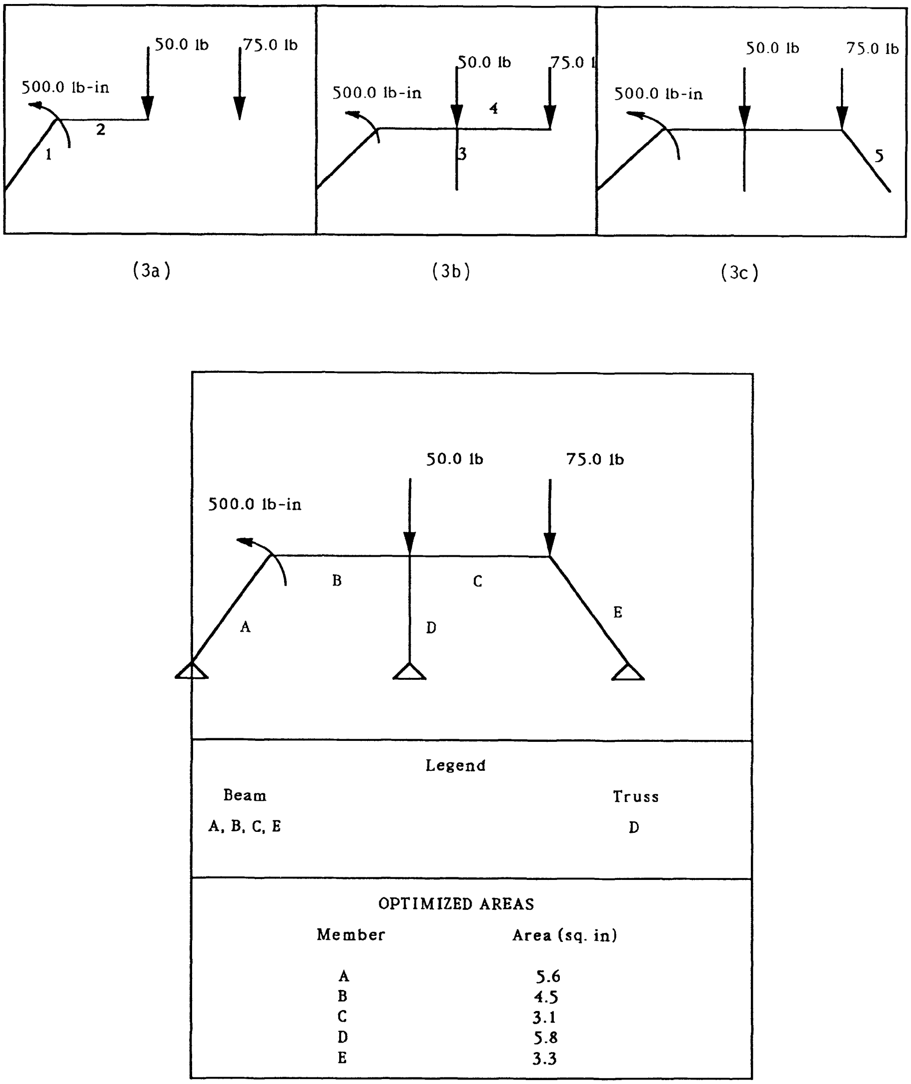


Figure 2. *Design domain with concentrated loads and moments.*



(3a)

(3b)

(3c)

Legend

Beam	Truss
A, B, C, E	D

OPTIMIZED AREAS

Member	Area (sq. in)
A	5.6
B	4.5
C	3.1
D	5.8
E	3.3

(3d)

Figure 3. *Heuristic generation of structural topology and member cross sectional properties.*

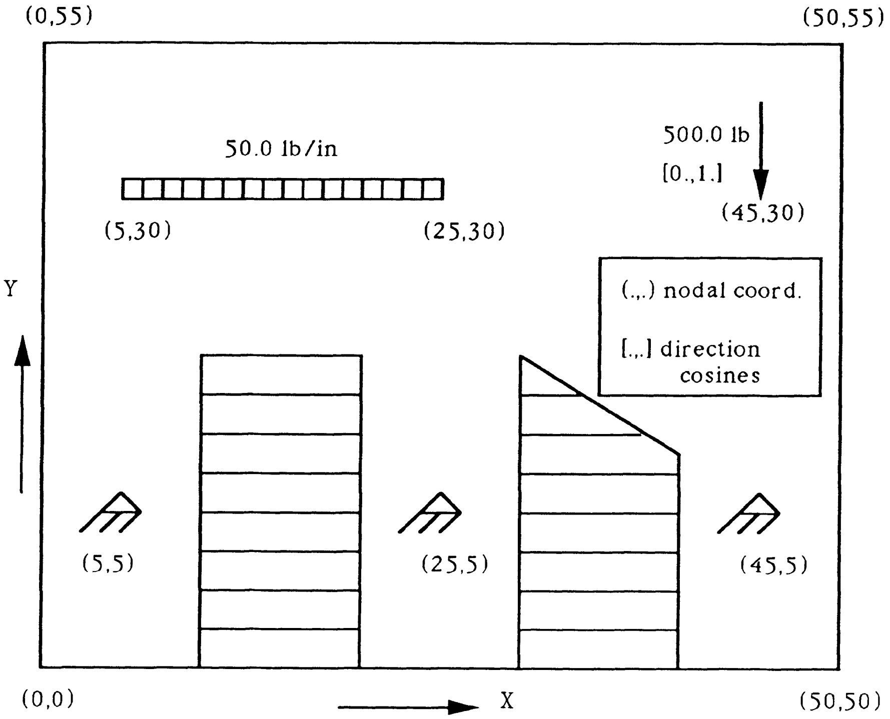
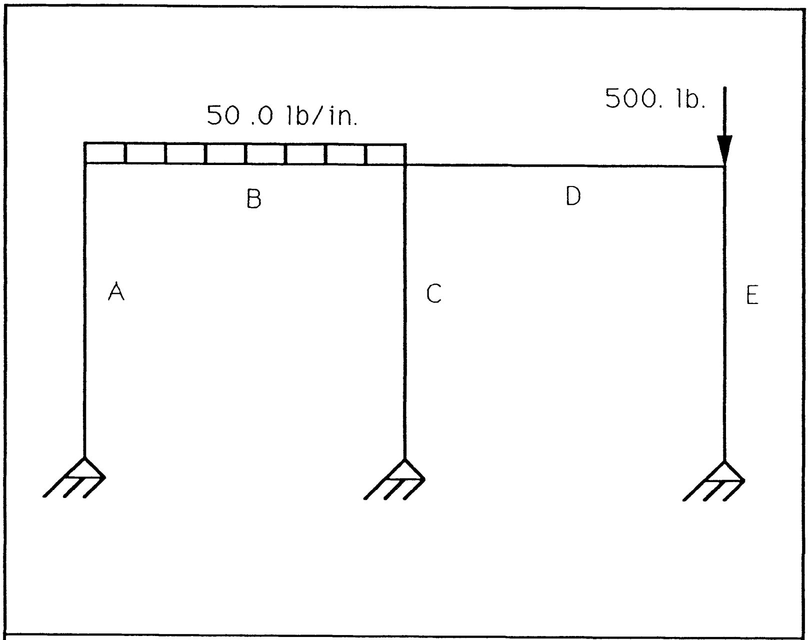


Figure 4. *Design domain with forbidden zones and a mix of distributed and concentrated loads.*



Legend

Beam	Truss
A, B, C, D	E

OPTIMIZED AREAS

Member	Area (sq.in.)
A	18.9
B	17.56
C	38.46
D	13.0
E	6.0

Figure 5. *Final topology and cross sections for design domain shown in Figure 4.*

APPENDIX A

Typical Rule for Topology Generation

A typical rule in the knowledgebase for topology generation is shown. A brief comment is provided following each statement.

```
(defrule rule6                          $name of rule
?k <-(new-id ?g $?a)                    $variable g,array a,
                                        statement k
     (?num)                             $variable num, this is
                                        the load#
=>
     (bind ?a (nth 7 $?a))              $7th value in array a is
                                        variable a,
                                        nearest support#
     (bind ?b (nth 8 $?a))              $8th value in array a is
                                        variable b, support type
     (bind ?c (nth 14 $?a))             $second nearest support#
     (bind ?d (nth 15 $?a))             $support type

     (if (= ?b 1)                       $if first support is
                                        clamp
          then
               (bind ?f 1)              $member # 1
               (format topo "%4d %4d %4d b %n" ?f ?num ?a)
                                        $writing the member#,
                                        load#, support# and beam
                                        type element to an
                                        output file

          else
               (if (= ?d 1)             $check if second support
                                        is clamp
               then
                    (format topo "%4d %4d %4d b %n" ?f ?num ?c)
                                        $write the member#,
                                        load#, support# and beam
                                        type to an output file

               else
(format topo "%4d %4d %4d b %n" ?f ?num ?a)
(format topo "%4d %4d %4d b %n" ?f ?num ?c)))
                    $write the member#,
                    load#, support# and beam
                    type elements to connect
                    the load to both the
                    supports, to an output
                    file.
```

APPENDIX B

Execution of Design Process

A detailed description of the execution process of the design process is documented below.

- Create an input data file. This file contains information in the following order.

```
# of loads
# of supports
# of distributed loads
# of forbidden areas
coordinates of loads
direction cosines of loads
magnitudes of loads
coordinates and type of supports
coordinates of the end nodes and magnitudes of
distributed loads
coordinates of the vertices of all forbidden areas
```

  The constraints are given in a separate file.

- Run the design domain evaluation module. This creates an extended database and a facts file of the form shown below.

```
(deffacts fact1
(new-id gen1 load#, load location, x component of load,
y component of load, moment, support#, type, angle,
distance, ......))
```

- Execute the topology knowledge base in the CLIPS environment. The procedure includes executing CLIPS, loading facts and rules files, and executing the rules. The output for the example shown in Fig. 2 is given below

```
1  1  4   b
2  1  2   b to nearest load
3  2  5   t
4  2  3   b to nearest load
5  3  6   b
```

The columns refer to member #, load #, support #/load #, member type respectively.

- Using the above output file, an EAL runstream is created to perform an analysis of the structure. The displacement values, first fundamental frequency and the stress values are extracted from the analysis.

- The displacement values are compared to the constraint allowables. If the constraint is satisfied, then frequency constraint is checked. Otherwise the topology is refined heuristically.

- An EAL runstream is created and the structure analyzed.

- The frequency values are verified for constraint satisfaction. If violated, then the topology is refined heuristically until the constraint is satisfied.

- The stress constraints are verified. If any violation is present then topology is modified by heuristics.

- The successful structure is analyzed using EAL. Each design variable is perturbed by 3% of its design value and the sensitivities of the constraints and the objective with respect to the design variable is evaluated. Design is updated on the basis of sensitivities to a feasible point. The sensitivities are recomputed.

- Heuristic optimization takes place in CLIPS environment. CLIPS is executed followed by the loading and execution of the rules file. The output file format is a $2 \times n$ array, with the design variables in the first column and its numerical value in the second column.

Representing the Ground

D. G. Toll
School of Engineering and Applied Science
University of Durham
Durham
United Kingdom

Abstract The representation of soils and rocks poses a particular problem for knowledge based systems (KBS) in geotechnical engineering. Geological materials are highly variable and considerably more complex than the man-made materials used in other branches of civil engineering. The knowledge base, and also a site specific database of ground conditions (which a KBS will need to access in order to make judgements about engineering problems) can be represented at a variety of levels of complexity. These will range from broad geological classifications, through detailed soil descriptions to quantitative parameters. A knowledge representation scheme is put forward which is able to deal with such a range of complexities.

A geotechnical KBS also needs the ability to reason about trends and variations across a site, using only localised observations at a limited number of locations. This ability to construct a visualisation of the ground conditions will be one of the key features of a useful geotechnical system.

1 Introduction

Toll (1990) has argued that for geotechnical systems to find acceptance among practising engineers, the criteria must be:

1. Intelligent enough to interact with an experienced engineer.
2. Can deal with both qualitative and quantitative data.
3. Can deal with uncertainty and imprecision.
4. Can construct a visualisation of the ground conditions.
5. Has a user-friendly natural language type dialogue.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 307–319.
© 1992 *Kluwer Academic Publishers.*

Systems are under development at Durham University which are aiming to meet these requirements. For a system to be 'intelligent' enough to interact with an experienced engineer it will need to be able to reason at all levels of knowledge. The structuring of the knowledge will therefore be a crucial element in achieving these aims. Another important feature will be the ability to reason about the changes in strata which occur between the points at which ground conditions have actually been observed. The ways in which these aspects are being tackled are discussed.

2 Knowledge Representation

In order to reason about engineering solutions a geotechnical expert system needs two main types of knowledge; *Geotechnical* and *Structural.* The geotechnical knowledge is that which represents the soils or rocks. The structural knowledge represents the type of construction. Both types of knowledge can be represented at different levels of complexity. To be able to operate at the appropriate level a system must be able to respond to different types of input data, and to use a model appropriate to that data type. Typical levels of geotechnical data are listed below with the appropriate responses.

Input Data	Response
Design brief (no geotechnical data)	Rules of thumb
Geological maps etc	Qualitative empiricism
Field data on soil/rock types	Semi-quantitative empiricism
Insitu field test data, laboratory classification tests	Quantitative empiricism
Material properties (Laboratory measurements)	Simplified theory
Interpreted fundamental properties	Theory

Rules of thumb are those which represent 'common sense.' They will, of course, be sweeping generalisations and may often not take all factors into account. They will therefore have a relatively low degree of confidence associated with them. *Empiricism* is knowledge which has little basis in theory, and which forms the core of engineering judgement. The term *Qualitative* is here used to refer to purely descriptive terms which have no quantitative significance. For example, the descriptive term 'Brickearth' has no immediate numerical association and would be described as qualitative. *Semi-quantitative* is used for more specific terminology, still descriptive, which has some quantitative association. For example, 'Stiff' is associated with a defined range of soil strengths or 'Clay' represents a defined range of particle sizes. *Quantitative* means that the knowledge is numerically based. *Simplified theory* represents methods of reasoning which have some basis in theory, even though empirical 'fudge factors' may well be incorporated. *Theory* represents the most fundamental level of knowledge, based on a deep understanding of soil or rock behaviour.

An example of how these sub-divisions operate is given below for a foundation problem. The example is, of course, highly oversimplified.

Rule of thumb:

If structure is house
  then foundation problems are unlikely.
If ground is rock
  then foundation problems are unlikely.

Qualitative empiricism:

If structure is office block
and ground is alluvium
  then foundation problems are likely.

Semi-quantitative empiricism:

If structure height is tall
and soil strength is soft
  then foundation risk is high.

Quantitative empiricism:

If bearing pressure is less than 75kN/m^2
and shear strength of soil is greater than 75kN/m^2
  then foundation risk is low.

Simplified theory:

If bearing pressure is P kN/m^2
and shear strength of soil is C kN/m^2
  then factor of safety $= 6.2 \times C/P$.

The degree of detail of the input data increases going down the list. So does the degree of detail, and of confidence, in the conclusion. Rules of thumb and qualitative empiricism will provide qualitative conclusions. Semi-quantitative and quantitative empiricism will be able to give semi-quantitative responses. Simplified theory and theory will give a quantitative result.

Rules of thumb and qualitative empiricism can be implemented relatively easily using production rules. Semi-quantitative and quantitative empiricial knowledge currently exist largely in the form of design charts and tables. This type of knowledge can best be implemented using look-up tables, which will often need more than two dimensions. Simplified theory and theory are best implemented using algorithmic routines.

By adopting such a structured system, it is possible to operate at all stages of a construction project, starting from the initial feasibility stage, when only vague descriptive data is known, through to detailed design using quantitative parameters. The majority of geotechnical design is carried out at the simplified theory/quantitative empiricism levels. A full theoretical approach using numerical modelling is rarely used in geotechnical design, except on very prestigous projects.

Often different types of knowledge need to be mixed. For instance, detailed quantitative knowledge of the structure, such as the bearing pressure, may be known but only qualitative knowledge of the soil may be available. Rather than develop mixed inference rules which can handle both data types, it is preferable to provide cross-over rules which can convert between data types, as this will avoid duplication of knowledge. Moving up the data structure (e.g. converting from quantitative to semi-quantitative or to qualitative) can be achieved with no loss in confidence. In the majority of cases the data structure will be filled in from the top down, so the less detailed data will already be available, and no inference is needed. Moving down the structure will involve either providing additional data, or alternatively assessing or guessing values. This can be done using established empirical rules for data assessment, or by using a knowledge base of 'typical' values.

3 Data Structures

For a KBS to be able to reason about a geotechnical problem it will need to have site specific knowledge of the ground conditions. The amount of data which will be available may be large (if an investigation has been carried out) and for this reason data input is best kept separate from the consultation process. To allow flexibility, systems must be able to accept data input during consultation, but should also provide easy database input and modification facilities as a way of informing the system prior to consultation.

Geotechnical databases have been implemented for site investigation data (e.g., Cripps 1978, Day *et al.* 1983). Use can be made of existing databases, although some structures are too restrictive in the style and amount of information to be very useful. A data structure which can handle geological data and interpreted fundamental properties, as well as the factual observed data, is given in Fig. 1. The structuring of the data is shown with reference to both the sources of information (Field investigation, Laboratory testing and Data interpretation), and also to the level within the knowledge representation scheme. The different components of the data are described below.

Profile: The sequence of ground conditions (with depth) at a single location within the site.

Geological Horizon: A specified layer within a geological sequence.

Layer: A specific soil/rock type within a geological horizon.

Samples: A discrete mass of material taken from the ground during field investigations.

Insitu tests: Tests which are carried out in situ during the field investigations.

Classification tests: Simple laboratory tests which help to identify soil/rock types.

Material Properties: Direct measurements of properties in the laboratory.

Fundamental Behaviour: The parameters of material behaviour interpreted within a fundamental framework.

Groundwater: Observations of ground water levels during, and after, the field investigation.

4 Visualisation of Ground Conditions

Geotechnical data, whether from a site investigation or from geological maps, is always incomplete. Observations of the ground conditions from borings or trial pits only gives information about discrete points which may be hundreds of metres apart. Geological maps only provide information on surface geology. The skill of the geotechnical expert lies in the ability to assimilate such different types of information, and to construct an 3-D visualisation of the ground conditions. This ability is based on an understanding of geological structures and processes.

Attempts have been made to develop interpolation techniques for site investigation data (Day *et al.* 1987). These simply correlate strata on the basis of the major soil type. Pattern recognition techniques have been used in petroleum exploration (Bois 1982, Wu and Nyland 1987). However expert knowledge of geological concepts must be incorporated with these techniques before a satisfactory system can be developed. Miller (1987) describes a system which embodies the geological concepts of tectonics, depositional and lithological sequences, but this has been developed for location of petroleum resources, not for engineering.

Fig. 2 shows some examples where changes in strata thickness may be observed in adjacent profiles. A particular horizon may simply thin or die out between the two profiles. However, a change in thickness may be due to the horizon being cut, either by the ground surface; an unconformity (an old eroded surface which has subsequently been overlain by younger deposits); a discordant body (a rock body, such as an igneous intrusion, which cuts through an organised sequence of rock layers); or a fault (a fracture with some relative displacement on either side of the fracture). The ability to recognise these different cases will be vital to understanding the ground conditions.

In addition to changes in thickness, different geological processes may result in changes in strata level. Fig. 3 shows some examples where changes in level can result from dipping, folding or faulting. Again the ability to distinguish between the three cases shown is essential for interpreting the ground conditions, and yet this cannot be done purely on the basis of strata levels in the observed profiles. Additional information on dip angles of the bedding planes or surface observations is required to differentiate between the different cases.

Some simple rules for changes in strata thickness are given in Appendix A, implemented in the form of a PROLOG program. Such ideas need expanding to introduce the concepts of stratigraphy. In the example this is based on a simple numeric code to identify a layer, and layers are assumed to be discordant if the numeric sequence is not continuous. Imprecision will also need to be dealt with; strata levels and thicknesses may often fluctuate within small ranges, without this indicating any significant change in ground conditions. Perhaps most importantly, multiple profiles will need to be considered, rather than looking at trends only between two profiles.

5 Conclusions

Knowledge based systems for geotechnical engineering need to be able to reason with all levels of information, from vague geological classifications through to detailed quantitative parameters. The knowledge base must be structured to reflect these different levels. This allows the system to accept a wide range of types of input data, and to respond to it at the appropriate level.

A key feature of a geotechnical KBS will be the ability to reason about trends and variations across a site. Geological concepts must be incorporated which can assist with inference of the ground conditions between the points at which observations are being made. Profiles observed during a site investigation need to be combined with other types of observation, such as surface mapping, to create a complete visualisation of the ground conditions.

References

Bois, P., (1982), 'Some comments on the application of pattern recognition to oil and gas exploration,' *Geoexploration* **20**, No. 1/2.

Cripps, J. C., (1978), 'Computer storage of geotechnical data for use during urban development,' *Bull. Int. Ass. Eng. Geol.* **19**, 290–299.

Day, R. B., Tucker, E. V. and Wood, L. A., (1983), 'The computer as an interactive geotechnical data bank and analysis tool,' *Proc. Geol. Assoc.* **94**, 2, 123–132.

Day, R. B., Tucker, E. V. and Wood, L. A., (1987), 'A quantified approach to the lithostratigraphic correlation of site investigation borehole logs,' *Computers and Geosciences* **13**, 2, 161–184.

Miller, B. M., (1987), 'The muPETROL expert system for classifying world sedimentary basins,' *Geol. Surv. Bull. (US)*, 1810, 87pp.

Toll, D. G., (1990), 'Do Geotechnical Engineers need Expert Systems?,' *Ground Engineering* **23**, 3, 32-36.

Wu, X. and Nyland, E., (1987), 'Automated stratigraphic interpretaion of well-log data,' *Geophysics* **52**, 12, 1665–1676.

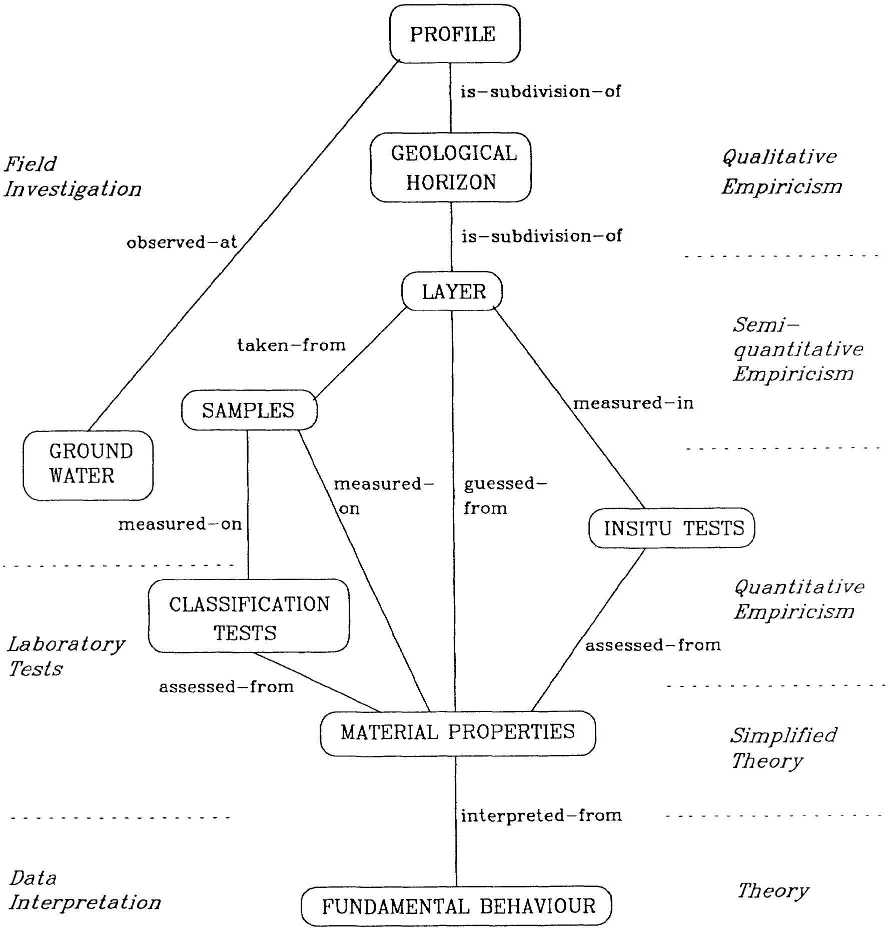
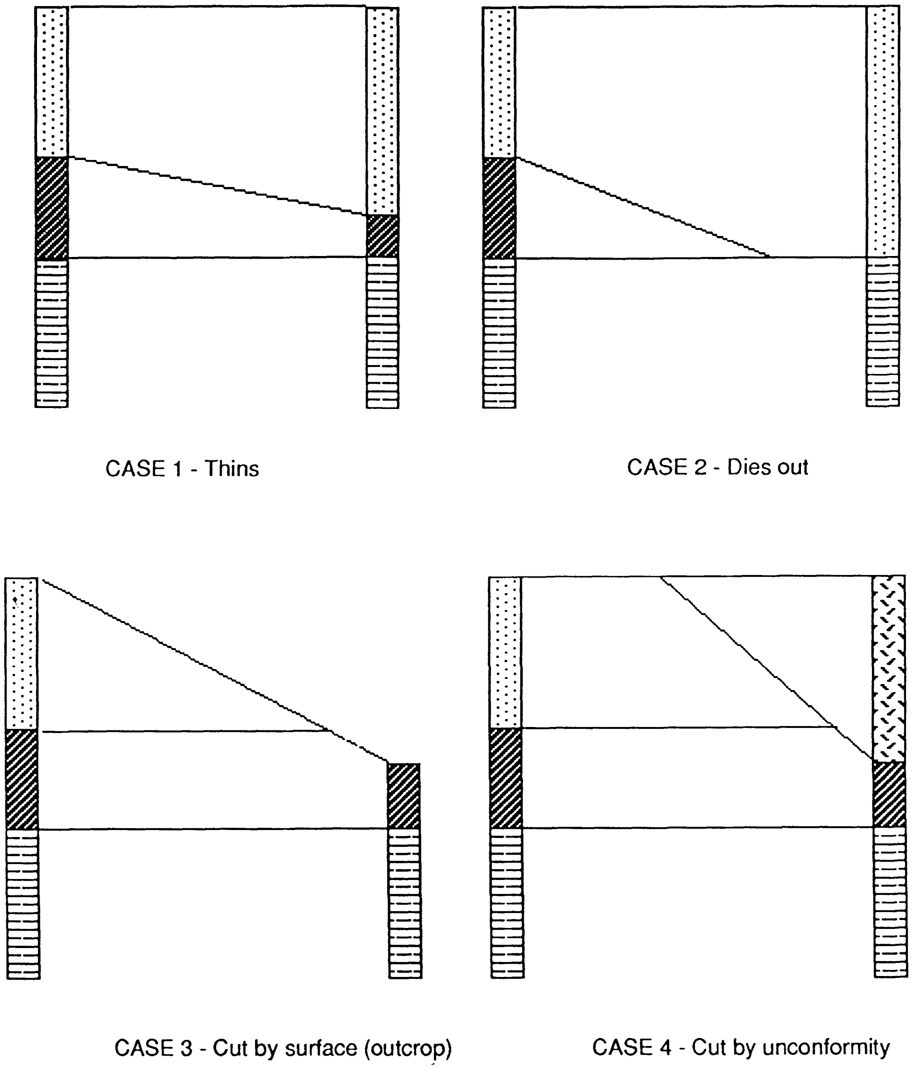


Figure 1. *A Geotechnical knowledge representation system.*



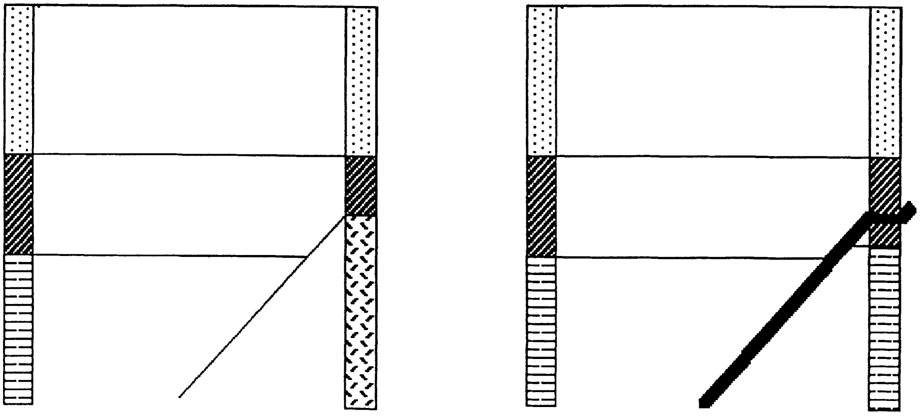
CASE 1 - Thins

CASE 2 - Dies out

CASE 3 - Cut by surface (outcrop)

CASE 4 - Cut by unconformity

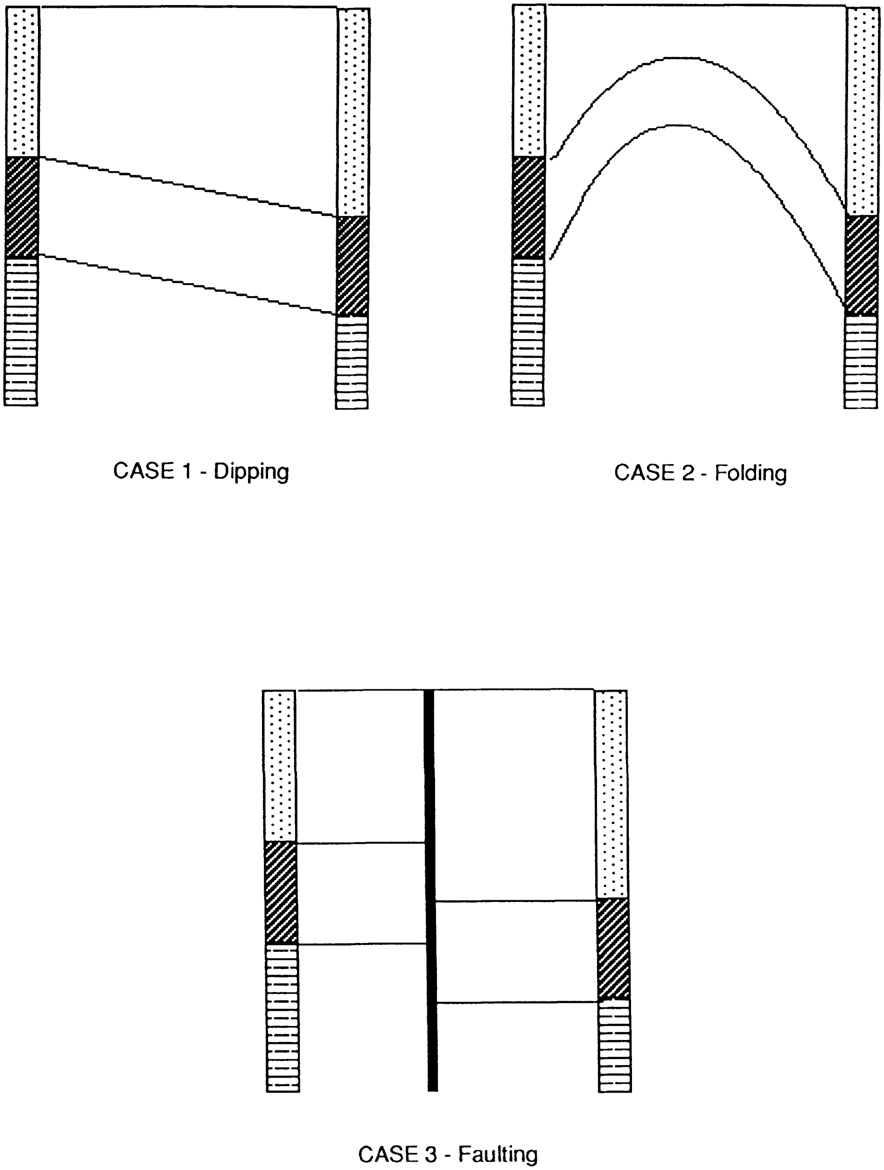
Figure 2. *Changes in strata thickness.*



CASE 5 - Cut by discordant body

CASE 6 - Cut by fault

Figure 2. (continued) *Changes in strata thickness.*



CASE 1 - Dipping

CASE 2 - Folding

CASE 3 - Faulting

Figure 3. *Changes in strata level.*

APPENDIX A

A PROLOG program which identifies stratigraphic trends between two profiles

```
trend :-
  write("Enter layer code "),nl,
   readint(Layer),
  get_levels(Layer,Top1,Base1,Top2,Base2),
   thickness(Top1,Base1,Thick1),
   thickness(Top2,Base2,Thick2),
  change_thickness(Layer,Thick1,Thick2,Top1,Top2,Cutoff,Thick_trend,Profile),
   write("Layer ",Layer," ",Thick_trend),
   write_profile("at",Profile),
  change_level(Cutoff,Top1,Base1,Top2,Base2,Level_trend,Slope),
   write(" and ",Level_trend),
   write_profile("towards",Slope),nl.

write_profile(_,Profile):-
  Profile=0.
write_profile(String,Profile) :-
  write(" ",String," profile ",Profile).

change_thickness(_,Thick1,Thick2,_,_,"","has constant thickness",0):-
  Thick1=Thick2.
change_thickness(_,_,Thick2,_,_,"","dies out",2):-
  Thick2=0.
change_thickness(_,Thick1,_,_,_,"","dies out",1):-
  Thick1=0.
change_thickness(_,Thick1,Thick2,Top1,Top2,"top","is cut by ground surface",Profile):-
  compare_thick(Profile,Thick1,Thick2,Top1,Top2,Top),
  get_ground(Profile,Ground),
  Ground=Top.
change_thickness(Layer,Thick1,Thick2,Top1,Top2,"top","is cut by unconformity",Profile):-
  compare_thick(Profile,Thick1,Thick2,Top1,Top2,_),
  get_layer(Layer,Profile,"above","no_fault").
change_thickness(Layer,Thick1,Thick2,Top1,Top2,"base","is cut by discordant
body",Profile):-
  compare_thick(Profile,Thick1,Thick2,Top1,Top2,_),
  get_layer(Layer,Profile,"below","no_fault").
change_thickness(Layer,Thick1,Thick2,Top1,Top2,"fault","is cut by fault",Profile):-
  compare_thick(Profile,Thick1,Thick2,Top1,Top2,_),
  get_layer(Layer,Profile,"","fault").
change_thickness(_,Thick1,Thick2,Top1,Top2,"","thins",Profile):-
  compare_thick(Profile,Thick1,Thick2,Top1,Top2,_).

change_level("",Top1,Base1,Top2,Base2,"is horizontal",0):-
  Top1=Top2,
  Base1=Base2.
change_level("base",Top1,_,Top2,_,"is horizontal",0):-
  Top1=Top2.
change_level("top",_,Base1,_,Base2,"is horizontal",0):-
  Base1=Base2.
change_level("",Top1,Base1,Top2,Base2,"top slopes down",Profile):-
  compare_level(Top1,Top2,Profile),
  Base1=Base2.
change_level("",Top1,Base1,Top2,Base2,"base slopes down",Profile):-
  compare_level(Base1,Base2,Profile),
  Top1=Top2.

change_level("base",Top1,_,Top2,_,"dips",Profile):-
  compare_level(Top1,Top2,Profile).
change_level("top",_,Base1,_,Base2,"dips",Profile):-
  compare_level(Base1,Base2,Profile).
change_level("",Top1,Base1,Top2,Base2,"dips",Profile):-
  compare_level(Top1,Top2,Profile),
  compare_level(Base1,Base2,Profile).
change_level("fault",_,_,_,_,"dip cannot be inferred",0).

get_levels(Layer,Top1,Base1,Top2,Base2) :-
  write("Profile 1"),nl,
  levels(Layer,Top1,Base1),
  write("Profile 2"),nl,
  levels(Layer,Top2,Base2).

levels(Layer,Top_level,Base_level) :-
  write("Enter top level for layer code ",Layer," : "),
  readreal(Top_level),
  write("Enter base_level for layer code ",Layer," : "),
  readreal(Base_level).

thickness(Top_level,Base_level,Thick) :-
  Thick=Top_level-Base_level.

get_ground(Profile,Ground):-
  write("Enter ground level for profile ",Profile," : "),
  readreal(Ground).

get_layer(Layer,Profile,"above","no_fault"):-
  write("Enter code for layer above layer code ",Layer," in profile ",Profile," : "),
  readint(Layer_got),
  not(Layer_got=Layer-1).
get_layer(Layer,Profile,"below","no_fault"):-
  write("Enter code for layer below layer code ",Layer," in profile ",Profile," : "),
  readint(Layer_got),
  not(Layer_got=Layer+1).
get_layer(Layer,Profile,_,"fault"):-
  write("Does a fault exist in layer ",Layer," at profile ",Profile," (yes/no) : "),
  readln(Answer),
  Answer="yes".

compare_thick(Profile,Thick1,Thick2,Top1,_,Top):-
  Profile=1,
  Thick2>Thick1,
  Top=Top1.
compare_thick(Profile,Thick1,Thick2,_,Top2,Top):-
  Profile=2,
  Thick1>Thick2,
  Top=Top2.

compare_level(Level1,Level2,Profile):-
  Profile=1,
  Level1<Level2.
compare_level(Level1,Level2,Profile):-
  Profile=2,
  Level2<Level1.
```

The Inductive System: A New Tool in Civil Engineering [1]

Tomasz Arciszewski
Civil Engineering Department
Wayne State University
Detroit, Michigan
United States of America

Abstract An inductive system is a computer program that uses learning from examples to conduct a qualitative data analysis: the extraction of decision rules from examples, determination of redundant attributes, and the analysis of relationships between different groups of attributes. Five potential civil engineering applications of inductive systems are discussed in this paper, including the extraction of decision rules from examples, problem-solving, shallow modelling, learning about a domain, and learning expert systems. Examples of individual applications are given. These were obtained using a class of experimental inductive systems based on the theory of rough sets.

1 Introduction

The inductive system is a new tool for qualitative data analysis which can be used for different purposes in civil engineering. An inductive system is understood here as a computer program that uses learning from examples to conduct a qualitative data analysis, including the extraction of decision rules from examples, determination of redundant attributes, and analysis of relationships between different groups of attributes. From from the civil engineering point of view, an inductive system can be considered as a black box, as a new tool which can be used for different purposes in knowledge acquisition and decision making. This new tool has significant advantages over human experts. Humans are very good at deduction, at using available general knowledge for dealing with individual problems. However, we have very limited inductive abilities. (By induction, we mean the development of general knowledge from examples.) Humans can handle only a very limited

[1] This paper is an updated and extended version of an invited paper, 'Inductive Learning in Civil Engineering,' published in *Proceedings of International Colloquium on Expert Systems in Civil Engineering*, Bergamo, Italy, October, 1989.

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 321–332.
© 1992 *Kluwer Academic Publishers.*

number of examples and attributes of a problem at the same time. Sillen [17] describes a human brain as a computer. He notes that an average human can handle only seven attributes and seven examples at one time, while a computer can deal with large numbers of both attributes and examples, limited only by the available working memory. This comparison clearly explains why computers are better than humans at learning from examples.

Inductive systems have already been used in the space industry for the extraction of decision rules from examples to be used in an expert system [14], and for different industrial problem-solving applications [17]. However, in civil engineering, applications of inductive systems are still mostly experimental in character [3,5,6,19].

The approach to computer learning from examples must be different in civil engineering than in computer science. Computer scientists are interested only in the internal workings of an inductive system. As civil engineers, we want to know the potential applications of inductive systems, and we want to know how to use different types of inductive systems. For these reasons, an engineering methodology of inductive learning has been developed at Wayne State University [4,7]. This methodology deals with the process of using inductive systems in knowledge acquisition, with applications for different civil engineering purposes. This work is intended to close the present gap between engineering and computer learning, and to stimulate engineering applications of inductive systems.

The engineering methodology of inductive learning is defined as a subarea of computer learning dealing with the process of inductive learning from the user's point of view. In the proposed methodology, the following three components have been distinguished:

1. the inductive learning process,

2. selection of examples,

3. control criteria.

Its initial outline is given, in [7]. This methodology was prepared for engineering applications, and should be useful for anyone interested in the practical application of inductive systems.

Our research indicates that an inductive system can be used for various civil engineering purposes. At least five possible applications of inductive systems in civil engineering have been distinguished:

1. Extraction of decision rules from examples for application in rule-driven expert systems.

2. Inductive problem solving.

3. Inductive shallow modeling.

4. Learning about a given domain through the process of gradually extracting decision rules from examples.

5. Learning expert systems for engineering applications, for example for conceptual design or for control.

These potential applications are discussed and examples of individual applications given. These were obtained using a class of experimental inductive systems based on the theory of rough sets and developed by Voytech, Inc. of Regina, Canada. The experimental examples were developed in the Intelligent Computers Center of Wayne State University's Civil Engineering Department.

2 Extraction of Decision Rules from Examples

This application of inductive systems is the best known, and it requires only a very few comments. It is well known that knowledge acquisition is a bottleneck in the development of many expert systems. Knowledge elicitation from domain experts is usually very time-consuming and requires the involvement of high-priced knowledge engineers. The process of knowledge elicitation is particularly difficult in all cases where decision rules are complex and are based on many attributes. In such cases, traditional methods of knowledge acquisition are ineffective, and only very rarely are the expected results obtained on schedule and within budget.

The application of an inductive system can change this situation drastically. Very complex decision rules can be generated, involving a large number of attributes. Traditional development of such rules would be very difficult, if not impossible.

The developed methodology of inductive learning [7] can be used to guide engineers through the process of extracting decision rules from examples. This methodology is currently available, and inductive systems can now be used as expert system building tools.

3 Inductive Problem Solving

Inductive problem solving is a process of extracting decision rules from examples to find one or several decision rules which are crucial to solving a problem. Its potential applications are much broader and more interesting than the simple extraction of decision rules from examples to be used in an expert system. Inductive problem solving can be considered also for immediate application, and should be particularly attractive to all civil engineers dealing with complex problems.

Inductive problem solving can be used as a supplementary technique in knowledge acquisition when traditional knowledge elicitation has not yielded the desired results. One or two complex decision rules cannot be identified by human experts, and an inductive system has to be used to find these rules. Such situations often occur when dealing with a problem which cannot be solved because of its complexity. A number of decision rules governing such a problem may be known, but there is still one, or several, rules missing. This rule, is the missing link, and it is crucial to the solution of the problem. The missing link cannot be found using traditional methods of analysis, because of the large number of examples, the large number of attributes, or both.

The limitations of human working memory have been mentioned above. These limitations explain why human experts are very bad at dealing with such problems. The application of an inductive system can improve the situation immediately. All examples may be analyzed by an inductive system, and all decision rules, including our missing link, may be found immediately.

There are known engineering applications of inductive problem solving; Novacast of Sweden has a very impressive record here. For example, this company has used inductive systems for solving complex problems related to the production of margarine and the determination of its melting point. Another successful application was the determination of the factors causing cracks in welds in off-shore drilling platforms [17].

A simple problem from the area of quality control in the manufacturing of steel beams will illustrate the concept of inductive problem solving. The problem is described by several attributes, including the conclusion, which represents the quality of a steel beam. These attributes and their values are given in Table 1.

Table 1. *Manufacturing of steel beams: attributes and their values.*

		Attribute values		
No.	Attributes	1	2	3
1	Type of stiffener	Standard	Experimental	
2	Type of welds	Fillet	Double fillet	Groove
3	Welder's experience	Low	Average	High
4	Humidity	Low	Normal	High
5	Temperature	Below average	Average	Above average
6	Product quality	Good	Bad	

It was noted that in some cases the quality of the beams was bad. Unfortunately, human experts could not find the reason. An inductive system was used to analyze all available 22 examples, based on 5 attributes, given in Table 2.

The inductive system immediately extracted a rule which provides the solution to the problem. This rule is given below:

When:

A1 = 2, stiffener is experimental,
A3 = 1, welder's experience is low,
A4 = 3, humidity is high,
A5 = 3, temperature is high,

Then:

A6 = 2, the product is faulty.

This problem is relatively simple, but it shows the potential applications of inductive problem solving.

4 Inductive Shallow Modeling

Inductive shallow modeling is a process of building a shallow model of an engineering system, physical or abstract, using an inductive tool.

Traditional or deep modeling is based on the assumption that we understand the behavior of an engineering system, and that we have its conceptual model. This conceptual model can then be used for building a formal mathematical model using available experimental results. Very often, however, our understanding of the behavior of engineering systems is incomplete. In this case building formal mathematical models based on their predicted behavior is very subjective, and simply incorrect.

Shallow modeling is based on the system's observed behavior. An understanding of the system is not required. Obviously, such modeling has significant advantages over traditional deep modeling. It is particularly useful in the modeling of very complex systems of unknown structure. Our initial experience in this area indicates that inductive shallow modeling may become very important, especially in engineering research. This experience and our initial methodological suggestions are presented in [9].

To demonstrate the use of shallow modeling, the results of an inductive experiment conducted about two years ago [9] will be briefly described here.

In the experiment, the results of only 15 tests of steel beams under bending were used. An inductive system was applied to confirm the existence of well-known relationships between different groups of variables.

In particular, we were looking for answers to the following questions:

1. Is the moment of inertia (V3) of our beams related to their depth (V1) and thickness (V2)?

2. Is the ultimate beam capacity (V6) related to its moment of inertia (V3)?

3. Is the ultimate capacity of the beam (V6) related to the measured strains (V7, V8) and calculated strains (V10)?

Table 2. *Manufacturing of steel beams: Examples.*

Example	Attributes					
No.	1	2	3	4	5	6
1	1	1	1	1	1	1
2	1	1	1	2	1	1
3	1	1	1	3	1	1
4	1	1	1	1	2	1
5	1	1	1	1	3	1
6	1	1	1	2	2	1
7	1	1	1	2	3	1
8	1	1	1	3	2	1
9	1	1	1	3	3	1
10	2	1	1	1	1	1
11	1	2	1	1	1	1
12	1	3	1	1	1	1
13	2	1	2	1	1	1
14	2	1	3	1	1	1
15	2	1	1	2	1	1
16	2	1	1	3	1	1
17	2	1	1	1	2	1
18	2	1	1	1	3	1
19	2	1	1	3	3	2
20	2	2	1	3	3	2
21	2	3	1	3	3	2
22	2	3	3	3	3	1

The modeling was conducted as a progressive learning process, and the results were recorded after each added example. These results are shown in Fig. 1.

It can be easily seen that the answer to the first question is a very strong Yes. In the theory of rough sets, the dependency factor measures the strength of the relationship between given variables and a group of variables. In this case the dependency factor equals unity, indicating a functional dependency.

The answer to the second question is more complex. There is a relationship between the moment of inertia and the ultimate capacity, but this relationship is not functional, and is relatively weak. The results obtained for the third question indicate that the learning process has not been completed, but there is definitely at least a weak relationship.

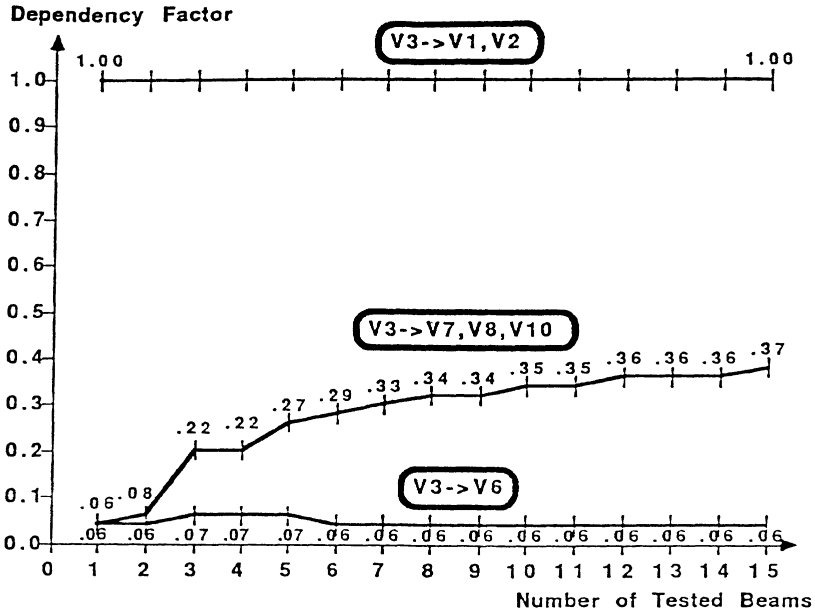


Figure 1. *Inductive shallow modelling: learning curves [9].*

Inductive shallow modeling is still in the experimental stage, but even now it could be useful for practical purposes, particularly in cases where traditional methods of deep modeling are not sufficient.

5 Inductive Learning about Domain

Inductive learning about domain s a systematic and monitored multistage learning process in which an inductive system is used as a learning tool. The objective of this process of learning is to improve the understanding of a given domain through the systematic development of a system of decision rules governing this domain. This initial learning is necessary in the process of knowledge acquisition regarding a new domain. It can also be used when a given domain is well known, but the number of examples, or attributes, precludes the possibility of a human generating decision rules.

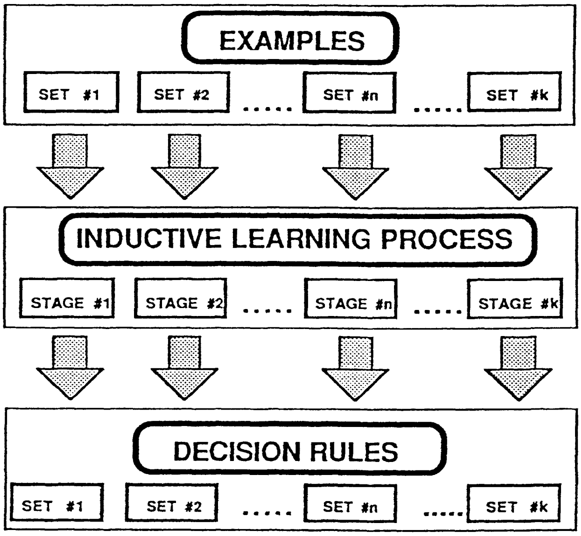


Figure 2. *Inductive learning about domain: a multistage process.*

Very often in civil engineering we have a large body of known examples. We spend months or even years on studies of a given domain, but because of its complexity we do not really understand it. We may identify several simple decision rules governing this domain, but we still need a more fundamental understanding. We simply need more fundamental decision rules governing our domain. This is a typical situation in the research and development of new engineering systems. We identify all known solutions and we want to understand all these solutions, which are our examples. In this case we can use an inductive system as an engineering learning tool, a new tool which can be used by a human expert to learn about a complex domain. This new tool is used in a multistage learning process (Fig. 2). At each stage of this process an inductive system is used to extract decision rules from a different collection of examples. The decision rules and parent examples are recorded. A human expert analyzes all examples and related decision rules, and tries to relate these decisions to examples and to improve his understanding of a given domain.

This potential application still needs a lot of research and experiment, but it looks very promising.

6 Learning Expert Systems

A learning expert system is an expert system with a learning component. Such a system has the ability to learn, that is, to modify its decision rules to follow changing conditions. A learning expert system can be developed for the purposes of conceptual design, diagnosis, or control [3,5].

To explain the concept of a learning expert system, a system for conceptual parametric design will be briefly described here [3]. By 'conceptual parametric design,' we mean an early stage of the design process. In this stage, design needs and available knowledge are analyzed and a number of concepts of a future civil engineering system are generated. In parametric design, a system under consideration is described by parameters and the design process is a sequence of operations on these parameters, including the identification of feasible values of these parameters and the determination of the optimal combination of parameter values. In conceptual design, the parameters considered are mostly qualitative in character, and a compatible combination of their values, when for all parameters one value is taken at a time, identifies a concept of a civil engineering system [3].

A learning expert system for conceptual design can be used for the production of concepts. These concepts can be selected from the generated combinations of values of qualitative parameters, using internally produced compatibility rules. In this case the system must be used in two stages: learning and production. The objective of the first stage is to extract, from given examples, a system of decision rules governing a given domain represented by the examples. In the second or production stage, these decision rules are used for the evaluation of combinations of values and the selection of compatible combinations, which represent the concepts being sought. More details on learning expert systems for conceptual design and the results of structural experiments are given in [5].

In the case of conceptual design, the use of inductive learning has many advantages. It enables us to deal with a large body of examples, and also leads to the development of very complex decision rules, which otherwise would not be prepared because of their complexity and unusual character. A learning expert system for conceptual design has the ability to produce standard, well-known concepts, but there is also a possibility that it will produce innovative, patentable concepts. The present research on learning expert systems for conceptual design is still in its early stages, but its potential is enormous, and very interesting developments can be expected in the future.

7 Conclusions

Five applications of an inductive system in the development of civil engineering decision support systems have been discussed. Two applications, extraction of decision rules from examples and problem solving, can be considered for immediate use, particularly now that the methodology of inductive learning has been developed and can be used to support these applications. Two other applications, shallow modeling and learning about a domain, still require research, but might be considered for experimental use. Learning expert systems, still require a great deal of research.

It should be noted that inductive learning in civil engineering is still in its experimental stages. The available experience is very limited, but initial results indicate that inductive systems may very soon become powerful tools in civil engineering, useful for different purposes, as proposed in this paper.

Acknowledgment The author acknowledges financial support for his research from Wayne State's Institute for Manufacturing Research.

References

[1] Adeli, H. and Balasubramanyam K. V., (1988), *Expert Systems for Structural Design*, Prentice Hall, Englewood Cliffs.

[2] Adeli, H., (1988), *Expert Systems in Construction and Structural Engineering*, Chapman and Hall.

[3] Arciszewski, T. and Ziarko, W., (1987), 'Adaptive Expert System for Preliminary Engineering Design,' *Revue Internationale D. E. CFAO et D'Intographie* **2**, 1.

[4] Arciszewski, T., Mustafa, M. and Ziarko, W., (1987), 'A Methodology of Design Knowledge Acquisition for Use in Learning Expert Systems,' *International Journal of Man-Machine Studies*, No. 27, 1987.

[5] Arciszewski, T. and Ziarko, W., (1988), 'Adaptive Expert System for Preliminary Design of Wind Bracings,' *Second Century of Skyscrapers*, Van Nostrand Publishing Company.

[6] Arciszewski, T., (1988), 'Inductive Learning in Engineering,' *Proceedings: USACERL/ASCE First Joint Conference on Expert Systems*, June 29–30.

[7] Arciszewski, T. and Mustafa, M., (1989), 'Inductive Learning Process: The User's Perspective,' Chapter 3 in *Machine Learning*, R. Forsyth (Ed.), Chapman and Hall.

[8] Arciszewski, T., (1990), 'Teaching Expert Systems Techniques at Wayne State University,' Chapter 2 in *Expert Systems Education in Civil*

Engineering, M. L. Maher and S. Mohan (Eds.), American Society of Civil Engineers.

[9] Hajdo, P., Arciszewski, T., Ziarko, W. and Aktan, H., (1988), 'Inductive Shallow Approach for Generation of Engineering Models,' *Proceedings of the Ninth European Meeting on Cybernetics and Systems Research*, Vienna, Austria, April.

[10] Fenves, S. and Garrett, J., (1986), 'Knowledge-Based Standards Processing,' *International Journal for AI in Engineering* **1**, 1.

[11] Kostem, C. N. and Maher, M. L. (Eds.), (1986), 'Expert Systems in Civil Engineering,' *Proceedings of a Symposium sponsored by the Technical Council on Computer Practices of the American Society of Civil Engineers*, Seattle, Washington, April 8–9, American Society of Civil Engineers.

[12] Maher, M. L., (1987), *Expert Systems for Civil Engineers: Technology and Application*, American Society of Civil Engineers.

[13] Maher, M. L., Fenves S. J. and Garrett J. H., (1988), 'Expert Systems for Structural Design,' Expert Systems in Construction and Structural Engineering, Chapman and Hall.

[14] Modesitt, K. L., (1987), 'Space Shuttle Main Engine Anomaly Data and Inductive Knowledge-Based System: Automated Corporate Expertise,' *Proceedings of the Conference on Artificial Intelligence for Space Applications*, Huntsville.

[15] Mohan, S. and Maher M. L., (Eds.), (to appear), *Expert Systems for Civil Engineering: Education*, American Society of Civil Engineers.

[16] Rychener, M. D., (1988), *Expert Systems for Engineering Design*, Academic Press.

[17] Sillen, R. V., (1987), 'Building PC-hosted Expert Systems Using Induction Methods,' Research Report, Novacast Expert Systems AB.

[18] Sriram, D., (1987), *Knowledge-based Approaches for Structural Design*, Computational Mechanics Publications.

[19] Tcheng, D. K., Lambert, B. L. and Lu, S. C-Y., (1989), 'Integrated Relation Learner, System for Optimization Inductive Bias,' *Proceedings of the Joint International Conference on Artificial Intelligence*, Detroit.

Preliminary Foundation Design Using EDESYN

S. Meyer
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh
United States of America

Abstract Foundation design requires experience and engineering judgment due to the unpredictable properties of soil and the empirical methods used in selecting a foundation type. A knowledge-based approach is used to facilitate the representation of design knowledge. The knowledge-based system generates a small number of foundation designs which are feasible for the given site conditions and building configuration. The building is considered as a whole to identify economical alternatives. The potential designs are selected from the full set of major foundation types. These preliminary designs can then be rigorously investigated by standard methods in order to produce a final design.

1 Introduction

This paper describes a Knowledge-Based System (KBS) implemented using the expert system shell EDESYN. The system addresses the preliminary design of foundations for multi-story buildings. Foundation design is a domain requiring experience and engineering judgment due to the unpredictable properties of soils underlying a proposed building and the empirical nature of the techniques used in analysis and design. Various methodologies may be applied to the problem of foundation design. This implementation uses the hierarchical decomposition and constraint directed search technique employed by the design shell EDESYN.

An expert system for foundation design can perform a useful function within building design since the first tasks on a construction site are site preparation and foundation work. Because of this, the foundation engineer is under pressure to produce a complete design as quickly as possible. A tool which uses preliminary soil data and building characteristics to reduce the set of feasible foundation designs would allow designers to begin their design process earlier and to concentrate on a

B. H. V. Topping (ed.),
Optimization and Artificial Intelligence in Civil and Structural Engineering, Volume II, 333–354.
© 1992 *Kluwer Academic Publishers.*

higher level of design. To provide such a tool is a motivation for the development of this knowledge-based system.

The system addresses preliminary foundation design—when preliminary soil data is available and after potential building configurations have been identified. Before extensive soil testing is completed, the system characterizes the underlying soil. This allows the system to generate, from incomplete data, a small number of preliminary foundation designs which are feasible for the given site conditions and building configuration. Only static axial loads are considered. The set of designs is selected from the full set of major foundation types—shallow, compensated and deep foundations. Additionally, the building is considered as a whole in order to identify economical alternatives rather than designing a foundation for a single column and repeating that design throughout the structural grid. Thus, the system will allow foundation designers to begin the design process with a set of preliminary designs which can be refined and extended quickly as more complete data becomes available. The preliminary designs can then be more rigorously investigated by standard methods in order to produce a final design. In this way, the KBS can be a part of the first iteration in the design process.

2 Designing Foundations

2.1 Three Phases of Foundation Design

This section is intended to outline a basic foundation design process. Foundation design is a complex subject for which no comprehensive, algorithmic procedure has been formulated by experts. The author does not purport to hold the key to such a unified design process but merely attempts to coalesce procedures for the more routine aspects of foundation design laid out in numerous texts on the subject. Any specific foundation design process must be customized to the soil conditions of that particular site, while this general design tool must admit to addressing only the most general soil conditions. The purpose of this outline is twofold, first to lay out a framework for a design process which will be paralleled by the KBS, and secondly to demonstrate the highly empirical nature of foundation design in general. As in other areas of structural design, many analytical methods have been developed for quantifying the various aspects of foundation design. However, how these tools are used and how the parameters involved are determined is based on engineering judgment—qualitative empirical information and experience.

In the most complete sense a foundation is the structural system which provides support for the structural loads. The foundation system includes the soil or rock on which the structure is founded and the portion of the structure which serves to transfer the loads to earth. More commonly the term foundation is used to refer only to the constructed transition member which rests in or on the supporting soil or rock. Nevertheless, the performance and hence the design of the transition member is dependent on the performance of the soil or rock. Thus, an accurate diagnosis of the engineering properties of the natural material on which the constructed material must rest is the first and most troublesome segment of foundation design.

The state of the underlying soil is independent of the building configuration, yet the load which the soil will be subjected to is determined by the size, shape and loading of the building. Therefore, the second step of foundation design must be the characterization of the building structure in terms of how it will effect the foundation system. These parameters give a picture of the intensity and physical extent of the stresses in the underlying soil induced by the structure.

The third stage is foundation synthesis. Synthesis may be defined as the selection, combination and adaptation of relatively standard components to generate a system which meets the prescribed functional requirements. In terms of foundation design, synthesis refers to the selection of one or more foundation types and materials, arranging the foundation types to transfer the loads properly, and sizing the components. This involves determining the depth below ground surface; the shape, size, and materials; and the construction methods for the optimal structure to transfer the superstructure loads into the earth. The foundation must satisfy the functional requirements of sufficient depth, safety against soil rupture or collapse, and the restriction of settlements to tolerable limits. Thus, foundation design is a three stage process composed of characterizing the three components of the soil-structure-foundation interaction which transfers the building loads to firm ground.

2.2 Site Characterization

Preliminary site investigation typically involves a visual survey and the drilling of a limited number of test borings at distances of 100 feet or more apart. The borings retrieve representative soil samples, perform penetration tests and determine the depth to ground water and bedrock. The boring logs, a surface description and geological maps may be all the geotechnical information that a foundation engineer has to work from when beginning the design process.

The visual surface inspection is used to identify topological indications of subsurface variations such as old stream beds or other factors which may guide the placement and number of bore holes. Geological maps, while helpful as another indicator of possible site irregularities such as old stream beds or ox bow lakes, are usually on a large scale and more concerned with rock formations than soils. The majority of the specific soil data is infered from the boring logs. This information is usually in the form of a profile of soil attributes as they vary with depth. The attributes include soil descriptions and SPT N values, and possibly water contents, unit weights, and/or plasticity characteristics.

The most common field test in North America, the Standard Penetration Test (SPT), is a dynamic penetration test for estimating a soil's strength and compressibility. The test involves recording the number of blows (N values) of a 140 lb. weight dropped 30 inches required to drive a standard split spoon sampler 12 inches into the undisturbed soil at the bottom of a bore hole. The SPT is an easily performed empirical test, but it can be affected by obstructions or lenses of soft soils. The SPT N values are of little direct value. However, when used in conjunction with the soil description a variety of heuristicts have been developed in the past for estimating many of the engineering properties of soils or foundation elements. For example, the capacity of a displacement pile driven into granular soils can be estimated from the SPT N values and the pile geometry. Heuristics also exist for estimating the bearing capacity, angle of internal friction or for determining footing dimensions based on the N values of a cohesionless soil.

On the other hand, the engineering properties of cohesive soils are not reliably estimated using SPT values. The compressibility of soft to stiff clays is dependent on a time frame of greater duration than the standard penetration test allows. Nevertheless, the N values can serve as a rough guide in correlating the known stratigraphy of a general locale with that of a particular site in that locale. From this correlation a geotechnical engineer with experience in the area can estimate the engineering properties of the soils at the site, considering them to be similar to those of a generalization of the strata of the area.

When the site both contains layers of cohesive soil and is not within a well studied, consistent area the soil properties of these clay or silt strata are estimated using purely empirical methods. The representative samples are retrieved from the borings and the soil is given a description such as 'soft grey silty clay with traces of fine sand.' The cohesive strength of the soil is estimated using a vane shear test or pocket penetrometer. The bearing capacity for shallow foundations is assigned via a table indexed on a soil description and stiffness using the cohesive strength as a bound. The values in these tables are based on an experientially founded correlation between stiffness, cohesion and compressibility so as to limit settlement to an acceptable amount.

The above initial soil parameters are sufficient for the geotechnical engineer to estimate such secondary soil parameters as the bearing capacity coefficients and the compression index of each stratum. From the initial and secondary parameters the engineer can then start identifying potential foundation systems. More accurate values for the parameters require time consuming laboratory tests such as triaxial tests or large scale field tests such as plate loading or test piles. The foundation engineer can use these values for later iterations of the design process.

2.3 Building Characterization

The second stage of foundation design, building characterization, is a fairly straightforward process. The column loads are computed in the same manner as when designing the building columns. Any distributed loads from external, internal, or shear walls are computed as well. The calculation of loads must also include the unloading produced by excavation, the weight of added fill and the stresses induced by changes in groundwater level. The column spacing is noted along with the total surface area of the structure. Finally, determining the depth of the basements completes the modeling of the structure. This process is a well structured task and hence could be done using an algorithmic procedure.

2.4 Foundation Synthesis

The third stage is the synthesis of the actual load transferring members, the constructed foundation structure. Foundations are classified into three basic types: shallow, compensated, and deep foundations as shown in Fig. 1. Shallow foundations extend below the building a distance less than or equal to their least width, and their support is derived wholly from direct bearing along the base. Spread footings, continuous footings, and mats or rafts are examples of shallow foundations. Compensated foundations work on the principle that if the weight of soil and water excavated equals the weight of the building added the soil is subjected to no additional stresses. The compensated foundation structure acts like the hull of a ship prompting the nickname floating foundation. Deep foundations, which include piles and caissons, extend below the building a distance much greater than their width. Piles can derive their support either from direct bearing at their base, through frictional resistance along their length, or through a combination of the two.

The selection of a foundation type is a task of matching the building loads to supporting strata and recognizing a foundation structure which can safely transfer the loads to the strata. If the soils are relatively strong a shallow foundation may be used. The dimensions of the foundation are determined so that the structural loads are distributed over an area sufficient to limit the unit load to within the unit bearing capacity of the soil. However, if the strong stratum is underlain by a highly compressible layer a shallow foundation may result in excessive or uneven settlements due to the compression of the indirectly stressed weak layer. Alternately, even if the building loads are light but no shallow bearing stratum exists the use of a shallow foundation may allow for too much settlement. Any circumstance of this sort should lead to the selection of a compensated or deep foundation.

Deep foundations are intended to transmit structural loads through weak or compressible soils to stronger soil or rock. When no strong layer exists the pile

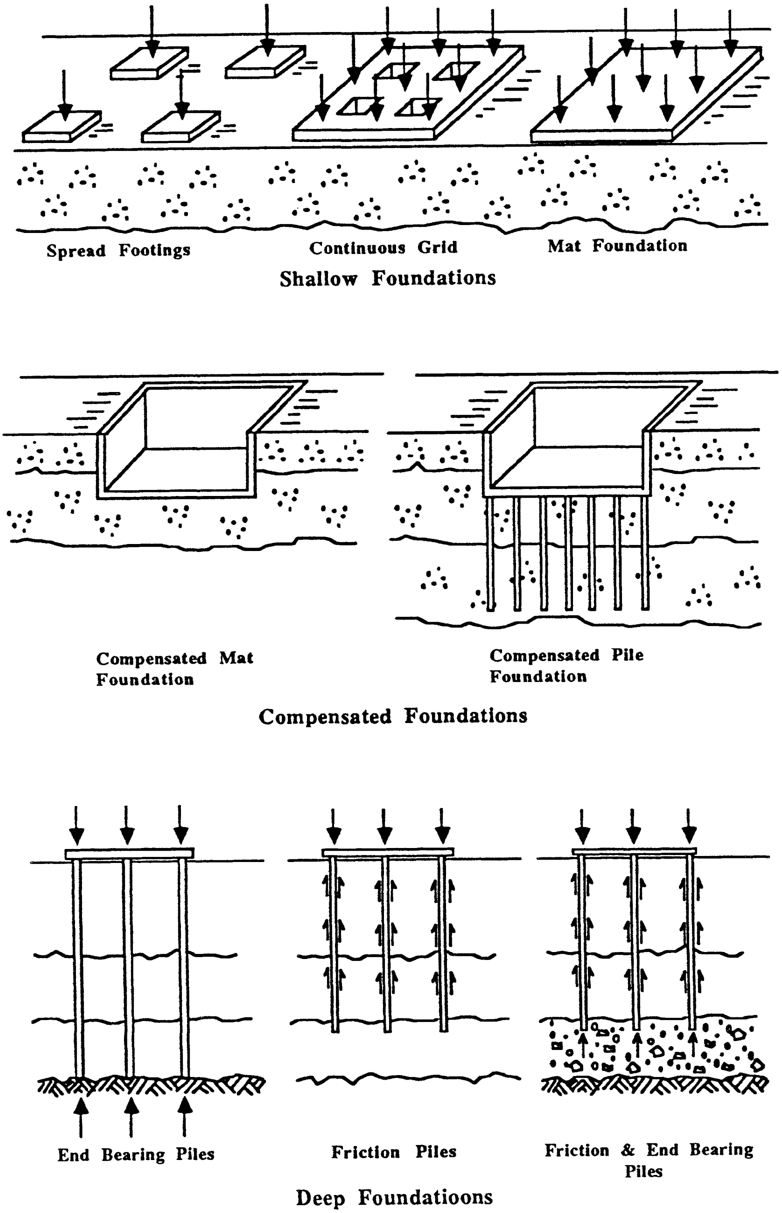


Figure 1. *Three major foundation types.*

group transfers the load to a larger area of soil than a shallow foundation can. The synthesis of a deep foundation system involves the selection of a pile material and construction method based on the magnitude of the loads, the nature of the soils that the pile must penetrate, and the length required to reach the supporting material. For each kind of pile the capacity of an individual pile is estimated. The intent of the estimation is to determine the length required to achieve the maximum safe axial capacity of the pile. Because of the bracing effect of even soft soils a pile may be considered as a short column. However, a building column is usually supported by more than one pile since the capacity of a pile is much less than that of a building column. Therefore, an efficiency or group effect for the pile group is computed based on the soils penetrated, and the number and spacing of the piles. The efficiency is a dimensionless factor between two and two thirds which is then multiplied by the capacity of the individual piles to arrive at a capacity for the pile grouping.

An additional choice when existing soils are found to be unsuitable from the standpoint of their bearing capacity or calculated settlements is the possibility of ground improvement. The objective of ground improvement is the strengthening of weak soils or the reduction of downdrag forces on piles. Ground improvement may take the form of the removal of the top few meters of poor soil and its replacement with compacted fill. Alternately, the existing soils may be compacted through mechanical or chemical means. A third improvement method is the reduction of groundwater to reduce compressibility or negative frictional forces on a pile foundation.

2.5 Ranking of Alternatives

As in most engineering design, the final choice among the feasible alternatives is based on economics. The economics of constructing various foundation types and their varieties can be as much a factor as the loading and soil conditions. The least expensive foundation type is typically a shallow foundation and therefore they are a preferred choice when feasible. If a shallow foundation is appropriate, a spread footing system which occupies close to 30% of the building area is likely to be less economical than a grid of continuous footings. When the footing area approaches half of the building area a mat foundation becomes economically attractive. The savings are in the formwork required and the amount of material needed to resist shear and moments in the foundation.

The selection of a particular deep foundation system is based on the overall cost of the candidates. The overall cost includes the unit cost per pile, the number of units required and the cost of installation. Installation problems are a considerable and highly unpredictable factor in the selection of a deep foundation system. Frequent or large obstructions or the presence of groundwater in cohesionless soils may rule out the use of certain types of piles. Therefore, the final selection of a pile type and installation method is usually not based on the few preliminary test borings. A more thorough site investigation is used to make the final decisions. This is but one more reason to generate a wide range of foundation systems during preliminary design.

3 Expert Systems and Foundation Design

Knowledge-based system applications have been classified along a spectrum ranging from derivation or interpretive tasks at one extreme to generative or formative tasks at the other [3]. Interpretive tasks use observed data to infer a problem state which is consistent with the initial conditions. The problem state contains more complete or more meaningful information than the initial statement. Two examples of interpretive systems are Prospector, a system for identifying ore-bearing geological formations [2] and Dipmeter Advisor, a system for interpreting oil well log data [1]. On the other hand, formative systems develop new object or process descriptions which satisfy all the applicable constraints. Beginning from an initial state and a set of constraints the formative system generates potential solutions and tests them against the constraint set. A formative system may be constructed to produce solutions which merely meet all the constraints or the system may form a solution which is optimal in some sense. Two examples of generative systems are R1 which configures VAX computer systems [8] and HI-RISE which synthesizes structural systems [6].

As stated earlier, the task of foundation design contains three stages: site characterization, building characterization and foundation synthesis. Site characterization is a derivation problem. An initial condition containing uncertain and incomplete soil property data is transformed into a wider and more useful set of soil parameters. The solution parameters are those which are sufficient for determining the dimensions and capacities of potential foundation designs. Which parameters are derived is dependent on the values of the parameters in the initial condition. Furthermore, the site characterization segment attempts to identify interactions between strata which may cause difficulties to the potential foundation designs.

The task of building characterization is a simple extrapolation from known data to more complete data. Building characterization is a derivation task, but one which could be performed using a standard algorithmic approach.

Foundation synthesis is a generative task. A description of the functional requirements is contained in the site and building characterizations. The limitations of the available foundation materials (including the soils or rock) and the construction processes form the set of constraints on the solution. Starting from the functional requirements and the constraints on the materials and processes involved, a foundation description is generated. Eventually the optimal solution is chosen since only one foundation can be built and the developer wishes to spend the minimal resources. Preliminary design should recognize the limitations on its domain and recommend multiple solutions. The multiple designs may be ranked according to an evaluation function to produce an ordered set of solutions.

Existing geotechnical knowledge-based systems tend to address a limited portion of foundation design. Systems exist which concentrate their efforts on pile design only or on the design of a spread footing or a single pile to support an individual column. The previous foundation design systems implemented in EDESYN follow the latter strategy. However, the exclusion of potential foundation types is likely to lead to an uneconomical design. Further, the restriction to one pile per column is an unrealistic constraint since the capacity of a column can be much greater than that of a pile. By considering the building as a whole the economics of various shallow foundations may be weighed against each other. Also, it is impossible to consider compensated foundations if the building as a whole is not taken into account.

Other systems concentrate on soil characterization. For example, CONE interprets soil characteristics from Dutch Cone penetrometer data [9] while SITECHAR serves as a component of an electronic workbench for geotechnical site characterization [10]. Each of these derivation systems provide a rigorous interpretation of soil properties within their domain. However, neither of these systems address the use of this information. The system described in this paper attempts to characterize the site conditions as fully as possible given the type of data normally available from preliminary site exploration and to use the information derived for synthesizing tall building foundation designs.

4 Overview of EDESYN

EDESYN (Engineering DEsign SYNthesis) is a domain independent shell for the development of expert systems in engineering design [7,5]. The architecture of EDESYN follows the current tenet of knowledge-based systems, maintaining a separation of knowledge and control. EDESYN defines a representation and organizational structure for the domain knowledge and provides the controlling inference mechanism or synthesis mechanism. The knowledge is provided by the developer of the particular KBS application in the form of system decompositions and plans, constraints, and functions. The resulting knowledge-base is used for a particular instance of the domain by a designer who provides the initial conditions and the mid-process guidance.

The decomposition describes how the main design goal is to be broken down into subgoals, and how each subsequent subgoal is to be addressed. Each subgoal can be satisfied in one of three ways: further decomposition, selection from an enumerated set of discrete solutions, or the evaluation of a LISP function. Eventually the results of all decompositions are assigned a value through the selection of a set member or as the result of a LISP function. The initial conditions and the synthesized values are stored in the design context in the form of a solution tree.

The planning rules guide the decomposition based on the values of attributes in the design context. Each planning rule is in the form of an **If** ... **then** ... statement. The antecedent of the rule specifies a state before the decomposition of the pertinent goal. If the antecedent is true the goal is decomposed according to the consequent of the planning rule. The consequent specifies a subset of the default decomposition to be used for the current instance of that goal's decomposition. If no planning rule applies, the current goal is decomposed according to the general decomposition. Thus, the planning of the decomposition is done during the design process. This allows the goals of a system to be organized in an order adapted to the values of particular attributes identified in the plan—adapting the process to the context.

The synthesis mechanism controls the generation of all design alternatives which are feasible for the initial conditions specified by the user. Following a depth-first traversal of the task decomposition, the algorithm seeks solutions to each goal. A goal which is solved through further decomposition is decomposed and a plan is applied if one exists. The plan is based on the application of the planning rules to the design context. A goal which is solved directly is assigned a value using the method prescribed in the decomposition. As the decomposition tree is traversed a solution tree is generated to contain each feasible solution. The solution tree branches at each goal solved via an enumerated set. After checking the constraints on an assigned value the search is continued with the next goal in the depth-first order. Each applicable constraint is checked against the current assignment before pursuing the next goal.

The constraints are used to prune the search and solution trees. Each constraint in the knowledge base describes an infeasible solution to a set of goals. When a value has just been assigned to a goal each pertinent constraint is checked to determine if there is a prohibition on the value assigned. The constraints are in the form of a set of logical expressions. Each expression compares an attribute of the decomposition or the initial conditions to its value in the design context. If every expression in a constraint is true the recently assigned value must be part of an invalid solution to the goal and the branch is pruned from the solution tree. When a constraint is fired the trees are pruned back to the previous branching node and the next branch in the search tree is pursued. Thus, the constraint serves to eliminate inappropriate alternatives to a section of the design based on the value of the combined attributes in the current solution path.

The design context maintains the knowledge about the state of the particular solutions to the current problem. Initially the context holds only the user supplied initial conditions. As the synthesis proceeds more information is added to the design context. The information forms a tree of alternate solutions, with each node representing an attribute and its value. When all the goal decompositions have produced leaf nodes the tree represents a complete set of feasible design solutions. Each path from root to leaf traces one feasible solution. At this point the context holds all feasible solutions to the design problem and the synthesis algorithm terminates.

5 Foundation Design In EDESYN

5.1 Overview

The implementation of a foundation design system in EDESYN involves decomposing the task into the three major phases discussed previously plus the subgoals for each phase, developing a set of constraints to guide each goal satisfaction search, and composing LISP functions to compute values for parameters which are not easily enumerated. The decomposition is organized into discrete tasks which depend on values available from the initial conditions or from previously performed tasks. The following sections describe an initial EDESYN implementation.

5.2 Decomposition

The foundation design system is first decomposed into systems corresponding to the three phases of foundation design: soil system, building system, and foundation synthesis system. In this implementation the building characterization is performed before the site characterization. Among the reasons for this choice is that this order facilitated the definition of a shallow bearing stratum in terms of the previously defined building characterization. The foundation design system decomposition is illustrated in Fig. 2, and Appendix I contains an example decomposition file.

The building system requires no further decomposition. All of the building system values can be derived directly from the initial conditions either through the constraint directed enumeration of the allowable values or through the evaluation of a function. Building system attributes include minimum column spacing, total building area, building weight per area, minimum foundation depth and maximum footing depth. The weight per area and the maximum footing depth are used later in the search for a shallow bearing stratum. The weight per area is also used in determining the depth of a compensated foundation so that building weight added and soil weight removed are balanced.

The soil system is further decomposed into stratum systems, the number of systems being determined by the initial conditions supplied by the user. Every distinct soil layer at the site corresponds to one stratum system. Each stratum system contains secondary soil parameters whose values are derived from the primary soil parameter values in the initial conditions. For example, cohesive soils are assigned a bearing capacity based on their consistency and compressive strength.

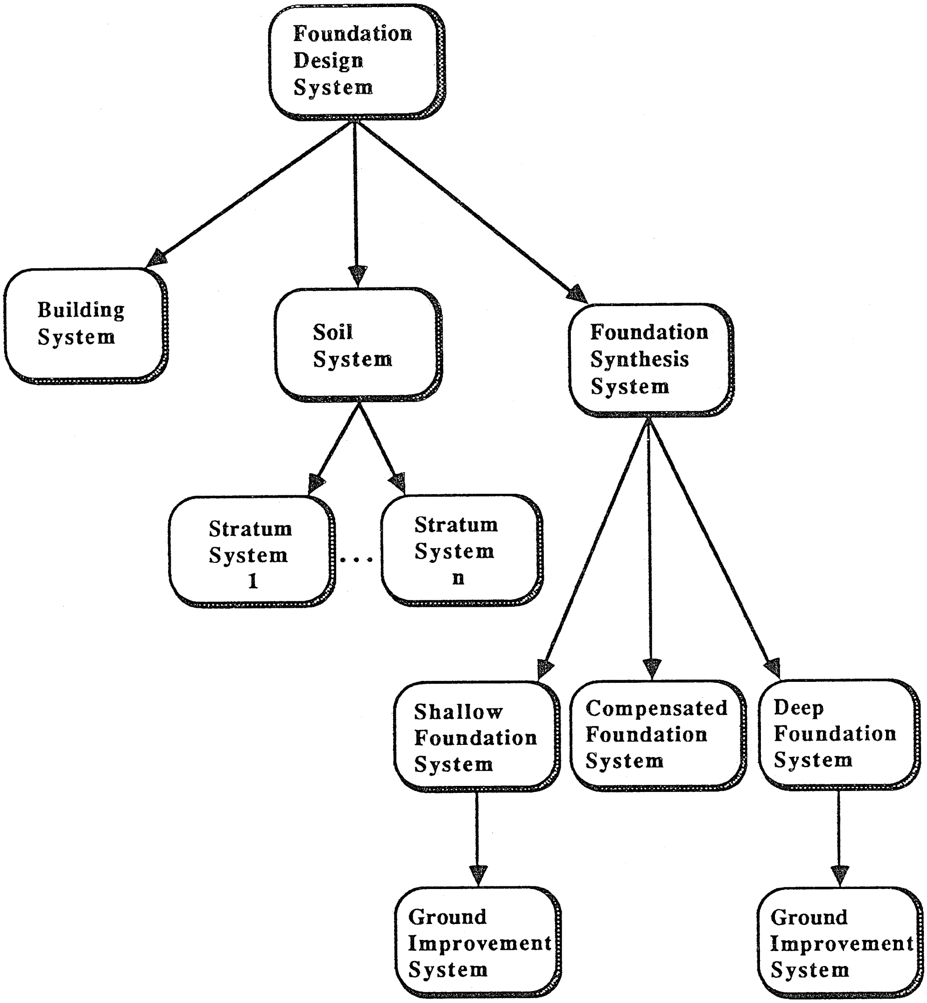

Foundation Design System
Building System
Soil System
Foundation Synthesis System
Stratum System 1
...
Stratum System n
Shallow Foundation System
Compensated Foundation System
Deep Foundation System
Ground Improvement System
Ground Improvement System

Figure 2. *Foundation design system decomposition.*

Cohesionless stratum systems use the SPT N values and the soil description to determine their bearing capacity value.

The foundation synthesis system is decomposed into the three major foundation types, a shallow foundation system, a compensated foundation system, and a deep foundation system. The first goal of the each of the foundation systems is a foundation variety within that type such as spread footing or prestressed concrete pile. This goal is satisfied through the constrained enumeration of the defined varieties within each type. Most of the other goals are given values through the evaluation of a function. For example, the thickness of a spread footing required to resist punching shear is computed by a specific function. The final value for the thickness of a spread footing is then calculated as the maximum of the thicknesses required to resist punching shear and wide beam shear.

An additional goal for shallow and deep foundation systems is a subsystem for considering ground improvement. For shallow foundations this subsystem contains goals which represent potential methods of improving the soil layers so that they may become bearing strata. For the deep foundations the improvement is intended to reduce excessive settlement or downdrag on piles. The systems use information from the soil stratum systems to assign method goals such as excavate-&-replace with a value of true or nil depending on whether the ground improvement method is deemed practical or not. When the method goal is given a value of true a new-bearing-capacity goal is given a value which supersedes the bearing capacity value of the original soil for the current solution branch. The decomposition of the shallow foundation system including its planning rule is illustrated in Fig. 3.

Planning rules within a system decomposition specify a condition when the default decomposition can be made more specific. For instance, if there is a shallow bearing stratum the decomposition of the shallow foundation system should be restricted to exclude a ground improvement subsystem goal.

5.3 Constraints

Each constraint in the knowledge base specifies a set of goal assignments which are invalid when combined in one solution path. The constraint set is developed within qualitative bounds. The problem is sufficiently constrained so as not to generate misleading or wasteful solutions. At the same time, it is loosely enough constrained to generate more than one design. In this way an indeterminate problem is turned into an advantage.

Constraints were used to to guide the assignment of a pile's resistance-type. For instance, timber piles are used only as friction piles, whereas compacted concrete piles are used only in end-bearing. The selection of pile diameters is also guided by constraints. Many pile varieties are available in more than one size, in addition to having a certain relationship between their grade and tip diameters. As an example, bored piles are used in a few standard sizes and the grade diameter is

```
Shallow-Foundation-System

  Type                          one-of      (spread-footings
                                              continuous-grid
                                              mat )

  Ground-Improvement?           subsystem   Ground-Improvement-System

  Depth-Below-Surface           function    Depth-Below-Surface

  Breadth                       function    Shallow-fdn-Breadth

  Punching-Shear-Thickness      function    Punching-V-Thickness

  Beam-Shear-Thickness          function    Beam-V-Thickness

  Moment-Thickness              function    M-Thickness

  Thickness                     function    Overall-Thickness


  IF
      stratum-system/shallow-bearing-stratum   =   True
  THEN
      (Type Depth-Below-Surface Breadth Punching-Shear-Thickness
         Beam-Shear-Thickness Moment-Thickness Thickness)
```

Figure 3. *Shallow foundation system decomposition.*

equal to their tip diameter unless it is belled. On the other hand, fluted shell piles are available in a different set of grade diameters and the tip diameter is smaller than its grade diameter.

Constraints also serve to eliminate partial design solutions when they become infeasible during the synthesis process. For instance, every pile type has a maximum capacity and most have a minimum and maximum length. Additionally, many pile types are limited to a particular soil type or damaged by obstructions in the soil. The following constraint specifies that if a thin-shell pile is being considered and the soil contains moderate or frequent obstructions, then that pile type is infeasible. A sample of the actual constraints used in this prototype are contained in Appendix II.

```
If
  casing = thin-shell
  obstructions in (list frequent-large frequent-small moderate)
then
  not feasible.
```

5.4 LISP Functions

The assignment of numerical values to most dimensional or capacity attributes is done using LISP functions. Standard analytical procedures have been used for dimensioning of foundation components such as footings and piles. Mat thicknesses are found in this implementation using heuristics. Whether the procedure is analytically derived or heuristically based the LISP function provides the EDESYN implementation with a direct algorithmic method of assigning continuous or discrete values. As described earlier, the thickness of a footing is computed using a set of functions. Three functions are involved in the process. Two separate functions calculate the thickness required to resist punching shear and wide beam shear in the footing. Each of these values is stored as an attribute of the shallow foundation system and the final thickness is computed as the maximum plus a three inch cover. Separate functions are used for each force both for modularity and so that it may be seen which is the determining force.

The capacity of each friction pile is assigned by a function which recursively loops over each stratum. The capacity of each stratum is added to the total capacity of the pile. The recursion terminates when the maximum capacity or the maximum length of the pile is reached, or when the pile reaches bedrock or another end-bearing stratum. If the maximum capacity is reached before reaching the maximum length of the pile type, the recommended length is adjusted in accordance with the maximum capacity.

6 Conclusions

The use of an expert system shell allows for the rapid development of a KBS applications. The design shell EDESYN provides a knowledge representation, and a hierarchical decomposition and constraint directed search technique for developing engineering design applications. The use of EDESYN for preliminary foundation design is appropriate both as a prototyping tool for the application and as a test domain for the shell itself. Most tasks are easily decomposed into loosely coupled systems, depending on information from previous tasks only. The use of enumerated solutions to a goal, providing a branching into multiple solutions, extends the system into domains which profit from the recommendation of more than one design.

On the other hand, the design knowledge contained in this implementation has been limited to the general information available in texts. In foundation design, even in preliminary foundation design, a controlling factor is often the special conditions of the intended site and construction team. The difficulty of quickly filling an expert system with the particular knowledge of a wide range of conditions, and using this knowledge efficiently is a separate research topic. Furthermore, information available from the performance of existing foundations in the immediate vicinity of a proposed building cannot be added to a fixed decomposition unless there are subsystems explicitly provided for this knowledge.

In this prototype, only a single soil profile is used in the site characterization. This simplification avoids the issue of managing large amounts of soil data in a design shell with limited database interfacing. A further difficulty in using EDESYN is when tasks are not easily decomposed into uncoupled subgoals. The friction pile attributes of capacity and length, when the maximum capacity is reached well before the maximum length, is an example of this situation. Pile length and capacity are most conveniently calculated in parallel, proceeding downward through the soil layers untill the maximum length or capacity is reached. In using EDESYN only one attribute is assigned at one time. Here the capacity is assigned first and then the length computation begins. With only the final capacity value available for use in length computation many calculations are performed redundantly for each pile type.

Nevertheless, the mixed formalism of EDESYN, combining decomposition, constraints, and algorithmic functions provides a means for tailoring the different aspects of the design process to an appropriate and convenient methodology. The separate formalisms provide a clarity to the process, dividing aspects into easily understood portions. This allows the developer to concentrate on particular situations or aspects of the design process. In summary, this implementation illustrates the potential for the rapid development of a knowledge-based application using the system shell EDESYN, that the constraint directed hierarchical decomposition technique is appropriate for preliminary foundation design, and that the developed

KBS can perform a useful function within the foundation design process.

References

[1] Davis, R. *et al.*, (1981), 'The Dipmeter Advisor: Interpretation of Geological Signals,' *Proceedings, Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., pp. 846–849.

[2] Duda, R. O. *et al.*, (1979), *A Computer Based Consultant for Mineral Exploration*, Final Report SRI International Project 6415, SRI International.

[3] Fenves, S. J., (1987), *Role of Artificial Intelligence and Knowledge-Based Expert Systems Methods in Civil Engineering*, Technical Report EDRC-12-17-87, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

[4] Hunt, R. E., (1986), *Geotechnical Engineering Analysis and Evaluation*, McGraw-Hill Book Company, NYC.

[5] Lord, J. A., (1989), 'EDESYN User's Manual,' Carnegie Mellon University, Department of Civil Engineering, Unpublished.

[6] Maher, M. L. and Fenves, S. J., (1984), *HI-RISE: An Expert System for the Preliminary Structural Design Of High Rise Buildings*, Technical Report R-85-146, Carnegie Mellon University, Department of Civil Engineering.

[7] Maher, M. L., (1988), 'Engineering Design Synthesis: A Domain Independent Representation,' in *Artificial Intelligence in Engineering Design, Analysis and Manufacturing* **1**, 3, 207–213.

[8] McDermott, J., (1980), *R1: A Rule-Based Configurer of Computer Systems*, Technical Report CMU-CS-80-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

[9] Mullarkey, P. W. and Fenves, S. J., (1987), 'Fuzzy Logic in a Geotechnical Knowledge-Based System: CONE,' *Civil Engineering Systems* **3**, 2, 58–81.

[10] Rehak, D. R., Christiano, P. P. and Norkin, D. N., (1985), 'SITECHAR: An Expert System Component of a Geotechnical Site Characterization Workbench,' *Proceedings, ASME Winter Annual Meeting Symposium: Applications of Knowledge-Based Systems to Engineering Analysis and Design*, American Society of Mechanical Engineers (ASME).

[11] Smith, G. N. and Pole, E. L., (1980), *Elements of Foundation Design*, Garland STPM Press, NYC.

APPENDIX I

Simplified Decomposition File for Foundation Design Using EDESYN

Each system is composed of triplets of the form attribute-name, solution method, method description. For a goal solved via decomposition or a function the description is the subsystem or function name. For a goal solved through the enumeration of a discrete set, the set itself is listed.

```
system fdn-design
        bldg-characterization          subsystem   bldg-char-system
        site-characterization          subsystem   site-char-system
        foundation-synthesis           subsystem   foundation-synthesis-system
end-system

system bldg-char-system
        bldg-area                      function    bldg-area
        weight-per-area                function    wt-per-area
        safety-factor                  function    SF-from-use-&-siezmic
end-system

system site-char-system
        soil-stratum-system            subsystem   stratum-system
        shallow-bearing-stratum        function    id-shallow-bearing-stratum
        min-fdn-depth                  function    min-fdn-depth
        max-footing-depth              function    max-ftg-depth
end-system

system stratum-system
        stratum-id                     function    stratum-id
        given-values                   subsystem   stratum-precondition-system
        thickness                      function    stratum-thickness
        stratum-top                    function    stratum-top
        friction-angle                 function    friction-angle
        Nc                             function    lookup-Nc
        Nq                             function    lookup-Nq
        bearing-capacity               function    allowable-bearing-capacity
end-system
```

```
system stratum-precondition-system
        stratum-id                     function    stratum-id
        bottom-depth                   function    bottom-depth
        Avg-N                          function    Avg-N
        density-consistency            function    density-consistency
        type-qualifier                 function    type-qualifier
        soil-type                      function    soil-type
        soil-grade                     function    soil-grade
        Cu                             function    Cu
        unit-wt                        function    unit-weight
        water-content                  function    water-content
        obstructions?                  function    obstructions?
        basement-depth                 function    basement-depth
        shallow-bearing-stratum        function    shallow-bearing-stratum
end-system


system foundation-synthesis-system
        shallow-foundation             subsystem   shallow-fdn-system
        compensated-foundation         subsystem   compensated-fdn-system
        deep-foundation                subsystem   deep-fdn-system
planning-rules
IF
        (precondition/bedrock-depth) < (site-char-system/max-footing-depth)
THEN
        (shallow-foundation-system)
IF
        (site-char-system/stratum-system/stratum-top
                (/shallow-bearing-stratum = /soil-stratum-system/stratum-id))
        < (site-char-system/max-footing-depth)
THEN
        (shallow-foundation-system)
end-system


system shallow-fdn-system
        shallow-grnd-improvement       subsystem   shallow-grnd-improvement-system
        shallow-fdn-variety            one-of      (spread-footing continuous-grid mat)
        breadth                        function    shallow-fdn-breadth
        punching-V-depth               function    D-punching-V
        beam-V-depth                   function    D-beam-V
        thickness                      function    max-D-punching-beam-V
        depth-below-grade              function    depth-below-grade
planning-rules
IF
        (stratum-system/shallow-bearing-stratum) <> nil
THEN
        (shallow-fdn-variety breadth punching-V-depth beam-V-depth moment-depth
        thickness depth-below-grade)
end-system


system shallow-grnd-improvement-system
        method                         one-of      (excavate-&-backfill compact surcharge
                                                   nil)
        depth                          function    grnd-imprv-depth
        imprv-bearing-capacity         function    imprv-bearing-cap
end-system
```

```
  system compensated-fdn-system
        compensated-fdn-variety       one-of        (compensated-mat compensated-piles)
        compensated-fdn-details       subsystem     comp-fdn-detail-system
  end-system

  system comp-fdn-detail-system
        depth-below-grade             function      compensated-depth
        mat-structure                 one-of        (solid hollow-tubed rect-cavities)
        mat-thickness                 function      compensated-mat-thickness
        pile-variety                  one-of        (H-sect pipe precast-concrete bored
                                                    prestressed-concrete thin-shell
                                                    fluted-shell)
        pile-diameter                 function      compensated-pile-diameter
        pile-length                   function      compensated-pile-length
  planning-rules
  IF
        (compensated-fdn-system/compensated-fdn-variety) = compensated-mat
  THEN
        (depth-below-grade mat-structure mat-thickness)
  end-system

  system deep-fdn-system
        deep-fdn-variety              one-of        (timber H-sect pipe precast-concrete
                                                    prestressed-concrete thin-shell
                                                    fluted-shell compacted-concrete bored)
        deep-fdn-details              subsystem     deep-fdn-detail-system
  end-system

  system deep-fdn-detail-system
        material                      one-of        (wood steel concrete)
        resistance-type               one-of        (friction end-bearing)
        shape                         one-of        (round-solid pipe square H)
        grade-diameter                one-of        (3.5 2.0 1.66 1.5 1.25 0.833)
        tip-diameter                  one-of        (4.0 3.5 2.0 1.75 1.5 1.25 1.0 0.833 0.66)
        construction-tech             one-of        (as-is cast-in-place cast-in-shell precast
                                                    prestressed compacted)
        insertion-tech                one-of        (driven vibrated bored)
        casing                        one-of        (nil corrugated thin-shell fluted-shell
                                                    pipe)
        max-length                    function      max-pile-length
        max-capacity                  function      max-pile-capacity
        individual-capacity           function      assign-pile-capacity
        length                        function      assign-pile-length
        efficiency                    one-of        (0.66 1.0 1.33)
        piles-per-column              function      piles-per-column
        group-capacity                function      group-capacity
  end-system
end-file
```

APPENDIX II

Portion of the Constraint File for Foundation Design Using EDESYN

Each constraint specifies an infeasible combination of attribute values. Within each elimination constraint the IF and THEN infeasible portions of the statement are implied.

```
constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/resistance-type) = end-bearing

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/shape) <> round-solid

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/construction-tech) <> as-is

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/insertion-tech) = bored

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/casing) <> nil

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/grade-diameter) > 1.75

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/grade-diameter) < 1.0

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/tip-diameter) > 0.833

constraint
      (deep-fdn-detail-system/material) = wood
      (deep-fdn-detail-system/tip-diameter) >= (deep-fdn-detail-system/grade-diameter)
```

```
    constraint
        (deep-fdn-detail-system/material) = wood
        (precondition/GWL) > (precondition/basement-depth)

    constraint
        (deep-fdn-detail-system/material) = wood
        (stratum-system/stratum-precondition-system/obstructions?
        (/stratum-top < (+ /stratum-precondition-system/basement-depth 40) ) )
        in (list sparse frequent-large frequent-small)

    constraint
        (shallow-fdn-system/shallow-fdn-variety) = spread-footings
        (* (shallow-fdn-system/breadth (/shallow-fdn-variety = spread-footings))
        (shallow-fdn-system/breadth (/shallow-fdn-variety = spread-footings) )
        (precondition/num-columns) ) > (* 0.4 (bldg-char-system/bldg-area) )

    constraint
        (shallow-fdn-system/shallow-fdn-variety) = continuous-grid
        (* (shallow-fdn-system/breadth (/shallow-fdn-variety = continuous-grid))
        (shallow-fdn-system/breadth (/shallow-fdn-variety = continuous-grid) )
        (precondition/num-columns) ) > (* 0.6 (bldg-char-system/bldg-area) )

    constraint
        (deep-fdn-detail-system/casing) = thin-shell
        (stratum-system/stratum-precondition-system/obstructions?
        (/stratum-top < (+ (precondition/basement-depth) 50)))
        in (list sparse frequent-small frequent-large)

    constraint
        (deep-fdn-detail-system/casing) = thin-shell
        (stratum-precondition-system/soil-type
        (/stratum-top < (+ (precondition/basement-depth) 50))) = gravel

    (deep-fdn-detail-system/casing) = thin-shell
        (stratum-precondition-system/soil-qualifier
        (/stratum-top < (+ (precondition/basement-depth) 50))) = gravely
end-file
```