

Some pages of this thesis may have been removed for copyright restrictions.

If you have discovered material in Aston Research Explorer which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our [Takedown policy](#) and contact the service immediately (openaccess@aston.ac.uk)

THE APPLICATION OF EXPERT SYSTEMS TECHNIQUES
IN THE DESIGN OF DURABLE CONCRETE

ANDREW JOHN WILKINS

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

June 1991

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior, written consent.

THE APPLICATION OF EXPERT SYSTEMS TECHNIQUES
IN THE DESIGN OF DURABLE CONCRETE

ANDREW JOHN WILKINS

Doctor of Philosophy

1991

Thesis Summary

This thesis describes work done exploring the application of expert system techniques to the domain of designing durable concrete. The nature of concrete durability design is described and some problems from the domain are discussed. Some related work on expert systems in concrete durability are described. Various implementation languages are considered - PROLOG and OPS5, and rejected in favour of a shell - CRYSTAL3 (later CRYSTAL4). Criteria for useful expert system shells in the domain are discussed. CRYSTAL4 is evaluated in the light of these criteria. Modules in various sub-domains (mix-design, sulphate attack, steel-corrosion and alkali aggregate reaction) are developed and organised under a BLACKBOARD system (called DEX). Extensions to the CRYSTAL4 modules are considered for different knowledge representations. These include LOTUS123 spreadsheets implementing models incorporating some of the mathematical knowledge in the domain. Design databases are used to represent tabular design knowledge. Hypertext representations of the original building standards texts are proposed as a tool for providing a well structured and extensive justification/help facility. A standardised approach to module development is proposed using hypertext development as a structured basis for expert system development. Some areas of deficient domain knowledge are highlighted particularly in the use of data from mathematical models and in gaps and inconsistencies in the original knowledge source Digests.

Keywords

Expert Systems, Concrete Design, Building Standards, CRYSTAL, Hypertext

Dedication

This thesis is dedicated to my parents, Brian and Sylvia who have encouraged and supported me through the many years of my education and to Siân who has to cope with me everyday.

Acknowledgements

The author would like to acknowledge the Science and Engineering Research Council and the Department of Environment Building Research Establishment for funding for the work on which this thesis is founded.

The author would also like to acknowledge the following individuals whose cooperation was essential to the work: Dr. J. Elgy and Prof. C. Page of Aston University; Mr. P. Walton, Dr. P. Nixon and Dr. R. Collins of BRE. Lastly, the author wishes to thank Mr. H.T. Tillotson of the University of Birmingham for his encouragement and advice.

Table of Contents

Chapter 1: Brief and Scope of The Project	8
1.1 Introduction to Expert Systems	8
1.2 Introduction to Concrete Design	11
1.3 BRE Digests as Knowledge Sources	12
1.4 Expert Cooperation	15
1.5 Specific Goals of the Project	17
Chapter 2: Overview of Expert Systems	22
2.1 Introduction	22
2.2 Representation/Inference Techniques	23
2.3 Knowledge Acquisition and Elicitation	29
2.4 Interface Techniques	33
2.5 Hybrid Expert Systems	35
2.6 Intelligent Expert Systems	37
Chapter 3: Engineering Design Theory	41
3.1 A Simple Design Task	41
3.2 A Model of Deliberate Design Activity	43
3.3 Design Prototypes	49
3.4 Generic Tasks	57
3.5 Expert Systems in Design	64
Chapter 4: Expert Systems in Engineering Design	67
4.1 Intelligent Front Ends	67
4.2 Mechanical Design Systems	70
4.3 Building Standards and Codes	72
4.4 Integrated Structural Design	78
4.5 Concrete Design	83
Chapter 5: Overview of Concrete Design	90
5.1 Introduction	90
5.2 Basic Mix Design	94
5.3 Sulphate Attack	98
5.4 Alkali Aggregate Reaction	100
5.5 Steel Reinforcement Corrosion	105
5.6 Other Problems	106
5.7 Conclusions from the Literature Searches	109
Chapter 6: Early Work: VAX - PROLOG	111
6.1 PROLOG as an Expert System Implementation Language ...	111
6.2 The Mix Design System	116
6.2.1 The Inference Engine	117
6.2.2 The Mix Design Knowledge-Base	122

6.2.3 The Knowledge-Base Editor	125
6.2.4 User Interface	129
6.3 Conclusions from the PROLOG Work	132
Chapter 7: Early Work: VAX - OPS5 and C	135
7.1 Sulphate Attack	136
7.2 Alkali Aggregate Reaction	143
7.3 Hydroxyl Ion Concentration	149
7.4 Conclusions from the OPS5/C Work	150
Chapter 8: Relevance and Evaluation of Expert System Shells	154
8.1 Speciality Shells	155
8.2 Evaluation of Shells	158
8.3 CRYSTAL	163
8.4 Conclusions	169
Chapter 9: The DEX Concrete Durability Expert System	171
9.1 Translation of Existing Work Into CRYSTAL	171
9.2 Evolution of Enhanced Features	175
9.2.1 User Interface	175
9.2.2 Justification and Explanation	178
9.2.3 Alternative Representations	182
9.2.4 Provision of Flexibility for the User	183
9.2.5 Representation and Use of Constraints	187
9.2.6 Further Knowledge Acquisition	190
9.2.7 Data Flow Analysis of Domain	191
9.2.8 Overall Organisation of Modules	194
9.2.9 Testing DEX	201
9.3 Potential for Rule Induction	204
9.4 Discussion of the CRYSTAL Prototype	205
Chapter 10: DEX Modules in LOTUS123	209
10.1 Mix Design Spreadsheet	211
10.2 Hydroxyl Ion Spreadsheet	217
10.3 Lotus - Crystal Interface	220
Chapter 11: The DEX Hypertext Support System	224
11.1 Hypertext as an Expert System User Support Technique	228
11.2 Use of Hypertext in the DEX System	230
11.3 A Methodology for Expert Systems Development	237
Chapter 12: Conclusions	240
References:	250
Appendix A:	278
Appendix B:	283

Table of Figures

2.1: Frame Representation of carbon element	28
3.1: A Model of Design as a Process	46
3.2: A Simple Design Hierarchy	49
3.3: Routine Design with Prototypes	54
3.4: Abstraction, Refinement and Heuristic Classification	62
4.1: Building Standards and Expert Systems	75
5.1: Relationship Between Durability and Design Life	94
5.2: Relationship Between Strength and W/C Ratio	96
7.1: Heuristic Classification in Sulphate Attack Prevention	139
7.2: General Control Regime for Concrete Durability Design	148
8.1: Sample CRYSTAL rule-tree	167
9.1: Sample CRYSTAL development system screen	173
9.2: Initial Screen of the DEX system	174
9.3: Exposure Conditions for Concrete	176
9.4: Sulphate Distributions in England & Wales	177
9.5: Mix Proportion Adjustments Form	178
9.6: Initial Screen of the SO3 module	184
9.7: Sulphate Risk Indicators	185
9.8: Soil Test Results Input Form	186
9.9: SO3 results screen	187
9.10: DEX Blackboard Organisation	195
9.11: Initial Input Results Screen	197
9.12: Design Dependencies	199
9.13: Rule Fragment of a Dependency Relationship	200
9.14: Rules produced by Automatic Induction from Examples	204
10.1: Modified Mix Design Table	213
10.2: Visual Interpolation in Mix Design	214
10.3: Ion Concentration Predictions	218
10.4: Effect of pfa on Rate of Reaction	220
10.5: LOTUS123-CRYSTAL Export file template	222
11.1: A CRYSTAL Rule Implementing Hypertext Links	233
11.2: A hypertext screen fomr the DEX User Guide	234
11.3: The result of selecting a Hypertext button	235
11.4: Hypertext as a Means of Obtaining User Input	236

Table of Tables

1.1: BRE Publications used as Knowledge Sources 13
3.1: Constraint Classifications 45
9.1: Cover Required for Steel In Various Exposure Conditions 191
9.2: Data dependencies in DEX 194
A.1: Contents of Na₂O and K₂O in Clinker Phases 279
A.2: OPC Hydration: Constant Values 280

Chapter 1: Brief and Scope of The Project

1.1 Introduction to Expert Systems

Expert systems are a by-product of Artificial Intelligence (AI) research. Briefly described, an expert system is a computer program that is competent in an area that would normally be thought to require some degree of human expertise. In addition expert systems usually comprise a number of distinct components that distinguish them from other types of programs. These components are usually identified as the *knowledge-base, inference engine, knowledge-base editor* and the *user interface*, [Kriz, 1987]. Expert systems are a maturing technology and there are countless numbers of journals, papers, text- and reference-books available. Chapter 2 of this thesis will describe some of the current research interest in expert systems and provide a number of suitable references.

The expert system project described by this thesis was undertaken at Aston University Department of Civil Engineering in collaboration with the Department of the Environment Building Research Establishment (BRE). The project was jointly funded by BRE and the Science and Engineering Research Council (SERC). BRE has itself instigated a number of expert systems projects: Fire Regulations, [Stewart, 1986]; BREDAMP for building damp diagnosis, [Allwood et al, 1988] and BREXBAS for Building Management, [Shaw, 1989]. In common with other expert systems

in the United Kingdom these applications are typically small and rule-based with less than 100 rules and implemented on a PC-based commercial shell, [Bundy, 1987].

As expert systems move from the academic to the commercial domain so it becomes more difficult to assess the extent of work in the field. Several authors over the last few years have attempted to survey the extent of expert systems uptake. Bramer, [1986], reported that the majority of the papers at the first technical conference run by the British Computer Society specialist group on expert systems (BCS-SGES) in 1981 were American in origin and were concerned mainly with suggestions for the uses of expert systems in the United Kingdom. By the time of the sixth conference, [*ibid*], many practical applications had been developed.

An estimate of what can be loosely termed the US expert systems 'market' growth of at least 30%, [Roesner, 1988], is supported by a survey of the size of market in the USA and Europe, again with 30% growth, [O'Neill & Morris, 1989]. This survey also reports on rates of expert system project completion and usage of various implementation methods, [*ibid*]. Typical size and completion rate of expert systems projects in Netherlands is given by Lippolt, [1990]. Comparison of these two reports shows that about 30% of software houses had initiated at least one expert systems project. Most projects are started in order to gain technical experience rather than at the request of an end-user and only 25-50% of such projects are carried through to full operational

status. Rada, [1990], conducts a survey of large ALVEY and EEC-funded projects but indicates that the numerical majority of projects (65%) in the United Kingdom are still small scale PC shell-based systems.

The immediate goal of most expert system projects (even those intended for simple exploration rather than user-need) is to provide a solution to some problem that is not amenable to traditional computational methods; data-base management or mathematical models, for example. Somerville, [1984], describes the domain of concrete durability itself as requiring "*knowledge from a number of disciplines*"; not lending itself to "... elegant theories or simple numerical calculations" and finally as intrinsically qualitative rather than quantitative. Chapter 2 will discuss when expert systems are an appropriate method for a given problem (see Palmer & Mar, [1988], for example).

Engineering design is seen as a fruitful subject for expert systems applications. Simmons, [1984], lists the efficient search of large design search spaces and the provision of expert system assistants for low-level (routine) design tasks among the potential benefits. Chapter 3 will review and discuss some of the theories of design that provide foundations for expert systems in the domain. Chapter 4 will give examples of expert system applications in engineering design in general.

There are, in addition, a number of generally accepted benefits that expert systems can bring to any non-trivial domain. Representation of the knowledge separately from the rest of the system entails that it can be easily incremented and checked for consistency and completeness.

The knowledge can then be corrected and effectively clarified for both the end-users (who may not themselves be experts in the field) and the original expert. Another obvious benefit is that once an expert system has been created it is, to some extent, an insurance against the loss of rare and expensive expertise and may also be an effective form of dissemination.

O'Brien et al, [1987], among others, comment on the need for a much greater degree of 'technology transfer' between researchers and the engineers who actually design concrete. The same author, [*ibid*], also proposes relieving the engineer of the necessity for performing the more mundane aspects of design by creating a number of standardised concrete 'alloys'. Expert systems could potentially address both these issues. Frohnsdorf et al, [1988], and Wager, [1987], see potential uses of expert systems within civil engineering as part of integrated systems incorporating databases, financial models and as intelligent front-ends to suites of complicated software such as finite element analysis packages.

1.2 Introduction to Concrete Design

Concrete design is typically guided by British Standards and BRE Digests. These provide advice on mix-constituent selection and proportioning in order to ensure that a mix achieves the strength and workability required and also has some protection against potential durability problems. These reference documents summarise the experience of many experts in the field and cover most commonly encountered

conditions. Popovics, [1987], provides a summary of the mechanisms of durability problems. Chapter 5 will describe methods of concrete mix design and each of the common factors relating to concrete durability.

Problems arise in practice from the difficulty of ensuring that a design complies with all relevant advice on potential durability problems. Firstly there is the practical difficulty of compiling advice from many diverse and cross-referenced sources. Rodway, [1985], for example, lists over 200 separate factors affecting concrete durability. Additionally, many of the materials used in concreting are changing, for example the strength properties of cement, [Cusens, 1984 and Pomeroy, 1987].

A second source of difficulty arises from the possibility of receiving contradictory advice from consideration of different potential durability problems. It is possible for example that prevention of alkali aggregate reaction by specifying a maximum cement content could conflict with a minimum cement content specified to prevent sulphate attack. Finally, situations can arise that are not covered by the existing standards and digests. In practice the response to this problem would be to consult with an expert on that particular area of difficulty.

1.3 BRE Digests as Knowledge Sources

The norm for expert system development is that the knowledge is obtained directly from an expert (or experts) in the domain. After some initial knowledge acquisition in the domain it was decided to avoid the imposition of large time commitments on the available experts by using

textual knowledge sources in the early stages of development. Thus the scope of the project was initially limited to those problems of designing durable concrete covered by Building Research Establishment digests, [BRE 1981, 1982a, 1988 and Teychenné et al, 1988]. These digests are listed below in Table 1.1.

Digest	Title
250	Concrete in Sulphate-bearing Soils and Groundwaters
263	The Durability of Steel in Concrete: Part 1 Mechanism of Protection and Corrosion
330	Alkali Aggregate Reactions in Concrete
	Design of Normal Concrete Mixes

Table 1.1: BRE Publications used as Knowledge Sources

There are a number of other advantages gained from using BRE digests. Each digest deals with a well defined sub-problem of the domain. This allows easy organisation of the system into naturally prescribed modules. In terms of McDermott's task domain classification scheme, the concrete durability knowledge contained in the digests is highly 'compartmentalizable' into sub-tasks, [McDermott, 1983]. This knowledge is also relatively complete, consistent and stable.

The quality of the knowledge is important not only for the ease of developing the knowledge-base, but also in the likelihood of user acceptance. The use of the digests emphasises one of the important aspects of expert systems; that the knowledge base contains compiled knowledge, [Hickman, 1986]. In this case the knowledge is pre-compiled

by the authors of the digests. BRE Digests come with an established reputation and confidence in their veracity which should communicate itself to users of an expert system based on the digests.

Finally, the digests provide sufficient variety and interest to demonstrate a wide range of topics in the application of expert system technology to concrete durability design. Chapters 6, 7, 9, 10 and 11 will describe the development of the expert system prototypes and the progressive exploration of some of the research issues in expert systems discussed in chapter 2.

An expert system that implements a set of codes or standards is different from an expert system that merely uses a set of codes as a knowledge source. Codes like those of the British Standards Institution (BSI) have legal weight - any artifact covered by a BSI standard must be demonstrably in compliance with it otherwise the designer or builder must face the consequences in the event of the artifact failing, [Saouma et al, 1989]. In the design of structures in particular the goal of design may be defined as being explicitly to ensure compliance with codes, [Fukuda, 1988].

An artifact as complex in make-up as a concrete mix design is inevitably covered by a plethora of standards for each of the constituents and for each stage of its design, manufacture and use. Each code may itself be complex and daunting in scope and language and contain logical errors and errors of omission. Additionally standards are frequently structured so as to contain numerous cross references to sub-sections and other

standards documents, [Stone & Wilcox, 1987 and Thewalt & Moskowitz, 1990]. There is therefore great potential for expert systems that automate standards compliance checking. Koskela et al, [1986, 1988], developed a system for checking Finnish fire regulations standards; Rasdorf & Wang, [1987], implement a system for American BOCA (building) codes and, similarly, Rosenman & Gero, [1985], a system for Australian AMBUC codes. More examples of code-based systems will be given in chapter 3. Chapter 11 is concerned with hypertext representation of complex textual material.

1.4 Expert Cooperation

In addition to the primary use of BRE digests, regular consultation sessions with actual experts were undertaken. The initial contacts with experts at BRE confirmed the suitability of the digests as useful knowledge sources. The early interviews also covered some wider issues such as the order in which a concrete design expert would tackle the durability problems raised by a particular design.

Since the digests formed the primary knowledge sources it was important that the original publishers of the texts were available to check that misinterpretation of the digests did not occur. This expert involvement consisted of sessions where the results given by each module for exhaustive test-cases were checked. Mistakes were followed through and clarification of the intended results lead to correction of the knowledge coded in the module.

The project relied on the BRE and Aston University concrete experts in a second sense; as potential users of the system. It was not felt to be necessary to answer the question of who should be the intended users of a concrete design system until the later stages of development when the feasibility of expert systems in the domain had been investigated. Thus BRE employees (not necessarily concrete experts) were used as judges of such implementational issues as screen ergonomics and the degree of help and justification required.

As will be discussed later in the relevant chapters, use of the digests as knowledge sources revealed some gaps and inconsistencies in them. Any extension of the expert system beyond the immediate limits of the digests would require a much greater degree of cooperation with concrete durability experts in order to rectify these problems. This would involve negotiating some of the obstacles familiar to knowledge engineers. Particular problems include the difficulty experienced by experts in articulating their knowledge, [Chung & Kumar, 1987].

A second potential difficulty is that experts may be reluctant to make their heuristics available to an expert system. This is understandable given that expert systems usually lack the 'common sense' grounding and paradigms underlying any human ability in a domain. This combination of basic training, experience and unstated assumptions is what enables an expert to know when a 'rule of thumb' is really

applicable,¹ [Collins et al, 1985]. Regular contact with the experts in the earlier stages of the project should allow any further development of the system to proceed from a basis of relative familiarity and certainty and help resolve some of these difficulties.

1.5 Specific Goals of the Project

The initial goal of the project was the creation of an expert system in the domain of concrete durability design. This ambition was inspired by DURCON an Expert System for durable concrete published by the American National Bureau of Standards centre of Building Technology, [Clifton et al 1985, Clifton, 1986, Kaetzel & Clifton, 1986, Clifton & Oltikar, 1987, Frohnsdorf et al, 1988 and Frohnsdorf, 1990].

The DURCON System is based on a code of practice "the Guide to Durable Concrete" published by the American Concrete Institute (ACI) Committee 201, [ACI, 1977]. The project therefore was intended to constitute a demonstration of the utility of implementing UK codes of practice as expert systems. The initial literature search revealed several concrete durability expert systems in addition to DURCON: COMIX, [Miller, 1985], and CONDURAS, [Rajkumar et al, 1987]². Chapter 4 will discuss the existing expert systems in the field of concrete design.

¹ More precisely, the reluctance of experts' may stem from an inability to be confident that all the preconditions of a heuristic rule have been made explicit in the expert system implementation of that rule.

² None of these systems treats the problems of concrete durability in sufficient depth to exploit the full power of expert systems.

Rowlinson, [1987], outlines a recommended development cycle for expert systems. In many ways this is an ideal for expert systems inspired by a commercial need. The endeavour to explore the use of expert systems in the design of durable concrete diverges significantly from this cycle by virtue of being academically rather than commercially motivated. One of the goals of the project is therefore to verify that the domain is suitable for an expert system solution³.

As part of the development of this expert system, it was necessary to conduct a thorough investigation of alternative implementation methodologies and fundamental theories. Chapter 2 will describe various knowledge representation and inference methods. Chapter 8 will discuss the criteria for evaluating alternative implementation methodologies and will describe several AI languages, tools and shell systems.

Most of the design theories described in chapter 3 were not obviously applicable to the design of concrete mixes and none were compelling enough to warrant use as the theoretical foundation for the project. There was a similar lack of evidence for the suitability for any particular knowledge acquisition method. Trimble, [1988], states as axiomatic that, in practice, most expert systems are assembled *ad-hoc* with no use of formal methodologies.

³ Laufman in his comprehensive discussion of expert system project feasibility comments that the suitability of an expert systems approach can often not be ascertained until after the application is complete [Laufman et al, 1990].

A literature review revealed a number of topics within the field of expert system techniques that showed interesting research potential and which could be addressed by the project. Particular topics of interest included the complexity of large modular systems, the reconciliation of conflicting advice and constraints; and the design of user and other interfaces, [Sauers & Walsh, 1983].

A possible development of the prototype systems into a full commercial system would be considered as a final goal of the project once the exploratory phase had been adequately concluded. The publications concerned with DURCON serve to illustrate the scale of development efforts required for such a system. Few actual implementational details have been given but the publication dates of the papers concerning DURCON cover a period from 1984 to 1988 when development can be seen to be in process. In addition, a total of 9 authors have contributed to publications describing DURCON, [Clifton et al 1985, Clifton, 1986, Kaetzel & Clifton, 1986, Clifton & Oltikar, 1987, Clifton & Kaetzel, 1988, Frohnsdorf et al, 1988 and Frohnsdorf, 1990].

Additional material from the literature confirmed concrete durability as a sufficiently complex and broad domain to potentially require much more than 2-3 man-years of expert system development effort. For this reason

it was decided to focus development on design and to ignore the problems of diagnosis and remedy of faulty concrete. In addition, the system was to be limited to routine design only⁴.

In addition to the exploration and development endeavours outlined above; it was hoped to use the products of the project to raise the awareness of the potential for expert systems in concrete design among civil engineering practitioners. Some aspects of the work have been published in the Magazine of Concrete Research, [Wilkins & Elgy, 1991a], and accepted for presentation at the Civil-Comp '91 conference, [Wilkins & Elgy, 1991b].

To conclude this introductory chapter the following list summarises the goals of the project:

- * Problem analysis and knowledge acquisition through interviews with experts and study of standard source texts - BRE digests.
- * Identification of current research issues in expert system applications
- * A thorough literature search into the use of expert systems in the broad domain of engineering design and concrete design in particular.

⁴ *Routine* design is defined in Gero's theory of design as the refinement of Prototypes as opposed to the enhancement or creation of new Prototypes, [Gero, 1988 and Gero et al, 1988]. Brown and Chandrasekaran denote this as *Class 3* design. Chapter 3 deals with this in more detail.

- * The development of a number of small expert system modules prototypes. Each module covering a sub-problem as defined by an individual BRE digest.
- * Evaluation of a number of different expert system development methodologies including languages, tools and shells and including different platforms - mainframes, PCs and work stations.
- * Exploration of the applicability of expert system techniques to the domain and of the utility of using BRE digests as knowledge sources.
- * The consolidation of individual modules into an overall system that would be a design orientated implementation of currently published BRE concrete durability knowledge. This system would be intended for Beta-testing at BRE and for possible further development.

Chapter 2: Overview of Expert Systems

2.1 Introduction

This chapter is intended as a brief description and overview of expert systems with the emphasis on those areas of expert systems technology that are of current research interest. Although expert systems are used in many domains and problem areas this chapter will pursue the central theme of the thesis by focussing on engineering expert systems rather than expert systems from other domains such as medicine, geology or law. It should be remembered however that most of the discussion in this chapter is equally applicable to expert systems in other domains.

As mentioned in chapter 1, expert systems are a rapid growth area of computer application development. Consequently there are a vast number of conferences, reference books and journals that have expert systems as a central topic. There are a number of older general references that are most often cited, [Hayes-Roth et al, 1983 and Alty & Coombs, 1984]. In addition there are a growing number of reference works with a narrower scope. For example, practical guides to expert systems programming and development, [Hu, 1987 and Vadera, 1989]; and publications targeted at general domains like engineering, [Taylor, 1988]. Taylor, provides a series of discussions and selected bibliographies on many aspects of AI in general and expert systems in particular.

Most descriptions of expert systems contain references to a typical architecture which will include three or four generally accepted components. These components are a knowledge base and editor; an inference engine and a user interface. A knowledge base is a collection of knowledge about the specific domain the system is concerned with. A

knowledge base editor is a program that facilitates the creation of a knowledge base, checking and enforcing the correct syntax for the representation used. An inference engine is a program that uses the knowledge base to respond to queries in a manner similar to that in which a human expert might respond. Finally the user interface is a program which makes it possible for people to use the system who are not themselves familiar with its basic mechanism and programming.

The basic theoretical and operational background of knowledge acquisition, knowledge representation, inference and user interface are discussed in depth in the various references mentioned above. The remainder of this chapter will attempt to provide a general discussion of each of the basic aspects of expert systems in order to place the work described in the later chapters within the context of some of the issues that are of current interest. Muller, [1986], provides a selective discussion of some of the practical concerns in expert system expected to become research subjects over the period covered by the subject of this thesis. Muller's concerns along with those raised by other commentators, [Taylor, 1986, for example] will be addressed in the discussions in later chapters.

2.2 Representation/Inference Techniques

Adeli, [1987], lists four main types of knowledge representation used in expert systems - logic, rules, frames and semantic nets. In practice rules and frames are the most common choice and it is these that Adeli recommends for expert systems in the domain of structures. These two forms of representation in particular are described in many references. Hu, [1987], for example, gives a practical treatment of how to implement

rules and frames as well as logic systems. In addition, the later chapters of this thesis will contain numerous examples of the use of rules, and rule-based inference in the design of durable concrete.

The representation of the knowledge in a knowledge base is declarative rather than (or in addition to being) procedural. Whether production rules, frames or some other representation is used; the representation can, to some extent, reflect the way in which humans actually conceptualize the domain. This makes the knowledge easier to implement and to understand. The use of an inference engine that is implemented apart from the knowledge base contributes to the clarity of the knowledge base.

The rule-based representation paradigm is easily described. Consider a simple rule of the following form:

IF <precondition> THEN <post-condition>

or:

IF <antecedent> THEN <consequent>

Inference proceeds by matching the precondition against a working memory of facts. If a matching fact is found; the post-condition is entered into working memory as a conclusion. Other rules may then be triggered by matching with the new fact. This is simple forward-chaining inference.

Some implementations also allow backward-chaining where the inference is triggered by a goal. To satisfy a goal the knowledge base is searched for a rule with the goal as its consequent. If a fact matches the

antecedent then the goal is satisfied, otherwise a sub-goal is created from this antecedent. Backward-chaining continues until the antecedent-consequent chain has been satisfied all the way back to the original goal.

Differences arise among implementations of rule-based systems from the diverse methods of dealing with a number of operational considerations such as those listed below:

- * mixing forward- and backward-chaining
- * the provision of an 'or' operator as opposed to multiple rules with the same post-conditions (rule-disjunction)
- * which rule to 'fire' if more than one matches the existing facts (conflict resolution)
- * whether to allow a given rule to fire more than once in an inference chain
- * the attachment of text that is used in explaining the rule or justifying consequent inferences to the user (so-called canned text)
- * the use of consequents with 'side-effects' such as running an externally programmed function or displaying a graphic as opposed to purely declarative consequents
- * allowing multiple antecedents and consequents (conjunction)
- * the ability to use non-atomic 'facts' such as arrays or other composite structures

Another facility offered by various inference mechanisms is the provision of some method of reasoning with uncertainty. Quinlan, [1983, 1986], describes three different methods of dealing with uncertainty in a domain and adds a fourth method - INFERNO. These four methods utilise subjective measures of probability (from the Prospector system); coupled

measures of belief and disbelief (from the Mycin system) and a range of degree of probability (Dempster-Shafer systems and INFERNO itself). The details and precise differences between the methods are not central to this thesis.

For a number of reasons, none of these methods of reasoning with uncertainty are relevant to the current project. The design of structures, of which concrete durability is a sub-domain, has potentially more profound consequences of failure than most other domains. Because of the implication of expert level competence, expert systems are more than mere tools. Therefore the liability relating to their use extends more firmly to the knowledge engineer and expert than is the case with traditional analysis packages for example, [Sales & Topping, 1985]¹. If the knowledge engineer, expert or user has to assign subjective measures of uncertainty to data or knowledge this exposes each to greater risk of liability in event of failure.

Allwood et al, [1987], and Bundy, [1987], advise against the use of uncertainty when the meaning of the measures of uncertainty is not clear. The meaning of the various measures of uncertainty can, of course, be made clear to the people involved in a project in order to make their use more consistent. The consistency required would however increase development costs and training costs for the end users.

As will be seen in chapter 5 there is some disagreement among experts in the domain of concrete durability. This could be reflected by a use of

¹ Gero, [1990], makes apposite reference to the code of Hammurabi:

If a designer-builder has designed-built a house for a man and his work is not good and if the house he has designed-built falls in and kills the householder, that designer-builder shall be slain (p27)

certainty factors in the knowledge with respect to the various expert sources. Reboh, [1983], proposes presenting the user with a choice between conflicting knowledge sources and thus passing on the responsibility for deciding absolutely between them.

Fortunately the existence of codes of practice for most aspects of structural design entails that in the majority of cases there is an ultimate authority. Thus the problem of conflicting and therefore uncertain knowledge only arises if the codes of practice are incomplete or self-contradictory. Chapter 9 will discuss these issues in the context of the use of BRE digests as knowledge sources.

Frames are used to represent groups of related facts, objects, and attributes. Each object belongs to a class and classes are organised in a hierarchy that relates objects. An object is represented by a frame. Each frame has a number of slots. Each slot can be another frame, description, specification, procedure, default value or relationship. Reasoning in a frame based system is based on inheritance - passing on a value from an object in a super-class to an object in a sub-class.

A simple example, (from Rychener, [1988]), illustrates the frame concept. From this example it is easy to see how the molecular weight of carbon-dioxide could be inferred from the atomic weights of carbon and oxygen and the structure of a molecule class frame.

The frame- and rule-based paradigms are not necessarily mutually exclusive. Frames can be seen as a representation that emphasises the organisation and relationship of objects when they have a homogeneous structure. Rules emphasise the knowledge used to relate objects when

```
{ { carbon
  is-a: chemical-element
  symbol: C
  atomic-number: 6
  atomic-weight: 12.01
  valence: 4
  family: IV-A
  similar-elements: silicon germanium tin lead
}
```

Figure 2.1: Frame Representation of carbon element

there is less structure. Rules could exploit structure and homogeneity and frames could make use of rule-based inference to supply values for slots.

From another perspective, frames could be viewed as meta-knowledge about a domain. Rosenman et al, [1986b], identify two specific types of domain independent meta-knowledge concerning the scope and the nature of knowledge in the domain. The scope, given in terms of a list of possible values for an object, clearly corresponds to the superclass of the object. The nature of the knowledge in the domain is simply whether a value is inferred (inherited) or must be supplied by the user.

Georgeff & Bonollo, [1983], describe a scheme for developing procedural expert systems. This scheme ensures that procedural knowledge is executed in a specific order and does not rely on some implicit ordering arising from the structure of the knowledge-base². Chandrasekaran, [1985], and Bundy, [1988], both comment on the problems of confusing

² Explicit use of procedural knowledge does not thereby transform an expert system into a traditional algorithmic program in the same way that conditional statements do not automatically make an algorithm into an expert system, [Clancey, 1989].

domain knowledge and control knowledge. This specific ordering of knowledge or distinction between procedural and declarative knowledge is another type of meta-knowledge. Each of these three types of meta-knowledge will be illustrated in the description of CRYSTAL in chapter 9.

2.3 Knowledge Acquisition and Elicitation

Knowledge acquisition is a crucial part of expert system development and is usually cited as the main bottleneck in progress, [Hayes-Roth et al, 1983, Chung & Kumar, 1987, de Greef & Breuker, 1985 and Hart, 1985]. Accordingly much expert systems research has focussed on improving the situation by various means. There are two fundamentally different approaches to the problem. Induction and other machine learning techniques are supposed to allow a system to acquire its own knowledge.

The other approach is to develop the techniques used by knowledge engineers to elicit human expertise. The collection edited by Gaines and Boose, [1988], contains essays concerning each of the major approaches to knowledge acquisition and detailed descriptions of some of the techniques. Forsyth, [1989], has edited a number of essays on different aspects of machine learning and Shapiro, [1987], describes the techniques of induction and conducts a number of case studies.

Induction methods attempt to draw general rules from a database of specific examples. Chapter 9 will contain an illustration of the limited use of induction relating the properties of sandstone aggregates to alkali aggregate reactivity, [Collins, 1986]. Another technique that could be included under the broad heading of 'machine learning' is the natural

language analysis of texts. Chapter 4 will describe some attempts to use this technique to build knowledge-bases from standards and codes of practice.

Knowledge elicitation is the task of retrieval and distillation of the expertise of the human domain experts into a form of knowledge that can be analysed, represented and finally used by expert systems. It is no exaggeration to say that an expert system is only as good as its knowledge-base and that therefore knowledge acquisition is the most important aspect of expert systems development. A number of formally structured techniques have been developed that purport to be more efficient for certain domains than the classic AI 'technique' of rapid prototyping.

Chung and Kumar, [1987], include rapid prototyping among their list of knowledge elicitation methods and rate this as highly as the other methods evaluated. Rapid prototyping is the unstructured gathering of knowledge from wherever it is available and representing it in whatever seems the best formalism. A run of the expert system 'tests' the new knowledge against the evaluation of a human expert and the process is then repeated. This is the so-called Run-Understand-Debug-Edit (RUDE) cycle, [Partridge & Wilks, 1987].

Descriptions of the various formal knowledge elicitation techniques can be found in the literature. Most are based on some variety of interview technique, [Burton et al, 1987, Chung & Kumar, 1987, Hart, 1985, Hickman, 1986]. The interview is structured according to a number of elicitation methods listed below. The written or taped results of the interview are analysed by whatever paradigm is being used.

This intermediate 'epistemological level' knowledge is then translated into the chosen representational implementation. KADS is one example where a structured approach has been implemented as a knowledge acquisition support shell written in PROLOG, [de Greef & Breuker, 1985 and Breuker et al, 1986]. One of the less obvious results of a more formal knowledge elicitation process is that the epistemological insight into the problem allows a level of specification of the system at a lower level than 'model what the expert does'. This lower level of specification is crucial if there is to be a rigorous attempt to validate the system once it has been completed, [St. Johanser & Harbidge, 1986].

Protocol analysis, goal decomposition, repertory grid and multidimensional analysis (card or concept sorting) are all methods that essentially observe the expert at work. At the extreme, Swaffield and Knight, [1990], propose using systems analysis techniques in the knowledge elicitation process. The techniques vary according to the differing emphasis on procedural or declarative knowledge; whether the expert is given real or hypothetical problems and the degree to which the expert or the interviewer control the session, [Hart, 1985, Hickman, 1986, Burton et al, 1987 and Chung & Kumar, 1987]. The suitability of each of the techniques therefore depends on the personalities of the interviewer and expert and on the nature of the domain knowledge, [Burton et al, 1987].³

Each of the methods listed above assume a situation of purity where there is an expert who is not also the knowledge engineer. This may not

³ As an example of the essentially subjective nature of these knowledge elicitation methods, Burton evaluates protocol analysis as being particularly slow and unproductive but Chung and Kumar come to the opposite conclusion.

be a realistic or useful assumption. Collins et al, [1985], conclude that it is in fact necessary for the knowledge engineer to be steeped in the 'tacit knowledge' of the expert and should undergo a period of apprenticeship in the domain if this background is lacking⁴.

Chung and Kumar, [1987], also report that the knowledge elicitation process is made easier by having a knowledge engineer who has undergone "at least three to six months" gaining familiarity with the general information about the domain before the first formal interviews with the domain experts. An obvious alternative approach is to create a knowledge-base editor or acquisition tool that is sufficiently user-friendly to allow the expert himself to build the knowledge-base, [Guida & Tasso, 1983].

No formal method can realistically promise to get the human expertise completely and correctly into the knowledge-base in one iteration. Describing standard software methodology, Lenat, [1986], points out that they have

"proven of little use in AI, a field which by definition tackles ill-structured problems" (p65)

Turner, [1985], states that being able to specify the problem solving techniques in advance of implementation is an indication that an expert system may not be a good solution. There is therefore a tension between the recognition of a problem as being suitable for expert system solutions if it is ill-structured and the desire to use structured methods

⁴ Using a colourful metaphor, Collins et al, distinguish the *dumplings* (specific knowledge items) from the *chicken soup* of tacit knowledge. This soup is lost in the transfer of knowledge from the expert to the end user via expert systems which are like *colanders*.

to facilitate knowledge elicitation and analysis. All methods must therefore use some degree of iterative prototyping and conversely in most situations prototyping should involve some formal interaction with an expert and analysis of the knowledge.

Finally, Allen, [1986], writing specifically about engineering expert systems recommends starting implementation "at the highest programming level possible". The process of knowledge elicitation results in increased awareness of the lower levels of the problem-solution space but this does not invalidate the basic tenet of developing the system in a top-down fashion.

2.4 Interface Techniques

Adeli, [1987], defines an expert system as:

... an "intelligent" interactive computer program that can play the role of a human expert by using heuristic knowledge... (p6)

It is true that most 'traditional' expert systems are interactive but this is not a necessary part of the definition. In fact, expert system users can be alienated by repetitive or protracted interactive sessions. It can be an advantage if an expert system is able to obtain its basic data from a database, for example, as suggested by Wager, [1987]. One of the possible uses of an expert system based on codes of practice is passive design or compliance checking, [Kalay et al, 1987]. In this type of system there is very little scope for human interaction with the system.

Given that most expert systems will be interactive with human users, the importance of user interface design should not be disregarded. Roesner,

[1988], notes that the user interface of applications programmed in traditional languages is still often less than satisfactory. The ergonomic demands of the user interface provided for expert systems by various languages and shells will be discussed in chapters 6 through to 9⁵. Although these issues are important in the consideration of the knowledge engineers convenience during development and in user acceptance of a system they are fundamentally cosmetic⁶. Rubin, [1988], provides a general reference for these user interface design considerations.

There are a number of areas where the user interface design can raise issues relating to the functionality of a system. For example, a system that is concerned with domains that require spatial reasoning such as traditional CAD or some aspects of structural design should have a graphics based interface. It would also need knowledge representation and inference that could interface to graphical representation and would therefore need to be implemented in a specialised shell and environment, [Balachandran & Gero, 1987]. The domain of concrete durability does not require such facilities but some ability to display graphics is desirable and several examples will be given in chapter 9.

One of the main functions of the user interface is the provision of support to the user by giving explanation as to why questions are asked; how questions should be answered and justifying the conclusions

⁵ A shell is a program combining the features of an expert system that are relatively domain independent such as the inference engine user interface functions and knowledge-base editor. A suitable shell may be used to create a domain specific expert system by the addition of rules or other items into the empty knowledge-base.

⁶ At the climax of his consideration of the users' needs in all five sensory media, Muir, [1987], proposes an ideal user interface device combining the functions of voice recognition and understanding with a fighter pilot's head-up display.

of inferences. These are the three main questions that a user might want answered during a consultation session. Hughes, [1986], and Gilbert, [1987], among others, discuss the need for various other explanation facilities corresponding to the six so-called 'w-words': what, why, when, where, how and who, [Savory, 1986].

There are two main approaches to accommodating this need for user support. These two methods are based on either canned text attached to rules or to rule-trace or frame-topology, [Savory, *ibid*, and Neches et al, 1985]. Both techniques have disadvantages and various researchers have sought means of providing improved explanatory capabilities⁷. For example, Savory, [*ibid*], describes how the TWAICE shell produces explanations from natural language transformations of the rule-trace facility. These issues will be further illustrated and discussed with respect to the DEX system in chapters 7, 9 and 11.

2.5 Hybrid Expert Systems

Many researchers now recognize that expert systems can make profitable use of many knowledge representation formalisms, [Sloman, 1984]. Dym et al, [1988], for example, describes a system for checking architectural code compliance that uses both frames and production rules. In addition many systems can interface to existing software such as spreadsheets, mathematical models or databases, [Wager, 1987]. In conjunction with the high degree of modularisation exhibited by most large expert systems this use of diverse data and knowledge sources often necessitates the

⁷ Canned text explanations add greatly to memory requirements in order to accommodate the text. Rule-trace explanations typically require some degree of expert systems know-how by the user. Both methods are inflexible and produced somewhat stilted results

use of a blackboard architecture. Kumar & Topping, [1988], for example, report a blackboard-based system for structural design that uses structural analysis programs written in FORTRAN 77.

Some recent work in the field of concrete durability has focused on the development of mathematical models to predict and explain some aspects of concrete performance. An example of this sort of model is Prof. H.F.W. Taylor's paper on the prediction of hydroxyl alkali concentrations in cement paste pore solutions, [Taylor, 1987]. There are also models of depth of carbonation and other properties of concrete and its constituents, [Parrot, 1986, Pomeroy, 1987 and Samarin, 1987]. Often the mathematics involved in these models require functions that are not available directly to AI languages or shells or, if available, are inefficient. This necessitates the use of modules written in languages like C or FORTRAN which have the necessary functions and which can be called by the expert system at the appropriate moment.

Corby, [1986], gives a concise definition of blackboards:

Blackboards are global memory-resident databases, accessible through a uniform protocol and tailored for expert source cooperation (p95)

Many examples of blackboard-based systems can be given. Derrington, [1987], describes a prototype blackboard system for control of the design process in CAD. Dixon et al, [1984], give a number of examples from mechanical design. Sriram, [1986, 1987b], reports on the DESTINY system that performs integrated structural design. DESTINY will be described in more detail in chapter 3. Many more examples and essays on blackboards are given in Englemore & Morgan, [1986].

Blackboards can specifically facilitate the control of interaction between many knowledge sources and the creation of explanations that involve multiple modules. Often these problems are handled by modules dedicated to control and explanation tasks. See for example, Corby, [1986], on the use of an explanation module in the SMECI shell and Gerstenfield et al, [1987], for an example of a system that manages other expert systems.

2.6 Intelligent Expert Systems

The use of the word "intelligent" in most definitions of expert systems begs many questions (for example, Adeli's definition quoted above, [*ibid*]). It is not within the scope of this thesis to attempt to define intelligence beyond two simple observations. Evans and Deehan, [1990], claim that the people who others find intelligent are "those whose minds work very like their own" (p9). Minsky on the other hand stipulates that:

Our minds contain processes that enable us to solve problems we consider difficult. "Intelligence" is our name for whichever of those processes we don't yet understand. (p71)

Thus intelligence is recognized in behaviour that is either understandable to the observer (for example a mathematician following the proof of another) or is mysterious to the observer (for example a layman watching the same mathematician).

This tension in the definition of intelligence has repercussions on the provision of explanation facilities. When a system exhibits intelligent behaviour this intelligence can be made less mysterious by examination of the rules that govern the behaviour of the system, that is, by

rule-tracing explanation as mentioned above. Conversely, rule-tracing, by resolving the mystery of the conclusion renders it un-intelligent. Simple rule-trace explanations are often thought to be an indication of a trivial expert system application.

Finn & Reinschmidt, [1986], in their description of expert systems, specify that:

An expert system is designed ... so that people ... can have access to the judgemental knowledge and experience of the human expert. (p813)

This implies that an expert system is potentially much more than a system that can simply solve a problem and explain its conclusions. Experts can act as references and repositories of relevant facts and knowledge that can be accessed by general enquiries, in addition to solving specific problems. Expert systems should be able to reproduce this capability since it would not seem to involve knowledge beyond that used in the problem solving task. This is not the same as using corresponding knowledge in both design and diagnosis, for example, since in these cases there is obviously some non-trivial use of meta-knowledge that is specific to the task.

Adeli, in his definition of expert systems, states that:

... an expert system can make educated guesses, recognizing promising approaches, and avoid blind search... (ibid)

Similarly Kriz, [1987], refers to expert systems exhibiting:

... problem solving in those areas and at that level of performance that is usually achieved by human experts... (p11)

A human expert can perform in the way Adeli and Kriz describe. However most expert system performance is at a level far below human expertise. To use Gero's description of different levels of design activity (described in detail in chapter 3), expert systems are only usually competent at routine design, that is, the refinement of design Prototypes. Expert systems can use heuristics to narrow search space, [Simmons, 1984]. Human experts by comparison are able to use "educated guesses" and "promising approaches" to enhance and generate new Prototypes, [Gero, 1988 and Gero et al, 1988].

Expert performance at this level is perhaps a valid goal for expert system developers but such systems would have to perform at the level of AM, [Lenat, 1982], which, it was claimed, could formulate new mathematical concepts and conjectures. The equivalent expert performance in design would be something like the invention of the air-supported roof to avoid the weight and cost of other methods of covering large spaces, [Gordon, 1978, and Gero, 1988], a good example of the generation of a new Prototype.

Clancey, [1989], views expert systems as qualitative models. This modelling is inherent in the pattern of inferences made by a system. From this viewpoint, knowledge acquisition should be more concerned with the adequacy of the model than emulating the reasoning process of the expert. The inadequacy of many expert systems as qualitative models stems from the lack of deep knowledge and reasoning ability. This failing is most often apparent in the brittleness of the expert systems' competence, [Lenat et al, 1986], and in the deficiency of the explanation facilities, [Dieng et al, 1987].

There is a difficulty in explaining exactly what is 'deep knowledge', [Cohn, 1985]. One type of deep knowledge is the causal knowledge that is summarised by the heuristics used by an expert system, [Cohn, *ibid*]. Kramer, [1987], cites mathematical knowledge as another type of deep knowledge. Such mathematical or causal knowledge can overcome brittleness by enabling an expert system to reason from first principles in cases that are not covered by existing heuristics.

It is for this reason that Lenat et al, [1986] and Steels, [1986], argue that deep knowledge can overcome the knowledge acquisition bottleneck⁸. The provision of deep knowledge is a potentially boundless task as the need to know 'why?' can extend indefinitely into the realms of physics or psychology. A number of undertakings are concerned with extending knowledge-based systems into these areas, [Lenat et al, 1986, and Hayes, 1983]. Chapter 9 will discuss the scope for such deep knowledge in the domain of concrete durability design.

⁸ If a system has access to the fundamental knowledge of a domain it does not need exhaustive case-specific heuristics. Deep knowledge improves explanation facilities by making the reasoning leading to a conclusion more explicit. The user can be shown the 'real' reason 'Why?' rather than being defrauded by mere restatements of shallow reasoning. In contrast, however, Coyne notes that the use of causal reasoning and explanation may itself be fraudulent.

Chapter 3: Engineering Design Theory

The central theme of this thesis is to explore issues in the development of expert systems that design durable concrete. The theory and practice of design, in itself, is not a primary concern of the work. The purpose of this chapter, therefore, is to consider the different perspectives and models of design revealed in the publications of various researchers concerned with the implementation of design-orientated computer systems¹. The primary benefit of such an undertaking is the definition of:

... sets of terms, models, intellectual structures and, ultimately, the controversies that enable us to talk about design. [Coyne, 1990] (p72)

Tong, [1987b], proposes a framework for evaluating knowledge-based models of design from three different perspectives: the *knowledge* represented; the *functionality* of the process and its *implementation* as a system. This discussion will follow this framework. The knowledge and functionality of design will be discussed in this chapter and some actual implementations will be illustrated in chapter 4.

3.1 A Simple Design Task

It is possible to conduct a design without being in possession of any conscious theory of what design is. It is also possible to design an artifact without any conscious theoretical grasp of the mechanics or physics of the artifact or of the task it is supposed to fulfil. At its

¹ The reader is referred to Coyne, [1988] and, Coyne et al, [1990], for an exhaustive and rigorous treatment of the themes outlined in this chapter.

simplest design need not be a deliberate task at all, [Coyne, 1990]. However, all artifice involves design, conscious or otherwise, by definition.

Consider the very simple design of a method of crossing a stream. The designer is faced with an objective - to create an artifact which will bear a given weight across a stream and which will be capable of repeating the performance a number of times. The designer also has some physical constraints on the design such as the depth, width and current of the stream and the local availability of materials.

Faced with these objectives and constraints the designer will be possessed of some relevant experience with the strength of various materials. There will also be experience of the basic techniques of bridging such as spanning a gap with a beam; placing a number of stepping stones or building a raft. The designer will proceed by adopting one of his basic techniques and attempting to implement it. This attempt may fail; perhaps the span is too broad for a tree-trunk bridge or the current too strong for the size of the available stepping stones.

The designer may try again with a longer tree trunk or heavier stones. He may use improved construction techniques like cooperating with another bridge-builder to use their combined strengths to obtain a longer tree trunk. He may decide that another bridging technique is more appropriate or develop a new technique such as 'bridge the smaller spans between a number of stepping stones'. If all else fails he may consider a different technology such as raft-building. There is no guaranteed solution for a given level of bridge or rafting technology.

If the bridge is successfully constructed it will be used for some time and then will fail due to material failure (wood rot) or unforeseen circumstances (heavy flooding or two people crossing at once). The next bridge designer will benefit from the experiences of the first bridge designer. For this to be possible, there must be a shared set of terminology and ideas such as types and properties of tree-trunks.

This simple example illustrates most of the factors present in the *common-sense* view of design. Most research in the application of expert systems to design emphasises one or more of the following:

Objectives

Constraints

Materials

Technology

Basic Prototypes

Redesign/Previous Experience

Terminology

Design description

The rest of this chapter will focus on some of the more prominent knowledge and inference characterisations in general and design models in particular. Examples from the domain of concrete durability design as they arise in the later chapters will refer to the discussions in this chapter.

3.2 A Model of Deliberate Design Activity

Deliberate, low-level, design procedures such as measuring the distance to be spanned and the lengths of the available tree trunks will reduce the effort wasted on trial-and-error attempts. The product of high-level

design activity is a design description. The design description should contain sufficient detail to enable the manufacture of the designed artifact, [Gero, 1990]. Deliberate design will also facilitate communication of successful design and construction principles.

Aldridge et al, [1986] describe design as *iterations of the process of balancing objectives against constraints* (p445). More specifically, Chandrasekaran, [1990], defines design as a mapping between a problem space and a solution space of assemblies of components by process of search among sub-assemblies. Gero qualifies this mapping search process as an exploratory operation with the provision that design involves learning and the discovery of emergent features of the on-going design task, [Gero, 1990]².

In one sense the specified objectives are constraints on the desired functions of the artifact under design. However these can be distinguished from another broad class of constraints in that objectives generate the search space for a design, [Gero, 1990]. For the purposes of this chapter a *constraint* will be loosely defined as any specification or other restriction that narrows the size of the potential solution space, [Taylor & Corlett, 1987, and Chandrasekaran, 1990].

Brown & Breau, [1986], and Sriram & Maher, [1986], among others, discuss the various types of constraint found in design. Classification of constraint types is possible along a number of axes. Some of these classification schemes are summarised in table 3.1.

² The essential difference between search and exploration is the lack of a well defined search-space in exploration. This characterization of design as lacking a defined search-space underlies the inclination of at least one research programme to abandon strong attempts to model design and to develop design expert systems, [Smithers et al, 1990].

<u>Origin</u> - [Brown & Breau, 1986] Implicit, In-Place, Inherited, Accumulation
<u>Origin</u> - [Kumar & Topping, 1988] Internal, External, Designer, Client/User, System-based, Regulatory
<u>Function</u> - [Sriram & Maher, 1986] Synthesis, Interaction, Causal, Parametric, Evaluation
<u>Function</u> - [Taylor & Corlett, 1987] Fixed, Variable, Relational
<u>Flexibility</u> - [Kumar & Topping, 1988, and Fukuda, 1988] Soft constraints, Hard constraints

Table 3.1: Constraint Classifications

As can be seen from the table, each commentator classifies different constraints as points along different axes. For example the *origin* of the constraint; the *flexibility* of the constraint or the *function* of the constraint. There is no overall compelling classification system. All that can usefully be said is that each domain, problem or implementation may require a variety of constraint types. In many cases, the constraint classification scheme depends on or determines the way the design task is divided into sub-tasks at an implementational level, rather than at the higher conceptual levels that are the concern of this chapter, [Brown, 1985, and Sriram & Maher, 1986].

From this initial discussion of the factors relevant to deliberate design activity it is possible to formulate a simple design model. The model can be used to illustrate the various activities implicit in the depiction of bridge design in the previous section. This model is illustrated in figure 3.1.

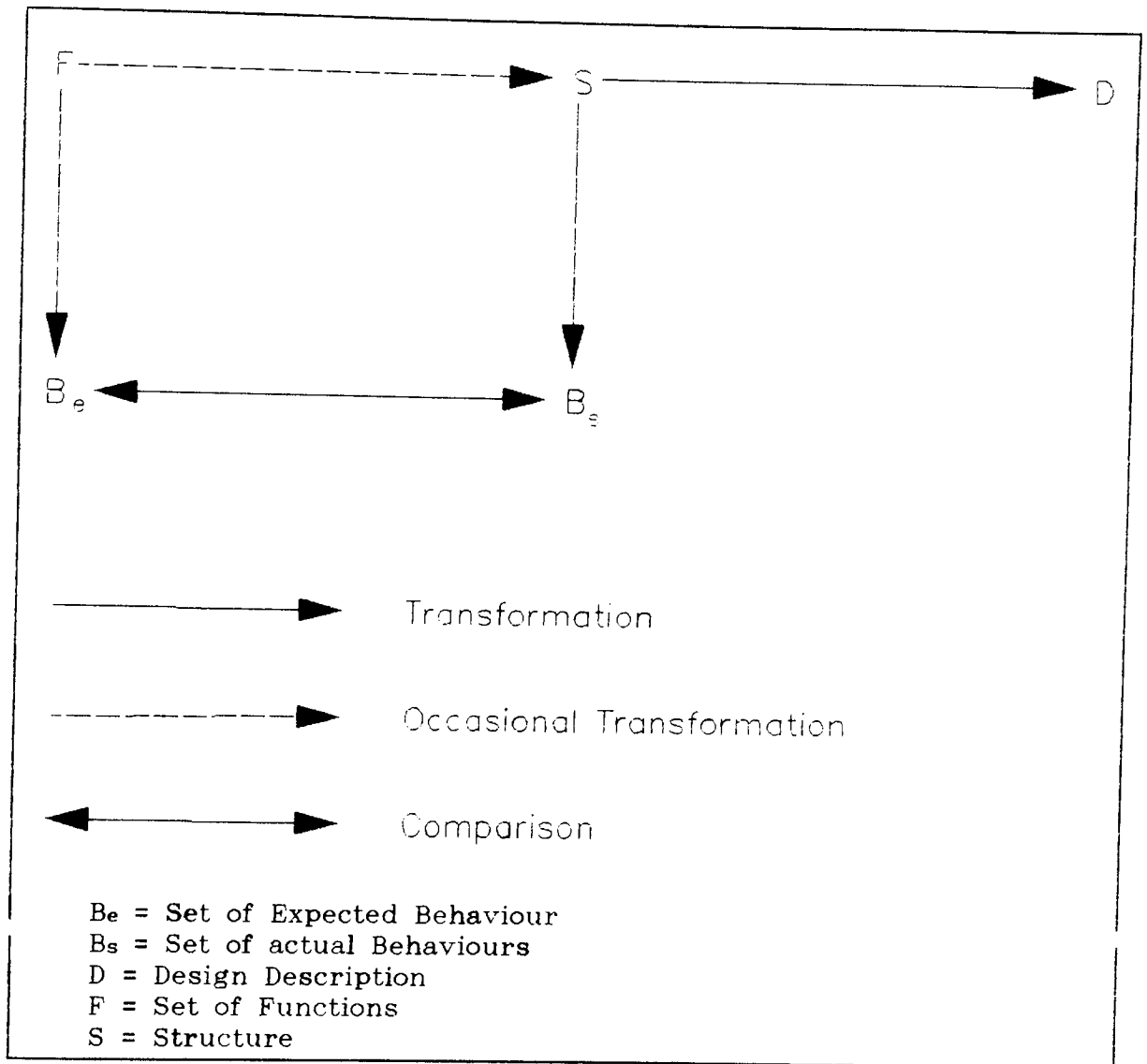


Figure 3.1: A Model of Design as a Process

(After Gero, [1990], p29)

The naive design model is the transformation $F \rightarrow D$. Here a set of functions is transformed into the design description (or the artifact itself). This corresponds to the building of a simple bridge by assembling a few components. The properties of these components are understood by what can be characterised as *common-sense*. The bridge is built in a certain way for a reason but these reasons are pre-conscious. Further analysis of these reasons requires a more developed design model.

The next design model is $F \rightarrow S \rightarrow D$ where S is the structure of the design elements and their relationships. The direct mapping of F to S is known as *catalogue lookup*. The mapping from structure to the design description is the province of computer-aided drafting systems, for example³. This model is still therefore describing unintelligent design⁴. The equivalent activity for the bridging example would be the use of a pre-fabricated bridge section.

Expected behaviour is the semantic interpretation of the function set F for a particular purpose, that is the *specification* of the design and the relevant constraints. The actual behaviour is that derived by analysis from the structure (or from testing of an actual prototype of the

³ Gero's use of the term S - *structure* is ambiguous. In at least one passage, the structure *is* the design description:

A design description represents the artifact's elements and their relationships; it is labelled structure S. [ibid] (p28)

The implication of this is that structure is a formal description and that the transformation $S \rightarrow D$ represents a semantic interpretation of the structure as a design description. An example of this transformation is a graphical interpretation by a CAD package.

⁴ Refer to the discussion of intelligence in expert systems in section 2.6 of the previous chapter. Catalogue lookup or drafting a description from structure are not intelligent in the sense that there is no recourse to a rich explanation or modification of the activity.

artifact). A more sophisticated model of design is given by $F \rightarrow Be \rightarrow S(Bs)$. Here knowledge relating structure to actual behaviour is used in conjunction with the expected behaviour in a process of *synthesis*⁵.

The comparison $Be \leftrightarrow Bs$ is the *evaluation* of the design. This evaluation can make a positive contribution to the design by revealing behaviours that were not part of the original specifications but which become desirable with hindsight. This *reformulation* is one of the reasons for the open-endedness of the design task.

If an artifact fails to meet the design objectives or some constraint is violated this may entail complete failure of the design process.

Alternatively *redesign* may occur. Redesign is often dependent on a hierarchical view of the design process, [Brewer & Gajski, 1986, and Brown, 1985]. Thus, in figure 3.2, a failure at the level of using a rolling method to transport the tree trunk may be solved by trying a different tree; by using a different transportation method or by redesign at a higher level, [Brown, 1985].

Some commentators suggest that most designs are redesigns, [Dixon & Simmons, 1983 and Dixon et al, 1984]. Redesign will be discussed in the context of Gero's design Prototypes in the next section. Design hierarchies will be described further in section 3.4 under the topic of the *object-synthesis* generic task.

⁵ See section 3.4 for a discussion of the generic task of *object-synthesis*.

Gero does not explain the significance of the $S(Bs)$ notation.

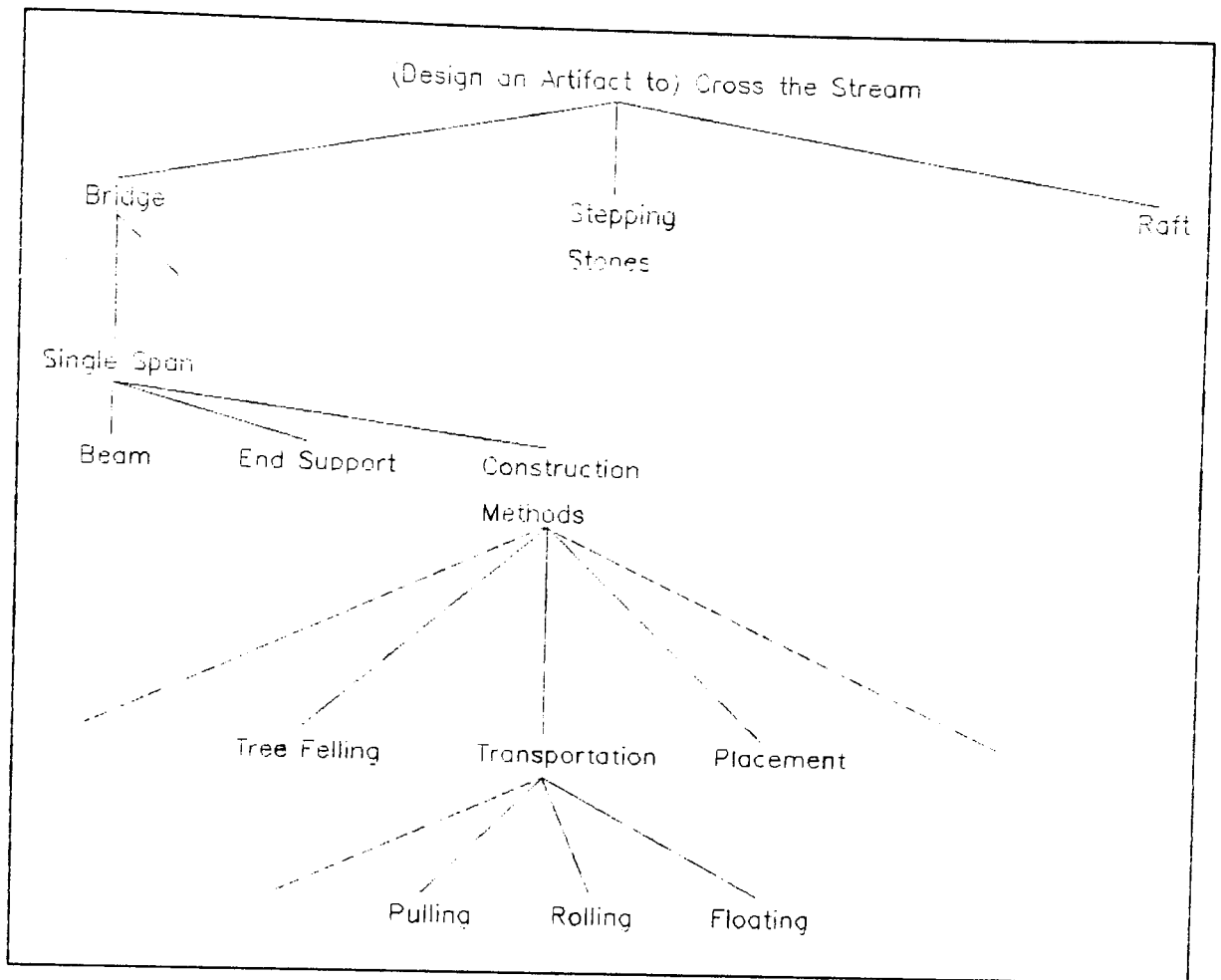


Figure 3.2: A Design Hierarchy

3.3 Design Prototypes

John Gero uses the concept of a design *Prototype* to illustrate the difference between design and other expert systems and to illustrate why existing design systems frequently fail to achieve acceptable levels of performance, [Gero, 1988]. Gero cites the reason for the failure of design expert systems as being due to a lack of understanding of the design process itself. Design Prototypes provide a basis for an understanding of the nature of design that complement the formal model

described in the previous section.

Design knowledge can be 'chunked'. In this form it becomes the conceptual schema for design. Conceptual schemas consist of

... knowledge generalized from a set of alike ... cases and form a class from which individuals can be inferred. [Gero, 1990]

(p30)

Within the schema of generalized knowledge about types; Gero distinguishes between *archetypes*, *stereotypes* and *Prototypes*. Examples of archetypes are specific designs such as the Taj Mahal and Ferrari sports cars which are unique examples of a type. As such, archetypes may be used as the analogues or inspiration for design. Stereotypes are copies without change of a type and their production does not constitute a design activity. Prototypes are the basis for the design of further instantiations of a type⁶.

This chunking of knowledge into Prototypes enables an experienced designer to create a specific design with relatively little problem specific information. A novice designer, lacking the Prototype knowledge, has more recourse to textbooks and problem specific information⁷. Design Prototypes are also instrumental in the ability of designers to initiate a

⁶ Prototypes therefore correspond to the classical Greek notion of *ιδέα* or *εἶδος* (ideas or forms). A Prototype for a chair is the same as the ideal chair which all real chairs reflect by virtue of their being chairs. See, for example book V of Plato's Republic. The distinction between Archetypes and Prototypes is therefore somewhat arbitrary. It makes equal sense to describe Ferrari designs as *instantiations* of a Ferrari *Prototype* or as *analogues* of the Ferrari *archetype*.

⁷ To some extent the difference between novice performance and expert use of knowledge chunked into Prototypes reflects the concerns mentioned in section 2.3. Collins et al, [1985], among others, maintain that knowledge engineers should undergo an extended apprenticeship in a domain in order to better appreciate the whole Prototype and not just specific knowledge items and performances.

design with incomplete specifications and constraints. Another example of the knowledge held in Prototypes is that which a designer uses to select the relevant behaviours which require analysis in the evaluation of a design (B_s in Gero's model [*ibid*]).

Gero (in an early version) formally defines a Prototype as:

$$P = (Dpg.I.V.K) \tag{1a}$$

where P is the Prototype

D_{pg} is a parameterized design description or a description generator

I is the interpreted goals and requirements of the design

V is the vocabulary and design elements

K is the knowledge relating the requirements and the design elements [Gero et al, 1988] (p5-6)

With the addition of the process model of design described in the previous section; the formal definition becomes elaborated:⁸

$$P = (F.B.S.D.K.C) \tag{1b}$$

where P is the Prototype

F, S and D are as above in figure 3.1

B = (B_s, B_e) as in figure 3.1 (B_e corresponds to I in (1a))

C = context

K = ($K_r, K_q, K_c, K_{ct}, K_p$)

$K_p = (T, P)$

P = partition

T = typology

(p33)

[Gero, 1990]

K_r is relational knowledge that links the variables of the function, structure and behaviour sets. K_q is qualitative knowledge that represents the effects of modifying the structure on function and behaviour. K_c is the computational or quantitative knowledge relating the F, S and B variables. The context C is the actual placement of the

⁸ A further elaboration of this model of design is given in [Coyne et al, 1990].

artifact being designed in the external world. Thus the contextual knowledge K_{ct} is concerned with exogenous variables imposed by the external situation.

K_p is knowledge about the actual Prototype. This consists of the type T of which the Prototype is an instantiation and the partition P which indicates what part of the higher level type the particular Prototype is representing. The vocabulary of the Prototype is no longer explicitly represented but is instead inherent in the actual names of the variables representing the other elements⁹.

The more developed formalism provides a sufficient basis for implementation. Examples are given of use of Prototypes for house plan design, [Oxman, 1990], window design, [Gero, 1990], and beam design, [Gero & Rosenman, 1990]. The earlier formalism is more tractable for explanation of the actual design process. Therefore model (1a) will be used for the rest of the discussion in this section.

Design can be classified into three modes: Prototype refinement, adaptation and generation¹⁰. Most everyday design is in the form of Prototype retrieval and *refinement*. Refinement is formally defined by:

$$D = \tau(Dpg.I) \tag{2}$$

[Gero et al, 1988] (p6)

⁹ Presumably the vocabulary is still represented explicitly by the knowledge-base editor in the shell implementations of this methodology. The benefits of such representation are discussed in chapter 8.

¹⁰ This classification corresponds to the class III, class II and class I types of design identified by Brown & Chandrasekaran, [1985].

In other words a transformation between the design objectives and the values of the parameterized design descriptors¹¹. This type of design is also known as *routine design*. Most of the design expert systems described in chapter 4 are limited to the routine design method of Prototype refinement.

The actual process of design by Prototype refinement proceeds recursively by the instantiation of Prototypes at successively lower levels in the design hierarchy. This successive Prototype selection and instantiation is depicted in figure 3.3. Returning to the bridge building example of section 3.1; the level 1 instantiation in figure 3.2 corresponds to the selection of the bridge branch of the hierarchy (rather than the raft or stepping stone branches). Further lower level instantiations in figure 3.3 continue down the hierarchy in figure 3.2.

The design is completed when it has proceeded without failure all the way down to the level where all the design variables are fixed. Figure 3.3 also shows the interaction points for the user and the effects of redesign and reformulation.

¹¹ If all the design descriptors are numeric then this process becomes amenable to numeric optimisation techniques, [Gero & Balachandran, 1986].

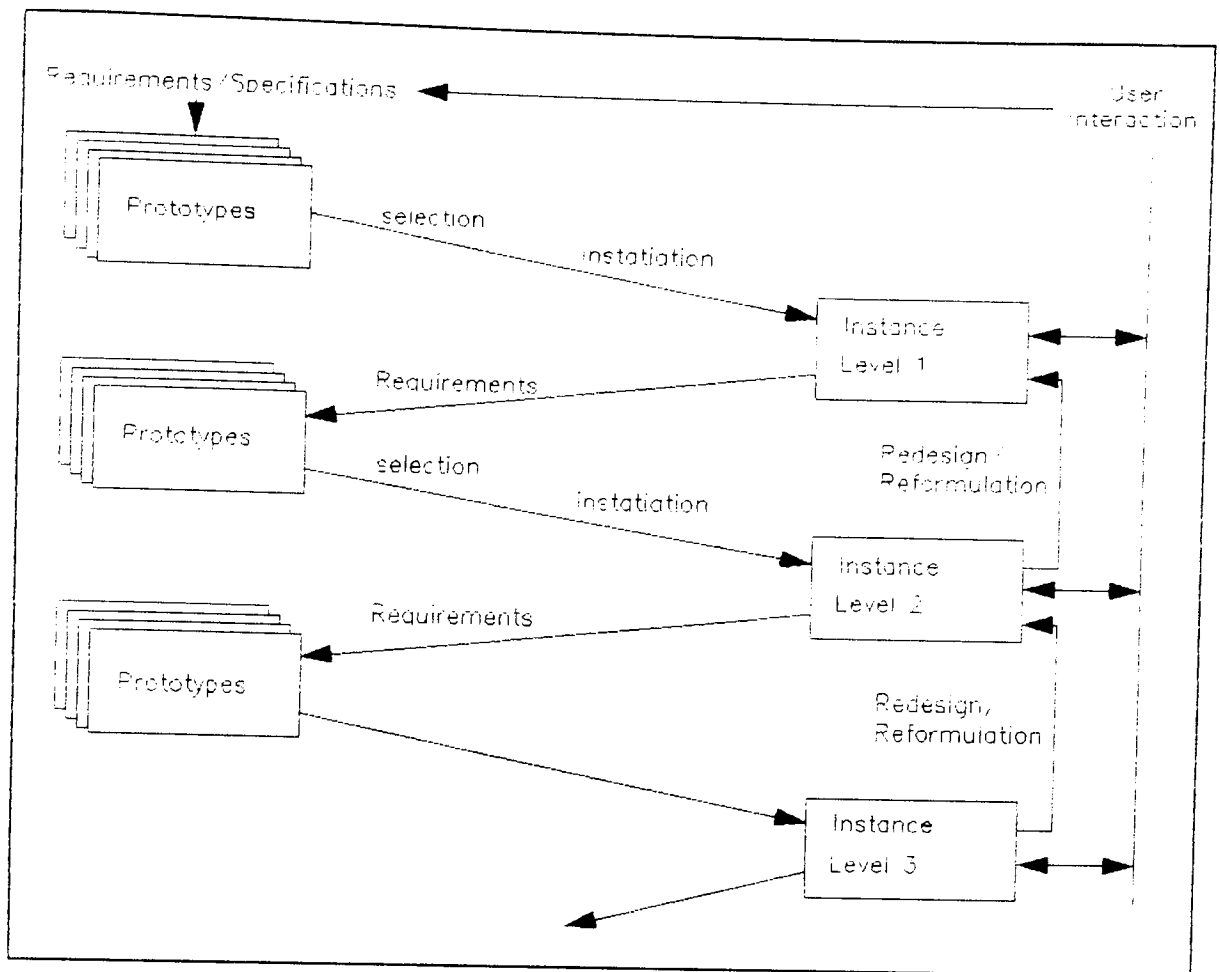


Figure 3.3: Routine Hierarchical Prototype Selection and Refinement
 (after Gero & Rosenman, [1990] p75)

Routine design amounts to the instantiation of a number of the predefined variables, slots, attributes or sub-Prototypes of a Prototype. In practice many of these values may be supplied as defaults or as a direct consequence of constraints. Therefore there is no fundamental difference between routine design and redesign. All design is redesign starting from a point where relatively fewer instantiations have been made.

It is tempting to view Prototype refinement as frame instantiation. Gero and the other authors cited, use object-orientated implementations. Gero is, however, insistent that Prototypes need not necessarily be represented as frames, objects or any other specific implementation, [Gero et al, 1988].

Prototype *adaptation* is necessary when the existing Prototype is inadequate for some design task. Adaptation involves some modification or extension to the knowledge, vocabulary and/or objectives and hence in the design descriptor. Once these changes have been made; routine design can recommence using the new Prototype. Using the example, Prototype adaptation would be illustrated by an addition to one of the lower level branches of the design hierarchy; perhaps the provision for dual- or multi-span bridges¹².

Prototype *generation* is the creation of a new Prototype. The process whereby this occurs is an example of truly *creative* design. Gero does not elaborate the concept of Prototype generation beyond the statement that it is often only recognized *post facto*. There may only be a difference of scale between Prototype adaptation and generation. Additional or new Prototypes at a lower level are adaptation whereas

¹² There is some similarity between this example of Prototype adaptation and the illustration of the transformation model described by Maher, [1990]. Furthermore, Maher cites the use of transformation grammars to capture *generative* design knowledge. Gero et al, [1985], also discuss design grammars but in this context a grammar is equivalent to a production rule and the examples given are of relatively simple design tasks. A more formal investigation of the relationship between Prototype adaptation and transformation grammars is beyond the scope of this thesis.

top-level additions are generation. An example of generation would perhaps be the jump from the use of the bridging to raft-building Prototypes¹³.

A number of research issues in the implementation of Prototypes have been raised. One obvious problem is how Prototypes can be indexed or otherwise retrieved¹⁴, [Gero, 1988]. The examples given in Gero, [1990] and Gero & Rosenman, [1990], use a simple method of indexing based on a symbolic identifier for each Prototype (given by T in (1b)) and for the part of the parent Prototype addressed (P in (1b)). This indexing method is therefore only of use in top-down design. There may be cases where the constraints and available materials are restrictive to the extent where bottom-up design becomes more appropriate.

The other main issue is in the completeness and consistency of the Prototype set, [Gero, *ibid*]. The completeness of the available Prototypes is limited to, but not determined by, the maturity of the relevant technology in the domain. There is a need for research into how design Prototypes become generated in response to new technologies.

The consistency issue is reflected in the nature of belief maintenance in human designers when faced with inadequate or inappropriate

13 A more convincing metaphor for Prototype generation would be the paradigm shift in scientific revolution, [Kuhn, 1970]. It must be pointed out however that Prototypes are effective on a much smaller scale than paradigms. Coyne et al, [1990], discuss the differences between science and design in chapter 1 section 1.2 of *Knowledge-based Design Systems*. However, to some extent the Prototypes in use by designers are part of the make-up of the engineering equivalent of paradigms.

14 This is a similar problem to the indexing in case-based design methods but on a different scale, [Chandrasekaran, 1990, and Maher, 1990]. A full treatment of case-based reasoning is given by Slade, [1991].

Prototypes. It is important to find an efficient and consistent method of determining a strategy for when it becomes necessary to abandon a line of Prototype instantiation and backtrack at a higher level¹⁵.

The model of design and the representation and use of Prototypes implies that there is a finite repertoire of tasks that are performed by design expert systems. Thus a final research issue in the use of Prototypes in design is the identification, analysis and implementation of these tasks.

3.4 Generic Tasks

Chandrasekaran, [1985], proposes a framework for expert systems development based on decomposition of problems into generic tasks. Each task has associated types of knowledge and control regimes. Generic tasks are therefore at a lower level of abstraction than characterisations such as 'design', 'monitoring' or 'diagnosis' for example. On the other hand, generic tasks are at a level of abstraction above implementation methods such as rules or frames.

Advantages arise from the use of a representation formalism at a higher level than the implementation language. Swaffield & Knight, [1990], illustrate how systems analysis techniques could be used to benefit expert systems development (see section 2.3). A high level representation or description of the design process can, for example, improve knowledge elicitation by focussing discussion¹⁶. Advantages also arise

15 Non-monotonicity in design is described in section 4.7 in Coyne et al, [1990]. Kumar & Topping outline a practical solution to the problem, [1990]. Another approach has been adopted for the DEX system and is described in chapter 9.

16 A model of the design process is, in effect, the results of knowledge elicitation that may be generalised to many specific design domains.

from the use of an explicit generic control regime. Differentiation between generic-control and domain-specific knowledge aids debugging by distinguishing logical errors from coding errors.

The generic tasks identified by Chandrasekaran in his 1985 paper are as follows:

State abstraction

Knowledge-directed information retrieval

Hypothesis matching

Assembly of compound hypotheses for abduction

Classification

Object synthesis by plan selection and refinement

These tasks will be described in the rest of this section with emphasis being given to those tasks most relevant to design¹⁷.

State abstraction relates changes in the state of a system to changes of the functions of the system and in the state of the immediate super-system. The knowledge is distributed among specialists in each system/subsystem. The control regime is bottom-up with changes at one level being followed up into higher levels; reflecting the system/subsystem relationships.

Knowledge-directed information passing relates attributes of one datum to some other datum. Examples are default values and inheritance. This kind of knowledge is typically represented within a frame hierarchy. Each frame specialises in the knowledge-directed data inferences for a single concept.

¹⁷ It is not clear that each of the generic tasks is mutually exclusive or that each occupies the same level of abstraction. For instance Clancey, [1985], in his discussion, shows how abstraction and refinement are sub-tasks of (heuristic) classification.

Hypothesis matching relates data describing a problem to a hypothesis. Evidence is organised into a hierarchy where each level supports the higher level hypothesis. Hypothesis matching would seem to be at a lower level than state abstraction and classification (see below), since both of these perform hierarchical hypothesis matching.

Abductive assembly of explanatory hypotheses uses the significances and relationships between hypotheses to generate a composite hypothesis for the purposes of explanation. This may be a recursive process for large numbers of hypotheses with the relatively superfluous explanatory hypotheses being dropped in favour of those with more significance.

The *classification* task specifies the place of a situation description within a classification hierarchy. The generic knowledge form is the declarative statement of a linkage between a partial description and evidence for a classification. This knowledge is typically organised among specialists that are concerned with proving or rejecting a particular concept. The control regime calls upon each specialist in a top-down order to attempt to establish its concept. If successful, a specialist calls its successors. If unsuccessful, all the successors are failed automatically.

Clancey, [1985], discusses *heuristic classification* in some detail. This description is largely compatible with Chandrasekaran's depiction of the classification task and arises out of a similar desire to find an abstraction for expert system description above the implementation

level¹⁸.

Clancey describes *simple classification* as identification of

... some unknown object or phenomenon as a member of a known class of objects, events or processes. (p293)

The classification classes are typically pre-enumerated stereotype¹⁹ solutions. This simple classification is hierarchical in that it places a specific instance within a structure of classes and sub-classes. Clancey gives three basic relationships used for this data *abstraction*: definitional and qualitative abstraction and generalization.

Definitional abstraction is classification of a datum by virtue of its essential features. Clancey gives the example:

*If the structure is a one-dimensional network, then its shape is a beam. [Clancey, *ibid*] (p294)*

Qualitative abstraction classifies a datum with respect to some value. To take an example from the evaluation of potential sulphate attack:

If the total SO₃ content of the soil is less than 0.2%, then the site has a class 1 sulphate attack risk²⁰

¹⁸ Steels, [1990], gives a thorough discussion of the interaction between inference structures such as Clancey's heuristic classification, and Chandrasekaran's generic tasks as well as problem solving methods and domain models. He also outlines the classification and cataloguing tasks required for a overall solution to the problems of relating domain tasks to generic expert systems solutions.

¹⁹ Although Clancey uses the word *stereotype*, this strictly *implies copies without change*, [Gero, 1990] (p30). What is really meant is closer to what Gero describes as *Prototypes* implying *the first on which others are modelled* [*ibid*].

²⁰ Although Clancey would describe this as qualitative abstraction it would also seem to be somewhat *definitional* in the more colloquial use of the word.

Generalization is similar to inheritance. Clancey uses the following illustration:

If the client is a judge, then he is an educated person. [ibid]

Evidence for an instance of a class is also indirect evidence that one of its sub-classes is relevant, as shown in figure 3.4 (*refinement*).

Uncertain evidence can lead to the creation of a ranked list of hypotheses.

In contrast to simple classification, *heuristic classification* relates

... solutions and solution features ... heuristically by direct, non-hierarchical association with some concept in another classification hierarchy. (p294)

Obviously the identification of one classification hierarchy as *different* from another is subjective. The inference is that there is some connection between the classes linked by the heuristic classification. This connection is made by *heuristic* rules of thumb rather than by more direct hierarchical abstraction or refinement. The relationship between simple and heuristic classification is shown in figure 3.4.

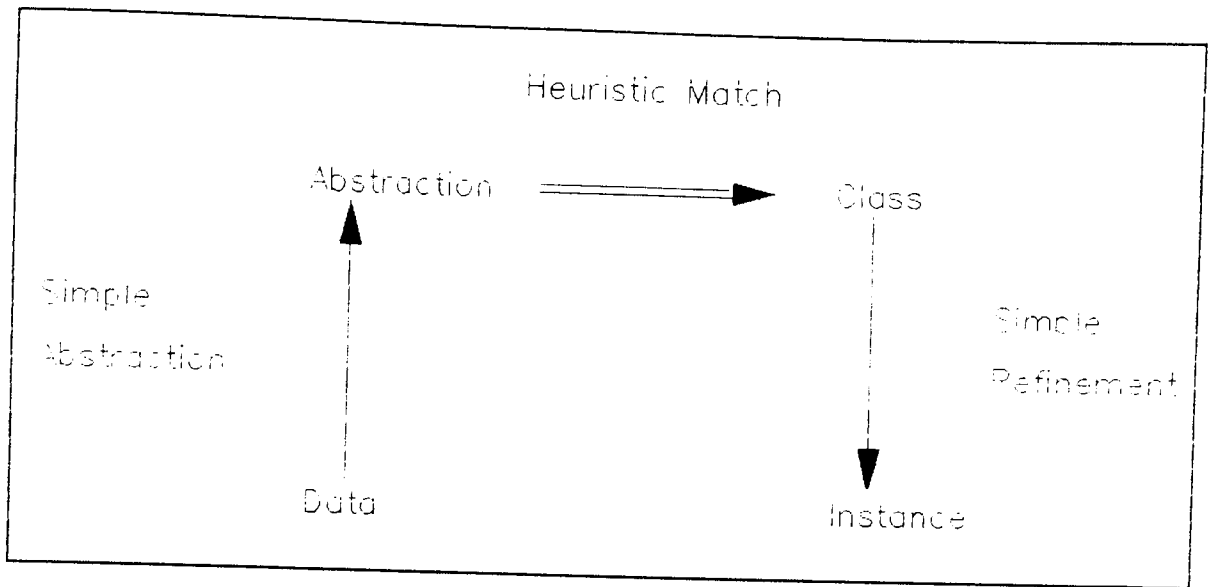


Figure 3.4: Data Abstraction, Solution Refinement and Heuristic Classification (after Clancey, [1985], p295)

Object-synthesis is the generic design task described in the previous sections. *Object* in this abstract sense can be a plan, for example, as well as more concrete artifacts. The knowledge used in object synthesis is concerned with the structure of the object at some level of abstraction and with the components and concepts required for refining this top-level structure.

The control regime is recursively defined as the top-down design of each of the component objects. At each stage a specialist for the object being designed uses a plan to execute object specialists to design the next lower level of detail. Design in this context follows the model $F \rightarrow B_e \rightarrow S(B_s)$ discussed in section 3.2. This model characterises synthesis as the use of expected behaviour (specifications) in the

... selection and combination of structure based on a knowledge of the behaviours produced by this structure. [Gero, 1990]
(p29)

Synthesis continues through the hierarchy until all the primitive object values are established, as illustrated in figure 3.2.

Brown, [1985], describes a design scheme with particular emphasis on failure handling²¹. As described above, design is organized as a hierarchy of concepts. Each concept specializes in some sub-problem of the design process, planning a sequence of design *actions*. An action may be a request for another specialist at a lower level in the hierarchy. An action may also perform a *task* or sequence of *steps* which result in the setting of some primitive aspect of the design.

Constraints²² may be used at any stage of the design in order to test its validity. Each *agent* (specialist, action, task, step) has its own knowledge relevant to handling failure in the form of a constraint violation or non-completion of a goal. As mentioned in sections 3.2 and 3.3, above, failure can result in redesign at any level. If all redesign attempts fail, the top-level agent may choose an alternative plan. To continue the example: 'design a raft' rather than 'design a bridge'.

Chandrasekaran, [1990], characterises design synthesis as a set of propose-critique-modify methods. Each sub-task is then further analysed in terms of generic methods. For example, design *proposal* can be achieved by recursive problem decomposition; by case retrieval or by

²¹ Brewer and Gajski, [1986], describe a similar hierarchical design paradigm.

²² see table 3.1 for a relevant list of constraint classifications [Brown & Breau, 1986].

simultaneous satisfaction of constraints. Verification that a design meets the functional objectives can be achieved by direct domain-specific calculation or by simulation.

Maher, [1990], in a similar vein, breaks design into the three phases of formulation-synthesis-evaluation. Maher also identifies three models for the synthesis/proposal stage: decomposition, case retrieval and direct transformation from design requirements to solutions²³. Obviously, these methods are not exclusive. Most significant design will involve some decomposition into sub-problems. Solution of these relatively simple problems can either be achieved by reasoning from similar cases or by direct solution at the most basic level of design.

3.5 Expert Systems in Design

Simmons, [1984], Tong, [1987a], and Rychener, [1988], give overviews of how expert systems can operationally benefit design in general. Most commentators include the following items:

- Packaging of expertise

- Improved consistency of design choices

- More exhaustive/intelligent search of large solution spaces

- Front end to analysis packages

- Management or automation of design/redesign cycles

These benefits generally apply to expert systems in any domain. The most obvious benefit of design expert systems is in the management and automation of the design/redesign process.

²³ Both direct transformation and simultaneous satisfaction of constraints are covered by Gero's simple $F \rightarrow S \rightarrow D$ model described in section 3.2.

Most of the commentators referenced above also cite a number of problems with the implementation of design-orientated expert systems. Although design is goal-orientated, this goal may be unspecific. The design goal may also change during the design process as simple proposals are rejected, [Gero, 1988, and 1990]. This entails that any solution is only relatively successful. The search for a solution to a design problem is dependent on all of the factors listed in section 3.1.

This indeterminacy entails a number of potential operational problems for expert systems. The other main problem with design is the large variety of knowledge that may be used in the design process. Additionally the designer may need knowledge from many different domains. These problems are summarised in the following list:

- Unknown size of problem/solution space

- No unique solution

- No way of determining optimum design

- Need for design decisions to be reported as well as actual results

- Need for many types of knowledge and reasoning methods

(quantitative, qualitative, geometrical, mathematical).

The models of design in this chapter are attempts to characterise design so as to minimise these problems. They can be summarized briefly:

- Design is directed towards the attainment of a number of objectives.

- There are generally a large number of potential solutions of these goals - the search space.

- The design space is restricted by a number of constraints reflecting various considerations.

- Generally there is no guaranteed optimal solution.

- The design process may be characterised as the design of

successive levels in a hierarchy.

At each level the design may progress by a number of methods such as the synthesis of lower level design problems, unique solution of a restrictive set of constraints, heuristic classification, use of existing solutions, or arbitrary decision on the part of the designer.

Failure at any level may occur by constraint conflict or inability to complete lower levels of the design.

Failure can be resolved by redesign at the current level or by backtracking to a higher level.

The design itself is the set of actual instantiations of the variables at the lowest levels of the hierarchy. Each possible permutation of these instantiations is a terminal node in the design space.

The knowledge used in design may be chunked as design Prototypes. These consist of the structure of the design; the plans for successive design at lower levels; defaults, ranges of permissible values; previous design solutions, methods for instantiating design variables and relevant terminology.

Each level of the design hierarchy may itself consist of lower level Prototypes.

Chapter 4: Expert Systems in Engineering Design

This chapter is intended to provide a brief overview of some implementations of expert systems in the broad domain of design. The examples given are not selected on any particular basis and do not constitute an exhaustive survey. In the last few years there have been several publications of bibliographies concentrating on expert systems in engineering, [Duffy, 1987, Chung et al, 1988 and AIAI, 1989]. Together, these provide a comprehensive listing of recent work in the field.

Section 4.1 will outline the earliest work on the application of AI to design in the form of intelligent front ends to analysis and simulation tools. Section 4.2 will describe a number of attempts to develop expert systems in the broad domain of mechanical design. Most modern design is subject to legislative constraints and safeguards in the form of standards and codes of practice. The use of these codes in computer systems is detailed in section 4.3

There are a number of expert systems applications in the domains of architectural and structural engineering which have more direct relevance to the subject of this thesis. Adeli, [1987 and 1988], gives extensive lists of expert systems in structural engineering. Some of the systems mentioned by Adeli will be described further in sections 4.4 and 4.5¹.

4.1 Intelligent Front Ends

Much of the earlier work on the application of expert systems to design concentrated on areas such as mechanical design. Here the emphasis is

¹ HI-RISE, DURCON and SASE; SACON is discussed in section 4.1.

on physical layout and dimensioning and properties like compressive and tensile strength. The development of many knowledge-based design systems arose out of a desire to extend design automation beyond the abilities of conventional systems such as CAD and analysis packages, [Dixon et al, 1984, and Taylor & Corlett, 1987].

The use of systems like finite element analysis (FEA) packages is a task that is often difficult and can require years of training to gain specific expertise. Fenves, [1985], cites the complexity of FEA use as a suitable domain for knowledge-based systems, specifically for intelligent pre- and post-processors or front- and back-ends. Fenves describes two such examples of intelligent front-ends, HYDRO and SACON, and outlines a number of important issues in the implementation of a general purpose FEA assistant.

Aldridge et al, [1986], describe two systems: PROCON in the domain of architectural design and Designer's Apprentice which designs nuclear weapons. Both these systems are interfaces to programs that simulate the performance of the artifact being designed. The paper describes the functionality required of useful interface systems. Typical interface system requirements are to check for input errors; execute the analysis or simulation model; filter and interpret output and manage a number of on-going projects².

Zumsteg & Flaggs, [1985], describe SACON in the context of the development of integrated analysis and design packages. In contrast to the limited ability of SACON to create input files for the MARC FEA

² Some discussion of the related issue of embedding non-AI programs in expert systems will be given in later chapters dealing with the use of the hydroxyl ion concentration prediction model.

package; the Composite Design Assistant (CDA), [Zumsteg & Flaggs, *ibid*, and Pecora et al, 1985], also incorporated rules that enabled it to interpret the results of the analysis and determine the behaviour of a structure. CDA did not implement any design knowledge itself.

The Buckling Expert, [Zumsteg & Flaggs, *ibid*], was a system that more fully integrated design and analysis. This was achieved by incorporating knowledge that was concerned with the qualitative assessment of the analysed structure and how to design for improved performance. The Bucking Expert was developed in the general domain of designing aerospace structures and of stiffened cylindrical panels in particular. The system was small (less than 100 rules) with limited user interface (scrolling screen dialogue) and unsophisticated explanation facilities (rule-trace).

Maher et al, [1988], develop ideas propounded by Fenves, [1985], and make use of the work of Pecora et al, [1985] and Zumsteg & Flaggs, [1985], as a prototype for a framework for an intelligent FEA assistant. This framework uses a blackboard³ to organise a number of specialist modules. Each module is concerned with a level of abstraction of the problem and posts its conclusions to the blackboard for the attention of modules at a higher or lower level. Like the Dominic system, described below, there are also a number of resource level programs.

Although Maher et al, [*ibid*] have proposed a framework for a fully competent generic intelligent front end for FEA packages there still remains a need for actual implementation of this ideal. Clark & Mac Randal, [1991], have developed a system (called IFE) that is described as

³ See chapter 2 for a discussion of blackboard architectures.

being competent in the tasks of dialogue with the user, input validation, data-model manipulation and the execution of applications. Furthermore the IFE system is able to perform these access tasks for a *plurality of advanced engineering software* [*ibid*] (p44) and to tailor its responses to a number of different user types, (Designer, Engineer, Modeller) and even individuals.

4.2 Mechanical Design Systems

Dixon et al, [1984], characterise design as an iterative cycle of three stages of decisions concerning the design process, technical design and acceptability of the design. This view of design is used as the foundation for a generic design system - DOMINIC, [Howe et al, 1986]. As mentioned in the previous chapter, Dixon and Simmons strongly emphasise the importance of redesign, [Dixon & Simmons, 1983, and Dixon et al, *ibid*]. The result of this emphasis is an architecture where the central organisation is based on redesign.

The design is initialised by retrieval of existing designs or from a general compliance with the most critical elements of a set of constraints and requirements. This initial design is carried out by a single module. Other modules perform further specialist tasks such as user interface, evaluation, acceptability and redesign. These modules are organised in a blackboard system, with a final module responsible for control tasks. Each module in the system is able to make use of *support resources* such as analysis and drafting programs.

The DOMINIC system is illustrated by the use of examples taken from the design of V-belts and of extruded aluminium shapes (heat sinks), [Dixon

et al, *ibid*, and Howe et al, 1986]. The use of DOMINIC in these domains is reported to compare favourably with human experts and with domain specific expert systems.

Mittal et al, [1985, 1986] report on the PRIDE system for the design of paper handling systems. PRIDE is ostensibly an attempt to explore issues in *non-routine* design. From the problem description given in their paper it is not clear that the design of pinch-roller paper transport systems is *routine* in the sense of being a transformation from objectives to the final design parameters within the scope of an existing Prototype. The problem is, however, large and complex and covers many areas of expertise.

The PRIDE system is object-based and uses a system of successive refinement through design plans. The system also uses a number of standard design methods that deal with types of goals and their respective knowledge, user-interface, computation and inference needs. Failure of a goal is handled by a global problem solver. The problem solver is able to analyse partial designs and suggest modifications. PRIDE is also able to maintain multiple designs simultaneously. The system was found sufficiently useful to be adopted by Xerox as a practical tool.

The DSPL language, [Brown & Chandrasekaran, 1986], uses a similar hierarchical refinement strategy. In contrast to PRIDE, goal failure is handled by the passing of failure reports to the parent goal. The parent goal is able to determine a strategy for solving the problem⁴. DSPL is

⁴ This method of failure handling is described more fully in Brown, [1985]. Brewer & Gajski, [1986], also use hierarchical refinement plans and a similar failure handling method in a system for VLSI design.

illustrated by an example system for the design of air-cylinders (AIR-CYL), [*ibid*]. In addition to the basic design ability of the DSPL language, a number of additional tools provide explanation facilities and knowledge acquisition, [Chiang & Brown, 1987].

The research program at Edinburgh university department of AI characterises design as an essentially mysterious, human exploratory activity, [Smithers et al, 1990]. The practical implication of this view is an emphasis on the development of support tools for human designers rather than design systems *per se*. A number of support systems have been developed and combined into the Edinburgh Designer System (EDS). EDS is illustrated by an example from mechanical design (water turbines).

The systems described in this and the following sections are limited mostly to *routine design*. There are some attempts at developing systems for non-routine design. For example the CYCLOPS system for innovative design, [Sriram et al, 1989]. This system uses relaxation of objectives and constraints to generate a much larger search-space in which it is hoped some *interesting* designs will arise *serendipitously*.

4.3 Building Standards and Codes

Standards and codes of practice (henceforth *codes*) are collections of knowledge about design. This knowledge has already been compiled,

tested and accepted by panels of experts in the creation of the codes⁵. Codes are therefore a potentially useful knowledge source for expert systems once the codified knowledge has been given a suitable structure. Another potential application of expert systems is in the automation of the process of authoring codes.

Many of the systems listed by Adeli, [1987, and 1988], are centrally concerned with the generation or the checking of designs against standards. Saouma et al, [1989], refer to the legal responsibilities that are inherent in codes. Rosenman & Gero, [1985], describe codes as being large and complex to use. Koskela et al, [1986, and 1988], cite the extensive use of technical jargon as another difficulty in using codes.

Stahl et al, [1983], note that design is often driven by codes, as opposed to being merely constrained by them. It is therefore seen to be desirable to allow CAD programs, for example, to have access to the provisions contained in the codes. The authors analyse the properties of codes and a standard for implementing a computer-representation is proposed. This standard is based on decision tables (DTs). Dts allow the clarity and completeness of a set of provisions to be examined.

The SASE system, [*ibid*], is based on this use of DTs to express codes. The system facilitates the creation of the DT data-base representing a code. SASE then enables a user to access and check the code provisions.

⁵ A code of practice is very close to Gero's idea of a Prototype. Consider the definition of a prototype:

... knowledge generalized from a set of alike cases [forming] a class from which individuals can be inferred. [Gero, 1990] (p30)

As a body of compiled knowledge, codes implicitly embody much of what Gero would include in a Prototype. In some cases it is possible to complete a design by inferring the design parameters from the code provisions.

The authors outline the SASE command language and suggest uses for the system and the code representation, for example, for knowledge-base construction.

Stone & Wilcox, [1987], depict a number of problems with existing codes in the form of inconsistency, incompleteness, circularity, redundancy and ambiguity. The authors describe a system which guides a user through the generation of a draft code. The text of each provision is input through standard forms depending on the type of the requirement and its place within the code.

The system interprets the input forms and represents the code as a body of PROLOG rules and facts. The logical consequences of this representation can be assessed by querying the system and providing ranges of values for the design variables. The process thus ensures the consistency and completeness of the regulations as a set of PROLOG rules. The system also includes facilities for consultation of the final code representation as a simple expert system.

Neither Stone & Wilcox, nor Stahl et al, discuss how the PROLOG rule-base or SASE DTs can be translated back into a readable form of English without either being stilted or re-introducing errors. This subject is addressed in a paper by Thewalt & Moskowitz, [1990]. Their text-generation system, ICARUS, addresses the computer representation used by the SASE system. ICARUS uses a number of generic templates to map each element of a decision table onto natural language (NL) sentences. Refer to figure 4.1 for a summary of the interactions between building standards and expert systems.

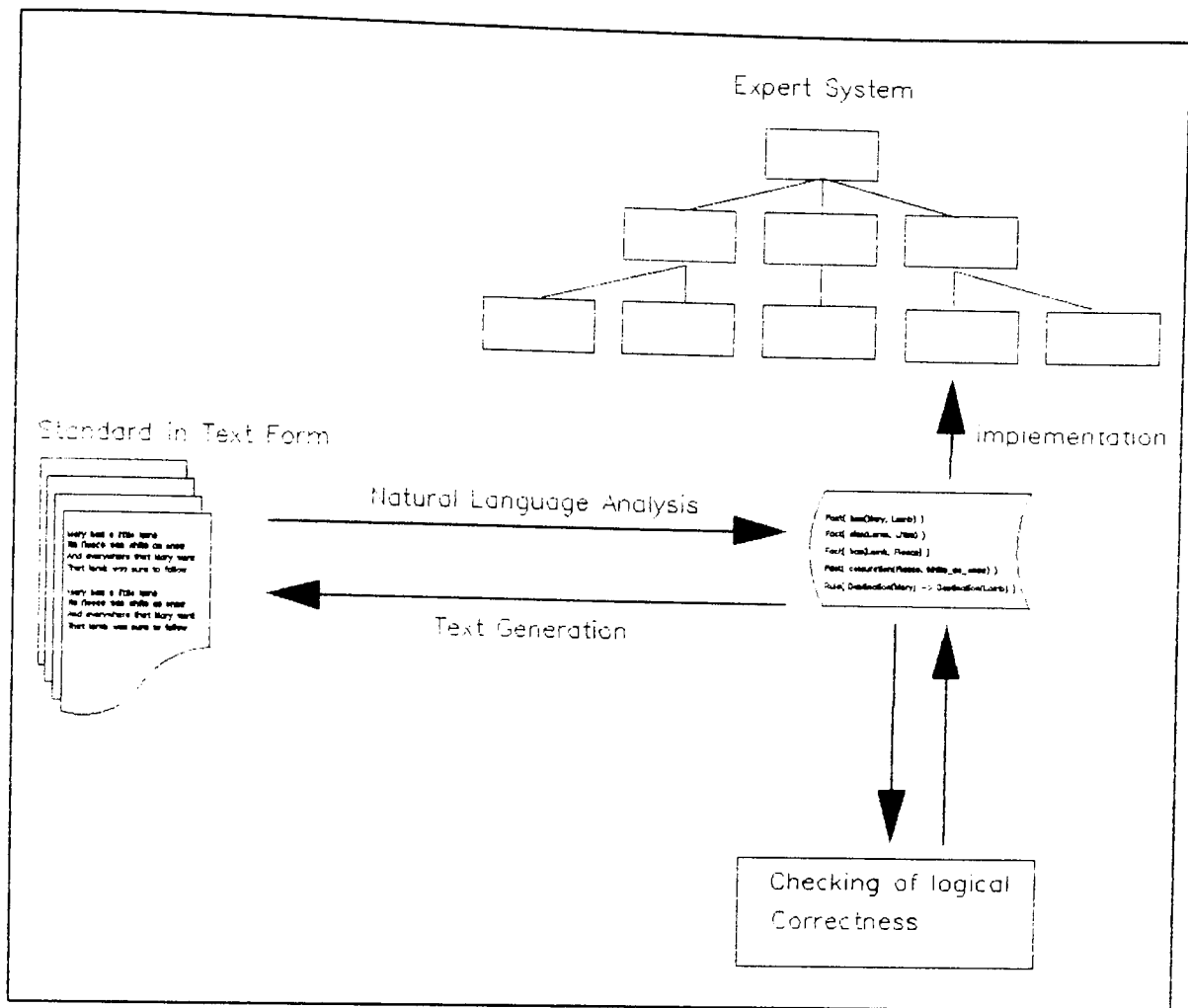


Figure 4.1: Building Standards and Expert Systems

Rasdorf & Parks, [1987], present a system that generates computer data-structures from a natural language (NL) analysis of a code. The system is illustrated by examples from the BOCA Basic National Building Code. The code is parsed and data items and relationships are identified. The user is consulted as the identified items are added to the data-base. The system can also be used to access the knowledge held in the codes by use of NL queries about specific provisions.

Two prototype systems for code compliance checking, Query Monitor and Roofload Checker, have been implemented by Rasdorf & Wang, [1987, and 1988]. The authors note the inefficiency of representing codes as production rules⁶. They cite as an example the need for 30 rules to represent a single BOCA table. SPIKE, is a development from these earlier prototypes. SPIKE is a generic code processor using a database of facts to represent codes [*ibid*]. Rasdorf and Parks planned to develop the NL codes analyser to generate input suitable for use by the SPIKE system.

Rosenman & Gero, [1985], also present an expert system shell for code implementation⁷. The shell is written in PROLOG and the example implements the Australian Model Uniform Building Code (AMBUC). The shell supports use of the code for *requirement finding*. Access to the knowledge contained in the code emphasises low-level queries rather than predefined high-level goals, such as would be needed for *compliance checking*. An example of a query to the system is given which indicates that the user is expected to have some ability with the shell's interface language.

*?infer true re 'fire-resistance rating' given building is_a hotel
and 'number of storeys' is 6. [ibid] (p403)*

This early prototype system was developed into the BUILD shell, [Rosenman et al, 1986a, 1988b and 1986c]. Examples are given of systems

⁶ This mirrors the preference of Stahl et al for DTs rather than rules for clear and complete representation of code provisions, [Stahl et al, 1983].

⁷ Descriptions of these systems use terminology and ideas that anticipate the design models and *Prototype* theory proposed in Gero's later published work and outlined in the previous chapter, [Gero, 1988, and 1990].

for the design of retaining walls and kitchen layouts. Several additional enhancements and extensions to the original are described. The provision of facilities for selection of components allows the system to actually perform design. The knowledge required for the design selection is not contained in the AMBUC codes but is retrieved from experts.

The example systems use graphics for passive description and for simple user interaction. The majority of the user interface is still of the scrolling-text *dialogue* variety and requires user ability with the syntax of the shell. For example:

?explain why_not 'fire resistance rating required ' is_none.

[Rosenman et al, 1986a] (p546)

Further examples are given for checking compliance with the AMBUC code (CODE); for the selection of algorithmic optimisation processes (OPTIMA) and passive energy design (SOLAREXPERT), [Rosenman et al, 1989]. These systems use interfaces to external programs such as CAD systems and optimisation routines.

DEST, [Fukuda, 1986, and 1988], is code-based expert system in the domain of design of oil storage tanks. DEST uses constraints from the relevant codes to restrict the search space of possible designs. This system is unusual in that it incorporates a variety of codes relevant to particular areas with differing structural design statutes.

Other code compliance checking systems include the Life Safety Code system (LSC). [Dym et al, 1988]. LSC is concerned with the architectural implications of fire-protection regulations for the design of hospitals. Like BUILD, LSC is interfaced to a CAD system. Saouma et al, [1989],

have developed a architecture for code-compliance systems. The method is illustrated by a prototype system that checks for compliance with codes for reinforced concrete design⁸.

Koskela et al, [1986, and 1988] describe several expert systems in the broad domain of construction. Like Rosenman et al, these authors implement building codes in two modes - constraint and requirement advice and code compliance. Like Dym et al, they have developed a system for fire-protection codes. Other relevant examples given in these papers include the repainting of wooden facades, pavement design and selection of ready-mixed concrete.

4.4 Integrated Structural Design

Code compliance is only one facet of building design. DESTINY, [Sriram, 1986, 1987a, and 1987b], is a framework of a system for fully integrated design⁹. DESTINY consists of a number of knowledge modules incorporating a variety of textbook, heuristic and deep knowledge. These modules are organised using a blackboard architecture. Overall control of the system is handled by TACON which is described as a *strategy level* module. TACON determines the correct execution of the lower level modules.

Specialist level modules are individually concerned with design tasks such as synthesizing structural

⁸ This system uses an interface to LOTUS123 for initial entry of the large volumes of data required by the system. Refer to chapter 10 for further discussion of the use of spreadsheets as utilities for expert systems.

⁹ Sriram [1987b], initially describes DESTINY as a *model* (p37) and ALL-RISE as a *framework* (p57). The description then proceeds as if the modules had actually been implemented. It is, therefore, unclear as to whether or not it is best to regard DESTINY as a functioning system or merely as a model.

configurations (ALL-RISE), analysing configurations (MASON), detailing structural elements (DATON) and evaluating the current design (CRITIC). Each of the specialist modules can call upon *resource level* modules which carry out functions such as data-base management and conventional analysis and costing. A final resource module is a database containing the relevant building codes.

DESTINY uses a hierarchical decomposition represented by schemas (frames). The building is represented as a grid of beams, columns and walls. The grid has two components, the lateral- and gravity-load resisting systems. Each of the two component systems are further decomposed into subsystems and components.

ALL-RISE is a development of the HI-RISE system that performs preliminary design of a high-rise building from the initial space plan [Maher & Fenves, 1985, Sriram & Maher, 1986, Maher, 1988, and Maher et al, 1988]. ALL-RISE is able to handle low- and medium-rise as well as high-rise buildings¹⁰. ALL-RISE makes use of constraints as active goals. Previous systems using constraints in design, as described above, merely emphasised their role as passive design checks, [Sriram & Maher, 1986].

Maher, [1988] describes three other systems in the domain of the structural design of buildings. FLODER designs alternative floor plans from input data concerning space and functional requirements. LOCATOR locates the lateral-load resisting systems as part of the

¹⁰ The relationship between the HI-RISE and ALL-RISE systems is described further in [Sriram & Maher, 1986]. The latest development of the HI-RISE/ALL-Rise prototypes in the CONGEN system, [Sriram et al, 1989]. CONGEN is a system under development for domain-independent preliminary design.

three-dimensional grid. These two systems use the generate-test paradigm of design. STRUPLE uses partial descriptions of requirements for a building to retrieve a structural configuration from a data-base of previous designs.

Finally, Maher et al, [1988], describe a system for the detailed proportioning of structural components. This system, SPEX, therefore performs at a similar level of the design process as DATON, in the DESTINY system. SPEX itself is organised as a blackboard and makes use of knowledge-based and algorithmic modules. Design standards are represented within SPEX as causal (deep) knowledge¹¹. Maher does not describe or speculate on the possibility of linking these systems into an overall integrated design package¹².

The decomposition model of design described in the previous chapter has been used as the basis of a shell for engineering design, EDESYN, [Westerberg et al, 1990]. The EDESYN shell has been used to implement three systems. STRYPES designs structural configurations. STANLAY generates layouts based on a given structural configuration. FOOTER generates foundation types based on building loads and soil types, [ibid]. These three systems are some of the modules of the integrated building design environment (IBDE)¹³.

¹¹ This is unusual in that standards are typically characterised as containing shallow knowledge only and as explicitly omitting deep supporting principles, [Stone & Wilcox, 1987]. Kumar & Topping also discriminate between Design Codes on the one hand and the 'theory of structures' on the other.

¹² In fact, SPEX is the prototype for the proposed DATON module, [Sriram, 1987b (p52)]. Fenves also refers to HI-RISE as the *first version* of DESTINY, (Discussion to [Maher & Fenves, 1985]).

¹³ Other modules include the SPEX system described earlier as well as ARCHPLAN for conceptual planning of the building CORE, designing the service core elements; and a construction planner.

Kumar et al, [1987a, and 1987b], have used the HI-RISE, SACON, SPEX and DESTINY systems as prototypes for a system for the design of industrial buildings, called INDEX. INDEX has been implemented using the Edinburgh PROLOG blackboard shell¹⁴. INDEX is consistent with the DESTINY model but with a few differences. INDEX, for example, maintains a full range of currently feasible designs rather than the best only. Another difference is that INDEX uses a fixed *specialist agenda* rather than a strategy level module like TACON. Finally, INDEX, includes modules for optimisation that are not incorporated into DESTINY.

Details of the INDEX system are given for the ALTSEL module which is concerned with preliminary selection from alternative structural systems. ALTSEL therefore performs at the same functional level as the HI-RISE and ALL-RISE modules. The sub-modules of ALTSEL correspond, roughly, to the subtasks of the HI-RISE system (synthesis, preliminary analysis, parameter selection/design, and evaluation).

The STRANEX module is concerned with the use of FORTRAN analysis routines. Kumar et al, [1987a], outlines two alternative methods of interfacing the FORTRAN programs with the PROLOG implementation of STRANEX. Briefly, the first and more straight-forward method is to simply use PROLOG to write input files for the FORTRAN programs and to

¹⁴ INDEX is an on-going research project. Like DESTINY it is still a model rather than a working system in some senses, [Kumar & Topping, 1991].

collect the results from output files. The second method uses an integrated interface between the FORTRAN and PROLOG code via a C interface. The C interface method is more involved but much faster¹⁵.

Another system, DESDEX (DETDEX), is described in [Kumar & Topping, 1988]. DESDEX is a prototype system for the detailed design of structures. DESDEX embodies knowledge concerning the theory of structures as well as design codes. The prototype occupies the same place in the INDEX system as DATON/SPEX in DESTINY.

The problem area covered by DESDEX is also used to illustrate a system for recovery from *non-monotonic* states, [Kumar & Topping, 1990, and 1991]. This system, DESCON, uses a network of dependencies between entities to control backtracking in the event of changes in specifications or constraint violation. The chain of dependencies is represented by a number of binary *supports* clauses:

supports(Tag1, Tag) [ibid] (p215)

Tags are identification numbers for entries on the global blackboard. This dependency directed backtracking enables the system to remove all entries (and only those entries) made invalid by events such as constraint violation¹⁶.

15 The success of the direct interface method is dependent on the languages involved being able to call sub-routines and functions from the other languages. The analysis program must also be designed to accept direct input rather than input solely from files. Chapter 9 describes a similar file-based interface. This method, however, uses a PC-DOS virtual (RAM) disk to improve file operation efficiency.

16 The module, DESCON, is an example of a qualitative model as described by Clancey, [1989]. Clancey cites failure recovery as one possible use of such models. DESCON could, perhaps, be developed to address other uses of qualitative models such as explanation.

The work of Kumar et al, along with the Maher-Fenves-Sriram systems, utilises a model of the structural design of buildings that progressively refines the accuracy of predictions of behaviour under loads, [Maher et al, 1988]. The initial prediction in the preliminary design (HI-RISE, ALL-RISE and ALTSEL) is inaccurate and components are only designed approximately. Finite element methods (FEM) allow much more precise analysis of loads and performance, but may require intelligent pre- and post- processor assistants (such as MASON, STRANEX or IFE).

The final stage is the detailed design. Here components can be individually and accurately sized and proportioned and checked for compliance with codes (SPEX, DATON, and DESDEX). It is at this stage that the design of component material such as the structural steel and concrete occurs.

4.5 Concrete Design

The most common structural components of buildings are steel and concrete. Concrete is itself a composite material. The design hierarchy for structures must, therefore, be extended to lower levels for the specific selection and proportioning of the constituents of the concrete mix. The main properties that are specified for concrete building material are composite strength and chemical protection of structural steel. Strength and protection of steel are relatively easy to design for. However, either of these properties may become compromised if the concrete suffers from durability problems. Therefore durability is the most important factor in the design of concrete mixes¹⁷.

¹⁷ Refer to the next chapter for a full description of some of the factors relevant to concrete durability.

Some work has been carried out on the application of computers to the design of concrete. One early system was described by Day, [1984]. This paper outlines a mix method based on the surface area of the aggregate. The method itself is novel and the emphasis from the implementational point of view is on *tuning* the mix to take account of necessary local empirical adjustment. No details are given of the language or hard-ware used, and there is no sample run of the program.

Hover, [1985 and 1987], describes another computer program for mix design. This program is based on recognized ACI 211 procedures. It is written in BASIC to be portable between a range of personal computers. The program consists of simple functions and procedures. The main use of the program is in teaching engineering students. The effect of altering some input parameter can quickly be seen in the quantities calculated.

Hover's simple mix design system was later developed into a small expert system for the selection of placement methods for concrete slabs, [Hover, 1987]. The CONSLAB system embodies knowledge from published sources, and from Hover's own experience in the domain. The system is of limited scope and still does not address any durability issues. However, Hover notes the rapid development from what was initially conceived as a simple problem into a relatively complex undertaking.

Brewer, [1987] and Dilly et al, [1987] describe the use of simple spreadsheets in the design of concrete. Brewer's spreadsheet is concerned with mix proportioning and is again based on the ACI 211 recommendations. The spreadsheet described by Dilly et al addresses the issue of durability. This method uses strength as an indicator of probable durability.

Using strength alone as an indicator of durability is a quick and relatively easy method to implement. The method was reported to have been implemented on a programmable calculator. However, this is not always a reliable method and in any case ignores aspects of durability that are not influenced by the designed strength of the concrete; alkali-aggregate reactions, for example¹⁸. Spreadsheets are useful for representing formulae and data-tables. The effect of changes in input can be seen very quickly. Furthermore, as more people become familiar with spreadsheets, many spreadsheet features become a norm for user interfaces.

Given the scope of recommendations contained in British and American Standards for concrete design, it is very unlikely that a spreadsheet could begin to cover anything but the most rudimentary aspects of durable concrete design. Much of the knowledge embodied in concrete standards is best represented as rules. Although spreadsheets can use conditional IF statements; these are not an acceptable method of implementing rules. Without the ability to use rules and structured representations such as frames, a spreadsheet would lack desirable facilities such as explanation of its conclusions.

There are several expert systems in closely related problem domains. Sacks & Buyukozturk, [1987], for example, describe a system for the design of reinforced concrete (RC) columns - EIDOC. As mentioned in section 4.3 above, Koskela et al, [1986, 1988] have written a number of systems: for ready-mixed concrete (RMC) selection and asphalt pavement design. Hozayen & Haas, [1990] have also developed a system for the evaluation and design of pavement mixes. However none of these systems

¹⁸ See chapter 5 for a more detailed discussion of the relationship between strength and durability.

bear crucially on the problems of concrete durability or add to the concepts of design in general described in this and the previous chapter.

There have been a number of attempts to develop expert systems for the design of durable concrete. DESIGN, [Miller, 1985], is a prototype inference engine intended to be useful for general design problems. The initial domain chosen for a DESIGN knowledge-base was concrete design. This knowledge-base, COMIX, contains both New Zealand concrete design standards and knowledge acquired from a Ministry of Works and Development concrete expert.

The COMIX system is based on the decomposition of design into sub-goals. However, Miller, regards each sub-goal as being necessarily a step on the right path towards completion. There is no possibility of evaluating the design at intermediate stages or of back-tracking in the event of failure. The system is therefore dependent on a view of concrete design that allows for a universally 'correct' and invariable path through the sub-goals.

The solution path is provided by the representation of the design data in a hierarchical network of nodes and frames. The domain knowledge is represented as production rules attached to the nodes. Additionally there are data-tables for *bulk-information* and a glossary containing definitions of terms.

Miller states that the COMIX system only implements a subset of the domain and omits, for example, environmental factors. COMIX can therefore be seen as an expert system for mix proportioning and limited code compliance. The system does not address any of the significant problems of concrete durability.

The expert system, CONDURAS, [Rajkumar et al, 1987], is able to conduct both backward chaining diagnosis of distressed concrete and to use forward chaining to make design recommendations. The deductive diagnosis of distressed concrete is described in some detail. The diagnosis is driven by hypotheses; data is obtained from the user in support of the alternative hypotheses and each is assigned a probability. No details are given of design using the system. Since the system is designed with the architectural emphasis on diagnosis, it is likely that the use of CONDURAS would be limited to the statement of design recommendations rather than detailed mix constituent selection and proportioning.

The most significant work with respect to the design of durable concrete is the DURCON system. DURCON is based on the "Guide to Durable Concrete" developed by the American Concrete Institute Committee 201. Additional knowledge is obtained from experts in the domain. DURCON has been under development since at least 1985, and is now commercially available.

To date, no overall description of DURCON has been published. From what is available it is known that DURCON is organized around various durability problems. The development cycle depends on the use of the ACI committee 201 recommendations. These meet three essential features likely to make a knowledge-based system for concrete design acceptable. In Clifton's words, they are:

- (1) agreed upon as the consensus of a team or committee of experts on the durability of concrete;*
- (2) published so that the concrete community could judge their reliability; and*
- (3) accepted by concrete practitioners. [Clifton et al, 1985] (p5)*

The knowledge contained in the guide was analysed as a tree hierarchy structure. Then the tree was represented as production rules. The knowledge-base is augmented by heuristics acquired from experts.

The first knowledge-base developed for the DURCON system was concerned with freeze-thaw exposure [*ibid*]. This module is small (34 rules)¹⁹. It was originally written in BASIC and later converted to PASCAL. The system is described as having the usual expert system architecture of knowledge-base, inference engine and working memory. DURCON is also able to explain its reasoning.

Examples of consultations with DURCON reveal that it has the standard menu-driven and dialogue user interface. Although implementational details are not given, the examples indicate that the explanations are of the *canned-text* variety²⁰.

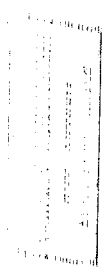
The next published description of the DURCON system outlines the steel corrosion module, [Clifton, 1986]. Again there are scant implementational details. The example session provided shows that the steel system uses a regime similar to the freeze-thaw system. Sulphate attack and alkali-aggregate reaction systems are similarly outlined by Clifton & Oltikar, [1987].

19 It is not actually stated that the system is implemented as a series of individual modules. It is however convenient to describe them as such even if in the actual implementation they are just elements of a single homogeneous system.

20 Canned-text is a text string or pointer to a text-file that is attached to each rule or goal and contains a simple textual description of the function of the rule. A request for explanation results in the associated text being displayed on the screen. The system is unsatisfactory for a number of reasons. It is inflexible and the 'correct' explanation must be anticipated *a priori*.

Frohnsdorf, [1988 and 1990]. provides further information regarding the wider context of the DURCON development work. Most significantly it forms part of a nationwide initiative on concrete research. This national programme utilises simulation models, data bases and a bulletin-board for expert collaboration, [Kaetzel & Clifton, 1986], as well as expert systems.

It is possible that DURCON will benefit from an extended period of incremental growth fuelled by expert collaboration on the bulletin-board system and through a sub-committee of ACI committee 201, [Clifton & Kaetzel, 1988], which has assumed control for future development of DURCON. It may therefore also be possible that the knowledge-representation needs of the system might extend beyond the current limited production rules. The system might therefore become limited by its PASCAL implementation and simple independent sub-modules.



Chapter 5: Overview of Concrete Design

5.1 Introduction

The purpose of this chapter is to provide a brief overview of the main factors contributing to concrete durability. There are, of course, very many references detailing the use of concrete in structures¹. The emphasis of these publications differ depending on their intended use. A very simple introduction is given by Shirley, [1985]. Blackledge, [1987], gives practical advice with the emphasis on actual use of concrete 'on-site'. Neville, [1981], is a standard undergraduate civil engineering level text-book. BS 8110, is the main British standard document dealing with concrete. Details of each of the constituent materials and the specification practices is given by BRE publications, [BRE, 1987a, and 1987b]. These publications are referenced in the appropriate places in the following sections.

Various authors have analysed and commented on the common causes of failure in buildings, [Cusens, 1984, Somerville, 1986 and Newman, 1987]. The figures from a French survey, [Paterson, 1984], reveal that poor design causes 37% of failure in structures. This figure is corroborated by similar surveys showing 34% to 57% of structural failures caused by design faults, [Cusens, 1984]. There is therefore obvious scope for improving concrete design practices.

Concrete design involves many different activities. The initial structural design of a building entails certain requirements on the designed

¹ Since the investigation of concrete durability is not, in itself, a goal of this thesis; there is no attempt to conduct an exhaustive literature review on the subject. The references cited are from a narrow selection of BRE publications, conference proceedings and journals.

strength and workability of the concrete and also some other factors such as maximum aggregate size. This in turn determines the types and quantities of the basic mix constituents - cement, water, aggregates and admixtures. Additionally, the environment in which the concrete is to be situated must be considered. This may entail modification of some of the quantities in the mix in order to minimise risk of frost-damage, steel-reinforcement corrosion, or attack by aggressive soils or groundwaters. Other factors are also relevant to the durability of the concrete such as the risk of alkali-aggregate reactions (AAR)².

The notion of design life of concrete structures is surprisingly under-developed, [Cusens, 1984, Newman, 1987, Somerville, 1984 and 1986]. The design life of a structure is a function of a number of considerations which are listed below:

- * definition of serviceability given the intended use of the structure
- * definition of the time scale over which the level of serviceability is required
- * specification of the maximum initial costs of construction
- * some element for maintenance and rehabilitation of minor failures
- * anticipation of environmental loading (in addition to the structural loading)
- * anticipation of the resistance of the concrete to the environmental and structural loadings

² See [Popovics, 1987], for a classification of deterioration based on mechanism

* consideration of the variable standards of material and construction

There are a number of problems in the consideration of these factors. The serviceability of a structure can only be defined with a corresponding measure of when the structure has failed the serviceability requirements: the performance parameters, [Cusens, 1984]. The time scale over which the structure is required to be serviceable can be rarely anticipated except with structures that are intended for very temporary use or intended to be 'permanent'³. Predicting the equivalent capital costs of the maintenance is a complicated economic calculation.

Many commentators describe problems relating to predicting durability performance. The difficulty of anticipating the full environmental loading is noted by Newman, [1987]. Predicting the long-term properties of the materials used is problematic because of inaccuracies caused by accelerated testing, [Sturup et al, 1987], and the lack of 'in-situ' monitoring, [Somerville, 1984]. These combine to deprive the research into mathematical models of a sound empirical base, [Page, 1986]. Somerville, [1986] describes the difficulties of quantifying the quality of construction standards and shows how construction practices can negate good design. These considerations make the entail the use of some margin of error for safety. This safety margin is implicit in the codes

³ Newman, [1987], gives three broad categories of design life: less than 10 years; 25 to 50 years and over 100 years. This implies that the design life of structures is not considered to be a continuum.

and made explicit in mix design methods⁴.

Different views on service life performance and design lead to a number of design strategies. Figure 5.1 (from Somerville, [1984 and 1986]) shows the broad spread of durability performance from completion of construction over time. Figure 5.1 also shows the three distinct design strategies. Curve (1) illustrates the strategy of over-specifying the concrete in order to ensure that there is no appreciable deterioration over time. This option entails very high construction costs and does not allow for any maintenance. Curve (2) shows the typical performance of a structure designed to a reasonable initial construction cost and with provision for periodic maintenance. Curve (3) represents the consequences of a design without margins for safety, failing catastrophically in unforeseen circumstances.

For the purposes of this thesis the 'design of durable concrete' is therefore loosely defined to be the design of concrete that will resist all durability problems at the minimum construction cost with an intended design life longer than that necessary for most durability problems to cause manifest failure, given some reasonable level of maintenance. The rest of this chapter will briefly describe the more common concrete durability problems.

⁴ This inability to design structures with respect to service life and predicted durability performance is in contrast to the design and maintenance of road networks where there are a number of commercial models in wide use. For example, HDM-III which is used for economic comparison of alternative construction and maintenance options and system BSM which manages road maintenance by consideration of performance indicators and critical deterioration levels, [Wilkins and Tillotson, 1991].

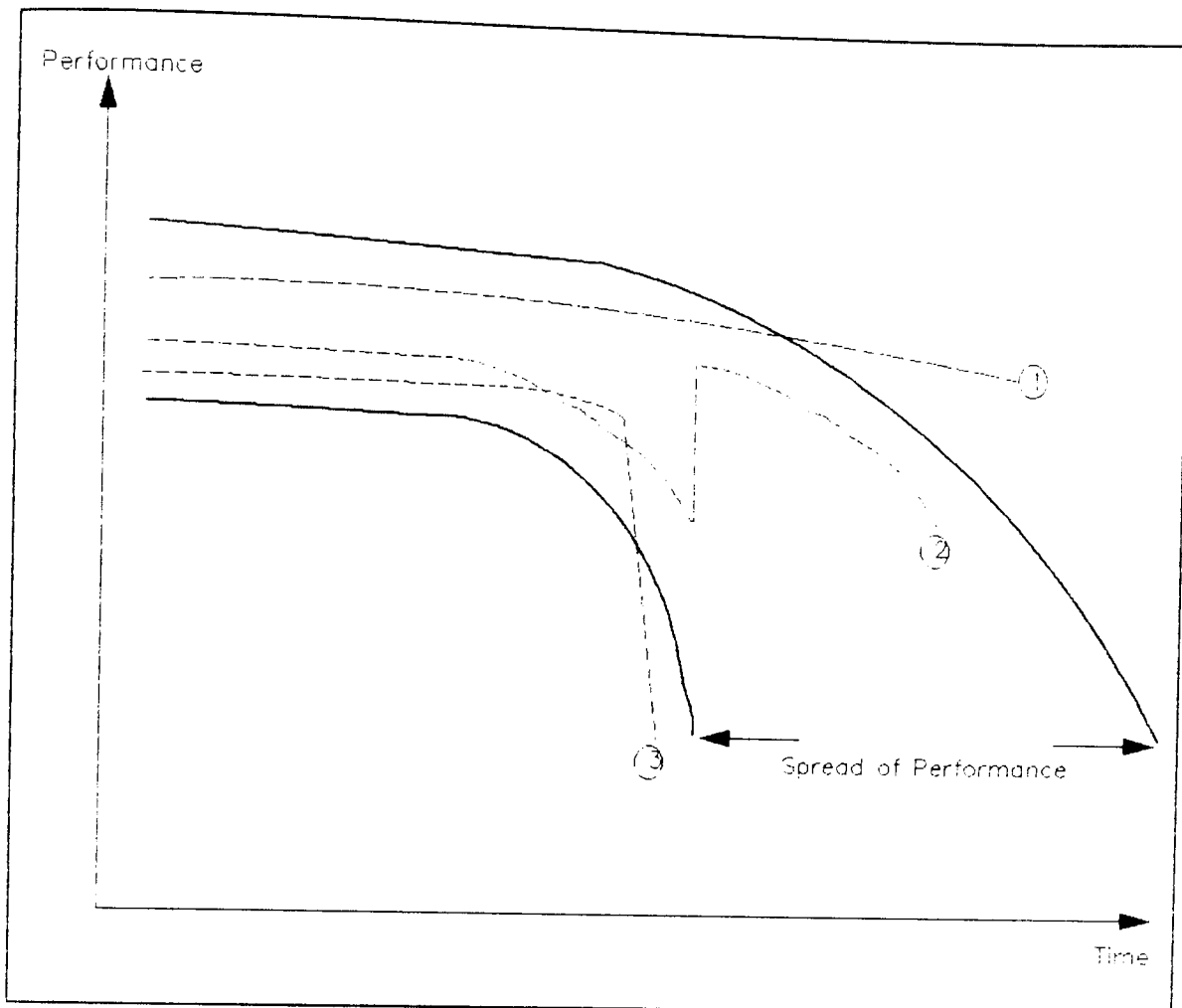


Figure 5.1: Relationship between Durability and Design Life of a Structure
(after Somerville, [1984 and 1986])

5.2 Basic Mix Design

The basic mix design of concrete is the process of determining the main quantities of the mix components - cement, aggregates and water. These components must be combined in proportions that result in concrete with the required properties of strength, workability and durability. There a number of mix design methods, [Shirley, 1985, Neville, 1981, and

Teychenné, 1988]. These are based on the relationship between strength and water/cement (W/C) ratio for a given set of ingredients. Figure 5.2 shows the relationship between compressive strength (in N/mm²) and W/C for a range of cements. The use of this diagram in calculating mix proportions is explained in detail in chapter 10.

Strength is the measured compressive strength of the concrete at a given age. The required strengths of the various structural components including specific concrete members is determined from the structural design. Workability is a relative measure of the ease with which the concrete can be placed into position and compacted before setting and curing. Durability will be discussed in more detail in the sections below. Most durability problems are better resisted by concrete with low permeability.

The compressive strength of concrete is influenced by a number of factors - the age, temperature of curing, W/C ratio and degree of compaction. Compressive strength is usually specified as that to be attained at a given age. The temperature of curing is limited to within a narrow range for the conditions in which concrete is usually laid. Concrete should always be fully compacted to avoid air voids.

The strength of concrete is therefore mainly influenced by its porosity and this is proportional to the W/C ratio as shown in figure 5.2 below. The curves in figure 5.2 represent contours of a surface. Each contour corresponds to the relationship between strength and W/C for a particular combination of materials.

The type of cement used will affect the rate of strength development due to the differing rates of hydration of various cements. The shape

Illustration has been removed for copyright restrictions

Figure 5.2: Relationship between strength and W/C ratio for a Range of Cements

(from Teychenné, [1988])

and type of aggregate will also have an effect, with crushed aggregates generally giving greater strength than uncrushed, smooth aggregates.

Day, [1984], describes an alternative mix design method that emphasises the influence of the surface area of the aggregate on water-requirements and the cohesion of the concrete.

Workability is a property of fresh concrete and reflects the ease with which the concrete can be transported, placed and compacted. Complete compaction is vital for the concrete to achieve its full potential strength, as mentioned above. Workability is measured by various means such as the slump test or Vebe time. The main factor influencing workability is the water content of the concrete. The secondary factor is the shape, grading and type of aggregates used.

In addition to the specifications of strength and workability and any additional specifications arising from durability considerations; location and budget will determine that various materials will be used preferentially. For example, aggregate is cheaper than cement so an economic design will seek to minimise the cement content of the mix. Generally, ordinary portland cement (OPC) will be cheaper than special cements designed to have particular properties such as rapid hardening (RHPC) or resistance to sulphate attack (SRPC). Since a large component of aggregate costs is transport, aggregates from a local source will be preferable to those from a remote source.

Concrete mix design therefore has various conflicting goals; maximising workability and minimising cost against maximising strength and durability. The process admits some of these goals as constraints which cannot be compromised (strength and durability) and the others as factors to be optimised given the satisfaction of the constraints. Since

many of the desired properties depend on factors of the various constituents that can vary over a short interval, the constraints should reflect expected variations in quality of the materials.

After the initial mix design phases, it is the usual practice to make several test batches. These can be tested to ensure that, for example, the strength and density requirements have been achieved. The design process should facilitate redesign to correct any failings of the test batches. It should also be flexible to allow design engineers to alter the model to reflect local conditions and materials, [Day, 1984].

5.3 Sulphate Attack

Solid sulphate salts are unable to react with concrete. In solution however sulphates can react with $\text{Ca}(\text{OH})_2$ and with calcium aluminate hydrate. This reaction forms gypsum and calcium sulphoaluminate. The reaction products have much greater volume than the original compounds. Thus expansion and degradation of the concrete occur.

Magnesium, calcium and sodium sulphates occur in some clays and the groundwater from these clays can contain sufficient sulphates to react with concrete. Additionally sulphuric acid solutions can occur in run-off from colliery tips and marshy country. Finally some building materials can contribute sulphates to the ground water. BRE digest 250, therefore lists several clays and other factors that can be considered to constitute a potential risk to concrete.

Risk from sulphate attack can be increased by groundwater movement and by mechanisms such as the migration of salts caused by concrete with one face exposed to sulphate solution with another face subject to

evaporation. Risk is decreased by the use of different cements with various constituents. Specifically, the amount of tricalcium aluminate in the cement is directly proportional to the potential sulphate damage.

Digest 250 and BS 8110: part 1 classify potential risk by reference to the quantities of sulphate salts measured by various methods of groundwater testing. Higher levels of sulphates indicate higher risk of attack. In addition the type of construction (mass or reinforced concrete and situation with respect to groundlevel) is used in the determination of risk.

The degree of risk of sulphate attack has several effects on the concrete mix design. In order to negate the risk of attack, digest 250 sets constraints on the type of cement used and the quality of the concrete. As mentioned above, cements with lower levels of tricalcium aluminate are less vulnerable to attack. Thus sulphate resisting Portland cement (SRPC) or supersulphated cement may be specified in areas of high sulphate-bearing groundwaters. Use of pozzolanas has the effect of removing some Ca(OH)_2 and can therefore also reduce the reaction⁵.

Good quality concrete in the context of durability means concrete of low permeability. Permeability is a function of the porosity of the concrete gel and this is in turn dependent on the cement and water quantities used in the mix and also the methods of placement. This subject is covered in more detail in section 5.3 on mix design. For the purposes of ensuring durability against potential sulphate attack, digest 250 and BS

⁵ Mathews, [1987a and 1987b], describes the properties of pfa and its use in concrete and outlines other theories of how pfa influences sulphate attack and other durability problems.

8110 set constraints on the minimum cement content and maximum W/C ratio. These limits are determined by the risk classification and type of cement used.

5.4 Alkali Aggregate Reaction

Alkali Aggregate Reaction (AAR) is a subject of confusion and controversy⁶. The main references used in this section are BRE digest 330, [BRE, 1988] and the guidance notes on prevention of alkali aggregate reaction produced by the Concrete Society, [1987].

AAR is a reaction between alkalis in the concrete pore solution and silicates in the aggregates. It would seem that higher levels of soluble alkalis imply greater potential reactivity, [Nixon & Page, 1987, Struble & Diamond, 1986 and Dobie, 1986]. The product of this reaction is a calcium alkali silicate gel. This gel can imbibe water with a consequent increase in volume. This increase in volume disrupts the concrete. Swamy & Al-Asali, [1986], report on the effects of AAR on compressive strength and other engineering properties. A similar reaction can occur with argillaceous limestone (alkali carbonate reaction).

Three essential conditions must be met before damage from AAR can occur. These three conditions and their consequences for prevention of AAR by design are discussed below.

The aggregate must contain reactive forms of silica. The most common forms of reactive silica are those with a disordered structure - opal for example. There is no British Standard test for susceptibility to AAR,

⁶ For example, BRE Digest 330 states that AAR affects only 170 structures in England. French, [1986], in contrast found AAR to be present to some degree in 40% of a sample of 300 structures.

[BRE, 1988 and Hudec, 1982]. Advice on avoiding reactive aggregates therefore tends to focus on determining the history of previous reactivity exhibited by a given aggregate⁷. Minerals recognized to present some risk in the UK are microcrystalline and cryptocrystalline quartzes, chalcedony and strained quartz. Thus flints and cherts are usually classified as potentially reactive. Note that because of the high transportation costs and increasing scarcity of aggregates, the use of reactive aggregates may be unavoidable.

In addition, the proportion of reactive silica is critical. A figure of 60% for flint and chert is given by BRE digest 330. At this proportion - the 'pessimum' - the reaction is 'overwhelmed' in some way and expansion decreases. The pessimum proportion varies proportionately to the reactivity of the mineral⁸. The practical effect of this is that a coarse and fine aggregate that are individually inert can become reactive in combination (and vis versa).

The second requirement for AAR to be possible is a sufficiently high level of alkalis in the concrete mix. Alkalis are mainly contributed by the cement as sodium and potassium oxides. Alkalis can also be found in sea-dredged and other aggregates, [Gutt & Collins, 1987 and Stark & Bhatt, 1986], and some types of admixture. The effect of NaCl, for example, on OH⁻ concentration is discussed in [Nixon et al, 1986, Nixon & Page, 1987, Canham, Ian, 1987 and Chatterji et al, 1986].

⁷ Dolar-Mantuani, [1983], for example, gives a relatively comprehensive list of reactive aggregates (in table 7.1). See also [Smith & Raba, 1986].

⁸ Struble & Diamond, [1986] reports a pessimum proportion of 2-4% with a narrow distribution for opal. They also report, [*ibid*], a pessimum of 30% for novaculite with a wider distribution.

Various limits on the level of alkali necessary to trigger AAR are given. The equivalent Na_2O level of the alkalis in the cementitious materials can be limited to 0.6%⁹. Such a low alkali cement should be unreactive with any aggregate. Either low-alkali cement can be specified or cement replacements like pulverised fuel ash (pfa), condensed silica fume or ground-granulated blastfurnace slag (ggbfs) can be used, subject to some considerations discussed below¹⁰.

If there are significant amounts of soluble chlorides in the aggregates or alkali bearing admixtures, a 3.0 kg per cubic metre limit is set for the Na_2O equivalent alkali content of the entire mix¹¹. The 3.0kg limit can be met by limiting the amount of cement used; by washing the aggregates; by using non-alkali admixtures or by using low alkali cement or cement replacement materials. There is, however, some controversy as to the amount of alkalis removed or contributed to the pore solution by various cement replacements.

From a brief literature survey covering a short period of time and a narrow range of publications; most commentators noted at least some reduction in the alkali concentration entailed by using pfa, [Idorn & Roy, 1986, Oberholster & Davies, 1986 and Durand et al, 1986]. The reduction in concentration is usually explained by the actual removal of ions from the pore solution rather than as a dilutant effect¹². Most of

⁹ A limit of 1.1% if ground-granulated blastfurnace slag is used.

¹⁰ In contrast to the generally detrimental effects of AAR; Idorn & Roy, [1986] note some benefits of alkalis in concrete such as improved rheological properties and increased density.

¹¹ Canham, [1987] reports that the effect of OH^- concentration is on the speed of reaction whereas the extent of damage is influenced by the total amount of OH^- available. He uses this as an argument for use of the 3.0kg limit in preference to the 0.6% limit.

¹² For example, the preferential reaction of pfa with potassium ions over sodium is reported in Canham, [1987] and Glasser et al, [1988].

these reports observe that the beneficial effect of pfa is dependent on such factors as the alkali content of the cement¹³, the levels of replacement of cement by pfa, and the chemical composition of the pfa, [Farbiarz et al, 1986, Dunstan, 1981, and Kollek et al, 1986].

Most commentators reported beneficial use of ggbfs, [Hogan & Meusel, 1981, Hogan, 1985 and Idorn & Roy, 1984]¹⁴. The effect of ggbfs on the alkali concentration is usually described as being due to dilution, [Canham, 1987 and Nixon & Page, 1987]¹⁵. Because this dilutant effect is not based on a reaction mechanism, ggbfs usually takes effect more quickly than pfa, [Canham, *ibid*, Nixon & Page, *ibid* and Glasser & Marr, 1984].

Micro silica fume is reported to decrease AAR expansivity, [Gjørsv, 1983]. This effect is either due to the reduction of alkali ion mobility, [Chatterji et al, 1986], or by decreasing the alkali concentration, [Nixon & Page, 1987]. The use of other cement replacement materials is less well documented although [Nixon & Page, [*ibid*] and Kollek et al, [1986], report marked benefits from using zeolitic natural pozzolanas.

In contrast, some researchers found that pfa and ggbfs make a significant contribution to the alkali content of the pore solution, [Barlow & Jackson, 1988]. These researchers are therefore more conservative about the potential benefits of cement replacement. Hobbs,

¹³ [Canham, Ian, 1987], [Nixon & Page, 1987], [Smith & Raba, 1986], [Gaze & Nixon, 1983], [Diamond, 1981], [Tenoutasse & Marion, 1986], [Nixon et al, 1986].

¹⁴ Kollek et al, [1986], report that ggbfs increases the OH⁻ concentration but still has a beneficial effect on AAR at higher levels of substitution.

¹⁵ Yamamoto & Makita, [1986], however, note an "inherent function of slag" beyond dilution.

[1986], recommends that one sixth of the alkali of pfa and half the alkali of ggbfs should count towards the total alkali content of the concrete. This recommendation is endorsed by BRE digest 330.

Such pozzolanic additions are siliceous materials and therefore have some contribution to the consideration of the pessimum proportion of silica¹⁶. The amounts of pfa and ggbfs are therefore constrained to a minimum of 25% and 50% respectively.

The third requirement for AAR damage potential is sufficient moisture to enable the reaction products to expand and cause cracking. A figure of 75% internal relative humidity has been given as a limit below which there should be no risk of AAR damage. This figure makes most concrete exposed to weather potentially at risk from AAR in addition to foundations, water-retaining concrete and concrete subject to condensation.

A secondary effect of the moisture conditions with respect to AAR is that of alkali migration due to moisture movement¹⁷. No figures are given for quantifying this effect. Instead some situations are simply classed as extra-risk. An example of a high risk situation is a foundation slab with one surface exposed to drying conditions. In these cases it is recommended that the 0.6% alkali limit be used rather than the 3.0kg limit. It is also recommended that the constraint on the alkali content includes a factor for the possible variation of alkali content in the materials over time.

¹⁶ See for example, Nixon & Page, [1987] and Farbiarz et al, [1986], on the effect of silica fume on the alkali/silica ratio.

¹⁷ For example see, [Nixon et al, 1979] on the effect of concentration of alkalis by moisture migration (possibly countered by the leaching effect of rain).

5.5 Steel Reinforcement Corrosion

Reinforcing concrete with steel provides it with tensile strength which it would otherwise lack. However corrosion of the reinforcing steel can cause damage to the surrounding concrete in the form of cracking and spalling. The concrete itself can protect the steel from corrosion by preventing the ingress of water and oxygen and additionally by maintaining the steel in an alkali environment to prevent oxidation.

There are two primary mechanisms by which the protective environment provided for the steel by the concrete can become compromised and corrosion ensue. The first of these is the process of carbonation. The alkaline materials in the concrete react with acidic solutions of carbon dioxide and sulphur dioxide. This reaction occurs initially at the surface of the concrete. This reaction progresses inwards through the concrete until the steel is exposed to the acidic solutions and corrosion commences.

The humidity and concentration of the acidic gases in the atmosphere directly influence the rate of carbonation. Any cracks in the concrete will expose the steel to attack immediately. The extent of carbonation is also dependent on a number of factors which can be controlled to delay the corrosion of the reinforcing steel. Decreasing the permeability of the concrete will provide better protection by slowing the rate of ingress of the acidic solutions and thus slowing the rate of progression of carbonation. Increasing the depth of cover provided by the concrete will also delay the onset of steel corrosion.

The second mechanism of attack is due to the presence of chloride ions in the concrete pore solution which can cause corrosion. The chlorides

present in the concrete during the early stages of hydration react with calcium aluminates and do not present a problem. However use of cements with low levels of tricalcium aluminate can leave sufficient free chloride ions in the pore solution to cause corrosion¹⁸. Chlorides can be introduced in a number of ways; as salt from sea-dredged aggregates or as a set accelerator. The risk from chlorides can therefore be reduced by limiting the chloride content of the mix.

Chlorides are also present in situations such as concrete exposed to sea-water or road de-icing salts. The threat from these external sources is subject to the same limiting factors as carbonation but the increased risk to concrete in use in marine or highway environments can be offset by more stringent requirements for depth of cover or quality of concrete.

In situations of severe risk there are a number of special measures that can be taken to prevent steel reinforcement corrosion. These measures include cathodic protection; coating the steel with some corrosion resistant or sacrificial coating and use of corrosion inhibiting admixtures to the concrete. These measures are subject to further research and are each recommended for different situations. There is also variability in protective ability between different commercial products.

5.6 Other Problems

Concrete may be subject to other durability problems. The three main categories not covered in the previous sections are attack by aggressive chemicals, frost damage and abrasion damage.

¹⁸ Note that this provision is in direct contrast to sulphate resistance which increases in proportion to the levels of tricalcium aluminate.

The main influences on the effect of chemical attack are usually the permeability of the concrete and the type of cement used. Neville, p444-445 gives a table of various aggressive chemicals and the normal order of ability of cements to resist these chemicals. Attack by sulphate-bearing groundwaters has already been discussed. The other main sources of chemical attack come from seawater, acidic atmospheric gases and soils and sewage.

Concrete in cold climates can suffer from frost damage. The mechanism of damage is the expansion of the pore water in the concrete as it freezes. The freezing water thus causes expansive damage of the concrete. The expansion of the concrete is not reversed when the water thaws thus successive freeze-thaw cycles cause cumulative damage.

The damage caused by freezing is dependent on a number of factors. The degree of saturation of the concrete prior to the freeze cycle is of primary importance. The freezing itself can cause more moisture to be drawn into the concrete by osmotic pressure which exacerbates the problem. This diffusion arises from differences in the concentrations of salts in different areas of the concrete¹⁹.

The second main influence on frost damage is the pore structure of the concrete. Large pores remain air-filled and therefore contain no water. The freezing point of the pore-water varies with the pore size due to the pressure of the surface tension which increases as size decreases. The very small gel pores thus prevent the freezing of the pore fluid at normal temperatures. The micro-structure of the pores is also influential

¹⁹ Note also the adverse effect of de-icing salts on steel reinforcement and AAR durability problems.

since discontinuous pores prevent the diffusion of pore water. This can lessen the movement of pore fluids towards the freezing areas and also decreases the initial saturation of the concrete.

Abrasion and erosion resistance are not easy to design or test for due to the variety of abrasive mechanisms. The main criteria seem to be the compressive strength of the concrete and the size and strength of the aggregate used. The quality of the surface layers of the concrete are also important so the method of curing and finishing are influential.

Other durability problems arise due to the use of new materials, for example, high-alumina cement, [Newman, 1987]. Several authors also comment on the possible implications of the changing properties of existing materials on durability like the increasing alkali content of cements, [Pomeroy, 1987, Newman, *ibid*] and Somerville, 1986]. O'Brien et al, [1987], give an example of the compaction problems arising from the use of high slump, self-compacting piling concrete.

A final point to note is that there are a number of attempts to simplify the design of durable concrete, for example, specifying by strength only [Deacon and Dewar, 1982], or by calculation of air-content from gravimetric and volumetric yields, [Dilly et al, 1987]. Although these methods would not address durability problems such as abrasion or AAR, they do have the advantage of being easily specified and tested for. The W/C ratio, for example, is easily specified but notoriously easy to ignore and impossible to measure in placed concrete. In Newman's words:

if you cannot measure it, don't specify it (p1263), [1987]

5.7 Conclusions from the Literature Searches.

In the light of the above discussion of the various aspects of concrete durability, it is possible to draw a number of conclusions regarding the subject as an expert system domain. The knowledge in the domain is, in some cases, not generally agreed. The life span of concrete structures in relation to its history of use entails that there is no firm basis for life-cycle prediction of structures, even those using common materials.

The specifications of the basic material, cement, have changed significantly over the previous decades. In addition, new materials are constantly being introduced. Many of these materials have poorly understood properties, especially in the long term. Thus the domain requires exhaustive research to quantify the basic properties of concreting materials and their reactions with each other.

In the absence of such conclusive understanding there remains a wide range of results arising from different concrete research projects. In the light of the instability of concrete durability knowledge and the diversity of current opinion among experts, the only suitable knowledge source for concrete durability expert systems is the published documents of acknowledged authorities.

The use of such publications as British Standards, BRE digests and other codes of practice ensures that inferences based on such knowledge sources can be stated with some certainty within the scope of the document. However an expert system based on a code must limit its scope to the extent of that code. This entails that concrete durability expert systems can confidently be created for code compliance checking or as requirements generators.

There is currently no possibility of producing an expert system that could claim to cover the entire domain of concrete durability design with acceptable authority. The literature search of current concrete design systems confirms this conclusion. Most design systems in general are limited to component synthesis and compliance checking within the scope of established prototypes and publicly accepted codes of practice.

Chapter 6: Early Work: VAX - PROLOG

6.1 PROLOG as an Expert System Implementation Language

PROLOG is a language developed at the Universities of Marseilles and Edinburgh in the 1970's. 'PROLOG' is derived from 'PROgramming in LOGic' and the language itself is based loosely on the ideas of logic programming¹. It is not necessary to describe PROLOG in detail here as various examples throughout this chapter will illustrate the pertinent features of the language. There are many books available about PROLOG emphasising its variant dialects, uses and levels of subtlety. The standard for the language is usually accepted as being set in "Programming in PROLOG", [Clocksin & Mellish, 1984]. Hu, [1987] and Vadera, [1989], give practical introductions to the language and Taylor, [1988], makes comparisons between PROLOG, LISP and C.

PROLOG was chosen as the language for early development of this project for a number of reasons; primarily its availability on the VAX main-frames at Aston University. The specific version of PROLOG chosen was that implemented under the University of Sussex/System Designers plc POPLOG system. POPLOG combines PROLOG, with LISP and POP-11

¹ The fundamental difference between PROLOG and logic is that in a logic system all 'true' propositions are true simultaneously whereas PROLOG processes its rules and facts sequentially. There other differences such as the meaning of 'false' which in PROLOG can only be interpreted as 'not-provable' in the current state.

into one integrated programming environment. It was felt that this would not restrict the choice of representation and inference strategy as drastically as a shell system, [Allwood et al, 1987]².

Various authors provide positive evidence that PROLOG is a useful language for expert systems work. Clark & McCabe, [1982], give several examples of existing or easily implemented features of PROLOG that facilitate its use for expert systems. Pereira & Oliveira, [1983], also illustrate the utility of PROLOG for expert systems work and describe a case study. Finally, familiarity with PROLOG and the versatility of the POPLOG VED editor made it a fairly productive medium to work in, thus enabling the system to benefit from a rapid development cycle.

It is possible to use PROLOG itself as a simple expert system. The basic elements of the language are facts and rules. For example, the following rule:

```
IF  the specification is for low workability,  
   AND water/cement ratio is 0.4  
   AND aggregate is uncrushed  
   AND maximum nominal aggregate size is 20mm  
THEN the required aggregate-cement ratio is 4
```

is the rule-based representation of a row of table 5, (p16), in "Introduction to Concrete", [Shirley, 1986]³. This rule could be implemented in PROLOG as:

² Allwood's main criticism of expert system shells is that they may reduce generality. Hickman, [1986], comments on the undesirable degree of determinism which may bias the way in which a knowledge engineer interprets knowledge and therefore advocates the use of object-oriented programming techniques (see the discussion of frames in chapter 2).

³ This document provides a simple mix-design method and was used as the knowledge source for the PROLOG system described in the chapter.

```
agg_cement_ratio(4):-
    workability(low),
    water_cement_ratio(0.4),
    max_nominal_agg_size(20),
    agg_type(uncrushed).
```

The rule could have both declarative and procedural interpretations. For example, if the water/cement ratio (W/C) exists as a fact:

```
water_cement_ratio(0.4).
```

Then the rule is, in part, declaratively stating a relationship between W/C and the aggregate/cement ratio (A/C) required for the mix. If there are no matching facts about W/C the rule might cause another rule to fire that calculates the desired ratio.

```
water_cement_ratio(X):-
    water_quantity(W),
    cement_quantity(C),!,
    X=W/C.
```

Or infers W/C from the strength grade specification, or from the user:

```
water_cement_ratio(0.4):-!,
    strength_grade(44).
```

```
water_cement_ratio(0.5):-!,
    strength_grade(33).
```

```
...
```

```
water_cement_ratio(X):-
    write("What is the water/cement ratio ?"),
    read(X),
    X>=0,X<=1,!. 
```

```
water_cement_ratio(X):-
    write("Error: water/cement ratio must lie between 0 and
    1"),nl,
    write("Please input a valid value"),nl,
    fail. 4
```

⁴ This rule is a simple example of one of the most powerful features of PROLOG - backtracking. If the third clause fails (by virtue of the value being outside the valid range) the fourth clause is called. This displays the appropriate error message and then fails itself. PROLOG then backtracks to the previous rule which it seeks to re-evaluate. If the first-rule succeeds the system is prevented from backtracking (in event of the failure of a later rule perhaps) by the cut (!). An alternative way of implementing the rule would be to make the second clause have a recursive call to the predicate itself in place of the fail statement.

PROLOG could use all of these rules in any order. However, in practice, the knowledge engineer would know at what stage in the consultation the W/C ratio is required. For example, the system would already have determined its value (and asserted it as a fact) from consideration of the strength requirement. Therefore in this case there would be no need for rules to ask the user for the value directly. Thus PROLOG can make use of meta-knowledge inherent in the ordering of the rule-base and the order of the clauses in the 'top-level' rule.

Without specifying the rest of this basic system it is possible to see how PROLOG could make use of this sort of direct implementation as a very basic expert system. In practice however there are many ways in which PROLOG can be extended. Sterling & Shapiro, [1986], for example, show how to develop the basics of the language into more advanced techniques such as meta-interpreters for expert systems (see below).

There are a number of advantages to be gained from not using PROLOG directly to represent the rules of the knowledge-base. It can be used as an interpreter of rules represented by some other method. To continue the example, the rule given above for determining the A/C ratio might be represented thus:

```
kb_rule( mixtab, [ low, w/c(0.4), mm20, uncrushed, a/c(4) ] ).5
```

PROLOG would be used to create an expert system shell that would consist of rules that could interpret this representation. This rule would be used in the context of needing a value for A/C ratio. The shell would interpret the rule so that when determining A/C the relevant kb_rules

⁵ It is possible to see from this example how PROLOG could be used to read this design 'knowledge' from a mix design data-base file that would more closely reflect the original tabular representation in the source text.

have "mixtab" as the first value (the context). This particular rule would have the interpretation that the A/C ratio is 0.4 if the other values are valid (matched to facts in working memory or otherwise determined).

Other rules might provide a simple 'canned-text' explanation facility:

```
explanation( mixtab, "The value is needed in order to determine
the aggregate/cement ratio")
```

This could be used to provide support in case the user was asked to provide a value for aggregate type, for example, and wanted to know why.

This representation method could be developed into a full frame-based system.

```
mix_design{ design_ID: ask_user,
workability: basic_spec,
strength_grade: basic_spec,
environment: basic_spec,
reinforcement: basic_spec,
cement: infer_cement_type(environment.SO3_content),
fine_aggregate: material_spec,
coarse_aggregate: material_spec,
water_quantity: calc_water_quantity(W/C_ratio),
alkali_quantity: calc_alkali_quantity(cement_alk,agg_alk),
expl_text: "You are determining the basic mix quantities" }.
coarse_aggregate{ classification: ask_user,
maximum_nominal_size: calc_max_agg(reinforcement.spacing),
chloride_content: test,
silica_content: ask_user,
crystal_structure: ask_user,
shape: ask_user }.6
```

One of the simplest extensions to PROLOG to facilitate its use as an expert system shell is the use of 'syntactic sugar' to make the rules

⁶ In practice, the specification of frames is only possible after knowledge elicitation and analysis in order to determine the correct structure for the hierarchy. These examples are not the result of such deliberations and are provided only as a rough illustration.

more easily read, [Clark & McCabe, 1982]. For example many PROLOG dialects allow the use of alternative reserved words and operators that would allow the following statement of the A/C rule:

```
agg_cement_ratio_is 4 IF
  workability_is low
  AND water_cement_ratio_is 0.4
  AND max_nominal_agg_size_is 20
  AND agg_type_is uncrushed.
```

Having briefly discussed the use of PROLOG in expert systems; the rest of this chapter will describe the first prototype system developed as part of the exploration of expert systems in concrete design and the conclusions from this initial phase.

6.2 The Mix Design System

The first practical problem tackled was the development of a system to derive the basic mix proportions from the strength and workability requirements of the concrete. This work was intended as a test bed for the exploration of the use of PROLOG for expert system applications.

The knowledge source chosen for the prototype system was "Introduction to Concrete", [Shirley, 1985]. This is an elementary publication with background material on cement and concrete in addition to a very simple mix design method⁷. It was felt that implementing the knowledge contained in the document would constitute a useful introduction to both expert system methods and the domain itself.

⁷ The method used is similar to but much simpler than that used in the later work, [Teychenné et al, 1988].

6.2.1 The Inference Engine

The method described in "introduction to Concrete" is an algorithm for determining concrete mix proportions based on specifications of strength grade, workability required, coarse aggregate size and nature and fine aggregate grading. This algorithm is implemented at the top level of the system and is one of the basic options, *designer*, open to the user at the start of a consultation⁸.

Designer first of all consults the knowledge-base⁹ to discover what must be specified for the problem. This is a list containing such items as strength-grade, workability and maximum aggregate size. The list of values that must be specified is *Spec_tags* in the listing given below.

Specifier performs the function of extracting the specifications from the user. That is, it obtains the actual value of the strength-grade and other *Spec_tag* items. These values are held in the *Spec_list*. *Describer* informs the user what it is he has specified. The knowledge-base is again consulted to discover what must be inferred and calculated from the specifications, (the *Calc_tags*). Finally, designer calls the predicates which actually calculate the mix proportions, describe the design to the user and save it to disk. Once all this is done the user is returned to the top level of the system.

⁸ The other options *editor* and *saver* are described below.

⁹ This knowledge base is in fact the PROLOG data-base or working memory. Its contents are the items of knowledge the user has created using the editor described below. These items include the explicit meta-knowledge about what data needs to be specified by the user and what must be calculated.

designer:-

```
fact(specifications,Spec_tags),      %part of the k-base
specifier(Spec_tags,Spec_list),
describer(specified,Spec_tags,Spec_list),
fact(calculations,Calc_tags),        %part of the k-base
design_describe_save(Spec_tags,Calc_tags,Spec_list),
start. 10
```

It is not necessary to describe specifier and describer in detail. They emphasise the use of knowledge-base items to provide lists of alternatives, range-values and text to be used in obtaining replies from the user. The intention was to create generic predicates that could be used in other modules and which had no domain-specific information content. For example there is just one *get_spec* predicate which is applied recursively to the list of things to be specified by the user in order to obtain appropriate values. *Get_spec* prompts the user; finds a list of optional responses and gets a valid response off the user.

```
specifier([Htag|Rtag],[Hspec|Rspec]):-
    get_spec(Htag,Hspec),
    specifier(Rtag,Rspec).
specifier([],[]).
```

```
get_spec(Tag,Spec):-
    write('Please specify '),
    write(Tag),nl,nl,
    findall(Opt,fact(Tag,[Opt|_]),Optlist),
    get_rep(Tag,Optlist,Spec),!.
```

Design_describe_save is called with lists of what must be specified; what must be calculated and what the user has actually specified. The first clause checks to see if a mix has already been saved that conforms to the given specifications:

10 This predicate and the following code fragments do not constitute an exhaustive program listing. They are reproduced in order to illustrate the essential features of the system. Some lines of code and entire predicate definitions have been omitted where they do not contribute to clear understanding. Comments to the code are identified by a % sign.

```

design_describe_save(Spec_tags,Calc_tags,Spec_list):-
    design(Spec_list,Results),!,
    write('Existing design consulted'),nl,
    describer(resulting,Calc_tags,Results).

```

The system therefore can use the results of earlier design sessions. This facility could have been made more useful by enabling partial matches and slight modifications. The code shown will only use the results from an existing design with identical specifications.

If a satisfactory design does not exist, the second clause for `design_describe_save` determines what must be calculated (according to the `Calc_tags` list) by calling `calculator`. Then the new design is asserted into the knowledge-base once it has been described:

```

design_describe_save(Spec_tags,Calc_tags,Spec_list):-
    calculator(Spec_list,Calc_tags,Results),
    describer(resulting,Calc_tags,Results),
    assert(design(Spec_list,Results)).

```

If it was not possible to calculate or infer the necessary characteristics of the design, the third clause informs the user that this is the case. Ideally the advice on how to re-specify a feasible design will be part of the domain knowledge and would be consulted by this final clause.

```

design_describe_save(Spec_tags,Calc_tags,Spec_list):-
    write('Your specifications are beyond the scope of this
system'),nl,
    write('Please try again. This time try specifying either
higher'),nl,
    write('workability or higher strength. '),nl.

```

Calculator is called when it is necessary to use the `calc` clauses in the knowledge-base to work out a mix design:

```

calculator(Spec_list,Calc_tags,Results):-
    setspecs(Spec_list),
    length(Calc_tags,L),
    scheduler(0,L,calc,Calc_tags),
    collect(Results).

```


This predicate makes extensive use of PROLOG facts in working memory rather than using a list containing the required elements. Calculator calls *setspecs* to create the working memory of items that the user has given specifications for. Given a list of specifications it simply asserts each element into the memory. Once all the calculations have been done, the results are collected from the working memory by *collect*.

The core of the calculator predicate is *scheduler*. This takes as its parameters two values that are used in its consideration of whether the attempt to perform all the calc clauses will ever terminate. The second two parameters are an atom (in this case *calc*) representing the tasks to be performed and a list of things that the task is to be performed on (in this case the list of things to be calculated).

The first definition attempts to call the functor on the parameter at the head of the things to be done by creating an instance of the general clause of the form, for example: *calc(w/c_ratio)*. If this call is successful, the *L* parameter is reduced by one. This parameter represents the length of the list of things to be done. Then scheduler makes a recursive call to itself with the rest of the list of things to be done:

```
scheduler(S,L,Funcator,[Param|List]):-  
    Clause =.. [Funcator,Param],11  
    call(Clause),!,  
    L1 is L - 1,  
    scheduler(0,L1,Funcator,List).
```

¹¹ This is an example of a PROLOG meta-predicate. It creates a new PROLOG clause from a data-item (or vis versa). This can then be executed by PROLOG. For example

```
?- Clause = [ foo, a, b, c].  
Clause = foo(a,b,c).
```

Most PROLOGs have this meta-programming facility which is extremely useful for programming expert system shells.

If the first clause fails, probably because the attempt to call the calc clause failed, the second definition checks to see whether the list of things to be done is longer than the S parameter. This parameter represents how many attempts have been made to call a clause since the last successful attempt.

If L is greater than S , it is worthwhile continuing. The item at the head of the list has already been the subject of the last unsuccessful attempt. Therefore it is not worth trying it again immediately. Hence, the head item is put at the tail of the list of things to be done¹². The S parameter is incremented by 1 and scheduler calls itself on the newly ordered list.

```
scheduler(S,L,Funcor,[Param|List]):-  
    L > S,!,  
    append(List,[Param],Newlist),!,  
    S1 is S + 1,  
    scheduler(S1,L,Funcor,Newlist).
```

The final clause is called when the S parameter finally becomes equal to the L parameter. If the number of unsuccessful attempts is equal to the length of the list of things to be done then all of the things have been attempted unsuccessfully since the last successful attempt and there is, therefore no point in making further attempts. The user is informed of which calls were unsuccessful:

```
scheduler(_,_,Funcor,Params):-!,  
    write('Unable to perform '),  
    write(Funcor),  
    write(' on '),  
    writelist(Params,1),nl, fail,!
```

¹² This is done by appending the rest of the list in front of the 'failed' element. It is possible that some items on the list are only capable of being calculated after other items have been calculated which is why a failed attempt results in pushing the failed item to the back of the queue rather than failing the entire attempt at calculation.

The scheduler predicate is a prototype inference engine for the PROLOG shell for mix design. The calc clauses that are interpreted by scheduler are of the form:

```
calc(Tag,Result):-Procedure.
```

This means that PROLOG code is still required for each knowledge-base rule. A fully independent shell implementation would require all of the code to be held as a data item:

```
calc(Tag,Procedure,Result).
```

This however would require a more sophisticated knowledge-base editor.

6.2.2 The Mix Design Knowledge-Base

Much of the early work on the system was concerned with experimenting with different utilities for manipulating a knowledge base; in particular developing a group of functions which together loosely comprise a crude knowledge base editor. During this development work, the intention was that a single data structure should be used to represent the data and knowledge of the system. As explained above the knowledge-representation should capture all the data-processing needs of the domain without requiring any specific PROLOG coded clauses.

The items in a knowledge-base for the system are ideally all of one form. The form used so far is that items should be single PROLOG facts i.e. unit clauses. This entails that the knowledge-base editor can use PROLOG data-base facilities such as matching, assert and retract. Each item has three distinguishing features: its *type*, its *tag* and a *list*. The structure used is:

```
Type(Tag,List).
```

or, as it is often accessed in the system:

```
Item =.. [Type,Tag,List]
```

The types currently used by the system are:

```
metafact
fact
rule
vocab
design
calc
```

For example:

```
metafact(rule, [' relationship exists between ']).
```

This states the metafact that, in context: 'rule' means that a 'relationship exists between' some properties or objects. These metafacts are used in describing newly created rules to the user.

```
metafact(properties, [ rate_hydration,
                        cement_fineness,
                        other ] ).
```

This metafact states that the properties currently existing in other fact and rule definitions in the knowledge-base are rate_hydration, cement_fineness and other.

```
design([ strength(15),
        work(high),
        size(mm40),
        nature(uncrushed),
        fine_grade(coarse) ],
      [ w / c(0.7),
        prop(40),
        a / c(7.5),
        fine_cont(3.0),
        agg_cont(4.5) ] ).
```

Design items are a little different because the tag for a design is a list of specifications rather than an atom as in the other types. The second

list in design items is the properties resulting from the list of specifications. It is these designs that are used to save the results of previous design sessions and which can be recalled by the *designer* predicate described in section 6.2.1.

The bulk of the items in the knowledge-base are usually facts:

```
fact(str_grade, [strength(30), w/c(0.4)]).  
fact(workability, [ work(high), proc(congested_reinforcement) ]).
```

These first two examples state that it is a fact that for strength grade of 30 you need a water-cement ratio of 0.4, and that for concrete to be used in congested reinforcement you need high workability.

To design a mix the *calc* rules in the knowledge-base need to be able to relate the specifications to the actual amounts of cement, water and aggregate. The knowledge-base therefore contains a large number of facts which effectively comprise a mix-table. The mix table in question is that on p16 of "Introduction to Concrete". The sheer bulk of this table when implemented as rules was the original motivation for much of the work on the editor functions described later.

There is provision in the knowledge-base editor for the creation and modification of facts that could have uses other than mix design. For example:

```
fact('RHPC', [cement_fineness(high)]).
```

This could be interpreted as stating that rapid hardening portland cement (RHPC) is of fine consistency.

Such facts could be used in conjunction with the rules such as:

```
rule(direct, [cement_fineness, rate_hydration]).
```

This rule is stating that a *direct* relationship holds between cement fineness and rate of hydration. This might allow the inference that RHPC has a high rate of hydration.

6.2.3 The Knowledge-Base Editor

The mix design system implemented a large amount of data and knowledge. During the development and refining of the system much of the knowledge inevitably has to be altered. These considerations make the provision for some fast method of adding to and altering the knowledge base of primary importance. Thus much of the work on the system has been concerned with developing a knowledge base editing facility.

The editor should be able to perform a number of tasks. Firstly it should facilitate the creation of new items for the knowledge-base. It should be able to access any item in the knowledge-base and alter or delete it entirely. Although this is not something that is necessary yet, the editor might also have some facility for checking for consistency in the knowledge-base, that is it should be able to check whether a new item contradicts, duplicates or supersedes an existing item.

The task of changing an item involves deleting it from the knowledge-base; extracting the changes in it from the user and then replacing it in the knowledge-base. The task of creating a new item could follow a similar routine but simply copying a blank template item (or indeed a more complete existing item that is similar to the intended

new item) and changing it. Ideally, different types of item should all be handled by the same procedure. This is one of the main motivations for having the items all in homogeneous form.

The *editor* is called by the user in response to a prompt by the top level of the system. Editor prompts the user for a task to perform then, relying on `interpret`¹³ to extract a meaningful response, then calls what the user asks for.

```
editor:-
    write('Do you wish to create a rule type (r) '),nl,
    write('          or a fact type (f) '),nl,
    write('or inspect the knowledge-base type (i) '),nl,
    write('          or quit type (q) ?'),nl,
    interpret(Rep),
    call(Rep).
```

Rule is one of the options given by editor, it creates rule items for the knowledge-base:

```
rule:-
    metafact(rule_types,Subj_list),!,
    write('These are the types of rule you can choose
from:'),nl,
    get_rep(rule_type,Subj_list,Reply),!,
    template(rule,Reply,Rule),
    write('You should specify the properties related by the
rule'),nl,
    metafact(properties,P_list),
    get_props(P_list,Prop_list),
    asserta(rule(Rule,Prop_list)),nl,nl,
    editor.
```

The possible varieties of rules and properties are represented by metafacts¹⁴. One option that is always available is 'other'. This is to allow for the creation of new varieties of rule items. This is essential, especially when the editor is being used to create an entirely new

¹³ see section 6.2.4 for an explanation of the `interpret` predicate.

¹⁴ These metafacts effectively comprise the domain lexicon - the names of the concepts, objects and properties, [de Greef & Breuker, 1985]

knowledge-base. Once the type of rule to be created has been chosen by the user, rule calls *template* to find out what to call it. If it is a new variety of rule, *template* prompts for a name for it and does the necessary book-keeping to keep the metafact up to date. *Get_props* obtains the properties related by the rule being created. The creation of facts is similar to the creation of rules.

A sample session is given below to illustrate the creation of a new fact using the knowledge-base editor¹⁵. The other editor functions to inspect the knowledge-base will not be detailed.

```
Do you wish to consult an existing knowledge-base (y/n) ?
|: n
Do you wish to make a design, edit the knowledge-base or stop
(d/e/s)?
|: e
Do you wish to create a rule type(r) or a fact type(f) or inspect
the knowledge-base type(i) or quit type(q) ?
|: f
These are the subjects you can choose from:
Please choose an option number option subject
                                1    other
|: 1
You are creating a fact for a new subject area !
What is the subject to be called ?
|: RHPC

What items should be on the property list ?
Item...
|: cement_fineness
Item...
|: USA_classification
Item...
|: <return>

These are the properties you should specify for the subject
1 cement_fineness
2 USA_classification

property is.....cement_fineness:
|: high
property is.....USA_classification:
|: typeIII
```

¹⁵ In this sample session the system prompt is ':' and the user's replies are **bold**

The fact you have just created implies that 'RHPC' has the properties:

```
1 cement_fineness(high)
2 typeIII
```

Do you wish to create a rule type(r) or a fact type(f) or inspect the knowledge-base type(i) or quit type(q) ?

! : **q**

Do you wish to make a design, edit the knowledge-base or stop (d/e/s)?

! : **s**

Do you want to save the current knowledge-base ?

! : **y**

under what name do you wish to save the Knowledge-base ?

! : **Example**

New File Knowledge-base saved and Session Ended

It is necessary to have some way of describing the various items in the knowledge-base to the user, for example during the creation and modification of items. It is not sufficient to simply copy the PROLOG code that the items are actually written in; the user is not expected to be able to understand PROLOG.

Describe takes three arguments comprising the three components of a knowledge-base item: the type, the subject and the list of properties. For each type of item there is a corresponding metafact that gives an interpretation for that type, for example: metafact(rule, [' relationship exists between ']). *Describe* determines the interpretation for the type of item it is dealing with by calling up the relevant metafact. Then it uses the interpretation to describe the item. For example, when called to describe the following item: rule(direct, [cement_fineness, rate_hydration]). the actual message written on the screen would be:

This rule implies that direct relationship exists between

```
1 cement_fineness
```

```
2 rate_hydration
```

Describer is a simple predicate used to inform the user of his choices when making design specifications and of the results of the design calculations.

```
describer(Tag,[H1;T1],[H2;T2):-  
    write(Tag),  
    write(' '),  
    write(H1),  
    write(' as '),  
    write(H2),nl,nl,  
    describer(Tag,T1,T2).
```

In use it would be called with *Tag* instantiated to either 'specified' or 'resulting'. The first list would be either the list of things to be specified or the list of things to be calculated and the second list would either be the actual specifications or the results of the calculations.

6.2.4 User Interface

During development of the system it became apparent that the major factor in the ease with which the system could be used by both the designer and other users was the amount of typing the user had to do to respond to system prompts and questions. Another factor was the response of the system to 'incorrect' replies. With this in mind, the system was given the facility to accept any of a number of replies from the user for each desired response. In most cases there is a single letter key for any response that the user wishes to make to the system. The key responses and the other alternatives are stored in a small data base.

The clauses for these 'vocab' facts are of the form:

```
vocab(Meaning,[ List of alternative responses]).
```

```

vocab(calc,[c,calc]).
vocab(design,[d,des,desi]).
vocab(editor,[e,ed,edit,editor]).
vocab(saver,[s,sav,save,saver,q,quit]).
vocab(rule,[r,rul,rule]).
vocab(fact,[f,fact]).
vocab(metafact,[m,met,metaf,metafact]).
vocab(vocab,[v,voc,vocab]).
vocab(inspect,[i,ins,insp,inspect]).
vocab(yes,[y,yes]).
vocab(no,[n,no]).

```

This vocabulary list is useful in determining responses to menus and other prompts as illustrated below. It would also have a use if the system were to require a natural language front end. The vocab clauses could constitute the basic terminal items of a language parser¹⁶. When a response from the user is required, `interpret(Response)` is called. If only one particular response is desired `interpret` can be called with the argument already instantiated to a constant. For example, `interpret(yes)` would read a reply from the user and fail unless its 'meaning' was yes.

```

interpret(Meaning):-
    readword(Reply),
    lookup(Meaning,Reply).

```

Readword was written to replace the standard PROLOG predicate: `read`. This was done because as already explained the effort required from the user in responding to the system is of paramount importance. During the development of the system it was found that even one extra key stroke made a crucial difference to the time it took to reply to prompts, especially when the reply could be as brief as a single letter or number. The standard PROLOG `read` requires the user to delimit his response

¹⁶ Natural language facilities were not implemented at any stage of the project development and will therefore not be described in detail. Clocksin & Mellish, [1984], (chapter 9) and Sterling & Shapiro, [1986], (chapter 16), give simple definite clause grammar implementations in PROLOG.

with a full stop before submitting it with <CR>. There is no reason why, for the brief responses used within this system, the response should not be delimited by <CR> alone. Thus readword was developed to allow this¹⁷.

Once the reply has been read its meaning is determined by lookup:

The 'Meaning' of an integer is that integer (used for selecting an option off a menu):

```
lookup(N,N):-
    integer(N),!.
```

Otherwise the meaning of a response is determined by checking vocab clauses until one is found which contains the response in its list of alternatives:

```
lookup(Meaning,Reply):-
    vocab(Meaning,Tokens),
    member(Reply,Tokens),!.
```

If no vocab clause is found to offer a meaning for the response then an error message is written on the screen and the user is prompted to try again.

```
lookup(Meaning,Reply):-
    write('Your reply: '),
    write(Reply),
    write(' was not understood. '),nl,
    write('Please try again. '),nl,
    interpret(Meaning).
```

Sometimes the reply from the user is in response to a menu supplied by the system. These menus are created and used by *get_rep*. This takes three arguments: a Tag which is a name for what the menu is offering a

¹⁷ Despite its apparent simplicity this readword predicate was very difficult to discover. This is a common difficulty with writing definitions for functions which closely resemble but modify PROLOG system predicates. In particular they are difficult to write because they are just too simple to provide useful trace information to aid debugging.

choice of; a list of options to be displayed on the menu and the third argument which is instantiated by the predicate to the choice made by the user from the menu:

```
get_rep(Tag,Opt_list,Reply):-  
    write('Please choose an option number'),nl,  
    menu(Tag,Opt_list),  
    response(Opt_list,Reply).
```

```
menu(Tag,Options):-  
    write('option  '),  
    write(Tag),nl,  
    writelist(Options,1),nl.
```

Menu itself is fairly self explanatory once an example has been seen:

Please specify str_grade

Please choose an option number

option str_grade

1 strength(15)

2 strength(20)

3 strength(25)

4 strength(30)

6.3 Conclusions from the PROLOG Work

Experimentation with the problems of a mix proportioning system brought to light a number of features. The knowledge involved in this task is best represented in tables as in the original texts describing the mix design method. Thus a rule-based system is not very appropriate. The system could have made use of a database containing the original mix-table knowledge in addition to those items better represented by rules.

The first conclusion of this work was therefore that a critical factor in the application of expert system technology to concrete durability would be the large volume and variety of types of knowledge that the domain involves. The mix design method used was itself extremely limited in scope and had only been adopted as an initial training exercise. As familiarity was gained with the problem it was seen that it did not require powerful inferencing abilities. The specific problem of fixing the initial mix quantities is largely a simple procedural algorithm.

The second conclusion was therefore that the problem of proportioning mix quantities is not a suitable problem for expert systems methods. This was also illustrated by the existence of systems that used much more simple computer methods for mix design such as traditional algorithms, [Day, 1984] and spreadsheets, [Brewer, 1987].

As development progressed, it became obvious that the majority of the work was focussed on developing a shell rather than on the expert system itself. The mix design system program was not originally intended to result in an expert system shell. The goal of developing a PROLOG program that performed mix proportioning necessitated a means of representing the knowledge required more concisely than PROLOG clauses. This pragmatic consideration entailed developing a set of editing functions and an interpreter for the representation.

The development of a scheduler to execute the procedural representations unintentionally transformed the system from a PROLOG program for mix design into a rudimentary expert system shell with a mix design knowledge base. The development of the shell facilities proved to be extremely resource intensive and could potentially have unbalanced the emphasis of the project. In particular it was felt that it

would be necessary to investigate another area of concrete design in order to be able to obtain a more general awareness of the inference and representational needs of the domain. There is little benefit from developing a shell which can only be used in one domain, [Miller, 1985]¹⁸.

Using PROLOG as the implementation language entails duplicating a lot of the features that are an integral part of an expert system shell. Other PROLOG implementations have library modules with facilities such as knowledge representation, inference engines and sophisticated user interface functions¹⁹. These were not available for PROLOG on the VAX mainframes at Aston. The PROLOG phase of the work was therefore concluded in March, 1988.

18 The methods used in this initial PROLOG prototype shell were later developed for use in the implementation of an intelligent front-end for various FORTRAN-based models such as the World Bank HDM-III model and TRRL's Transyt, [Wilkins & Tillotson, 1991]. Instead of having expert-system rules in the knowledge-base this system uses a knowledge-base of template clauses to create, edit and validate a set of input files for the FORTRAN models.

19 Allwood et al, [1987], lists several problems inherent in using PROLOG as an expert system implementation language such as inefficiency and lack of real number arithmetic. Many of these objections can not be applied to PROLOG implementations in general and especially not those which have large toolkit libraries.

Chapter 7: Early Work: VAX - OPS5 and C

OPS5 is an expert system development tool available on the VAX mainframe cluster at Aston University. OPS5 is based on the production-rule paradigm of expert system knowledge representation, [Brownston et al, 1985]. Brownston et al, [*ibid*], provide the definitive reference for the language. A summary of OPS5 and an example application for oil spillage management is given in [Hayes-Roth et al, 1983]. A detailed description of the language will not be given. Several examples of OPS5 rules and consultation sessions will be presented.

OPS5 rules obey the following syntax:

```
(p name-of-rule
  (condition-element)
  (condition-element)
  ...
-->
  (action)
  (action)
  ... )
```

Rules are executed by a recognise-act cycle. The condition elements of each rule are matched against working memory elements (WMEs). If a matching rule is found, the actions on the right-hand side of the rule are carried out. There are various actions, performing tasks such as making or removing WMEs, and general input and output functions. The usual *modus operandi* in OPS5 systems is to use *task* WMEs to factor the knowledge-base into groups of rules concerned with particular goals.

These tasks are not part of the OPS5 language but are defined by the user in the same manner as any other WME:

```
(vector-attribute agenda)
(literalise task mode      ;; task-group
  name                    ;; first task
  agenda)                 ;; rest of tasks in a list
```


OPS5 was chosen for the next phase of the project in light of the conclusions regarding the use of PROLOG to implement expert systems in the domain. OPS5 provides the bare minimum of necessary structure without imposing any unwanted limitations on knowledge representation or inference strategy. It was also available on the VAX and was relatively simple to use¹. It was decided to investigate a new sub-domain that was more central to concrete durability design. The domain chosen was that of sulphate attack. This is less complex than some of the other sub-domains and is treated very concisely by BRE Digest 250, [BRE, 1981].

7.1 Sulphate Attack

Following the protocol implicit in the digest, the sulphate attack system (SO3), had three main tasks in its consultation with the user. BRE digests are written with the intention of providing practical advice. They are therefore relatively easy to interpret in terms of rules and other provisions.

Firstly the OPS5 module determined the likelihood of risk of attack from sulphate-bearing soils and groundwaters. This simply involved asking the user whether certain soils or contaminants were present on the site. The user was assumed to be able to identify these factors but it was possible to implement a help system to provide advice².

¹ ENVISAGE, another shell available on the VAX, was rejected after initial consideration. Previous experience with this shell did not encourage its use in this domain, [Chidley et al, 1986].

² OPS5 does not itself provide any explanation facilities. Help is provided by rules that fire in response to certain user input such as 'help' or '?'. Given the context of the response these rules then present explanatory text to the user.

Consider the following paragraphs from BRE Digest 250, [1981]:



Aston University

Content has been removed for copyright reasons

An example of an OPS5 rule is given that implements the first of these paragraphs:

```
(p risks-soil          ;; establishes risk of sulphate attack from
soil
```

```
    (task ^mode so3 ^name site-risk)
```

```
    (risk ^source nil ^presence n)
```

```
-->
```

```
    (write (crlf) (tabto 10)
```

```
      is the soil-type of the site any of the following (crlf))
```

```
    (write (crlf) (tabto 20)
```

```
      london clay ? (crlf))
```

```
    (write (crlf) (tabto 20)
```

```
      lower lias ? (crlf))
```

```

(write (crlf) (tabto 20)
      oxford clay ? (crlf))
(write (crlf) (tabto 20)
      kimmeridge clay ? (crlf))
(write (crlf) (tabto 20)
      keuper marl ? (crlf))
(write (crlf) (tabto 10)
      answer {(y/n)} (crlf))
(make response ^doing soil ^resp (accept)) )

```

The first condition of this rule matches a working memory element (WME) *task* with the attribute *mode* matching *so3* and *name* attribute matching *site-risk*. The second condition matches a *risk* WME with *nil* as the *source* attribute and *n* as the *presence* attribute. In other words, the rule fires if the task is determining site risk for the sulphate attack module and if the risk has not yet been determined.

The action part of the rule presents a textual prompt to the user and obtains a response which is placed into the working memory by the *make* command. If response to this rule was "n" the system would then fire other rules asking similar questions about the risk from underfloor fill and run-off from colliery sites or marshy country.

If some risk was evident the system proceeded to quantify this risk by classifying the site by the results of tests on the soil for SO₃ content. Finally, this classification was used to specify types of cement and mix quantities of cement and water that would negate the risk of damage from sulphate attack.

This scheme of risk identification and classification and subsequent prevention illustrates Clancey's description of heuristic classification [1985]³.

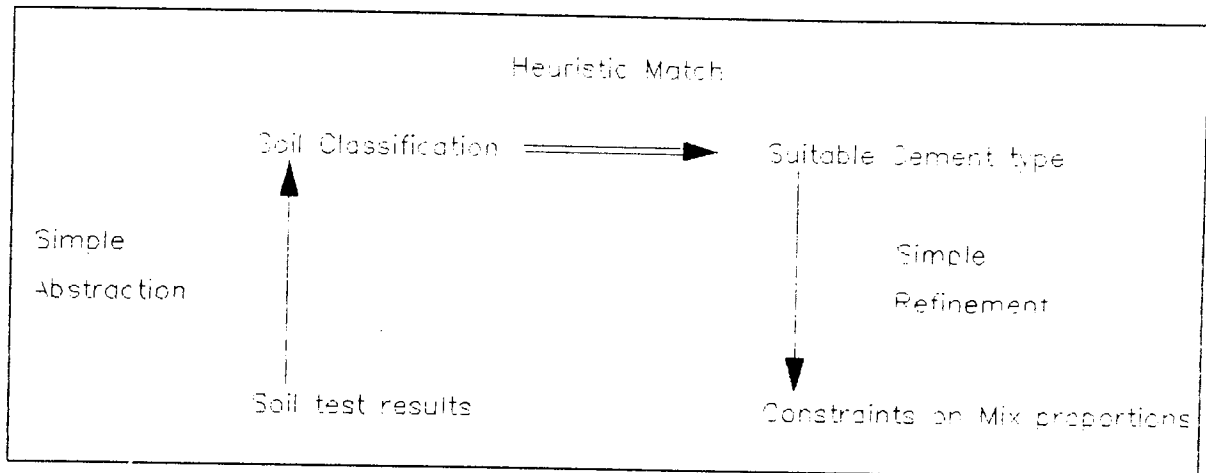


Figure 7.1: Simple and Heuristic Classification in Sulphate Attack Prevention

(After [Clancey, 1985] (p295))

The system could produce a justification of its conclusions. This justification was based on a combination of the canned-text and rule-trace methods. Each rule has attached justificatory text. If justification is required by the user, the *mode* attribute of the task WME is set to *explanation*. The rules are then fired in reverse order to provide a sequence of justifications of each step in the induction. The explanation starts from the current state of the consultation and proceeds backwards through previous stages.

³ See chapter 3 section 4 for a discussion of classification as a generic task in design expert systems.

An example of the output from this explanation technique is given below. In this and the following sample OPS5 prompts are in CAPITALS and the user responses are **lower case**.

IS THE SOIL-TYPE OF THE SITE ANY OF THE FOLLOWING
LONDON CLAY ?
LOWER LIAS ?
OXFORD CLAY ?
KIMMERIDGE CLAY ?
KEUPER MARL ?

ANSWER (Y/N):

?

THIS QUESTION IS BEING ASKED BECAUSE IF THERE ARE ANY OF THE FACTORS MENTIONED AT THE SITE IT WILL BE NECESSARY TO CONDUCT FULL TESTS FOR SULPHATE LEVELS IN THE SOIL

IS THE SOIL-TYPE OF THE SITE ANY OF THE FOLLOWING
LONDON CLAY ?
LOWER LIAS ?
OXFORD CLAY ?
KIMMERIDGE CLAY ?
KEUPER MARL ?

ANSWER (Y/N):

y

AS THERE SEEMS TO BE SOME RISK OF SULPHATE ATTACK, IT IS NECESSARY TO ESTABLISH THE AMOUNT OF SO3 PRESENT. DO YOU HAVE THE RESULTS READY ?

ANSWER Y OR N:

n

PLEASE PERFORM A SO3 TEST AND SUBMIT THE RESULT.
DESIGN SAVED.
SESSION TERMINATED.
THANKYOU

(At this stage the user conducts the tests or otherwise acquires the necessary results.)

DO YOU WISH TO RESTORE AN EXISTING DESIGN ?

ANSWER Y OR N:

y

AS THERE SEEMS TO BE SOME RISK OF SULPHATE ATTACK, IT IS NECESSARY TO ESTABLISH THE AMOUNT OF SO3 PRESENT. DO YOU HAVE THE RESULTS READY ?

ANSWER Y OR N:

y

WHAT WAS THE LEVEL OF SO3 FOUND ?
(EXPRESSED AS A PERCENTAGE): **0.6**

WHAT WAS THE SOURCE OF THIS FIGURE ?
RESPOND EITHER: 'S' FOR SOIL SAMPLE
 'W' FOR GROUNDWATER SAMPLE
 'M1' FOR 1:1 WATER:SOIL EXTRACT
 OR 'M2' FOR 2:1 WATER:SOIL EXTRACT

s

SITE CLASS IS 3

PLEASE CHOOSE ONE OF THE FOLLOWING CEMENT TYPES

CEMENT:	RESPOND:
OPC WITH 70-90% GGBFS	PCSLAG
RHPC WITH 70-90% GGBFS	RHSLAG
OPC WITH 25-40% PFA	PCASH
RHPC WITH 25-40% PFA	RHASH
SULPHATE-RESISTING	SRPC

PCASH

WHAT IS THE NOMINAL MAXIMUM SIZE AGGREGATE?
ANSWER 10, 20 OR 40: ?

IT IS NECESSARY TO KNOW THE AGGREGATE SIZE BECAUSE THIS AFFECTS THE CONSTRAINTS ON CEMENT QUANTITY ETC ENTAILED BY THE SULPHATE RISK.

WHAT IS THE NOMINAL MAXIMUM SIZE AGGREGATE? ANSWER 10, 20 OR 40: **20**

DOES THE JOB ENTAIL ANY OF THE FOLLOWING:

SECTIONS WITH ONE FACE EXPOSED TO THE
SULPHATE-BEARING WATER WITH ANOTHER SUBJECT TO DRYING
CONDITIONS ?

THIN SECTIONS EXPOSED TO THE SULPHATE-BEARING WATER
?

ANSWER (Y/N): **n**

THE RESULTS OF THIS CONSULTATION ARE AS FOLLOWS:

THE SITE WAS CLASSIFIED AS: CLASS 3

THE CEMENT CHOSEN WAS: PCASH

THE MINIMUM CEMENT CONTENT IS: 380 KG PER M3

THE MAXIMUM W/C RATIO IS: 0.45

NOTE ALSO THAT IT IS IMPORTANT TO AVOID SHARP EDGES AND
ROUGH SURFACES EXPOSED TO SULPHATE-BEARING WATER

DO YOU WISH TO SAVE THE STATE OF THE DESIGN ANSWER 'S'
QUIT ANSWER 'QUIT'

SEE THE CHAIN OF INFERENCE EXPLAINED ANSWER
'EXP'

OR CONTINUE ANSWER 'C' ? :

exp

THE EXPLANATION WILL PROCEED ONE RULE AT A TIME
STARTING FROM THE MOST RECENT. AFTER EACH RULE TYPE
'EXP' TO CONTINUE THE EXPLANATION. ANY OTHER RESPONSE
WILL HALT THE EXPLANATION.

RULE: CONSTRAINTS IF SITE IS CLASS 3 AND CEMENT TYPE IS
PCASH THEN MINIMUM CEMENT CONTENT IS 380 AND MAXIMUM
W/C RATIO IS 0.45

exp

FACT: CEMENT IS PCASH

exp

RULE: CLASS SITE IF SO3 LEVEL IS 0.6 % FROM A SOIL TEST
THEN SITE IS CLASS 3

exp

FACT: SOIL TEST

exp

FACT: SO3-LEVEL 0.6

exp

FACT: SOIL IS A KNOWN SULPHATE RISK THEREFORE TEST FOR
SO3 LEVEL NECESSARY

exp

EXPLANATION COMPLETE.

As can be seen from this sample session log, the OPS5 system has a number of disadvantages. The input of user responses and the display of results is crude in contrast to more modern shell systems many of which emphasise sophistication in this area. Although the explanation facilities were relatively easy to implement, they too are crude and inflexible.

This prototype sulphate attack module was capable of generating multiple solutions for a particular case. These multiple solutions were based on the different choices of cement that could be made for a given soil-classification. For the narrow scope of this particular problem this did not present any difficulties from the point of view of *combinatorial explosion*.

Another experiment with the SO3 system divided the code into further sub-modules each of which was concerned with a sub-problem of the prevention of sulphate attack. For example one module was solely concerned with identification of initial risk; another was concerned with

risk classification and so on. There was little practical difference between code-size or performance for the single system and the modular system.

The main benefit of sub-modularisation would be in the creation of a diagnostic system. The relevant sub-modules (risk indication and classification) could become part of a diagnostic system (through modular compilation). Only the knowledge concerning relevant cement restrictions and proportion constraints would then need to be rewritten specifically for a diagnosis system⁴.

The SOS prototype system based on the BRE digest was relatively simple to implement. The system illustrated the utility of using the digest as a knowledge source. It also suggested that modules based on similar publications would provide a suitable modular decomposition of the overall goal of durability design. Despite the poor quality of the user interface, initial feedback from the domain experts was encouraging. It was therefore decided to extend the experiment with OPS5 into a less trivial sub-domain.

7.2 Alkali Aggregate Reaction

The problem of alkali aggregate reaction (AAR) is significantly more complex than that of sulphate attack. There are three separate factors which determine the risk - the alkali content of the concrete; the type of constituents in the aggregates; and the likely internal humidity of the concrete. The knowledge source used for the AAR module, [Concrete

⁴ As mentioned below the implementation of the design knowledge as tables rather than production rules could also facilitate the use of the knowledge in diagnosis.

Society, 1987], outlines critical values or thresholds for each of these factors but there are many ways of specifying the critical values and of seeking to ensure compliance with them.

For instance, one recommendation is that the alkali content of the cementitious materials does not exceed 0.6% Na₂O equivalent - in some cases this merely involves specifying that the cement itself has a manufacturer's certified average reactive alkali content of not more than 0.6%. This limit can also be met by combining Portland Cement with other material, either ground-granulated blastfurnace slag (ggbfs) or pulverised fuel ash (pfa), so that the total reactive alkali content of the combined cementitious materials has a value of less than 0.6%⁵.

In other cases (as when sea-dredged aggregates are used), the other materials used in the concrete also contribute significant alkalis (that is, greater than 0.2kg per m³) and so the 0.6% limit becomes inapplicable. In this case a 3.0kg per m³ limit should be used unless the aggregate combination can be considered unreactive.

In addition, there is disagreement among the experts as to the net effect of cement replacement materials on the alkali content of the mix⁶. Unlike some expert system domains it is not desirable to use any computational or inferential method of dealing with this disagreement and uncertainty. Most inferential methods that utilize uncertainty factors do so as a means of handling uncertainty of data⁷. In this domain it is the

⁵ Note that the consideration of sulphate attack resistance also often entails the use of pfa-or ggbfs-portland cement combinations to a specific degree of replacement that has to be ensured in order to prevent sulphate attack.

⁶ Refer to chapter 5 for an illustration of the range of opinion pertaining to the effect of pfa cement replacement on AAR.

⁷ See chapter 2 for a discussion of uncertain inference techniques.

knowledge that is equivocal. There are methods for dealing with contradictory knowledge items, [Reboh, 1983]. However none of these are useful in the domain of design of structures.

Choosing between alternative views on the effect of pfa on alkali content is very different from choosing between different interpretations of geological data, [*ibid*]. In the domain of concrete durability design it is necessary to take the more conservative of two otherwise equally preceded alternative views. This is advisable in order to safeguard the legal liability of the engineer in the event of deterioration of the concrete.

The specification of mix parameters with respect to negating the risk of AAR is therefore not as simple a procedure as for sulphate attack. The system has to cycle through a number of measures suggested by the user; assessing each for its effectiveness until one of the critical values is complied with. However it is possible to provide a simple analysis of the alkali content of the mix as it is at any stage and make recommendations as to the most practical method of reducing it below the relevant threshold.

THIS IS A SIMPLE PRODUCTION SYSTEM TO GIVE ADVICE ON PRECAUTIONS NECESSARY FOR THE PROTECTION OF CONCRETE FROM ALKALI SILICATE REACTION THE RULES IT USES ARE TAKEN FROM THE 'REPORT OF A WORKING PARTY' CHAIRED BY M.R. HAWKINS PUBLISHED IN DEC 1986

AT ANY TIME A RESPONSE MAY ENTAIL THAT THE SYSTEM CONSIDERS THAT THERE IS NO FURTHER RISK FROM AAR

IN MOST CASES A RESPONSE OF 'X' WILL TEMPORARILY HALT THE SYSTEM,

'H' WILL INVOKE THE HELP FACILITY

'?' WILL TELL YOU WHY A QUESTION IS ASKED

WILL THE SECTION HAVE, AFTER CURING, AN INTERNAL RELATIVE HUMIDITY GREATER THAN 75 % ? ANSWER Y N

n

DOES THE SECTION EXHIBIT ANY OF THE FOLLOWING FEATURES:
CLADDING OR EXTERNAL MEMBERS WHERE CONDENSATION CAN OCCUR ?

EXPOSED CONCRETE FRAMES/PANELS ?
INTERNAL MEMBERS SUBJECT TO CONDENSATION OR HIGH
HUMIDITY ?

ANSWER Y OR N

y

DOES THE SECTION EXHIBIT ANY OF THE FOLLOWING 'HIGH RISK'
FEATURES:

- FOUNDATIONS (EVEN WATERPROOFED) ?
- WATER-RETAINING STRUCTURES ?
- MEMBERS PROTECTED UNDER BRIDGES ?
- BRIDGE COLUMNS, BEAMS OR PARAPETS ?
- OTHER HIGHWAY STRUCTURES ?
- MULTI-STOREY CAR PARKS ?

ANSWER Y OR N

y

PLEASE NOTE THAT IT IS HIGHLY RECOMMENDED THAT, WHERE
THERE IS RISK OF AAR, AGGREGATES CONTAINING OPALINE
SILICA SHOULD NOT BE USED

IS THE AGGREGATE TO BE USED CONSIDERED TO ENTAIL ANY
RISK OF AAR ?

ANSWER Y N

y

DOES THE AMOUNT OF FLINT OR CHERT IN THE AGGREGATES
AMOUNT TO MORE THAN 60 % BY MASS ? ANSWER Y OR N OR ?

n

WHAT IS THE TARGET MEAN CEMENT CONTENT OF THE CONCRETE
IN Kg/m³

330

WHAT IS THE MANUFACTURER'S CERTIFIED MEAN ALKALI
CONTENT IN THE CEMENT EXPRESSED AS NA₂O EQUIVALENT % ?

0.65

IS THERE ANY CEMENT REPLACEMENT INTENDED ?

ANSWER 'N' FOR NONE

'G' FOR GGBFS

OR 'P' FOR PFA

n

DOES THE FINE AGGREGATE CONTRIBUTE, IN THE FORM OF
CHLORIDE IONS, TO THE ALKALI CONTENT ? ANSWER Y OR N

n

DOES THE COARSE AGGREGATE CONTRIBUTE, IN THE FORM OF
CHLORIDE IONS, TO THE ALKALI CONTENT ? ANSWER Y OR N

n

DO ANY (OTHER) ADMIXTURES CONTRIBUTE ALKALIS TO THE MIX
?

ANSWER Y OR N

n

SINCE THERE IS RISK OF AAR IT IS NECESSARY TO CONSIDER
THE FOLLOWING ALTERNATIVE OPTIONS:

USE NON-REACTIVE AGGREGATE. OPTION: NRA

USE CEMENT WITH LESS ALKALI. OPTION: LAC

EXCLUDE MOISTURE FROM THE SECTION. OPTION: DRY

INCREASE CEMENT REPLACEMENT LEVEL. OPTION: POZ

OPTION: **poz**

REPLACE CEMENT WITH PFA OR GGBFS

IS THERE ANY CEMENT REPLACEMENT INTENDED ?

ANSWER 'N' FOR NONE

'G' FOR GGBFS

OR 'P' FOR PFA

p

WHAT IS THE TARGET MEAN CONTENT OF THE CEMENT REPLACEMENT IN THE CONCRETE. ANSWER IN KG/M3

75

WHAT IS THE MANUFACTURER'S CERTIFIED MEAN ALKALI CONTENT IN THE REPLACEMENT EXPRESSED AS NA₂O EQUIVALENT % ?

0.9

SINCE THERE IS RISK OF AAR IT IS NECESSARY TO CONSIDER THE FOLLOWING ALTERNATIVE OPTIONS:

USE NON-REACTIVE AGGREGATE. OPTION: NRA

USE CEMENT WITH LESS ALKALI. OPTION: LAC

EXCLUDE MOISTURE FROM THE SECTION. OPTION: DRY

OPTION:**lac**

IS IT POSSIBLE TO USE A CEMENT WITH ALKALI CONTENT LESS THAN 0.65 ANSWER Y OR N

y

WHAT IS THE MANUFACTURER'S CERTIFIED MEAN ALKALI CONTENT IN THE CEMENT EXPRESSED AS NA₂O EQUIVALENT % ?

0.45

WITH ALKALI CONTENT OF LESS THAN 6% AND INSIGNIFICANT OTHER ALKALIS THERE IS NO FURTHER RISK OF AAR

DO YOU WISH TO SAVE THE STATE OF THE DESIGN ANSWER 'S' QUIT ANSWER 'QUIT'

SEE THE CHAIN OF INFERENCE EXPLAINED ANSWER 'EXP' OR CONTINUE ANSWER 'C' ?

q

AAR FINISHED

Both the SO₃ and AAR systems follow the same general schema. This schema can be summarised as successive consideration of the following factors: risk indicators; risk classification; risk prevention measures. This schema is illustrated by the flow chart in figure 7.2. In some cases such as alkali aggregate reaction prevention it is necessary to iterate the cycle as each attempt to prevent the risk may not be completely successful. It is expected that other durability systems based on BRE digests (such as the digest on steel reinforcement corrosion [BRE, 1982a]) will also follow this regime.

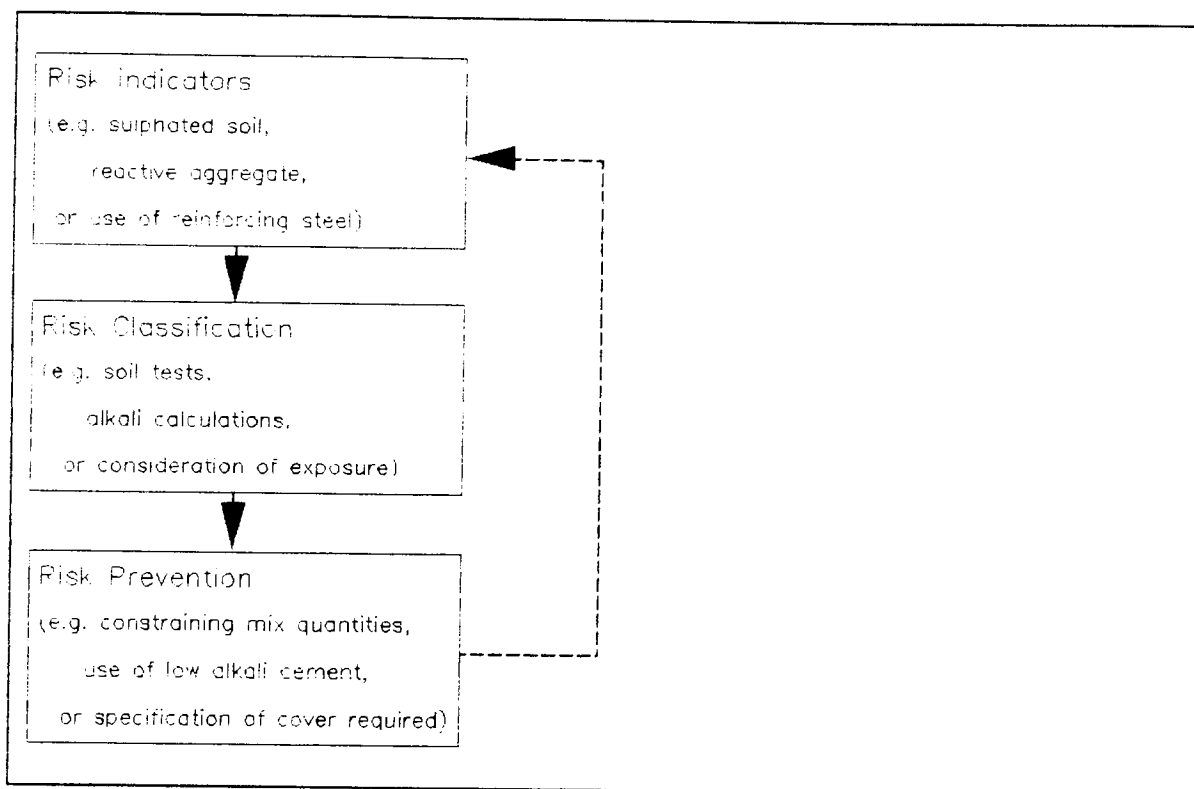


Figure 7.2: General Control Strategy for Concrete Durability Design

As with the SO₃ module, the AAR system was well received by the domain experts. It was felt, however, that the problems encountered with the state of the knowledge in this sub-domain illustrated the desirability of having some deep fundamental knowledge available to the system.

Recent research results in the domain could be used to provide this deep knowledge. Candidates for implementation included the reactivity of various aggregates [Dolar-Mantuani, 1983]; the prediction of hydroxyl ion concentration in the pore solution, [Taylor, 1987] and the effect of hydroxyl concentration on AAR risk [Nixon et al, 1986, Nixon & Page, 1987 and Canham, 1987].

7.3 Hydroxyl Ion Concentration

It was decided to proceed with an attempt to implement the Taylor model of hydroxyl ion concentration prediction, [*ibid*]. The intention was to use this hydroxyl ion concentration prediction in conjunction with knowledge relating hydroxyl levels to AAR risk to more accurately specify the AAR risk. This endeavour raised some representation difficulties in that OPS5 lacks the mathematical functionality required by the model.

The model does not, in itself, present any computational difficulties. However the interpretation of the paper outlining the model was extremely difficult and much experimentation was required before the program implementing the model was able to reproduce the four examples given⁸. It is not necessary to reproduce the program listing but an outline of the algorithm derived from the original paper is given in appendix A.

Having developed a program in C that could predict hydroxyl ion concentrations to the required accuracy, additional work was required to make this prediction available and useful to the AAR expert system module. This work was not felt to be central to the concerns of the project. A major problem encountered at this stage was the difficulty of directly interfacing C and OPS5⁹. Further development of the Taylor model implementation is discussed in chapter 10.

⁸ This is not meant to imply any deficiencies in the model or the paper *qua* technical research paper. There is however a difference between the mathematical rigour required for a research paper in chemistry and the rigour required for implementation of the paper as a computationally correct algorithm.

⁹ Alternative methods such as an indirect interface via disk-files was not conceived at this stage of the project.

7.4 Conclusions from the OPS5/C Work

The quality of user-interface provided by OPS5 is detrimental to the usability of the system through a consultation of any length. It was therefore necessary to investigate a shell which was more directly orientated to user-interface considerations.

The combination of a primary OPS5 module with sub-routines written in C is not itself an easy task and is highly impractical since it greatly reduces the portability and maintainability of the system. Thus a prime consideration in the evaluation of languages and shells for use in the domain is the ability to support a modularised approach to the problem and a high degree of mathematical functionality. Alternatively, the shell must provide a useful interface to other languages that can provide mathematical functionality.

Other issues brought out by the OPS5 phase of the project include the disadvantages of representing some knowledge as production rules, which is the only representation directly available to OPS5. The obvious alternative to production rules is decision tables (Dts) which in many cases would better reflect the original, tabular, presentation of the design knowledge in the source text¹⁰. These DTs could be stored in a data-base which would reduce the bulk of the relatively inefficient production rule system.

Tabular representation has the advantage of readability and thus maintenance of the system code. Tabular representation might also allow the modules to perform tasks other than design. The systems developed

¹⁰ See, for example, table 1 on p3 of Digest 250 on sulphate attack [BRE, 1981].

in OPS5 are dedicated to just one main goal: constraining a mix design with respect to an environmental loading. There are several other possible uses of the knowledge, held in the digests, for example in diagnosis of failed structures.

It would therefore, be highly desirable to have the same knowledge sources usable in different modes of consultation without duplication of knowledge in different representations. Tabular representation of the design knowledge would, perhaps, make this easier because a table does not have the explicit interpretation of a 'direction' of use as in 'IF...THEN' with production rules.

Another issue raised is the general nature of the knowledge in the domain. The codes of practice which have provided most of the knowledge incorporated so far are good sources of directly applicable design knowledge. However this knowledge is shallow in the knowledge engineering sense. All that is generally presented in the Standards are prescriptive rules for durable concrete. This is acceptable and even necessary in the standards where extensive explanation and justification of the prescriptions would detract from the ease with which they could be used.

In contrast an expert system usually is required to be able to provide sufficient explanation and support to allow confidence in its conclusions. In the domain of mix design the simple outlining of the chain of inference which most expert systems provide is not adequate to ensure this confidence. Secondly, the advice given in the standards, while extensive in its scope, inevitably can not cover every single case where durability advice is necessary.

The knowledge in the domain is subject to constant revision and growth. As new materials, techniques and environmental situations constantly arise the knowledge used by an expert system should reflect these changes. It is therefore desirable to incorporate extensions to the knowledge presented in the standards, wherever possible, to allow the system to provide in-depth justification and explanation; to reason about cases outside the scope of the standards and to find methods of incorporating recent research before it is universally accepted or even complete.

These extensions, using current research in the domain, can effectively increase the search space of feasible designs and give greater depth to the inference, explanation and justification facilities of the system. Experts in the domain are able to give advice on 'difficult' and marginal areas so they must have some knowledge that is not encoded in the Standards.

Apart from the usual problems of distilling expert knowledge there is an additional difficulty in obtaining this knowledge for an expert system in the domain of durability. As mentioned in chapter 2, legal liability makes the provenance of knowledge embodied in design expert systems crucial.

The OPS5 phase of the project has demonstrated the utility and feasibility of using BRE digests and other similar codes of practice as knowledge sources for design-orientated expert systems. A structure and theory of concrete mix design with respect to durability has been developed based on the decomposition of the mix design task into sub-tasks based on the specific problems covered by BRE digests and similar publications.

The advantages and disadvantages of OPS5 as an expert system development tool are listed below:

- * OPS5 is very easy to learn and use for reasonably large knowledge-bases
- * It provides a flexible development capability through the use of modular compilation
- * Some expert system features that are provided as standard in PC shells are not present. The most notable deficiencies include the lack of sophisticated user interface functions and facilities for explanation and justification
- * The language has insufficient mathematical ability for engineering applications
- * Interfaces to mathematical languages such as C are difficult to implement and would reduce the portability and maintainability of the combined package
- * OPS5 is based on production rules which are not appropriate for all the knowledge representation needs of a design system

Chapter 8: Relevance and Evaluation of Expert System Shells

Expert system shells offer a number of obvious advantages over lower-level languages. The deficiencies of the user interface facilities implemented in PROLOG and OPS5 have been illustrated in the previous chapters. Both OPS5 and PROLOG also require significant development effort to create the necessary tools required for expert systems implementation such as explanation and justification methods and knowledge-base editors. The need to program expert systems tool-kits for a language before satisfactory applications can be developed reduces the immediate utility of that language.

O'Neill & Morris, [1989], reveal that 65% of expert systems developers in the United Kingdom preferred to use micro-based shells and languages. Rada, [1990], also reports that most expert systems deliveries are of PC shell-based systems. In contrast the sales revenue is greater from non-PC language-based systems.

Because of the nature of the micro-computer software market, languages are more likely to have available expert system tool-kits. Shells are more likely to have source-code licenses¹. Many small shells are available in the public domain or as *shareware*. The existence of language tool-kits makes PC AI languages comparable to low cost shells. For example Turbo-Prolog is provided with predefined predicates and tool-kits that make expert system shell creation much easier than is the case with VAX-PROLOG, [Wilkins & Tillotson, 1991].

¹ Jackson, [1986], provides an example where the shell EXPERT was used in order to gain rapid development benefits. Availability of the EXPERT source code resulted in modification of all but 20% of the shell program.

As mentioned in chapter 6, there are also some possible disadvantages in using shells as an implementation methodology. The main potential problem is that a shell might limit the representational and inference strategies available, [Allwood et al, 1987]. Initial experience in the domain indicated that production rules (possibly in combination with a database representation for tables) and simple goal-driven inference were appropriate for a concrete design system. This entails that a wide choice of shell systems are potentially applicable in the domain.

This chapter will discuss the general relevance of shells for use in the broad domain of engineering design. The evaluation and choice of shells will be elaborated in section 8.2 and The CRYSTAL shell will be described in section 8.3. This section will also summarise the selection criteria and highlight those that are satisfied by CRYSTAL.

8.1 Speciality Shells

For any given expert systems project, if time and resources permit, it may be beneficial to create an expert system shell from scratch. This may be necessary if none of the commercially available shells are found suitable. Shell development is usually the result of a development programme for a specific application using a low-level language². Alternatively shells may arise out of top-down considerations of a wide domain. A number of such academic specialist shells have arisen out of design system research. Rychener, [1988], comments that the

² Most of the early shells had this origin, for example EMYCIN and the various shells based on the HEARSAY and PROSPECTOR paradigms.

specification and development of specialist design shells is still an *open research topic*; of necessity since design itself is still a research subject, [*ibid*] (p9).

An early prototype specialist shell is the General Engineering Problem Solving Environment (GEPSE), [Chehayeb et al, 1985]. One of the foundations of GEPSE is the recognition that there are three primary engineering tasks: analysis, design and diagnosis. Chehayeb et al maintain that each of the three tasks utilise a *common body of* knowledge. The tasks differ only in the order in which they are applied. Thus decomposition of the tasks at the correct level would allow the creation of a common set of generic tasks and knowledge³.

Knowledge in engineering is categorised as either *static* or *active*, [*ibid*]. Static knowledge consists of a number of elements: objects, descriptions (attributes or parameters), links (relating objects) and functions (procedural attachments). Active knowledge includes both algorithmic procedures and heuristic rules. GEPSE consists of representation methods for the static and active knowledge types and additional features such as user interface and meta-level control. GEPSE can therefore, be seen as a variety of frame- or object-oriented system with additional facilities for forward- and backward-chaining rules.

There have been a number of shells specifically targeted at design applications⁴. The BUILD shell followed a bottom-up development starting from the realisation of the importance of building codes in design and

³ Although it is true that design shares common generic tasks with analysis and diagnosis, it has one feature - synthesis, which is unique. As synthesis is the essential feature of design, any system that ignores it may have limited design ability.

⁴ Each of the remaining shells: BUILD, DOMINIC, and DSPL, has been mentioned in chapter 4.

the difficulties inherent in using them, [Rosenman & Gero, 1985, and Rosenman et al, 1986a, 1986b, 1986c, 1989]. Early versions of the BUILD shell facilitated the use of codes for requirement finding and therefore were used for creating design assistants rather than design systems *per se*, [Rosenman & Gero, 1985].

Later developments added the facility for compliance checking and the use of knowledge from experts in addition to code provisions. Systems built under the later version of BUILD could also use interfaces to external CAD and analysis packages. The ability for the user to select from alternatives completes the functionality required of a simple design system⁵.

DOMINIC, [Howe et al, 1986], is based on the model of design as redesign, [Dixon & Simmons, 1983, and Dixon et al, 1984]. The interesting feature of DOMINIC is its use of a blackboard architecture that anticipates the DESTINY integrated design model, [Sriram, 1986, 1987a, and 1987b]. DOMINIC therefore facilitates design that extends across more than a single narrow concern. DOMINIC, however lacks the explicit hierarchical representation of the structure of domain tasks and objects proposed by the DESTINY model.

DSPL (Design Specialist and Plans Language) is a language based on the *generic tasks* idea developed by Brown & Chandrasekaran, [1986]. The other primary feature of DSPL is the facilities for failure handling. Failure at any level in the task hierarchy is resolved by the ability of

⁵ The SO3 and AAR systems described in the previous chapter followed a similar development path. From an initial intention to capture the compliance requirements embodied in a code it can be a simple step to create a design system by the addition of facilities to select from viable alternatives (either by user choice, expert heuristics or optimisation).

each specialist to utilise an alternative plan for the execution of lower level design agents or to fail itself and thus pass the failure back to a higher level which may in turn execute a different plan or fail itself, [Brown, 1985].

As a language DSPL lacks many of the features desirable in a full shell. However a number of research programmes have been instigated in order to provide tools for the language. Examples include explanation and knowledge acquisition facilities, [Chiang & Brown, 1987].

8.2 Evaluation of Shells

During the duration of the project described in this thesis, facilities were available within the systems group of the civil engineering department for the evaluation of a number of PC-based shells. The Science and Engineering Research Council made available an IBM PC-AT and several shells: KES, SAVOIR and XiPlus⁶. Additionally the department had access to a number of shells such as, Sage and Envisage, Super-Expert and TIMM⁷.

6 Each shell was reviewed by a different member of the systems group. The collective report of the results of this evaluation has been submitted to SERC but has not been published. In light of the following discussion of shell evaluation it would not be useful to present the results in this thesis.

7 The department has been sufficiently encouraged by the results of initial expert systems projects to purchase a more advanced integrated expert system development tool - Leonardo.

A number of different projects within the group had made significant use of these shells and so there was significant experience of shell use, see, for example, [Chidley et al, 1986]. The eventual decision to use CRYSTAL was made after a further evaluation of this shell on the recommendation of an experienced shell user⁸.

There have been several published reviews evaluating various shells. The criteria used in these evaluations and in the shell reviews within the civil engineering department at Aston University are described in the following section. Appendix B offers a list of shells and references where they have been described or evaluated.

There are a number of problems with attempts to evaluate shells objectively. Firstly it is rare that any organization can make a wide selection of shells available for comparison. Commercial shells range in price from hundreds to tens of thousands of pounds. Although shell suppliers make demonstration versions available, these perforce emphasise finished applications in trivial domains. Thus the collection of sufficient legal copies of shells for evaluation of development facilities can be prohibitively expensive.

Another important factor is the amount of human effort required to test a shell. Allwood et al, [1987] review 9 shells. As a rough estimate the evaluation took the 3 reviewers 12 months. This implies that a period of approximately 4 months per shell per reviewer is required. Similarly, Vedder, [1989] reports a review of 5 shells. Each 2 man evaluation team covered a single shell and the programme took 14 weeks. This results in the approximate figure of 3¹/₂ months per shell per evaluation.

⁸ Dr. Alan Harget, the examiner for the first year project report leading to this thesis.

It is important that the reviewers are not experienced in using the shell and that a single test case is used for all the shells. Furthermore each reviewer must only conduct a single test. A final condition is that each of the reviewers should be equal in terms of experience and skill as knowledge engineers and have equal expertise (or lack of) in the target domain. If any of these criteria are violated then the evaluation is potentially devalued due to bias, unequal test conditions or the benefits that always arise from a second implementation of a single test case.

In common with most problems of software selection the choice of shell is, in theory, determined by both objective and subjective factors. Previous experience is probably the most important factor in shell selection. There are no firm conclusions to be drawn from the comparisons listed in appendix B other than to reinforce the essential subjectivity of the choice. However, if the prerequisites for evaluation can be achieved, it is possible to make some objective statements about the criteria that can be used to guide the choice of shell. The following sections will describe these criteria then provide an illustration of the

The most obvious evaluation criterion is that a shell must support the knowledge representation and inference strategies required by the domain, [Palmer & Mar, 1988, and Vedder, 1989]. Thus for non-trivial domains it may be the case that the best shells are those that use a number of representations, [Sloman, 1984]⁹. Additionally the shell should support the mathematical functionality required or interface to appropriate languages, [Palmer & Mar, 1988].

⁹ Hickman, [1986], cites the need for a variety of knowledge representations as support for the use of object-oriented languages rather than specific AI languages or shells. Jackson, [1986], notes that integrated tool kits such as ART provide access to a plurality of formalisms.

Another prominent criterion is the suitability of the hardware required and the development environment provided, [Palmer & Mar, 1988, and Vedder, 1989]. For example, the environment should provide version control, trace, session-save and -logging facilities, [Born & John, 1986]. Another useful feature is the ability to use graphic tree representations of rule-trees or frame topographies, [Nguyen et al, 1985].

The syntax of the representation should be easy to read so that non-specialists, such as the domain experts, can read and understand the knowledge-base, [Born & John, 1986]. Another useful feature in this context is an editor that structures the knowledge-base, [*ibid*]. A structured representation and presentation allows the developer to move around the knowledge-base in a manner that reflects the actual structure of the rule-tree. This would allow, for example, movement directly from a parent rule to a child rather than the rule immediately below or above the current one which is all that is possible in a 'flat' knowledge-base.

All of these features contribute to a rapid development cycle. The speed with which an expert system can be used to develop a knowledge-base is vital. The power of expert systems lies in the extent of the knowledge-base. Therefore, all else being equal, the shell that facilitates more rapid development generates more competent expert systems, see for example, [Roesner, 1988, and Vowler, 1989]¹⁰.

¹⁰ The proviso for 'all other things being equal' can not be overstated. It is not the case that any given problem domain can be turned into a successful expert system by the creation of a massive knowledge-base. Given that successful knowledge structures, representations and inferencing methods can be found; a larger knowledge-base will generally improve and broaden competence.

The shell should provide a user interface that is easy to use, [Palmer & Mar, 1988, and Vedder, 1989]. It should therefore insulate the user from the command or programming level, [Born & John, 1986, and Muir, 1987]. Additionally there should be facilities such as several different types of help, and *what if* capability, [Born & John, 1986, and Savory, 1986]. The user interface should also emphasise ergonomic factors such as the minimisation of screen clutter and keyboard use and the ability to use graphics for input and output, [Muir, 1987].

The shell should have the ability to exchange data with external software, [Born & John, 1986, Palmer & Mar, 1988, and Vedder, 1989]. This is particularly important if the expert system application is to share data with existing software. Potential users of an expert system are much more likely to be appreciative of its power if they can see that it makes use of dBase3 data bases or LOTUS123 templates that they are possibly familiar with, [Powell, 1989]. In many cases the application will have data-processing needs that are more applicable to other software paradigms and so will benefit from interfaces to such software even if the system is not to be embedded in existing systems.

Finally, selection should be guided by factors such as the training required to use the shell, [*ibid*]. Other considerations include the quality of documentation provided and any *hidden* costs such as run-time licenses and add-ons such as induction engines or interfaces, [Vedder, 1989]¹¹. All of the above criteria need to be weighed against the price of

¹¹ On the other hand, Born & John, [1986], note the attractions of a shell with separate development and delivery executives. The delivery system does not need many of the facilities required by the development system. The delivery system may therefore be smaller, more efficient and cheaper (but may be an additional hidden cost).

the candidate shells and hardware requirements. This consideration is complicated by the need to calculate the potential value of the target applications and the long term value of possible future developments.

With the benefit of the experience gained from the PROLOG and OPS5 work it was possible to confirm that most of the above criteria are relevant for the shell selection in the domain of concrete durability design. In addition it was possible to identify the particular knowledge representation and inference needs.

Consideration of the knowledge source texts showed that often the prevention of a possible risk could be reduced to a simple set of procedures. In general, design requires goal-directed inference. Additionally, design usually requires some cycling through design and redesign and some complex interaction between sub-modules¹². Design does not usually require use of degrees of certainty or other probabilistic methods.

8.3 CRYSTAL

CRYSTAL4 is the latest version of a shell produced by Intelligent Environments¹³. CRYSTAL is a low price (c£350 with academic discount), PC-based shell emphasising simple goal-directed inferencing, rule representation, a sophisticated user interface and mathematical functions. In addition, CRYSTAL has a very easy learning curve despite the indifferent introductory documentation.

¹² See chapters 3 and 4 for discussions of computational models of design.

¹³ Intelligent Environments Ltd., Northumberland House, 15-19 Petersham Road, Richmond, Surrey, TW10 6TP

CRYSTAL is based on a development system that includes a dedicated knowledge-base editor. It is possible to use any other editor to create a knowledge-base and to then compile the result. The built-in editor, has the advantage of imposing the CRYSTAL syntax and rule-base structure directly on the user.

The format of CRYSTAL rules is:

```
                <NAME OF RULE>

IF <condition>
    AND <condition>
    AND ...

OR <condition>
    AND <condition>
    AND ...

...
```

Thus the syntax has a primarily declarative interpretation: the rule is true if one of the alternative condition sets is true.

The top level of every CRYSTAL rule-base is the CRYSTAL MASTER RULE. This is the rule that CRYSTAL seeks to prove when the rule-base is run. Each condition may itself be a rule. To create a new rule, the developer types its name as a condition in a higher level rule. The new rule may then be expanded into supporting conditions. Each rule can be placed as a condition in another rule by accessing a pop-up dictionary

containing all the current rule-names. If a rule is not expanded it may still be used by the run-time system. CRYSTAL asks a default yes/no question requesting the user for the truth-value of the rule.

CRYSTAL has a large number of functions and commands. Commands execute functions and are used as rule conditions. Each command and function is selected from a dictionary. Functions can test or assign values to a variable. Assignments and tests can utilise a wide variety of numeric, string, file and time/date functions. Commands also allow the system to interact with the user through a variety of types of form (yes/no, input, output, help, print, menu, graphics displays). These commands and functions can be seen as *side-effects* of the declarative rule-base that give it a strong procedural bias¹⁴.

A final feature of CRYSTAL that enhances development of large and complex rule-base is the ability to display and navigate through a graphic representation of the rule-tree¹⁵. This sample rule-tree for the SO3 module is reproduced in figure 8.1. This figure has been enhance to clarify the illustration. In normal use only a portion of the tree is displayed at one time and the developer moves to the level of interest for greater detail. Each rule is only expanded once to aid clarity.

¹⁴ It is possible to use CRYSTAL as a straight-forward procedural language with traditional control mechanisms such as for- and while-loops. The only thing the shell lacks in this regard is the ability to use rules as procedures and functions with direct parameter passing. It is, however, possible to use the system's working memory of global variables as an alternative to parameters. Georgeff & Bonollo, [1983], stress the importance of a methodology that allows:

formal algorithmic knowledge to be uniformly integrated with heuristic declarative knowledge. (p151)

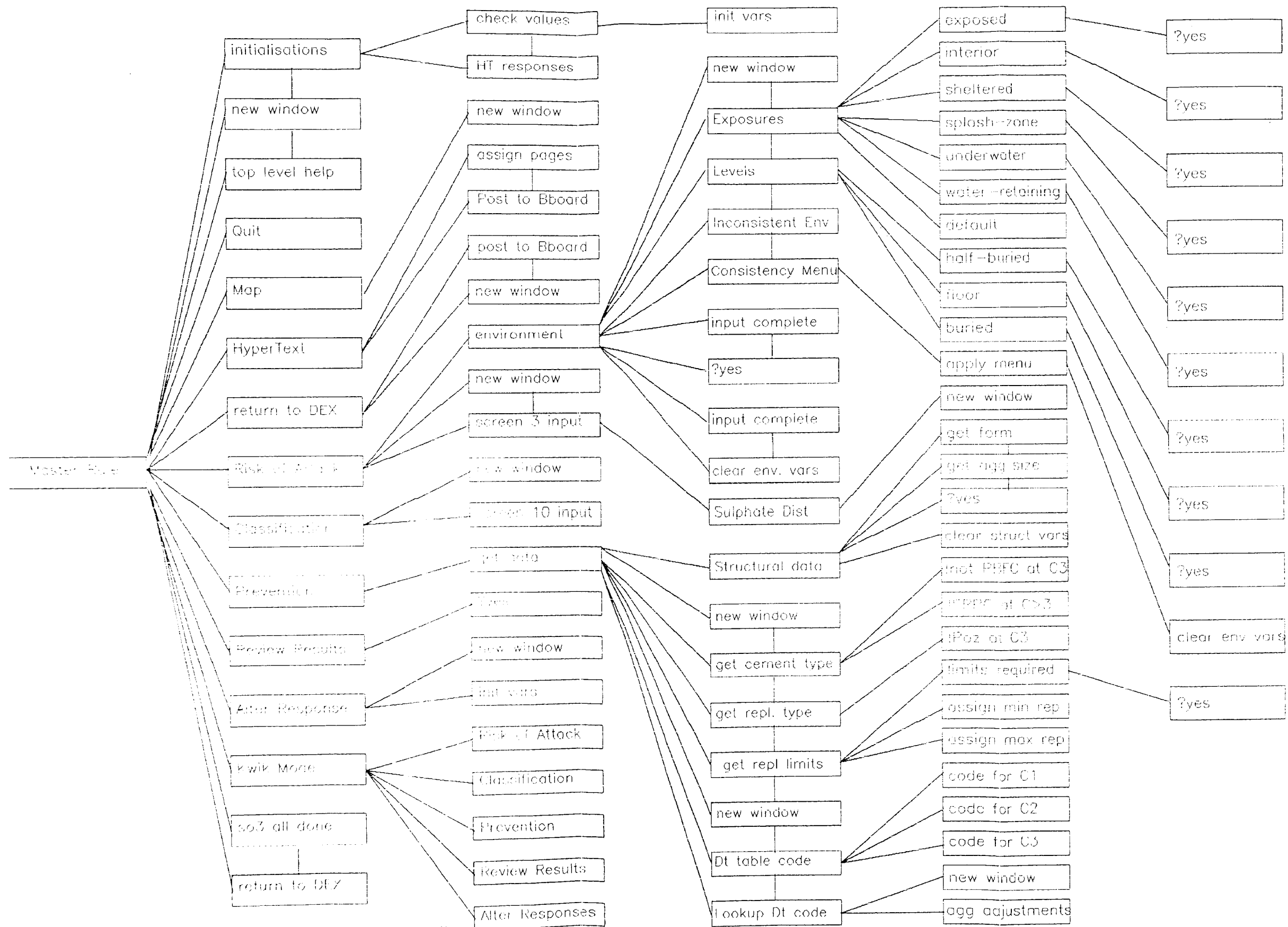
¹⁵ The screen and printer drivers required must be purchased and installed separately.

Additionally the system only displays rules and not CRYSTAL commands. Each horizontal line joins parent-rules to children. A vertical line indicates rules linked by the AND operator.

The rule-base editor itself explicitly checks for and eliminates circular rules. The context dependent movement through the rule-tree structure instils awareness of the structure of the knowledge-base in the developer. In combination with the rule-tree display, this helps eliminate most of the other potential logical problems of a rule-base such as inconsistency, incompleteness and redundancy¹⁶, [Nguyen et al, 1985, and Saouma et al, 1989].

¹⁶ CRYSTAL maintains a list of unused rules that have been expanded into conditions but which have been *pruned* from the rule-tree.

Figure 8.1: Sample CRYSTAL Rule-Tree Display



The criteria for selection of shells for design systems are summarised in the following lists. The criteria that are met by CRYSTAL are marked with an asterix.

General features required in design systems:

- * Suitability of inference engine with respect to design
- * Ability to implement a design/redesign cycle
- * Ability to easily integrate modules to reflect problem decomposition
- * High degree of mathematical functionality
- * Ability to interface to other packages and languages

Development:

- * Environment that automates the edit/compilation/run cycle
- * Editor tailored to the syntax of the knowledge representation
 - Automatic version control
- * Trace facilities at varying depth of detail
 - On-line help for shell commands and functions
- * Graphic representation of the rule-tree or frame-topography
- * Context dependent movement through knowledge-base file

User Interface:

Provision of facilities for a majority of the six *w-words* especially -

- * Why (is this question being asked)?
- * How (can I answer it)?
- * Explain (the meaning of this question)!
- * Justify (those conclusions)!¹⁷
 - What IF (I had answered otherwise)?
- * Windows to allow formatted and overlaid questions and answers
- * Menus and Yes/No questions to aid and constrain answers

¹⁷ All of the responses to these questions should preferably be tailored to each particular user's level of expertise.

- * Data-input screens to allow question shortcuts
- * Colour and graphics to highlight important input and output
- * Minimal screen clutter and keyboard use

Overall speed of operation

Costs:

Initial cost of shell versus long term value of application(s)

Hardware requirements

Training costs

Hidden costs such as 'essential' add-ons (induction engines, interfaces)

Run-time licenses

8.4 Conclusions

The selection of a shell rather than a low level language and the consequent choice of a particular shell is a significant decision point in the development of an expert system application, [Rowlinson, 1987].

Unless there is an existing source of experience with a number of shells this choice will probably be determined by purely financial considerations or by accident. The financial resources required to conduct a comprehensive survey of commercial shells are rarely available. The number of available shells and the rate of development of up-grades and new products will limit the period of use of any survey.

Even if the human and financial resources are available, use of a shell for evaluation purposes is not sufficient to develop a true understanding of its potential and limitations. It is possible to find significant new methods of using the features of a shell even after

several years of experience developing a number of different applications with it. All the above considerations limit the utility of the reviews of expert system shells published in the literature.

If objective and comprehensive evaluation is possible, a number of criteria can be specified. Within the limited scope of this project it was possible to experience several shells. CRYSTAL was found to embody a significant number of the criteria thought to be desirable for shells in the domain of concrete design. CRYSTAL was therefore chosen for further development. The CRYSTAL version of the concrete durability modules will be described in the following chapter.

Chapter 9: The DEX Concrete Durability Expert System

A broad literature search, combined with initial experience in implementing expert systems for various concrete durability design problems, suggested a number of salient features of the domain. These features entailed a number of characteristics required for an expert system in the domain.

The most important of these features and requirements are simple, goal-directed, deterministic inferencing; a combination of production rules, decision tables and possibly frames; access to mathematical functions and external programs; modular decomposition of the design process and a sophisticated user interface. In addition there is also a critical requirement for a development environment that facilitates rapid prototyping.

In the light of the above considerations the CRYSTAL shell was chosen as the main tool for the further investigation of the use of expert system techniques in the domain of concrete durability design. CRYSTAL has been briefly described in the previous chapter.

9.1 Translation of Existing Work Into CRYSTAL

With the benefit of the experience gained with PROLOG and OPS5 it was possible to produce improved versions of the sulphate and AAR modules very quickly - 1 man-day for the prototype sulphate module (as opposed to a week in OPS5) and 1 man-week for the AAR module (2 months for OPS5)¹. The AAR module was slightly modified in order to incorporate the

¹ Koskela et al, 1988 report a similar speed-up in development time for a translation of a system written in ES/P translated to Insight 2 (2 months initial development time and 2 weeks translation time).

recommendations of BRE digest 330, [BRE, 1988]. This made the AAR module consistent with the intention of the project to embody the knowledge contained in BRE publications.

It was also possible to implement a CRYSTAL version of the hydroxyl ion concentration model without using embedded C procedures. All three modules were capable of producing context-dependent help, explanation; justification and hard-copies of results. The next phase of the project concentrated on the exploration of various expert system methods within the CRYSTAL implementation and the development of a prototype system for integrated concrete design - DEX.

An example of the first few conditions of the CRYSTAL master rule from the SO3 rule-base is given below in figure 9.1. The fragment shown tests a rule called *initialisations* with the side effect of instantiating variables and importing results from the blackboard file². The next rule *#new window* is a *tool-box* rule (common to all rule-bases in the system) that sets up a standard window configuration for the system. The *top level help* creates a help window that may be popped-up by using the **F1** key.

The help screen details the default help for the module that outlines the users options (including how to get context dependent help). The next line contains a menu command. This displays a screen of menu fields; prompts the user and obtains a response. In this case the response is the string variable *r\$* which represents the user's choice of top-level

² The blackboard memory is necessarily a file because of the inability of CRYSTAL to directly access RAM. However this is not to imply that the file needs to be stored on disk. It is possible to configure PCs with sufficient RAM so that part of RAM is reserved for a *virtual* disk drive. This entails that access to the blackboard file can be almost as efficient as direct use of RAM.

actions. The *quit* rule exits the rule base if the user's choice is to quit. If the user's choice is other than quit the first disjunction of the master rule fails and CRYSTAL attempts to prove the second.

The second disjunction tests for a response indicating that the user wishes to see a map. The map shows the user's current location in the network of modules. Each of these may be called in an arbitrary order or not at all. The map is therefore of great potential comfort to the user. A general view of this map is reproduced as figure 9.10.

If *map* succeeds, the *restart* command returns the user to the top level menu³. If *map* fails, the system attempts the next disjunction (to call the relevant *hypertext* resource), and so on.

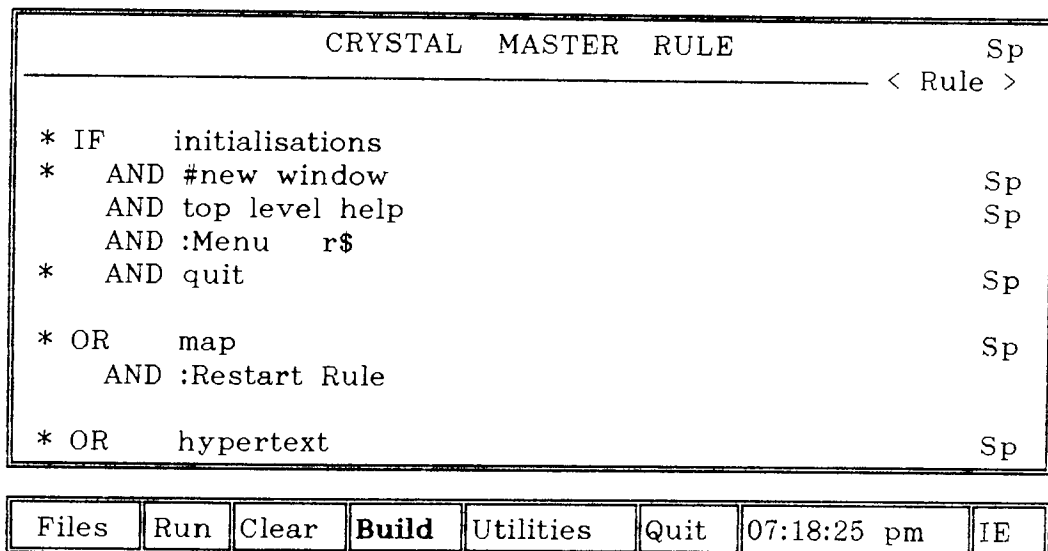


Figure 9.1: Sample CRYSTAL development system screen

³ In this and other CRYSTAL rule displays the following conventions should be noted. A colon : is used to denote a CRYSTAL system command. A bullet * indicates a child rule. The # sign before some rules is a convention that ensures that often used or *generic* rules occur first in the rule dictionary. The *Sp* markers indicate that a rule can be called more than once. Without the *Sp*, designation the *initialisations* rule, for example, will only be called the first time the master rule fires.

The run-time menu presented by this CRYSTAL master rule is illustrated in the following figure. This figure also shows the other options presented by the system. These are described in more detail in section 9.2, below.

```
Design example _____ DEX 05.1
Options:
  Continue with Durability Design
  Hypertext system
  Review results
  Alter input
  File operations
  Map of system
  Pick a module to consider
  Output to printer

  Log errors/bugs encountered
  Quit

      status:
Structural Data input:
Sulphates:
Steel:
Mix Design:
Trial Mixes:
AAR:
```

Figure 9.2: Initial Screen of the DEX system

Although CRYSTAL rule bases are tedious to describe they are very easy to develop and debug. The provision of automatic dictionaries for rules, variable and functions profoundly decreases the amount of typing the developer has to perform and consequently reduces mistakes⁴. These,

⁴ Allwood et al, [1987], cites the facility for such variable dictionaries as being essential for the development of large knowledge-bases.

along with the other features of the editor such as context dependent movement through the rule-base and automation of the edit-compilation-execution cycle, allow for very fast prototyping.

9.2 Evolution of Enhanced Features

9.2.1 User Interface

The initial enhancement offered by the translation between OPS5 and CRYSTAL was the implementation of an improved user interface. The CRYSTAL output system is constructed around discrete screens rather than continuously scrolling text. Input is based on a variety of screen-forms and input fields. The forms can present menu, yes-no or ordinary input fields. Forms with input or menu fields provide a much friendlier interface than a system based purely on commands or sequences of questions, [Saouma et al, 1989].

Another advantage of menus is that the list of acceptable responses is effectively an item of concise and explicit meta-knowledge concerning the limits of competence of the system. The addition of windowing facilities allows the developer a great deal of flexibility in screen design. A single background screen can have windows superimposed on it in positions and order that can be determined at run time. Figure 9.3 illustrates the use of windows on a schematic diagram. This diagram summarises the possible placings of concrete with respect to various exposure conditions.

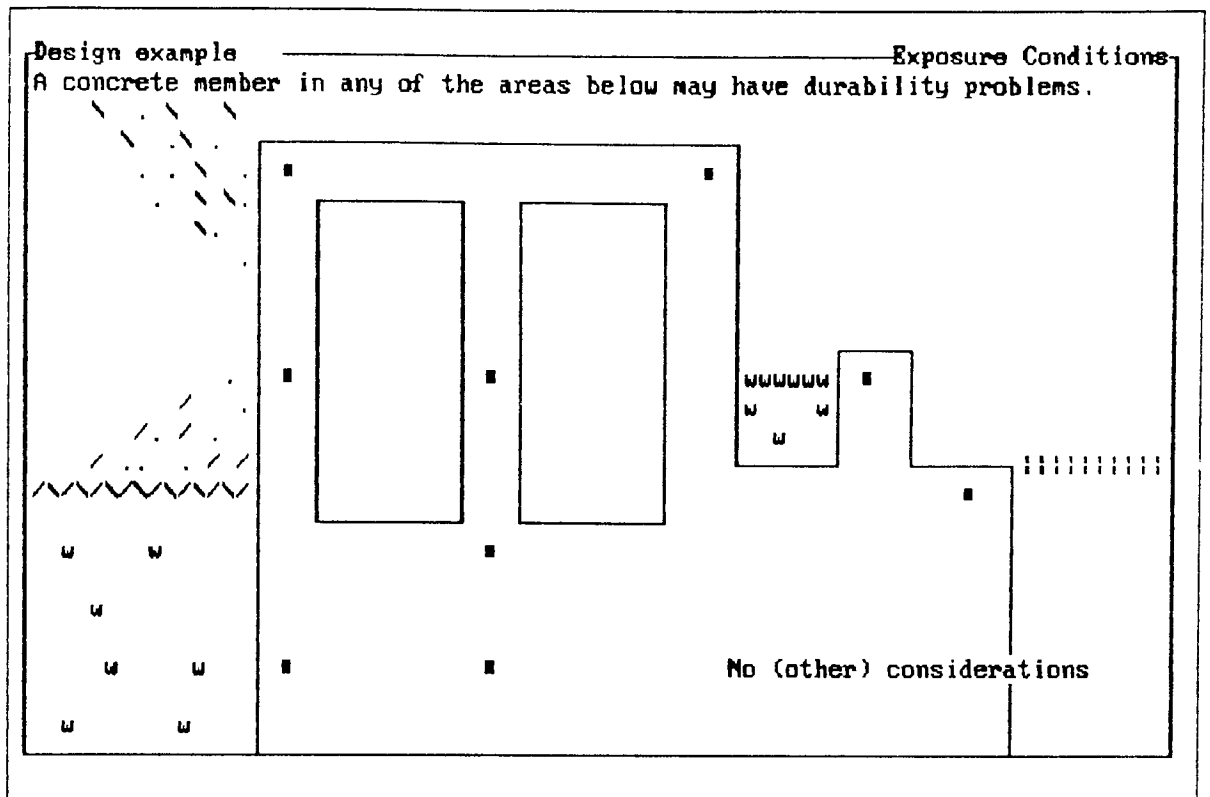
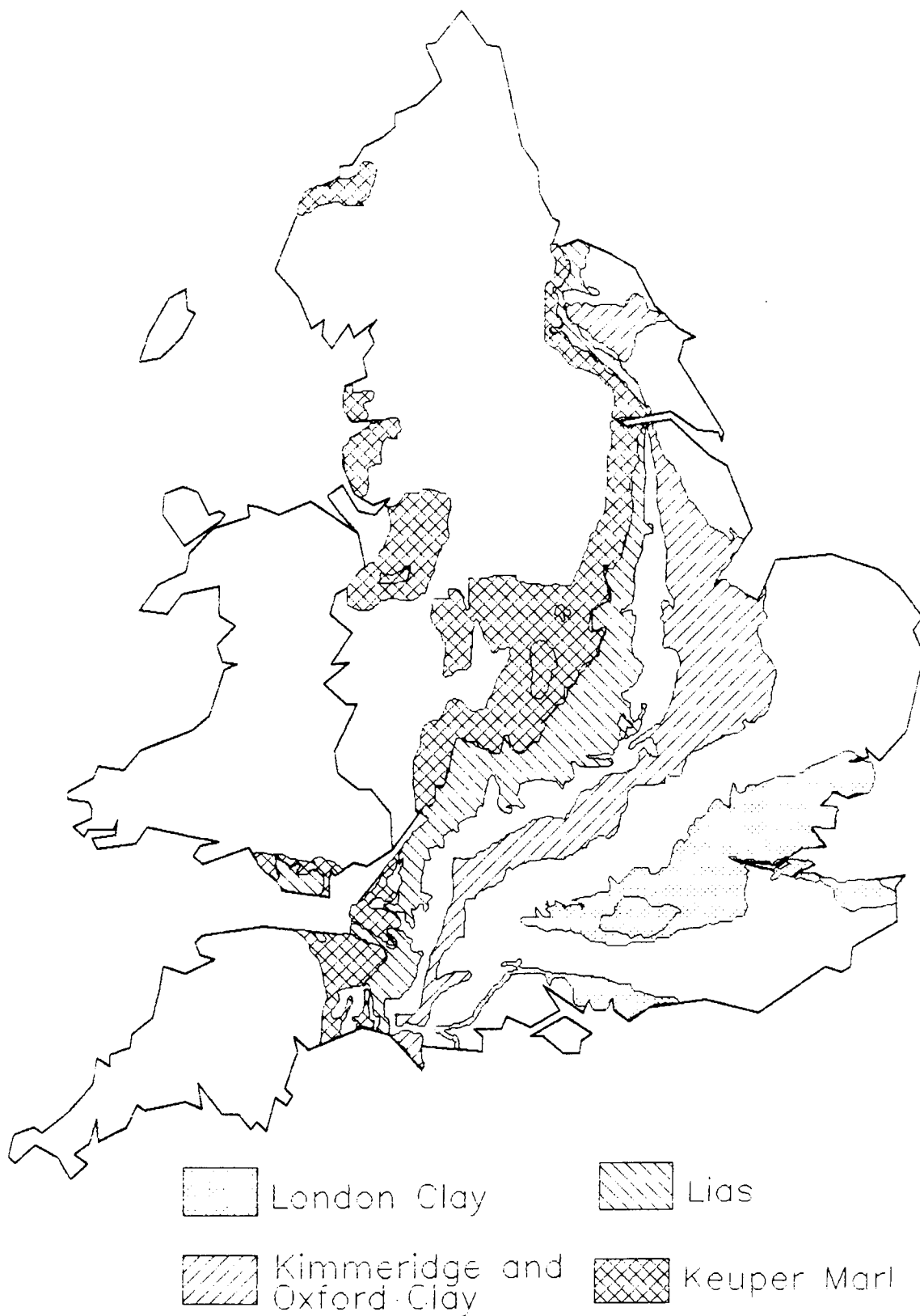


Figure 9.3: Exposure Conditions for Concrete

Windows can also be super-imposed onto graphic displays. This allows use of graphics for active input. Figure 9.4 shows a map of the distribution of sulphate-bearing soils in England and Wales. DEX is able to display this map to the user and have the user indicate the proximity of the construction site to sulphate-bearing soil deposits.

Figure 9.4: Sulphate Distribution in England and Wales



It is also possible to use analogue indicators, such as sliding scales, for relative and imprecise input such as certainty factors. Figure 9.5 shows the use of a sliding scale to indicate the required adjustments to mix quantities in response to trial mix results. In this diagram the figures "175" and "205" represent the extremes of the allowable range and "190" the currently selected value that changes as the bar is altered.

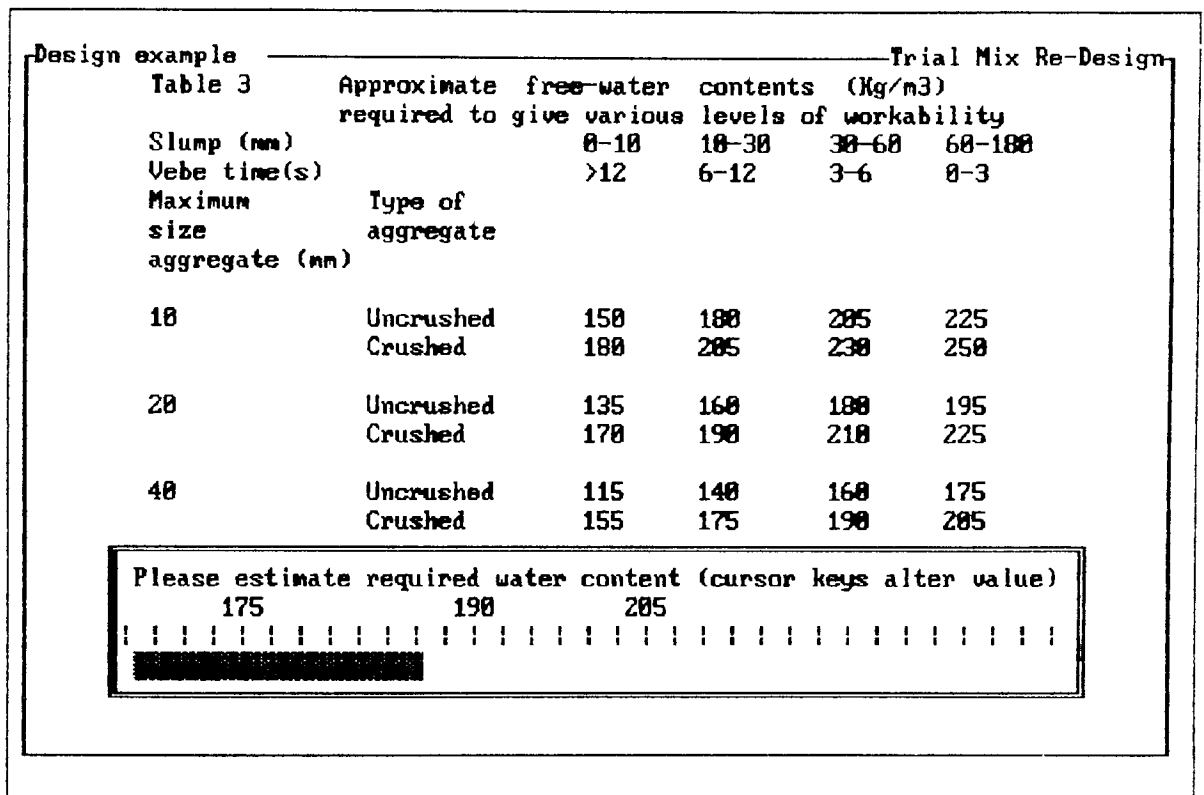


Figure 9.5: Mix Proportion Adjustments from Trial Mix Results

9.2.2 Justification and Explanation

Once the translation of the OPS5 modules had been completed to the extent of their previous competence, the SO3 module was used as a test-bed for various user-support systems.

CRYSTAL supplies a trace-like justification system that allows the user to move backwards (and forwards) through the rule-tree from the current position. The use of this method allows the user to answer for himself questions such as "WHY is this being asked?". The rule-trace method depends greatly on the use of significant and *meaningful* rule and variable names. Also, the user is expected to be able to use CRYSTAL commands to navigate the rule-tree⁵.

CRYSTAL also provides the facility for help-screens and help-file display so that each question or conclusion that the system puts to the user can be supported by text screens either detailing how to answer or what the significance of a conclusion is. This is a slightly more sophisticated version of the canned-text support that was provided for the PROLOG and OPS5 modules. Various researchers note the inadequacy of the rule-trace and canned-text explanation facilities in general, [Neches et al, 1985, Dieng et al, 1987].

There are various methods of enhancing explanatory abilities such as filtered rule-trace or natural language enhanced rule-trace. However, the CRYSTAL development system does not provide access to source-code. This limits the ability of the developer to build or enhance low level tools such as the trace facility. The first attempt to provide a better user-support system therefore by-passed the CRYSTAL system trace facility. The CRYSTAL help system was still used to provide *flat* support in the form of a screen of relevant text.

⁵ The CRYSTAL animation of the rule-tree answers the conceptual dependency aspect of user queries but leaves the user to fathom the details of the particular question category of interest (that is WHY, WHAT, HOW, WHEN, WHERE, WHO etc, the so-called w-words), [Savory, 1986, Hughes, 1987, and Gilbert, 1988].

The rules in the knowledge-base were modified so that each rule utilised in the inference chain created a simple report of its significance. Each rule-report was appended to a file. A request for justification of a conclusion from the user caused the report file to be displayed. On conclusion of each module the report was appended to the overall report file for that design consultation. The user was thus able to call for justificatory explanations at two levels of detail: 'local' and 'global'. A sample fragment of a local justification report is shown:

Risk Result: Site class 2

User Data-input: concrete with 40mm aggregate
 OPC cement with pfa replacement between 25 and
 40 %

Therefore: maximum water/cement ratio is 0.55
 minimum cement content is 310 Kg per m³
 adjusted for 40mm aggregate size

Therefore: maximum water:cement ratio is 0.63
 minimum cement content is 270 Kg per m³

This method proved adequate at the local level but could only reflect the state of the consultation as a linear sequence. Thus the result of the determination of the minimum cement content would be remote from other considerations concerning the constraint such as minimising alkali levels in the AAR module. Other problems arose when it became necessary to be able to backtrack through previous module

consultations. The use of a static text file to summarise the inference chain proved too inflexible to permit reconsideration of earlier responses and conclusions⁶.

This use of inference reports still failed to address the wider issues of user support for questions other than simple justification. The structured approach to user-support was later abandoned in favour of a hypertext system that provided an easy means for the user to explore the conceptual linkages of the current theme for the consultation from any relevant viewpoint. The DEX hypertext system and the implications of hybrid hypertext/expert systems is described in detail in chapter 11.

Hypertext support provides a number of valuable features for an expert system. For a complex domain such as concrete durability, hypertext can be used to provide user support for all the various w-word questions. This support is possible without provision of distinct program code for each type of question or consideration of other factors such as user-expertise as is necessary using established expert system techniques.

Additionally it was found that hypertext could be used as an alternative and possibly superior interface to the expert system itself. Special *input* buttons on the relevant pages of the source document constitute an acceptable user-input alternative to expert system generated questions or forms.

Finally, the creation of hypertext documents makes demands on the developer that are similar to those of knowledge elicitation from textual

⁶ These problems could have been addressed by keeping each module report as a separate file and using hypertext browsing abilities to explore the conceptual dependency relationships.

sources. In the case of BRE digests the creation of hypertext pages and links is easier than the extraction of rules and therefore can serve as a useful introductory exercise for the knowledge engineer.

9.2.3 Alternative Representations

One result of the growing size of the DEX system was a need to explore alternative representations for some of the design knowledge. Much of the knowledge embodied in the digests is tabular⁷. The inefficiency of rules for representing tabular knowledge was discussed in chapter 4. Various methods of tabular representation were tried using the various interface facilities provided by CRYSTAL. These interfaces include direct use of files such as structure ASCII files and files produced by dBase3 and LOTUS123. Each interface used adds greatly to the RAM required by the CRYSTAL environment.

The memory requirements of the CRYSTAL interfaces was extravagant in comparison to the modest amount of knowledge that required tabular representation. The final solution was to use CRYSTAL rules to determine the value of an index to a simple array of values. These arrays are created by a *once-only* CRYSTAL process and then stored in a permanent write-protected export file⁸. Various other code-based systems such as SPIKE, [Rasdorf & Wang, 1987, and 1988] and COMIX, [Miller, 1985], have emphasised use of tables wherever possible to replace rules.

⁷ See, for example, table 1 on p3 of Digest 250 on sulphate attack [BRE, 1981].

⁸ Because of the heterogeneous nature of the tables in digest 250 and the mixture of numeric and textual criteria it is not possible to directly calculate the values held in the table other than by the use of an individual rule for each entry.

9.2.4 Provision of Flexibility for the User

As with the OPS5 systems, the CRYSTAL sulphate attack and AAR modules were initially split into 3 sub-modules. The first sub-module was called by the user (or blackboard) to check for the possibility of risk. If necessary, the second sub-module quantified this risk. Finally the third sub-module produced a set (or sets) of constraints on the mix design. It was later seen to be more efficient to combine the sub-modules so that the SO3 and AAR systems each comprise a single module.

Within each area of durability consideration the design is organised using the original OPS5 regime (as illustrated in figure 7.2 in a previous chapter). At various points during a consultation with a specific durability module, the user is presented with the initial menu shown in figure 9.6. This offers several options in addition to continuing with the design task. The user is able to review results from the module, alter previous responses and to enter the hypertext sub-system.

In order to provide a comparison with the OPS5 user interface a sequence of screens from a consultation with the CRYSTAL version of the SO3 system are given. As explained above, figure 9.6 shows the *top-level* control options presented by the module.

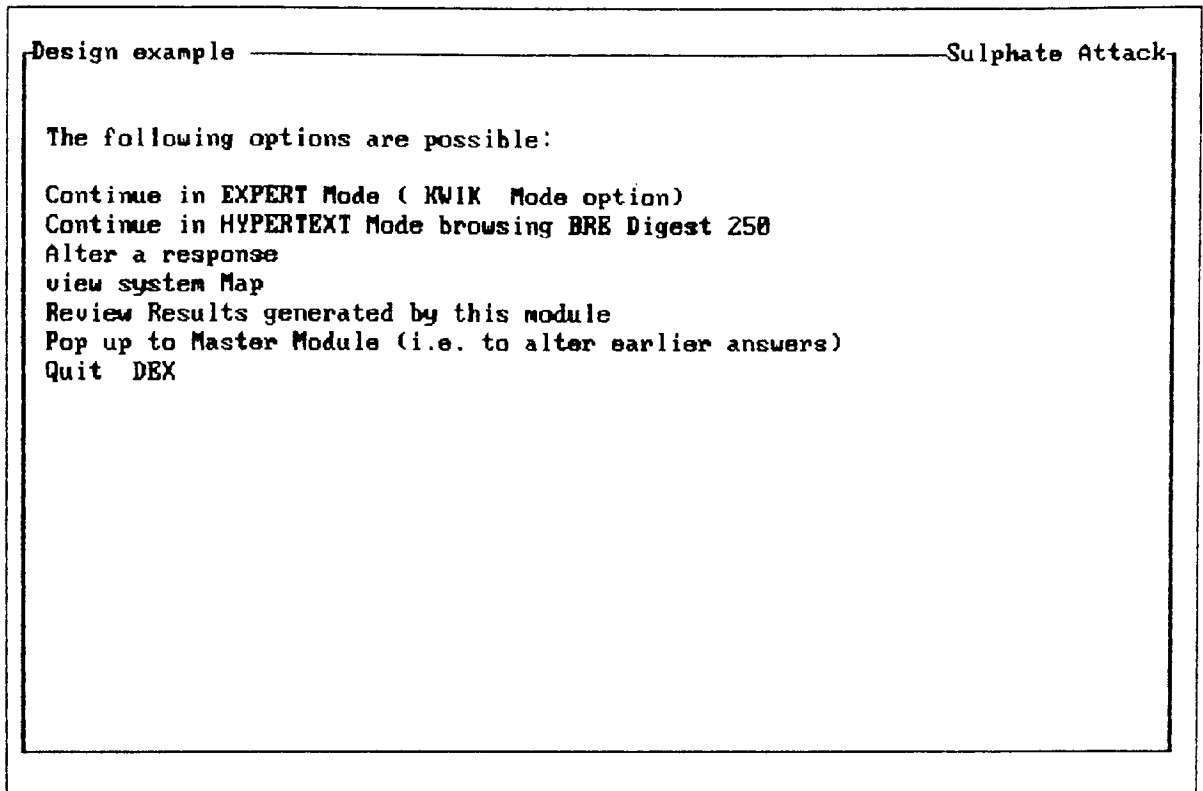


Figure 9.6 Initial screen of the SO3 module

The normal sequence of tasks for the system is executed by the Expert Mode. In this mode the user is returned to the top level after each stage. This allows the user, for example, to review or alter intermediate results. In contrast, the Kwik Mode accesses the stages in an uninterrupted sequence. This makes the consultation faster for a more experienced user but also reduces the opportunities to change responses and utilise the hypertext system.

Figure 9.7, illustrates the potential indicators of sulphate attack risk. The user simply uses this menu to select one or none of the options. If the user is unsure it is possible to provide further help, in this case in the form of the sulphate distribution map shown above in figure 9.4.

Design example	Sulphate Attack
In order to determine if there is any risk of sulphate attack the following information is required: ?	
Is the soil any of the following?	
London Clay	
Oxford Clay	
Kimmeridge Clay	
Lower Lias	
Keuper Marl	
Is the site proximate to either a colliery tip or Marshy country ?	
Are any of the following used as under-floor fill?	
Colliery Shale	
brick-rubble	
ashes or	
industrial/mine waste	
No Risk	

Figure 9.7: Sulphate Risk Indicators

If a particular module requires an input result, it will ask the user for a value. If, however, the particular variable has already been instantiated, the module will check the value for validity (with respect to an appropriate range) and conformity with any relevant constraints. Thus the modules may be used for *passive* (code compliance checking) as well as *active* design (design decision making), [Kalay et al, 1987].

Figure 9.8 shows the input screen for the results of a soil test. This screen closely reflects the actual layout of the left-hand side of table 1 of [BRE, 1981]. This formatted input screen is a better alternative to the original use of a number of questions to obtain the test type and test results.

Design example		Sulphate Attack			
Select appropriate test result to determine site classification risk from Kimmeridge Clay					
Class	Total SO ₃ (%)	SO ₃ in 2:1 water:soil extract g/l	SO ₃ in 1:1 water:soil extract g/l	In ground- water g/l	
1	< 0.2	< 1.0	< 1.0	< 0.3	
2	0.2-0.5	1.0-1.9	1.0-2.5	0.3-1.2	
3	0.5-1.0	1.9-3.1	2.5-5.0	1.2-2.5	
4	1.0-2.0	3.1-5.6	5.0-10.0	2.5-5.0	
5	over 2.0	over 5.6	over 10	over 5	Continue

Figure 9.8: Soil Test Results Input Form

Finally, figure 9.9 illustrates the results of the SO₃ consultation. The user is presented with a form detailing his responses, earlier information from other modules and the resulting constraints on the mix constituents and proportions.

Design example	Sulphate Attack
The following information is required for the prevention of sulphate attack risk:	
Cement type: OPC	
Cement Replacement Material: pfa 25% - 40%	
Max nominal aggregate size: 40mm	
These values entail the following constraints on the mix design at class 3:	
Minimum cement content: 340Kg/cu m	<input type="checkbox"/> Yes
Maximum W/C ratio: 0.50	<input type="checkbox"/> No
<p>You have entered these as relevant factors in the sulphate attack prevention for the concrete. Is this correct?</p>	

Figure 9.9: SO3 Results Screen

9.2.5 Representation and Use of Constraints

Once the degree of risk, from sulphate attack for example, has been classified, appropriate measures can be taken to prevent that risk. This consists of setting various constraints on the mix parameters as illustrated above in figure 9.9. The CRYSTAL SO3 module was used to explore various methods of representing these constraints. It was initially thought advisable to use a general method of constraint representation and handling.

Each constraint produced was given a unique number (in sequence) this number was an index to a number of arrays representing the various

aspects of the constraint. Initial attributes that were thought to be potentially useful included the name of the parameter being constrained, the value being imposed, the type of constraint (either minimum, maximum or identity) and the source module. Example constraints from consideration of sulphate attack are given below.

Constraint:	Parameter:	Value:	Type:	Source:
1	cement\$	OPC	ident	so3
2	agg	40mm	ident	so3
3	cement	270kg/m ³	min	so3
4	w/c ratio	0.6	max	so3
5	rep\$	pfa	ident	so3
6	rep	40%	min	so3
7	rep	25%	max	so3

As each area of potential risk generated a set of constraints, this set was compared with the total set previously generated by other risk prevention modules. Conflicting constraints were either reconciled logically or by heuristic strategies. For example, if there were two constraints both setting a maximum water/cement ratio, from structural-strength and sulphate attack prevention considerations, the smaller maximum value becomes the operative constraint.

If this logical reconciliation is not possible it becomes necessary to find an alternative strategy such as re-running one of the two conflicting modules in order to generate a new constraint that can be reconciled

with the old. In practice, analysis of the possible conflicts should exhaustively identify a small number and a heuristic can be written detailing how to deal with each conflict in its probable context. This method is therefore similar to Ferrante's characteristic error approach:⁹

information about characteristic error classes enables the resolution of a specific conflict to be dependent on the immediate context of the problem.

The correct response in terms of identifying which module to rerun is represented by a heuristic rule. In the case of a clash between sulphate attack considerations and workability requirements it might prove easier to rerun the sulphate module specifying a more resistant cement that requires less stringent w/c ratio controls. Since each module has the ability to run with any of its input and output variables already instantiated it is a simple matter to have the reconciliation heuristic supply a suitable *compromise* value to a contradictory parameter.

This conflict resolution strategy can therefore be seen as collecting a set of consistent constraints¹⁰. There were a number of problems with this resolution method. The success of a particular design depends on the inclusion of all pertinent constraints in the consistent set. The process of comparing each constraint in a new set against all the others required a separate dedicated module and was itself time-consuming. CRYSTAL only has arrays of single type, either string or numeric. This

⁹ [Klein & Lu, 1989] also recognize the need to use or develop domain dependent conflict resolution expertise.

¹⁰ Quinlan, [1983], describes PONDEROSA, a similar system for finding consistent sub-sets of a model containing possible contradictions. Quinlan compares the method with alternatives such as Bayesian and non-Bayesian uncertainty, (PROSPECTOR and INFERNO).

entails that the array handling functions must be duplicated and executed in parallel. This ensures that both string and numeric constraints can be accommodated.

9.2.6 Further Knowledge Acquisition

Knowledge acquisition for the earlier OPS5 modules was a simple matter of deriving rules from the text. These rules were limited in scope and appropriate for the narrow concerns of the individual modules. For example, the section quoted above in section 7.1 can be interpreted in a number of ways depending on the requirements of a particular system.

The general significance of the text is that there are a number of factors that may be present at a particular construction site that may indicate possible risk from sulphate attack. The OPS5 sulphate attack system made use of this knowledge in a number of rules that simply succeeded or failed consideration of the potential risk indicators.

Later versions of DEX have more global data needs. Thus simple affirmation or otherwise of risk became inadequate. The CRYSTAL version of DEX, therefore, uses a menu to record the actual source of sulphate attack risk. Figure 9.7 above, illustrates this menu from the user's point of view.

Unlike the OPS5 system, the CRYSTAL implementation uses a global variable to record the actual source of risk, such as soil, ground-water run-off or underfloor fill. This information may be used later in the consultation by other modules. For example, the consideration of steel reinforcement corrosion risk is based on exposure conditions. These are given in the following table taken from BRE Digest 263 [1982]:

Nominal cover to reinforcement

Condition of exposure	Nominal Cover <i>mm</i>				
	Concrete Grade				
	20	25	30	40	50 and over
<i>Mild</i> : e.g. completely protected against weather or aggressive conditions, except for brief period of exposure to normal weather conditions during construction	25	20	15	15	15
<i>Moderate</i> : e.g. sheltered from severe rain and against freezing whilst saturated with water. Buried concrete and concrete continuously under water.	-	40	30	25	20
<i>Severe</i> : e.g. exposed to driving rain, alternative wetting and drying and to freezing whilst wet. Subject to heavy condensation or corrosive fumes.	-	50	40	30	25
<i>Very severe</i> : e.g. exposed to sea-water or moorland water and with abrasion	-	-	-	60	50
<i>Subject to salt used for de-icing</i>	-	-	50	40	25

Table 9.1: Cover required for Structural Steel in Various Exposure Conditions (from [BRE, 1982a]).

Thus moorland water is a potentially very severe corrosion risk. It is therefore possible to use the consideration of sulphate risk to assign a value to a global variable that indicates the location of the construction site with respect to marsh or moorland.

In a similar manner, the presence of corrosion risk from de-icing salts may be used to infer that the concrete under consideration is a highway structure. This is then used in the consideration of AAR risk to indicate that the structure is particularly vulnerable and that the 3.0 kg Na₂O equivalent alkali content limit is inapplicable, [BRE, 1988].

9.2.7 Data Flow Analysis of Domain and Potential Constraint Conflicts

Further analysis of the potential interactions among the modules covering each risk area resulted in a realisation that there was a optimum sequence that would avoid potential conflicts. The optimum agenda is as follows:

Sulphate Attack

Steel Corrosion

Mix Proportioning

Alkali Aggregate Reaction

The sulphate attack and steel corrosion modules are independent of each other. Both of these modules must run before the mix design module because the mix design is dependent on their results. The AAR module must run after the mix design because it needs to calculate alkali content of the mix and this requires the mix proportions. In some circumstances, the AAR module may set a maximum cement content constraint. In this case the mix design must be rerun with this additional constraint.

Because of the above considerations the generalised constraint representation was abandoned in favour of the use of a single variable for each individual constraint. The results of the data-flow analysis of the relevant modules is summarised in table 9.2.

Table 9.2 ignores local control variables and only includes variables that are used in more than one module. A *U* indicates user input, a *C* indicates the result of an inference or calculation. Conflict is only possible where there are two modules which calculate or infer the same value. There are conflicts of this type where SRC and MIX may both specify an admixture; and SRC (steel reinforcement corrosion) and AAR both specify protective coatings. Further knowledge elicitation will be required to gather expertise in order to resolve these possible conflicts¹¹.

A final conflict may occur when AAR requires a reduction in cement quantity (*cem_q*) or increase in cement replacement quantity (*rep_q*) in order to lower alkali contents. In this case it becomes necessary to rerun the MIX module with new maximum cement and minimum replacement constraints. It is possible to run the module with the relevant constraints set automatically so the user need not be directly involved in this redesign phase. The mix design module is described in the next chapter.

As other modules are added to the system, it may be possible that other conflicts arise. It is not anticipated that these will require any techniques beyond those in current use. This method of *ad hoc* solution places emphasis on the need for additional knowledge acquisition in cases of further conflict in order to determine methods of resolution, [Klein & Lu, 1989].

¹¹ The domain of concrete durability design is not deep in the sense of a hierarchy of sub-tasks. The entails that it would not be viable to use the method of failure handling developed by Brown, [1985]. This method is dependent on a rich use of hierarchical decomposition of the design task and plans for execution of sub-tasks.

User Input	SO3	SRC	MIX	AAR
U Agg	Agg	C Admix\$ U admix_q	C Admix\$ U admix_q Agg	Admix\$ admix_q Agg
	C Cement\$	U air_p	U air_p Cement\$	
U Exposure\$	C Coatings\$	Exposure\$	C Cem_q C Fines_q	C Cem_q C Coatings\$ Exposure\$ Fines_q
U Floor\$	Floor\$	Floor\$		Floor\$
U Form\$	Form\$			Form\$
U Grade		C Grade	Grade	
U Humidity\$	U G_water\$	G_water\$ Humidity\$		Humidity\$
	C max_Rep		Max_cem	C max_Cem max_Rep
	C max_WC		max_WC	
	C min_Cem		min_Cem	
	C min_Rep			C min_Rep
			C q10mm	q10mm
			C q20mm	q20mm
			C q40mm	q40mm
	C Rep\$		Rep\$	Rep\$
U Workabil\$			C Rep_q Workabil\$	Rep_q

Table 9.2: Data Dependencies in DEX

In a few cases there is no definitive determination of a mix quantity, for example, the amount of cement replacement material is constrained but not to a unique value. When a figure is required for the mix design method, the user can be asked to provide a value between the established limits. Another method would be to subject the alternatives or range of valid values to a cost analysis and use the cheapest.

9.2.8 Overall Organisation of Modules

With a number of separate sub-modules each contributing advice for concrete mix design it was decided to organise them in some manner

whereby a series of sessions with various modules could produce a coherent set of specifications for an actual mix. The obvious architecture for such a task was a blackboard.

Blackboard architectures are discussed in chapter 2. A number of blackboard-based systems such as DESTINY, [Sriram, 1986, 1987a, and 1987b], are described in chapter 4. The organisation of the integrated DEX system is shown in figure 9.10.

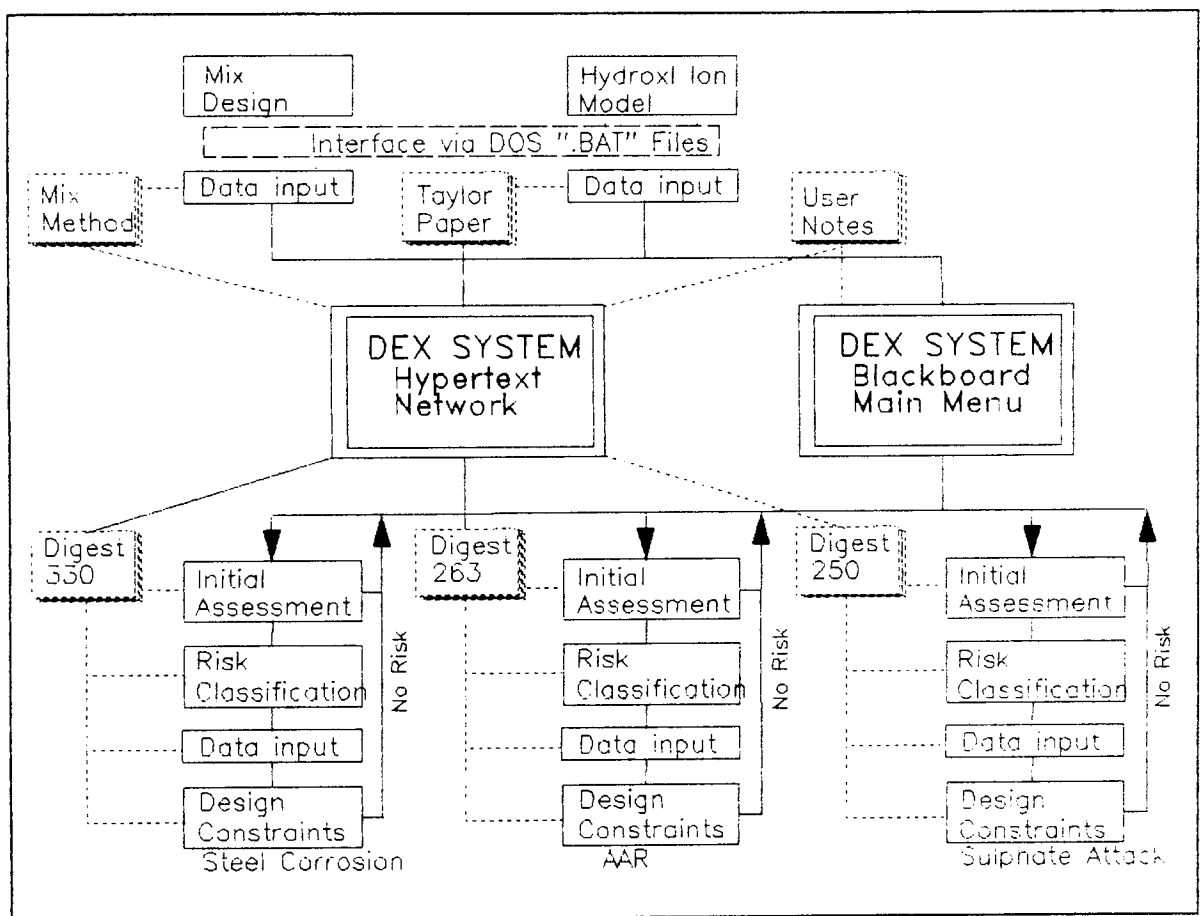


Figure 9.10: DEX Blackboard organisation

The durability modules such as SO3 and AAR correspond to the *specialist* level in Sriram's model. The modules for hypertext support, mix design

and hydroxyl ion concentration are the equivalent of Sriram's *resource* level modules, [*ibid*]. Communication between cooperating modules is performed via a number of blackboard files. Each on-going design project has a separate file summarising the current state of the design.

In addition there is a *working memory* file for the actual design in process that is used by each specialist module to retrieve required input and to post results to the rest of the system. These files are CRYSTAL system *export* files. The LOTUS123 resource modules require individual communication files that present data in formats required by the LOTUS123 spreadsheets. The interface with LOTUS123 will be described in the next chapter.

It is possible to configure a PC so that spare RAM is utilised as a *virtual disk*. A virtual or RAM disk has the same operational characteristics as a magnetic disk in that files are written to or read from it in the normal manner. However the entire disk structure is stored in RAM. This entails that RAM disk access is very much faster than a magnetic disk. Thus it is possible to use a file for the common blackboard *memory* without the usual drawbacks of file interfaces.

Overall control of the consultation, file management and other general user support such as facilities to consult individual specialists are dealt with by the DEX blackboard module. Refer to figure 9.2 for an illustration of the options that are presented by the blackboard controller. The DEX module itself, corresponds to Sriram's *strategy* level module, [*ibid*].

The DEX blackboard module initiates a new design by requesting information from the user that is required by a number of the durability

and resource modules. This information includes the placement and exposure of the concrete being designed. This input form was shown in figure 9.3. Additional information required results from earlier stages of the building design process that are not addressed by DEX. These include considerations such as strength grade specifications, maximum nominal aggregate size and the use of reinforcing steel or mass concrete.

A typical collection of this initial information is shown in Figure 9.11 which illustrates a DEX result report.

Design example	Review Results
The following information resulting from structural design considerations was input by the user:	
Form of construction: Mass concrete	
Strength grade: 35 N/mm ² at 28 days	
Workability: med-high	
Max nominal aggregate size: 40mm	
The following environmental placement considerations have been indicated:	
half-buried	
exposed	

Figure 9.11: Initial Input Results Screen

DEX schedules the specialist modules in accordance with rules that reflect the optimum order. DEX has rules that embody a prototype¹² dependency network among the modules' global variables. This network arose from the data-flow analysis work presented in section 9.2.7. The network allows the system to be used for general consultation of any module in any order.

This ability to allow the order to pursue non-optimum paths through the specialist modules follows the specification and implementation by Kalay et al, [1987], of a system which combined passive assistance with active design. Like Kalay's system, DEX allows the user to choose any module but then proceeds to complete any unresolved issues in the optimum order.

The network also allows the user, at the strategy level, to alter responses given at the specialist level. As the user calls for a particular module or alters a response, the system uses the dependency rules to undo any results further down the dependency chain. This entails that the integrity of any on-going design is not compromised by conflicting input or results. Figure 9.12 depicts the pop-up window that informs the user of the forward-effects of a call to a specialist module out of sequence.

¹² That is *prototype* meaning *not yet fully developed* rather than *prototype* as a *schema for design*.

Design example	DEX U5.1
Options:	
What would you like to change: ?	Effecting:
Exposure	Corrosion risk,AAR risk
Ground level	Sulphate risk,Corrosion risk,AAR risk
Concrete type	Sulphate risk
Strength grade	Corrosion resistance,Mix quantities
Workability	Mix quantities
Max Agg size	Sulphate resistance,Mix quantities
Soil type etc	Sulphate risk
cement type	Sulphate resistance,Mix quantities
cement replacement type	Sulphate resistance,Mix quantities,AAR resistance
alkali content	Corrosion resistance,AAR resistance
aggregate type	Mix quantities,AAR risk
No changes	
Mix Design: done	
Trial Mixes: not reqd	
AAR: risk from quartzites - negated by low alkali	

Figure 9.12: Design Dependencies

An example of a rule implementing the dependency network is given in figure 9.13. This rule disjunction is executed if the user is altering the form of construction. The condition-rules *sure?* and *?yes* ensure that the user is aware of the consequences of altering the form of construction. If this is the case, the rule sets the *form\$* variable to a blank string¹³. The dependency rule then calls a child rule that initialises all the variables set by the SRC module.

¹³ The initialised *given\$* variable is used to represent the fact that it is now necessary to rerun the rules that gather the structural design input specifications from the user.

If there is a site classification of 1 (from the sulphate attack module) the rule initialises all of the variables set by the SO3 module. This rule is only necessary if the site is class 1 because the form of construction is only relevant in class 1 soils. Both the dependency rules call further rules to initialise variables set in modules further down the chain such as the mix design.

Alter Inputs		Sp
		< Rule >
OR	:Test r\$="Concrete"	↑
	AND :Assign r\$="Sulphate4"	
*	AND sure?	Sp
*	AND ?yes	Sp
	AND :Assign given\$=""	
	AND :Assign Form\$=""	
*	AND initialise SRC results and down	Sp
	AND :Test Class=1	
*	AND initialise SO3 results and down	Sp
		↓

Files	Run	Clear	Build	Utilities	Quit	07:53:01 pm	IE
-------	-----	-------	--------------	-----------	------	-------------	----

Figure 9.13: Rule Fragment of a Dependency Relationship

Although mechanisms for managing the dependency network have been created it is not clear that rule-based representation is the best method. In addition, it will be necessary to conduct further knowledge elicitation from the domain experts to confirm the results of the original analysis.

As has been explained, it is not obvious that any conflicts will arise between modules. If however, later additions or alterations do make conflicts possible, the dependency network might be used to reconcile these conflicts. Kumar and Topping describe a more flexible method of representing a dependency network that facilitates *dependency directed*

backtracking, in order to resolve non-monotonic states such as constraint violation, [Kumar & Topping, 1990, and 1991], (see chapter 4 for further discussion of this method).

The dependency rules effectively comprise the ability of the system to perform redesign at the global level. This is in addition to the redesign abilities of the individual modules at the specialist levels. In practice, in the domain of concrete durability, redesign has only been required to offer flexibility to the user rather than as an enforced response to conflicts.

9.2.9 Testing DEX

Different commentators identify different evaluation criteria for completed expert system applications. Berry & Hart, [1990], and Preece, [1990], include *maintainability*, *reliability*, *validity*, *user acceptability* and *verification*. Each of these were considered during the development of DEX.

Verification in conventional software development consists of the formal proof that the system satisfies its specifications. In practice, formal verification is impractical for all but the most simple conventional algorithms. Informal verification determines whether or not the software produces the output that would be expected from a given input.

Testing of output is an acceptable substitute for formal verification to the extent that the full range of possible input is considered. The verification of a system is therefore the primary responsibility of the

developer. Each distinct task within each DEX module has been exhaustively tested and therefore verified during development through rapid prototyping.

At a higher level, verification entails the consideration of all the permutations arising from execution of each module in every possible order and must also encompass the full range of output possible from each module. The testing of the system would also encompass extraneous factors such as operating system errors. This level of verification is appropriate for a system intended for public distribution but is not necessary for an academic project.

Validation of a system determines the success of the system in meeting the real-world requirements of the original project. Validation therefore requires explicit high-level system specifications. These specifications may evolve during system development.

Validation is therefore the responsibility of the body that commissioned the system development. This *Beta-testing* is only possible after the *Alpha-testing* or verification phase conducted by the developer.

Validation of DEX by BRE is appropriate if the prototype is a potential basis for further development and use with more definitive specifications as a design system, [St. Johanser & Harbidge, 1986].

A second aspect of the periodic Beta-testing at BRE has been the identification of misinterpretations of the digests. For example an early version of the AAR module included the contribution of alkalis from aggregates in the consideration of the 3.0 kg alkali content limit but the alkalis from aggregates were omitted from the calculation of the binder alkali content for compliance with the 0.6% limit.

The relevant passage from digest 330 does not make it obvious that aggregate alkalis should be considered when calculating the cement alkali content:

...Portland cements with an equiv. Na₂O (acid soluble) of less than 0.6% are unlikely to suffer from ASR even in combination with a reactive aggregate. (p5)

Other examples of possible misinterpretation arise from cross-module inferences such as the equivalence of marshy and moorland ground water for steel reinforcement corrosion and the assumption that the presence of exposure to de-icing salts entails that the concrete in question comprises part of what digest 330 refers to as a *highway structure*.

User acceptance is obviously determined with respect to the likelihood that potential users will find the system easy to consult. As a feasibility study, one of the central concerns of the DEX project has been the user interface facilities. To some extent the degree of user-friendliness required of a system can be determined by the developer. If a system is tedious for the developer to test then it is likely that a user would also object to the interface.

Beyond these considerations the evaluation of user-acceptance can only be conducted by further Beta-testing among the intended users. The DEX project has implemented a variety of interface techniques from simple dialogue in the early OPS5 systems through forms, menus and graphics as well as hypertext in later versions. Actual evaluation of the resulting interface is beyond the scope of the DEX project as an academic exercise.

9.3 Potential for Rule Induction

Induction techniques are attempts to automatically create rules from tables of example data. As an experiment, the shell Super-Expert¹⁴ was used to induce a set of rules from a table of data relating AAR performance to sandstone aggregate properties such as density, moisture retention and shape, [Collins, 1986].

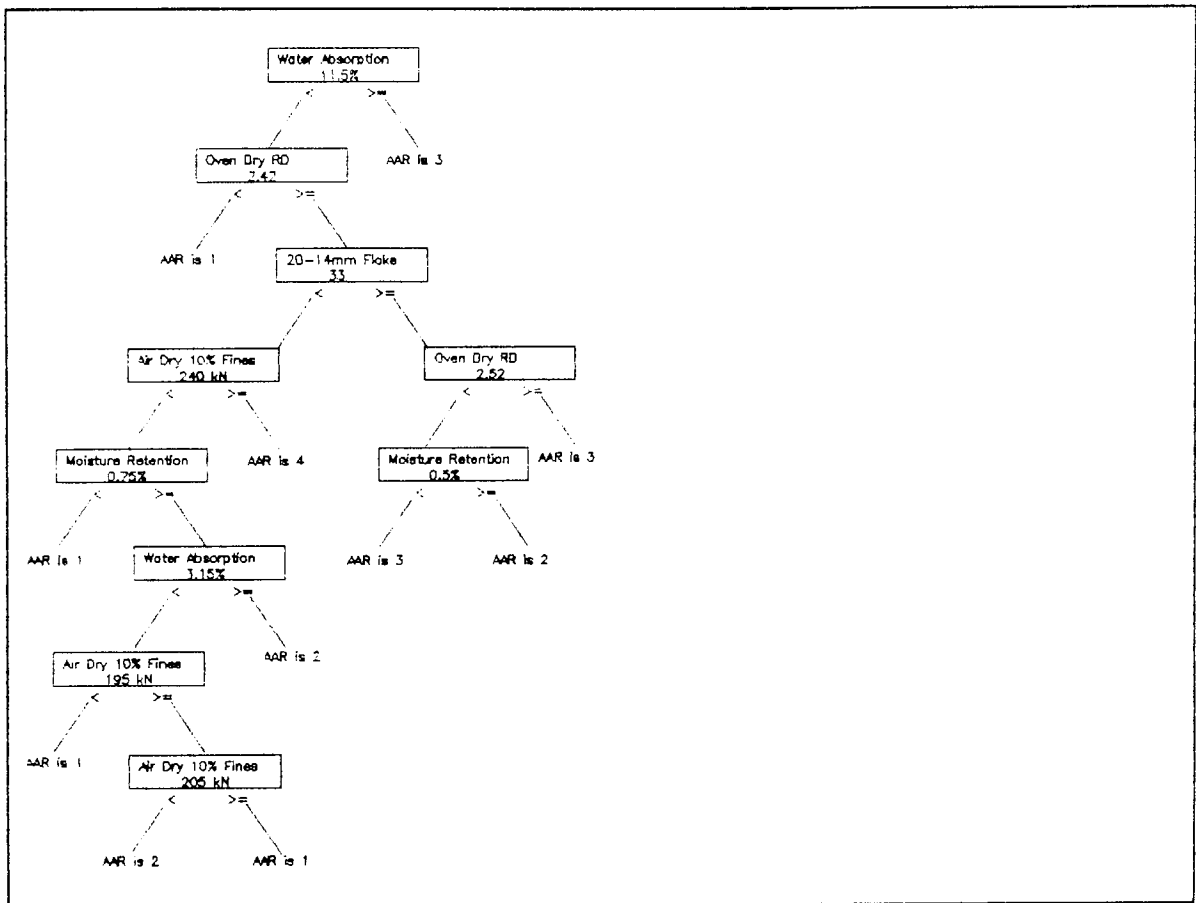


Figure 9.14: Rules from Induction

¹⁴ CRYSTAL has its own induction engine but this must be purchased separately and the experiment did not justify the expenditure.

The rule-tree resulting from the induction session is shown in figure 9.14. Each node of the tree represents a decision based on the value of an attribute. For example, the top node entails that AAR risk classification is 3 (high) if water absorption is greater than or equal to 11.5%.

A technical and philosophical discussion of induction is beyond the scope of this thesis. Dr. Collins is in possession of these results but has not yet been able to express an opinion of their provenance. The general nature of the inferences implied by the rule-tree is unconvincing. The most obvious criticism is that the use of exact limits on the values is unrealistic. The immediate conclusion of this induction experiment is that, for this data at least, inducted rules should be evaluated by a domain expert.

9.4 Discussion of the CRYSTAL Prototype

The work described in this chapter was intended as an exploration of the use of expert system techniques in the domain of concrete durability design. One practical result of this work is a framework for building cooperating expert systems in the domain. This framework includes a set of rules implementing techniques for user-interface, explanation, durability design control and the use of external software¹⁵. This infrastructure was used for the implementation of the module for the

¹⁵ CRYSTAL lacks facilities for modular compilation. Therefore the commonly used rules must be replicated in each module. As much as 25% of the rule-base of each module could be implemented as a single shared object file in a more traditional language such as Turbo-Prolog or C.

prevention of steel reinforcement corrosion. The rapid development of this module demonstrates the utility of the DEX blackboard architecture and common rule-set.

The development of the DEX prototype has demonstrated the utility of the BRE digests as knowledge sources and as models for durability design. The extension of the system to cover new areas of durability and less common materials should utilize similar sources. The use of such material addresses the concern of Clifton et al, [1985], that user acceptance of expert systems in the domain of concrete durability design crucially depends on their prior acceptance of the knowledge sources.

Further development of DEX could seek to extend the capabilities of the system within the limits of the BRE digests. In some cases these digests have been shown to be incomplete. For example, digest 250 on sulphate attack, [BRE, 1981], includes a provision that allows for *further relaxation in the cement content and water/cement ratio* when the concrete is to be used for strip foundations and trench fill for low-rise buildings in class 1 soil. The digest omits to specify the extent of this permissible relaxation. The digest also fails to provide explicit advice for increased protection for those cases that are identified as *most troublesome* such as ground floors.

As a prototype, the system has potential for use as a teaching system. Hover, [1985], describes a traditional program for mix design which is used for training students. The main benefit of a computerized mix design system is in the ability of the user to make changes in the input parameters and see the results immediately. This capacity for rapid exploration of the relationship between input parameters and mix

proportions emphasises *results rather than procedures*, [ibid]. In addition, expert system training aids allow the novice to explore the logical reasoning underlying expert decisions.

Within the scope of the cases covered explicitly by the digests, DEX may have practical application as a system anywhere that the source documents are already used. As a reference or compliance checker the system would become more useful as more modules are added in other areas and from other sources such as British Standards. In addition to this passive use as a reference system, DEX is also able to perform active design.

At a practical tool for concrete designers DEX is therefore a useful implementation of design codes in a vein similar to the DURCON system described in chapter 4. DEX is a valuable development in that it embodies UK design knowledge. From the details published about DURCON and the other systems described in chapter 4, DEX would seem to use more sophisticated knowledge representation and user interface than existing systems in the domain.

Further additions to the DEX modules could address issues such as resistance to frost, abrasion and those aggressive substances not already covered. In addition to extending the knowledge taken from the digests, and the number of digests covered, DEX could also incorporate established knowledge about materials such as high-alumina cement, light-weight aggregates and silica-fume used as a cement replacement material.

One of the most interesting results of this later work is the analysis of the data-flow implicit in the digests. The conclusion that arises from this

analysis, that there is little potential for conflicting advice between the digests so far studied, is important in that it further illustrates the utility of the digests as knowledge sources. This result also entails that there is no need to develop general purpose techniques for conflict resolution in the domain.

The realisation of the power conferred by hybrid hypertext/expert systems is the most valuable result of the project described in this thesis. The use of hypertext in conjunction with an expert system provides greatly enhanced user support in complex domains. Hypertext also delivers an attractive alternative to user input through questions or forms. The final benefit is the advantage to be gained from the development of hypertext documents as a supplement to knowledge acquisition. The brief summary given in this chapter anticipates the detailed discussion given in chapter 11.

Chapter 10: DEX Modules in LOTUS123

CRYSTAL includes a wide variety of mathematical functions. It is therefore possible to use CRYSTAL to program most aspects of the areas of durability covered by the DEX system. For example, CRYSTAL was used to implement the hydroxyl ion concentration model described in chapter 7. This was possible without the use of embedded sub-routines written in FORTRAN or C.

Although CRYSTAL has this mathematical functionality, the production rule representation is not the most appropriate method of implementing some aspects of concrete durability design. Chapter 4 included a number of references to projects that use tables to implement some forms of design knowledge taken from codes of practice, for example, [Stahl et al, 1983].

It was therefore decided to investigate the ability of CRYSTAL to interface with external software applications. CRYSTAL provides standard interfaces to dBase3, LOTUS123 and structured ASCII files. All three of these interfaces were used to implement the table relating sulphate soil classification to mix proportion constraints, from page 3 of Digest 250 on sulphate attack [BRE, 1981].

As was explained in the previous chapter, the use of the built-in CRYSTAL interfaces made unacceptable demands on random access memory (RAM). Although an alternative method was found for representing the design table, LOTUS123 provided a number of features that were of potential use in concrete design.

The nature of spreadsheet use is such that most worksheets are created by individuals for private, limited circulation¹. Fischer & Rathke, [1988], describe spreadsheets as *convivial* tools. Conviviality emphasises software that is both useful and usable by *common* users rather than programmers. A number of references describe the use of spreadsheets in a variety of applications in the broad domain of design.

Brewer, [1987], reports a spreadsheet for concrete mix design². Dilly et al, [1987], use a spreadsheet to calculate durability based on strength considerations. These applications exploit the ease with which spreadsheets can be used incrementally to generate solutions to complex calculations. Spreadsheets also provide a means for exploring the effects of altering input parameters. Intermediate and subsidiary results can be displayed easily whereas a procedural language tends to obscure these by-products.

Saouma et al, [1989], note that LOTUS123 spreadsheets constitute a convenient data entry option. The authors describe a system for code compliance checking. This system requires a lot of input data. Entry of this data through a series of questions as generated by an expert system proved to be tedious to the user. LOTUS123 provides a means of capturing the initial data through input forms. Spreadsheets are one of

¹ Spreadsheets are a mature technology and therefore it is not necessary to describe common spreadsheet facilities in this chapter.

² This spreadsheet is similar in scope to that described in section 10.1. Brewer's system is based on the ACI committee 211 standard: "Standard Practice for Selecting Proportions for Normal, Heavyweight and Mass Concrete".

the most common computer programs, [Fischer & Rathke, 1988]. Saouma et al therefore assume that most potential users of the system will be familiar with the typical spreadsheet user-interface³.

The utility of spreadsheets for training and *what-if* analysis is described by Wenzel, [1987]. Skwara, [1987], notes several other advantages of spreadsheets for structural engineering design. In addition to those features already mentioned in this section, Skwara lists the speed of spreadsheets and the ease with which results can be presented as a variety of charts and other graphical forms.

10.1 Mix Design Spreadsheet

It is possible to view mix proportioning as the top-level goal of concrete design with durability consideration being relatively minor sub-tasks. The DEX system takes the alternative view that the proportioning is a comparatively trivial final procedure that is only possible after exhaustive durability design. This reflects the emphasis on durability that was revealed by initial consultations with experts at Aston University and at BRE.

Each durability module described in chapters 7 and 9 sets constraints on mix constituents and proportions with respect to potential durability problems. Teychenné et al, [1988], describe a mix design method which uses these constraints to provide a basis for the mix proportioning procedures. This method is based on the same relationship between

³ The safety of this assumption is demonstrated by the fact that the LOTUS123 user-interface is widely copied by other spreadsheets. The LOTUS system has become a *de facto* spreadsheet standard. Therefore if the expert system user has any spreadsheet experience it will probably be of relevance to use of LOTUS123.

specified strength and water/cement ratio as that used for the initial PROLOG mix design system, [Shirley, 1985]. The Teychenné method is considerable more sophisticated and broader in scope than the Shirley method.

The method is described as a number of procedures for determining the various mix proportions based on specified strength and workability. These procedures are mostly simple arithmetic and data table look-ups. The method is summarised by a mix design form. A modified version of this form is given in figure 10.1. The form is intended to be copied and to be filled in by the engineer as the design proceeds. The mix proportions presented at the end of the mix design method are the final results of the entire concrete design process.

The nature of the mix design form suggested its implementation as a computer spreadsheet. The worksheet itself is divided into a number of different areas. The presentation area closely resembles the original design form and it is this area that is reproduced as figure 10.1⁴. Figure 10.1 shows the state of the worksheet before any data has been input. The rest of the worksheet is a hidden area containing the formulas and tables embodied in the mix design procedures.

The initial implementation of the method as a spreadsheet was relatively complex. The method uses simple arithmetic and data tables for most of the mix proportioning procedures. However, Stage 1, [ibid], (p13), uses

⁴ The presentation area differs from the original form in that it includes the modified tables for air-entrained concrete and concrete with replacement of cement by pfa or slag. The additional lines of this enhanced table can be conditionally hidden or displayed depending on the use of entrainment or cement replacement.

Concrete Mix design form		Job Title:	
Stage:	Item:	Reference or Calculation Values:	
1.1	Characteristic spec Strength N/mm ² at Proportion defective days %
1.2	Std dev	F3 Samples..... N/mm ² N/mm ²
1.3	Margin	(k=)	x = N/mm ²
1.4	Targ Mean Str	C2 +..... = N/mm ²
1.4.1	Air Content%	
1.5	Cement Type	spec	OPC/SRPC/RHPC +
1.6	Agg Type: Coarse		Crushed/Uncrushed
	Agg Type: Fine		Crushed/Uncrushed
1.7	Free-w/c ratio T2 & F4		} use lower value
1.8	(Max w/c)	(Spec) }
2.1	Slump or Vebe Time	Spec	slumpmm or Vebe timesecs
2.2	Max Agg Size	Spec mm
	Prop of pfa	Spec	
2.3	Water Content	T3	kg/m ³
3.1	Cement Content	C3	/ = kg/m ³
3.2	(Max Cement)	(Spec) kg/m ³
3.3	(Min Cement)	(Spec) kg/m ³ use 3.1 if ≤ 3.2 use 3.3 if > 3.1
3.4	Modified free-w/c ratio		kg/m ³
3.6	pfa Content	C7	
4.1	Relative Density of Agg (SSD)assumed	2.7
4.2	Concrete Dens	F5	271 kg/m ³
4.3	Total Agg	C4	271 - - = kg/m ³
5.1	Fine Agg	% passing 600µm sieve%
5.2	% Fine Agg	F6	%
5.3	Fines Content	C5	x = kg/m ³
5.4	Coarse Content	C5	- = kg/m ³
=====			
Quantities per m ³ Cement pfa Water Fine Agg Coarse Agg (kg)			
(kg) (kg) (kg) (kg) 10mm 20mm 40mm			
(to nearest 5kg)			

Figure 10.1: Modified Mix Design Form
after [Teychenné et al, 1988] (Table 1)

visual interpolation from a graph for the basic relationship between strength and water/cement ratio (W/C). The use of this graph is summarised in figure 10.2.

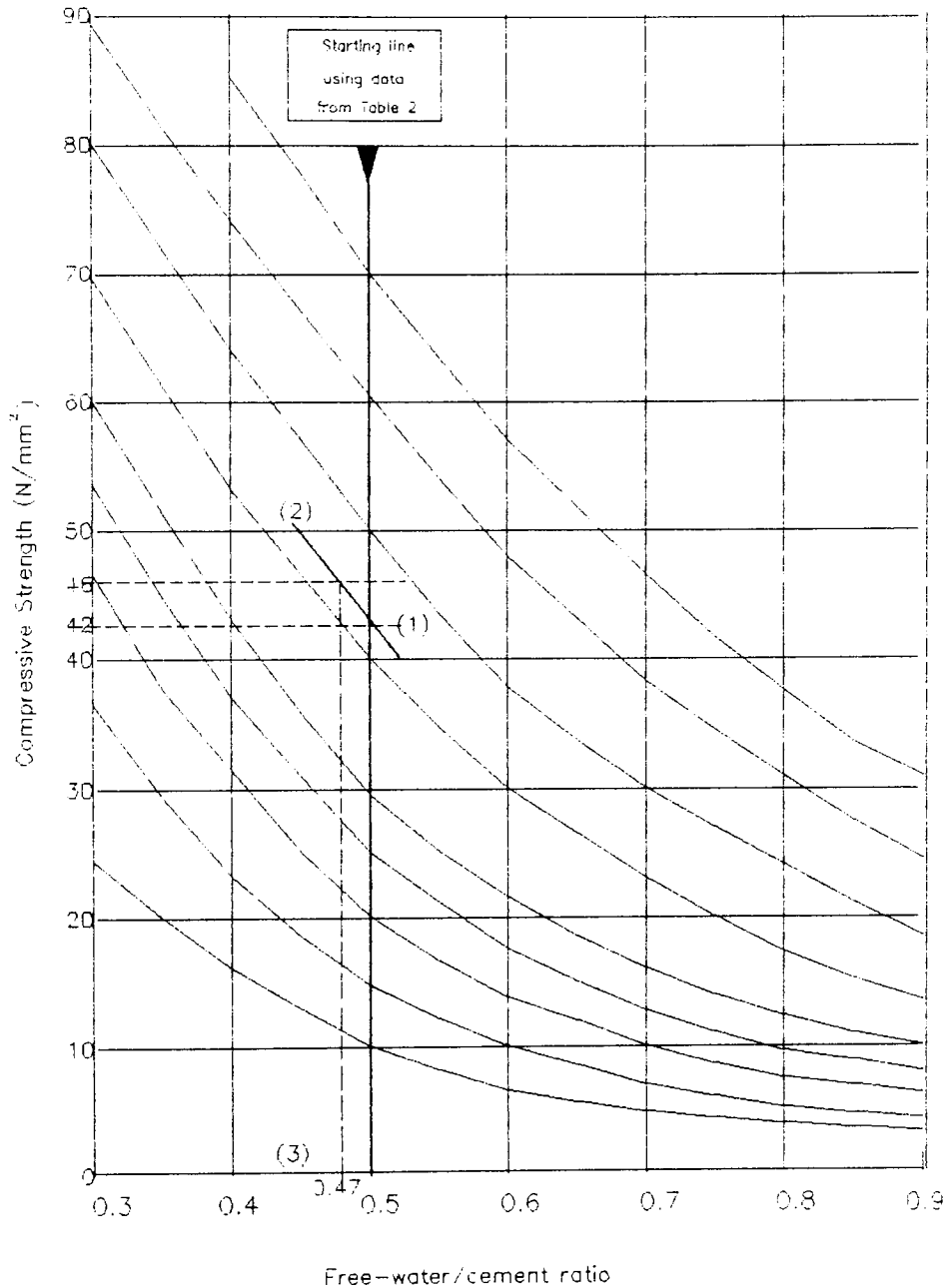


Figure 10.2 Visual interpolation in mix design
after [Teychenné et al, 1988] (Figure 4 p16)

Firstly, the strength value is obtained for concrete with W/C of 0.5, (1). This value is plotted on the graph. The engineer must then draw an *imaginary* curve parallel to the existing curves that passes through the desired strength value, (2). Finally, the corresponding W/C ratio is read off from the imaginary curve, (3). Although this procedure is easy to describe and to perform, it does not lend itself to computer representation.

The curves shown in figure 4 of the method and reproduced here as figure 10.2 are based on empirical data. From the text, it is not clear how many different factors are included in the relationships. The relevant factors include specified strength and age, type of cement and type of aggregate. These factors are therefore qualitative summations of many lower level quantitative attributes. Thus it is not possible to represent the curves as contours of some n -dimensional surface.

The strength-W/C relationship was therefore represented by exhaustively reading the data-points from the given curves. These points were then tabulated. The spreadsheet makes the assumption that the given data points are sufficiently close so that intermediate points can be approximated by linear functions. The engineer's imaginary curve can then be approximated by a simple linear interpolation from the given data-points. The results of this approximation is satisfactory for the degree of accuracy required by the method and achieves perfect agreement with each of the examples given by the authors.

The mix design spreadsheet uses a similar linear interpolation for the procedure given in stage 5. This procedure relates maximum nominal aggregate size, workability, fine aggregate grading, W/C ratio and the proportion of fine aggregate. This, more complex, interpolation also

achieves near-perfect accord with the given examples. The data-base facilities of LOTUS123 that are required for the interpolation entail that these procedures would be extremely difficult to implement using CRYSTAL.

The mix design spreadsheet can be run as a stand alone application. As such it would be useful to any engineer who uses the BRE mix design method described by Teychenné et al. The user is able to run the spreadsheet in two modes. The first mode relies on the user to move through the spreadsheet supplying the required input values in the appropriate cells⁵. The second mode makes use of extensive macros to automate the input requirements and to ensure the correct completion of the design.

In the automatic mode it is possible to disguise the worksheet to the extent that the user need not be aware that he is actually running a spreadsheet. The DEX system uses the mix design spreadsheet in this automatic manner. In addition DEX includes a CRYSTAL module that provides the necessary pre-processing of the mix design input values. The effect of this pre-processing is to make the spreadsheet completely invisible to the user. The system runs from CRYSTAL to the spreadsheet and back again with the spreadsheet simply performing the calculations and posting the results to the blackboard.

Part of the input requirements of the mix design spreadsheet is the constraints on various proportions established by durability considerations. In many cases these constraints are sufficient to

⁵ As explained above, it may be valid to assume that most users of a PC-based system would have the require familiarity and skill with LOTUS123 spreadsheets.

determine a unique mix design solution. In other cases the user is left to decide, for example, on the type of cement to be used and the use and extent of cement replacement materials. The options are presented in an order that implicitly mirrors the relative cost implications. Addition of information regarding the price of materials would allow the spreadsheet to calculate the optimum cost solution, [Brewer, 1987]. This optimisation is therefore a candidate for future development of the system.

10.2 Hydroxyl Ion Spreadsheet

A VAX C program to implement the Taylor ion concentration model, [Taylor, 1987], has been described in chapter 7. It was possible to translate this model as a CRYSTAL module. In both cases, the complexity of the concentration prediction algorithm provoked some difficulty in its realization as a program. During the development of the proposed CRYSTAL program, LOTUS123 was used to provide a means of analysing the intermediate steps of the algorithm.

The LOTUS123 hydroxyl ion concentration model was relatively easy to create and reproduced the results for the four examples given in the paper. This worksheet was also used to provide sensitivity analysis for the ion concentration model. Using the spreadsheet it was possible to graphically illustrate the effects of incrementing various input parameters. The increase in concentration of the main cement ions over time is shown in figure 10.3. This figure shows significant agreement with the calculated results and observed data given by Taylor ([*ibid*], p9).

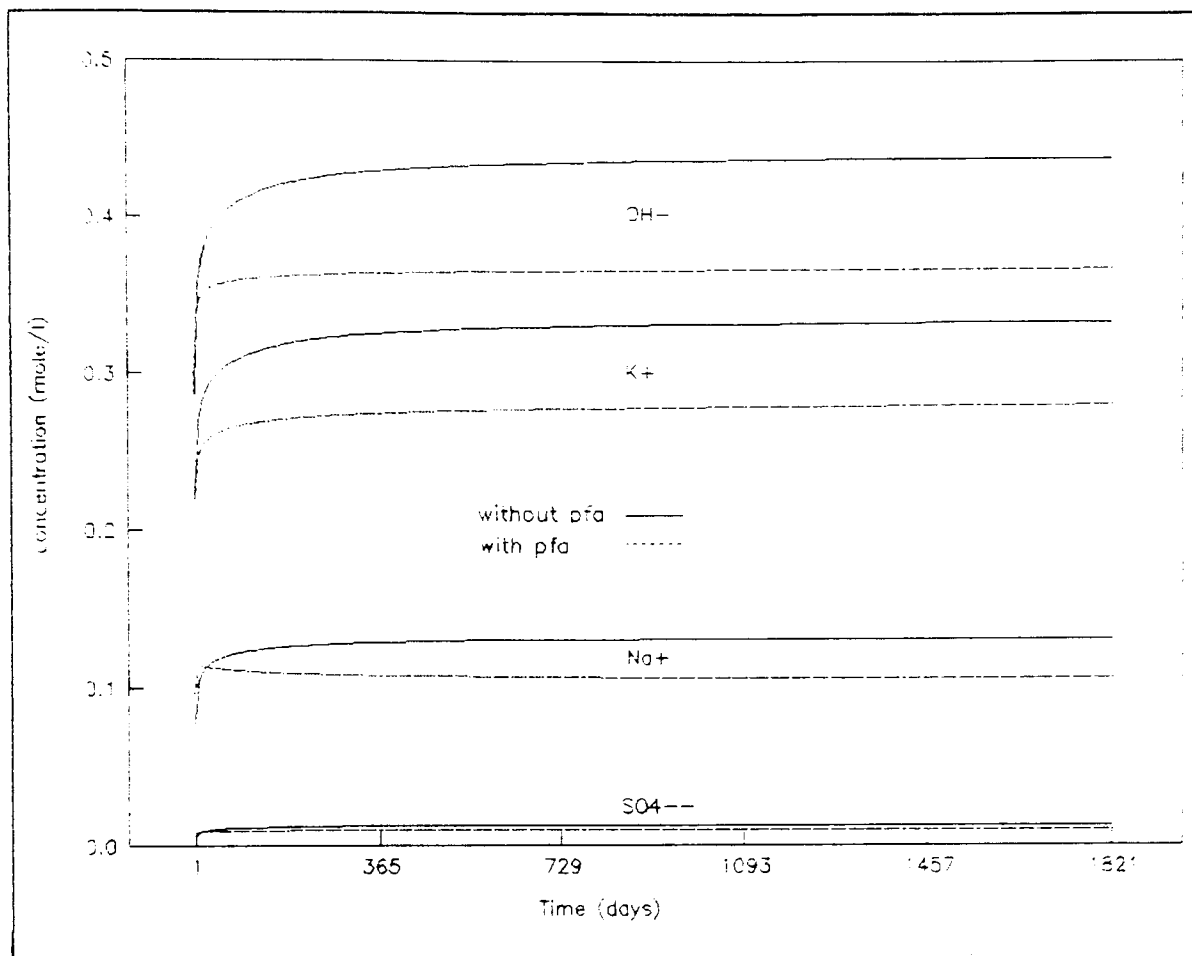


Figure 10.3: Ion Concentration Predictions

In a similar manner it is possible to use intermediate results of the calculations to investigate other features of the prediction model. For example equation (5) of the Taylor paper: $F = 1 - \exp[-k_2 * (T-k_3)^{k_1}]$, is used throughout the model to calculate the proportion of a phase that has reacted within a given time period, (T).

Each phase has different values for the three k constants. Additionally the constants are altered by the use of pfa. Hence the effect of pfa

cement replacement on rate of reaction can be graphically illustrated by iterating the model over increasing time periods. The results of such an iteration are shown in figure 10.4.

These results are available from the spreadsheet with no additional calculation or output procedures. To obtain the same results from a program written in a procedural language such as FORTRAN or C would have required some modification of the original program for ion concentration prediction.

The success of the LOTUS123 ion concentration model determined that it should be used directly by the DEX system. Thus the attempt to implement the model as a CRYSTAL module was abandoned. In common with the mix design spreadsheet described in the previous section, the ion concentration model can either be run directly by a user or executed by a CRYSTAL module which gathers and validates the required input.

The potential use of the hydroxyl ion concentration prediction requires the development of heuristics that match the predictions and aggregate type to potential AAR performance. It is also necessary to test and calibrate the implementation of the model against published experimental results. This work is outside the scope of the expert system research described in this thesis⁶.

⁶ For example, Kollek et al, [1986] report mortar bar results relating OH⁻ concentration to AAR damage. These results imply that a OH⁻ concentration of 0.3M at 180 days prevents AAR with Opal. These results would have to be repeated and verified. The implications for mortar bar tests would have to be decided for concrete in real environments. Similar results would have to be obtained for other aggregates

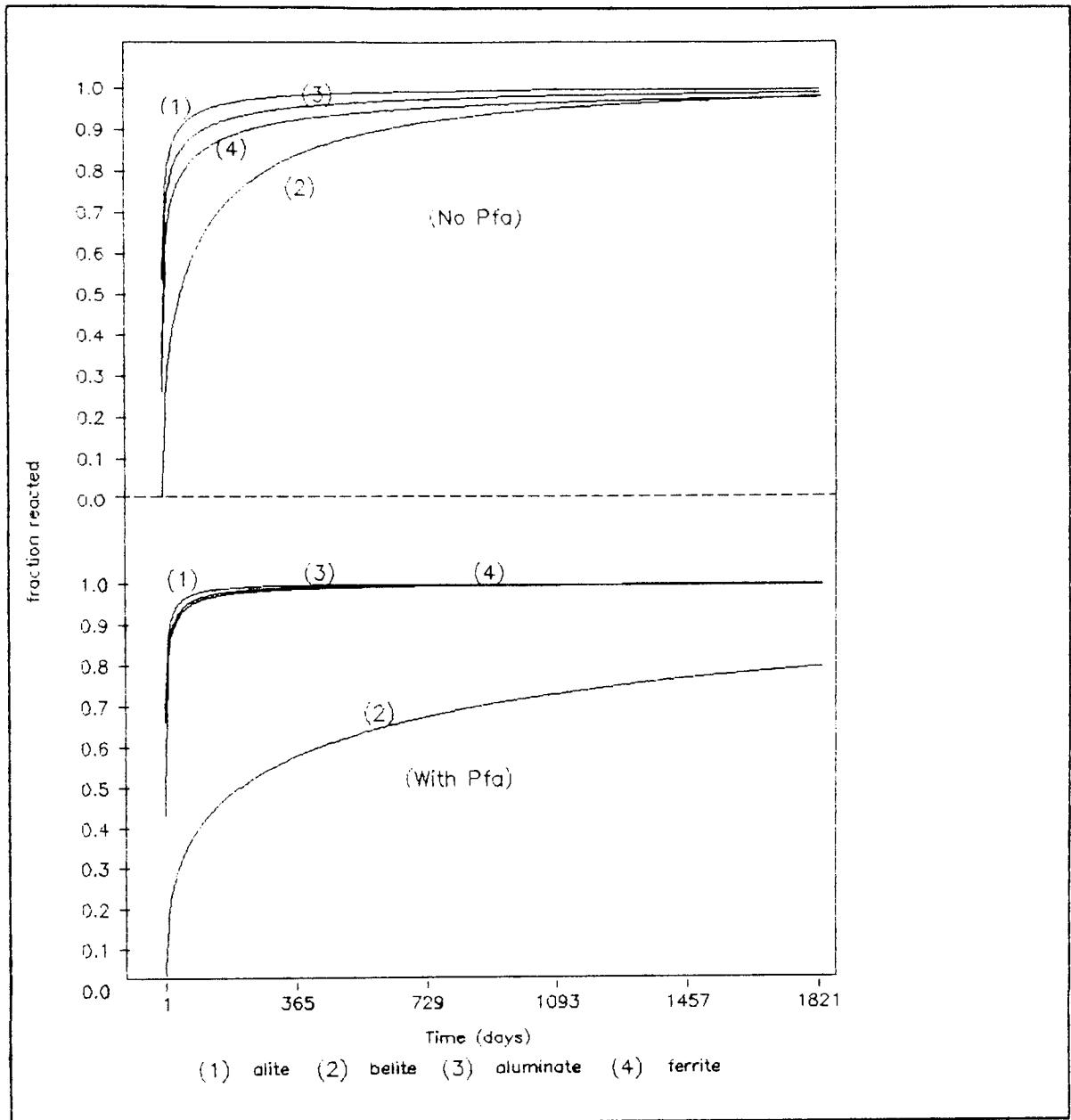


Figure 10.4: Effect of Pfa on the Rate of Reaction of the Four Main Concrete Phases

10.3 Lotus - Crystal Interface

Both the mix design and ion concentration spreadsheets utilise the same method for interfacing between LOTUS123 and CRYSTAL. Initial

development of these spreadsheets made use of the proprietary interface supplied with the CRYSTAL development system. This interface supplies CRYSTAL functions that can be used to read and write values in cells contained in the Lotus spreadsheet file. These interface functions were therefore used in both the spreadsheets and additionally in the experimental use of LOTUS123 to implement lookup tables.

The CRYSTAL-Lotus interface provides many functions other than those required for simple data-passing. The interface therefore uses a significant amount of RAM. On a *standard* PC system with 640 kbytes of memory, the interface reduced available RAM to the point where it became impossible to load and run the durability knowledge-bases.

The simple interface requirements of the LOTUS123 spreadsheets can be resolved by alternative methods. Communication from CRYSTAL to LOTUS is accomplished by means of a simple ASCII file created by the CRYSTAL modules. This file contains the values required by the spreadsheet in a specific order with one value per line. A simple *autoexec* macro reads these values into the spreadsheet through a file import command.

Communication from the spreadsheet back to the CRYSTAL modules is achieved by another Lotus macro that copies the mix design results into a template that is formatted as a CRYSTAL exportable file.

A spreadsheet *template* is simply a group of spreadsheet cells that performs general operations on changing data. To illustrate, the interface template is reproduced in figure 10.5. When printed, this template produces a file in the CRYSTAL export file format. The relevant CRYSTAL module can then import the values into the appropriate variables.

```

*
MIX.kb                Thu Oct 25 19:08:06 h1990
EXPORT RULES

EXPORT VARIABLES
  Admix$ = "none"
  Admix_q = 0
  Agg = 20
  agg_dense = 2.60
  air_p = 0.00
  Cement$ = "PBFC"
  Cem_q = 360
  coarse_f$ = "Uncrushed"
  defectives = 0.0
  fine_f$ = "Uncrushed"
  fine_grade = 70
  fine_q = 490
  Grade = 35
  KB$ = "mix"
  margin = 12
  max_Cem = 0
  max_WC = 0.60
  min_Cem = 300
  name$ = "Example"
  q10mm = 465
  q20mm = 935
  q40mm = 0
  Rep = 0
  Rep$ = "none"
  Rep_q = 0
  slump$ = "10-30"
  strength = 47
  strT = 28
  str_s = 0
  Water_q = 155
  wet_dense = 2405
  work$ = "med-low"

```

Figure 10.5: LOTUS123-CRYSTAL Export File Template

Each entry in this template is a LOTUS123 formula. For example the entry for the `Water_q` variable is `+ "Water_q = "&@string(water_q,0)`. This formula concatenates the string "Water_q" (the name of the CRYSTAL variable representing water quantity) along with the relevant formatting

string (" = ") and a string representation of the value held in the spreadsheet range called water_q (formatted to zero decimal places because the CRYSTAL variable is an integer.

Kumar and Topping compare two methods of providing an interface between PROLOG and FORTRAN, [1987a]. The first of these methods is the file method adopted for the CRYSTAL-LOTUS123 interface. The authors found that the method using a direct interface via C was significantly more efficient than the file method. A direct interface is not feasible for the DEX system since source code access is not available for either CRYSTAL or LOTUS123. Fortunately the interface via files does not appreciably delay the system.

In the present implementation it is necessary to return to the DOS operating system in order to execute CRYSTAL and LOTUS123 in alternation. Overall control of the DEX system is therefore accomplished by means of a DOS batch file. This file iteratively calls either CRYSTAL or LOTUS123 according to the current stage of the design consultation. The design stage is deduced from the presence or absence of various files used to communicate between the two packages.

It is theoretically possible to realise a better command interface. If sufficient RAM can be reserved, for example by use of expanded memory, either CRYSTAL or LOTUS123 can load a copy of the DOS command shell in order to execute the other package. This would enable a direct and integrated interface between the two systems.

Chapter 11: The DEX Hypertext Support System

Hypertext is a well established technique for the presentation of textual and other information. Hypertext can be best understood by contrast with traditional text representation methods. In a traditional text such as a book or a plain ASCII file, the information is organized and accessed linearly. Access to information in a linear text is achieved by reading through the text in sequence. Cross references are found manually by use of a table of contents and an index. References may also be found automatically by searching the document for the required reference string.

In a hypertext system the text itself may be stored in any convenient manner. Access mechanisms make use of active links between words or concepts in any area of the text and related concepts, definitions, notes, digressions or illustrations from any other area of the text. Additionally, links may be formed for references between different texts or even executable code or sound or video devices. Thus hypertext embraces a range of applications from the help facility of LOTUS123 to full hypermedia systems.

In his paper introducing and surveying hypertext, Conklin, [1987], quotes an early definition of hypertext as:

a combination of natural language text with the computer's capacity for interactive branching, or dynamic display ... of a nonlinear text ... which cannot be printed conveniently on a conventional page. [ibid] (p17)

Conklin then describes hypertext as being implemented as windows on a screen. The windows are associated with objects in a database. The objects in this database are linked by pointers. Details of hypertext database implementation vary from system to system. In general a hypertext object-record will consist of a word, phrase or page (or page-number). In addition there will be fields for links to various other objects. There may be more than one type of link and therefore more than one object to link to.

The user of a hypertext system is presented with a series of screens of information. Each screen is called a *page*. On each page there will be a number of key words or phrases that can be selected by moving a highlight. These words are called *buttons*. The highlight may be moved using the cursor keys or mouse or by some other method such as typing the first letter. The user selects a particular button and the corresponding link is retrieved from the database and the appropriate action, or action script, performed. In addition, there will be buttons on all screens to allow linear movement from page to page.

For example, a screen could have a button highlighting a formula name: *equation(5)*. This button represents a database object. The object could have a number of links to different objects:

the general definition of the formula: " $F = 1 - \exp[-k_2 * (T-k_3)^{k_1}]$ ",

a legend defining the terms used in the formula,

the values of the various constants depending on the current location in the linear text,

a link to another page that provides an explanation of the origin of the formula,

general information regarding the use of the hypertext facilities.

The actual object-link used for the button selected by the user can be determined by a number of factors. The required link may be decided by the current *location* of the user in the linear text. Alternatively the system may represent different links by different *hot-keys*. For example, the **F1** key has come to be the *de facto* standard in many systems for a request for help. Other more intelligent hypertext systems may use some form of *user model* to determine the type of link generally required by a particular user. An ideal hypertext system would allow individual users to define and retrieve their own personal links, [Carando, 1989].

One of the attractions of hypertext is that links can be pursued in either direction. For example, it is possible to follow a link from a reference into that document. It is also possible to move back from the referenced document to the original document. Additionally, it is possible to store an arbitrary number of links steps and trace back through the entire sequence.

Windows are a natural choice for displaying linked text items. Most hypertext systems use a full screen to display the text in the standard linear form. Smaller pop-up windows are then used, for example, to show definitions or illustrations of a selected button. The use of windows entails that in most systems the selective disclosure of a number of textual or graphical objects is executed at a relatively low level by screen-handling functions.

There are a number of disadvantages to hypertext. The most commonly noted drawback is the possibility of the user getting *lost* in the document, [Conklin, *ibid*]. In other words, the user becomes disorientated

by a sequence of links between linearly distant sections of text¹. Other problems arise from circular references or sections of the text that become disconnected from the whole. Carando, [1989], notes that it may be difficult to retrieve precise facts efficiently from a hypertext system.

These problems are not necessarily intrinsic to the hypertext methodology and it should be possible to develop systems that avoid them. For example, the hypertext system may be developed from an original linear text and incorporate the ability to follow links in either direction and use an index. In this case most of the drawbacks such as disorientation and redundant sections of text are circumvented.

Hypertext is therefore a general purpose tool that is suitable for representing many varieties of textual or graphic material. Hypertext is particularly suitable for animation of large amounts of reference material especially if this material is usually distributed among a number of separate paper publications². The complexity and scope of codes of practice such as British Standards documents and BRE digests make these an ideal candidate for hypertext representation³.

One of the most compelling conclusions arising from the investigation of the expert system implementation of BRE digests is the realisation that

1 This lost feeling may be no more than a bias towards linear thought styles that dictate that a book necessarily begins at the beginning and ends at the end.

2 As evidence for the potential benefits of hypertext representation of reference material it is possible to cite Hillier & Freeman, [1984], who report a survey that found that engineers spend approximately 70% of their time searching for source material. A comprehensive hypertext system tailored for the particular needs of individual engineers could dramatically reduce this inefficiency.

3 The extensive diagrams, tables, formulae, footnotes, technical terms, internal and external references in a PhD thesis would be an ideal candidate for hypertext animation.

both the BRE and British Standards Institute would have much to benefit from developing a standard hypertext system for publication of codes of practice.

11.1 Hypertext as an Expert System User Support Technique

Chapters 2, 8 and 9 included discussion of issues in the provision of user support. The conclusion of these discussions is that explanation facilities are often flawed. User support can be made more sophisticated by provision of a rich variety of question and answer types; by modelling the user's level of expertise or by dressing the explanation in more natural language, [Savory, 1986, Hughes, 1987, and Gilbert, 1988]. However, in the provision of such facilities it is possible to undermine the original use of the system to perform design, make diagnoses or as a front end.

The requirements of a user who presses the **F1** button or responds "Why?" to a question cannot be anticipated and resolved in general, even by a sophisticated explanation or justification facility. In addition, the concept of explanation may itself be invalid. Coyne, [1990], discusses the difference between classical *cognitivism* and *connectionism*. Explanation as used in expert systems, is dependent on the cognitivist idea that reasoning can be decomposed into logical steps and that it makes sense to explain reasoning by enumerating and describing each of these steps⁴.

⁴ Although Coyne uses the idea of explanation to illustrate cognitivism, he states that most AI research is based on the cognitivist ideal. This is most obvious in the use of rules and inference in expert systems. To some extent his discussion of cognitivist explanation can be extended to AI in general.

Coyne, [*ibid*], points out that there are in principle several difficulties with the cognitivist view of explanation. The first difficulty arises from explanations based on unsound or controversial theory. Coyne also mentions the problem that each step in an explanation can itself require explanation. There may be in principle no final explanation of anything.

For example, the explanation for the use of sulphate resisting portland cement (SRPC) in areas with high levels of sulphate in the soil is that SRPC contains less than 3.5% tricalcium aluminate (C₃A)⁵. The utility of cements with low C₃A is explained by the propensity for the hydrates of C₃A to react with sulphates and cause damaging expansion. The explanation can continue by describing the reasons why C₃A should react with sulphates. This explanation can then proceed into a physical explanation of chemical reactions in general and so on, possibly *ad infinitum*⁶.

The alternative view of explanation is based on the connectionist view of language as enabling a *cooperative domain of interactions* [*ibid*] (p75). Coyne relates that discourse occurs in the context of groups of people influencing other groups. The result of this discourse is the underlying matrix of a particular discipline - the kuhnian paradigm. A more scrupulous mode of explanation therefore relates a particular decision to its surrounding milieu.

⁵ C₃A is cement chemists shorthand notation for 3CaO.Al₂O₃.

⁶ Even a full explanation of cement chemistry would still omit to explain why expansion occurs and why this is harmful to the concrete. Furthermore, new discoveries in the physics of elementary particles could add lower levels to the explanation. Although it is not likely, a scientific breakthrough could falsify the higher levels of the explanatory theory.

In the case of the decision to use SRPC, the only sound explanation may be to restate the actual heuristic in its context. The heuristic, as expressed in Digest 250, is the simple provision for the use of SRPC in areas with high risk of sulphate attack because SRPC tends to lessen the occurrence of sulphate damage. Causal theory is therefore used neither in the decision to use SRPC, nor in the explanation of that decision. This mirrors the original emphasis of expert systems that they embody heuristic reasoning rather than causal reasoning.

The milieu of design decisions is often embodied in codes of practice. These codes summarise the group experience of committees of recognized experts, [Clifton et al, 1985]⁷. The emphasis of the codes is the statement of heuristic provisions rather than causal theories. Therefore the justification of design decisions based on codes of practice is grounded in the code provisions themselves. There is no need for further elaboration. If an expert system can make explicit reference to the relevant section of a code or standards document this is sufficient *explanation* of its conclusions from inferences based on the standard.

11.2 Use of Hypertext in the DEX System

Much of the knowledge embodied in the DEX system described in the previous chapters is derived from BRE digests. Initial attempts to provide user support within DEX made use of large chunks of the original text to answer queries. In effect much of the justificatory ability of the early versions was of the variety "X is the case because digest Y says so".

⁷ Refer to chapter 4 where the similarity of codes of practice and Gero's Prototypes is noted.

The creation of numerous text extracts was made more efficient by capturing the original document with a computer scanner device. The entire text was then stored on disk as an ASCII file. Help screens and other user support material was placed into the CRYSTAL modules by elementary cut-and-paste methods. The use of large volumes of textual material increased the size of the CRYSTAL knowledge-bases beyond acceptable limits.

In order to reduce the memory demands of the textual support material, the help screens were removed from the actual knowledge-base and stored as disk files, with a single screen per file. It was then necessary to create rules in CRYSTAL to retrieve the appropriate screen of information in response to particular user queries. The difficulty of anticipating user needs motivated the creation of CRYSTAL rules that allowed the user to view screens in any desired order.

The browsing facilities became the basis of a rudimentary hypertext system: DEX-HT⁸. Although Conklin, [*ibid*], describes hypertext in terms of the data-base paradigm, any other formally equivalent programming methodology can be used. Conklin himself refers to the resemblance of hypertext to semantic networks. Thus CRYSTAL rules can be used to represent the hypertext implications of selecting a particular screen button.

DEX-HT contains a module for each of the digests used as knowledge-sources for the durability expert systems, [BRE, 1981, 1982a, and 1988]. Additional modules cover the BRE mix design method, [Teychenné, 1988], and the DEX expert system user guide. Each

⁸ The details of the further development of the browsing facilities need not be described here.

document is split into screen sized pages with each page typically corresponding to one or two paragraphs of text. For example, the text fragment quoted in section 7.1 from BRE digest 250 corresponds to a single page of the hypertext version of the digest (this actual page is illustrated in figure 11.4 below).

The DEX hypertext system incorporates most of the facilities described in section 11.1 that are commonly found in hypertext systems. The required techniques have been fully developed and a common core of rules has emerged that facilitates the addition of further modules.

An example fragment of a CRYSTAL rule is given in figure 11.1. The rule is part of the knowledge of the hypertext representation of the DEX system user guide. This rule is fired once the user has selected a button. The buttons are represented by the string stored in the r\$ variable. Thus each rule disjunction tests for a particular response and executes a particular link action.

The first rule shown in figure 11.1 opens a link to another document (in this case digest 330 on alkali-aggregate reaction, [BRE, 1988])⁹. The next link rule opens a small window on the screen and displays an appropriate form. The final rule moves the user to page 2 of the current document.

⁹ The hypertext system stores the page location of the user in each opened document. If the document is opened for the first time or in a specific context the system can open an appropriate page.

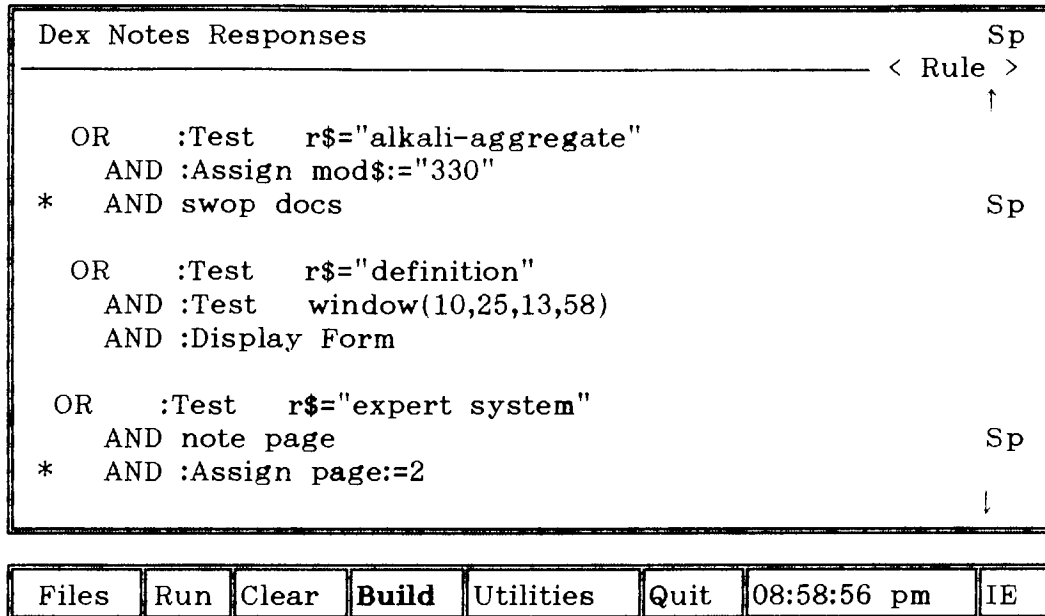


Figure 11.1: A CRYSTAL Rule Implementing Hypertext Links


Figures 11.2 and 11.3 show an example DEX hypertext page from the DEX user guide. The first figure shows the page as it would appear when first displayed. The user can either move on to another page in sequence or to an index for the current document. The other available buttons can be displayed by pressing the return key .

Figure 11.3 shows the result of selecting the *definition* button (this is the link action represented by the second rule in figure 11.1).

3 What a Hypertext system is

A hypertext system is a computer program that 'animates' a body of text so that, for example, distant but related items can be directly accessed from each other and key concepts can have directly accessible definitions and explanations. Typically a hypertext system arranges the document as a number of 'cards' or screens each of which will contain a number of 'buttons' which can be activated in order to access other areas of the document; expand a word into its definition or display a table or diagram.

Hypertext is thus very useful for making large and complex documents easier to read. It is particularly useful where an activity requires the use of a number of such documents with many technical terms and cross references. Because of the very large quantity of textual material in a significant application, hypertext systems tend to make heavy demands on computer memory.

Press Return for screen buttons and space for index

Figure 11.2: A Hypertext Screen From the DEX User Guide

The use of CRYSTAL to represent links entails that DEX-HT is able to use all the facilities offered by expert systems in its determination of appropriate links. It would therefore be possible to make the CRYSTAL hypertext system *intelligent* to a greater degree than would be possible with a purely data-based system. The use of CRYSTAL also entails that DEX-HT has a number of disadvantages. In many cases the rule-representation is less efficient than a standard hypertext system.

In order to keep the expert system knowledge-bases within acceptable size limits, all the hypertext rules are contained in separate rule-bases. This means that there is some delay while the system moves between

3 What a Hypertext system is

A hypertext system is a computer program that 'animates' a body of text so that, for example, distant but related items can be directly accessed from each other and key concepts can have directly accessible definitions and explanations. Typically a hypertext system arranges the document as a number of 'cards' or screens each of which will contain a number of buttons which can be activated in order to access other areas of the document; expand a word into its definition or display a table or diagram Map

Hypertext is thus very easier to read. It is use of a number of such references. Because of the v significant application, hyp computer memory

dèfin'ition, n. statement of the precise meaning of a term

ex documents
ity requires the
rms and cross
material in a
eavy demands on

Next	Screen	Previous
------	--------	----------

Press Return for screen buttons and space for index

Figure 11.3: The Result of Selecting a Hypertext Button

expert system and hypertext modes. It would be preferable to have the hypertext system run as a memory-resident program, simultaneously with the expert system proper.

The main advantage of using CRYSTAL to implement DEX-HT is that the hypertext can be used as an alternative to the standard sequence of questions in the consultation with the DEX expert system (DEX-ES). The hypertext facilities are supplemented by an additional type of button that instantiates one of the global variables used in DEX-ES. This variable can then be posted to the blackboard file using the same techniques as the expert system modules.

Figure 11.4 shows a page from digest 250 , [BRE, 1981]¹⁰. The user is able to select buttons that instantiate variables for soil-type, run-off, ground-water contaminants or underfloor fill. In the figure the user has highlighted the button indicating the possibility of contamination of ground-water by run-off from a colliery tip. Once the user selects a button, DEX-HT pops-up a small question window that ensures the user is aware that he is providing input that may be used in inferences.

DIGEST250

Occurrence of Sulphates

Sulphates occur mainly in strata of London Clay Lower lias Oxford Clay Kimmeridge Clay and Keuper Marl . The most abundant salts are: ?

- Calcium sulphates (gypsum or selenite)
- Magnesium sulphate (Epson salt)
- Sodium sulphate (Glauber's salt)

Sulphuric acid and sulphates in acid solution occur much less often but might be found where pyrite in the soil is being slowly oxidised, for example near **colliery tips** and in marshy country As well as occurring naturally, sulphates are sometimes present in materials such as colliery shale used as fill beneath solid concrete groundfloors. Moisture from the ground carries the salts to the underside of the concrete which is then attacked leading to expansion and cracking of the floor and the surrounding structures (see Digest 222). Brick rubble particularly with adhering plaster, ashes and some mine/industrial wastes are potential sources of damage by sulphate attack.

Next	No Risk	Previous
------	---------	----------

page 3

Press Return for screen buttons and space for index

Figure 11.4: Hypertext as a Means of Obtaining Input Values From the User

¹⁰ The data input buttons on this screen are green on a colour screen.

It would be possible for the user to use DEX-HT almost exclusively for the consideration of durability risks. DEX-ES would still need to be consulted to determine the constraints required for risk prevention.

A final year undergraduate project in the department of civil engineering at Aston University has recently addressed the possibility of using the Apple Macintosh Hypercard language to create a hypertext system with design capabilities, [Howard, 1991]. This project demonstrated the feasibility of such a system in a domain restricted to partial consideration of sulphate attack, alkali aggregate reaction and steel reinforcement corrosion.

The results of such a brief project cannot be conclusive. However, the system demonstrated the feasibility of using the hypercard language for the representation of digests. This representation included photographs which could not be reproduced in the DEX-HT system. The hypercard project also succeeded in the determination of design constraints from digest provisions.

This system was, however, entirely passive. The onus was on the user to provide appropriate input and to access the cards where the resulting constraints were indicated. In addition the hypercard system has limited calculation abilities and would have needed to use a spreadsheet or other external language for the calculations required for the mix proportioning.

11.3 Hypertext as A Methodology for Expert Systems Development

The creation of hypertext documents entails that the developer becomes familiar with the organisation and conceptual structure of the original

text. Additionally, the creation of the hypertext links instils an awareness of the inferencing and computational implications of the knowledge contained in the document.

Links are identified as the developer becomes more aware of the key concepts of the document. For example, links may be suggested by the multiple occurrence of several words and phrases in various parts of a digest (and perhaps in other documents). Therefore the creation of links in turn increases the familiarity of the developer with the domain by unfolding a vocabulary of key concepts.

Breuker et al, [1985], note the need for an interpretative framework to facilitate the transformation of knowledge into an expert system. Collins et al, [1985], state that it is necessary for a knowledge engineer to undergo an apprenticeship in a domain before developing an expert system. To some extent the creation of hypertext links provides both the required framework and the apprenticeship.

The practical result of the realisation of a fully animated hypertext document is that the developer acquires some degree of expertise in the domain covered by the document¹¹. To the extent that codes embody much of Gero's Prototypes, the knowledge engineer is absorbing the Prototype in the creation of the hypertext document. Thus hypertext development serves as a foundation for the knowledge acquisition required for the creation of an expert system in the domain. This is particularly true if the hypertext facilities are implemented with the intention of subsequently developing an expert system.

¹¹ A possible supplement to the manual extraction of knowledge from code texts would be the automatic creation of a knowledge-base through natural language analysis of the text, [Rasdorf & Parks, 1987].

The major conclusion arising from the creation of the DEX-HT modules is that hypertext can make several important contributions to expert system development. Firstly, hypertext can provide a philosophically sound means of supporting the user in the expert system environment. Hypertext animation of support text can be used in addition to or instead of standard expert system explanation and justification facilities.

Hypertext can also be used as a supplement or addition to standard expert system interface techniques such as inference driven questions or input forms. In certain cases it may even be preferable to have the expert system knowledge representation and inferencing completely embedded in a hypertext system. In this case the user conducts a consultation by moving through the appropriate hypertext pages and selecting the necessary buttons.

Finally, hypertext development is a valuable addition to standard knowledge acquisition techniques. The discipline of identifying and creating links provides the knowledge engineer with a sound basis for further specific knowledge acquisition.

Chapter 12: Conclusions

The introductory discussion in chapter 1 outlined a number of goals that would guide the completion of this project. Later chapters have described the attainment of these substantive goals. These goals are summarised by the following list:

- * Knowledge acquisition and analysis of data and task flows for the domain of concrete durability design.
- * Identification of research issues in expert system applications
- * Realization of an expert system for concrete design through the successive development of prototypes in number of sub-domains.
- * Evaluation of expert systems methodologies through successive prototypes developed in an AI language; a higher level language and an expert system shell.
- * Demonstration of the utility of BRE digests as knowledge-sources.

A description of the basic elements of expert systems in terms of architecture and techniques was developed into a selective enumeration of issues of current interest in expert systems applications. These issues include:

- * Use of uncertainty
- * Combined use of rules and frames and other representations
- * Use of mathematical and other deep knowledge
- * The knowledge acquisition bottleneck

- * Techniques for user support facilities such as explanation
- * Provision of sophisticated user interface including minimal keyboard use and uncluttered screens and the use of both passive and active graphics

A number of theories and models of design have been described. This discussion was intended to serve as a background for the description of existing design systems given in chapter 4. The material in chapters 3 and 4 also provides a foundation and context for the development of a concrete durability expert system.

Several commonly noted features of design are listed below:

- * Goal directed search or exploration
- * The search space is generated by the design objectives
- * This space is then restricted by design constraints
- * There is no guaranteed optimal solution.
- * The design process may be amenable to analysis in terms of generic tasks.
- * The use of these generic tasks could simplify the creation of design systems.
- * The synthesis of design solutions within the search space is characterized as the recursive synthesis of lower level components of the design
- * The design itself is described by the set of instantiated variables at the lowest levels of this component hierarchy

- * The knowledge required for design can be chunked into prototypes which include various types of knowledge such as heuristics, previous experience, vocabulary, evaluation criteria and design plans.
- * Design systems incorporating the above features of design are often based on a blackboard architecture.
- * Within the blackboard, modules can be classified at three levels.
- * Strategy modules plan and control the design.
- * Specialist modules embody expertise in particular areas of the design process.
- * Resource modules can be used either by the system or by human designers to perform low level tasks such as compliance checking and interfacing to analysis routines.
- * Additional modules may be required at any level to handle failure of any design step, for example through constraint violation or conflict.
- * Another hierarchy within such design systems reflects the relationship of structural components.
- * The lowest level of this hierarchy is selection of specific materials.

A brief overview of concrete durability issues concentrated on the three main problems covered by the project: sulphate attack (SO₃), steel reinforcement corrosion (SRC), and alkali-aggregate reaction (AAR).

Additional sections cover mix design and other considerations such as design life. The overall impression of concrete durability knowledge is that it is unstable and fragmentary.

The conclusions of the chapter on concrete design therefore prescribe the potential uses of expert systems in the domain. Concrete durability systems should be limited to the scope of the provisions set out in accepted codes of practice. These codes embody the sum of the relatively certain knowledge in the domain. Extension beyond these limits would reduce the confidence of the system's conclusions and expose both the knowledge engineer and the domain expert to unacceptable legal liabilities, [Sales & Topping, 1985, and Blum, 1986].

Chapter 6 presents the development of simple PROLOG expert system for basic mix design. Chapter 7 describes other prototype systems written in another AI language, OPS5. These systems incorporate knowledge from BRE digests on sulphate attack and AAR. These prototypes established a number of considerations for further work in the domain.

Although PROLOG could be used to write a program that creates mix designs, this program could not then be used for another problem such as ensuring resistance durability problems. Each durability problem would require either a different system or significant extension of the mix design system. The development of the PROLOG mix design system, therefore, entailed the development of most of the features normally expected in an expert system shell. This was required in order to give the system greater generality.

Each of the OPS5 durability modules incorporates a common sequence of tasks. This sequence is implicit in the BRE digests used as knowledge

sources. Each durability module therefore conducts a consultation by seeking initial indicators of risk; evaluating the risk; and then setting appropriate constraints on the mix design to negate the risk. Additionally the consultation can include a redesign cycle. This can either occur because the precautions do not immediately negate the risk completely or because the user wishes to alter a response or a specification.

OPS5 was used in the hope that this would reduce the amount of development effort required to investigate the use of expert systems techniques in concrete design. Despite being a tool designed specifically for expert systems development, OPS5 provides limited expert systems facilities. In particular OPS5 lacks the sophisticated user interface functions that are increasingly expected of computer applications in general and expert systems in particular.

Both VAX PROLOG and OPS5 lack the mathematical functions that many engineering expert system applications require. It is possible to interface these languages to other languages that provide mathematical capabilities, [Kumar et al, 1987a]. However, the use of more than a single language on a mainframe can entail considerable implementational difficulties. Even if indirect interfaces are used, through disk files for example, the solution is still unsatisfactory because of the potential maintenance and distribution problems.

It was possible to draw on the conclusions of previous chapters in order to consider those features that are desirable in a shell for use in design systems. The problems of objective evaluation of shells are discussed. Despite these problems it is possible to establish some criteria for shell evaluation. In the light of the experience with shells available within the

research group and the specific evaluation of a number of shells the CRYSTAL system was chosen for the further investigation of expert system techniques in the domain.

The most important of the technical criteria are included in the following list:

- * Suitability of inference engine with respect to design
- * Ability to implement a design/redesign cycle
- * Ability to easily integrate modules to reflect problem decomposition
- * High degree of mathematical functionality
- * Ability to interface to other packages and languages
- * Development environment that automates the edit/compilation/run cycle and generally facilitates rapid prototyping
- * User environment that encourages user acceptance specifically
- * Colour and graphics to highlight important input and output
- * Minimal screen clutter and keyboard use
- * Overall speed of operation.

The remaining chapters recount how CRYSTAL has been used to develop an integrated system for the consideration of the major durability problems covered by BRE digests. The DEX system includes enhanced CRYSTAL versions of the original OPS5 durability modules as well as an

additional SRC module. These modules are organised under a blackboard architecture. The final expert system module is responsible for controlling the sequence of the durability and other modules.

In addition to the durability modules the system incorporates a LOTUS123 spreadsheet that implements the BRE mix design method. This method combines the constraints arising from the consideration of durability problems. This integrated system, DEX, has progressed to the Beta-test stage. Evaluation and possible development of the system has now been passed on to BRE.

The development of the DEX system is described through a series of experiments with various methods of implementing some of the expert system facilities described in chapter 2. Other implementational investigations sought to capitalise on the positive features of the shell that correspond to the criteria for evaluation and which enhance its use in the domain.

An extensive analysis of the data flow properties of the DEX system revealed that there was little possibility of constraint violation or conflict. The potential for conflict can be handled on a *ad hoc* basis with the domain experts providing rules for conflict resolution where this is not obvious from the context of the conflict.

Chapters 7 and 10 outline the implementation of an algorithm that embodies Professor Taylor's model of hydroxyl ion concentration, [Taylor, 1987]. The LOTUS123 version of this model can be interfaced to the DEX-ES system and could, in principle, be used as a resource module by the AAR specialist. This would presuppose the successful testing of the

model against a significant volume of experimental results. Additionally it would be necessary to incorporate expertise that related the ion concentration prediction to potential AAR performance.

Chapter 11 discusses the conceptual and implementational issues of hypertext. The combination of hypertext and expert system paradigms is the paramount achievement of the DEX development project. Hypertext is an ideal medium for BRE, BSI or some third party to use in the publication of codes of practice.

Hypertext is also proposed as a sound method for providing user support within expert systems. The use of hypertext for user support reflects the realisation that the concrete durability knowledge contained in digests and codes of practice is essentially heuristic¹. It is therefore disingenuous to develop explanation facilities, for example, that go beyond the level of restating the code provisions.

Within the expert system environment, hypertext provides a means of supplying the user with the background information to just the level of detail and in any order he requires within the scope of the source *matériel*. Furthermore the development of hypertext animation of a document provides the knowledge engineer with some degree of expertise in that portion of the expert system domain covered by the document. Thus the creation of a hypertext system is good preparation for the development of an expert system based on the same material.

¹ Codes of practice are summaries of the design experience of the members of the committee responsible for establishing the code. Codes are therefore compiled heuristic knowledge, [Hickman, 1986]. This is precisely why they make good knowledge sources.

The DEX system therefore includes both expert system modules (DEX-ES) and hypertext animations of the source digests (DEX-HT). DEX therefore exhibits a higher degree of sophistication in areas such as knowledge representation and user interface functions than existing systems such as DURCON. This hybrid system is proposed as a useful tool for the training of students. The most useful feature of the system in this respect is the ability to explore rapidly the global effects of changing input parameters. DEX-HT provides useful support for students investigating the problems of concrete durability.

Additionally, DEX could be used in design practice as an alternative, or as a supplement, to the source digests, [Clifton & Oltikar, 1987].

Obviously the use of DEX in such circumstances presupposes that it successfully passes the assessment by BRE. Further development of the system could then be undertaken to increase the number of digests covered by DEX-ES and DEX-HT.

Additional documents could be included in DEX-HT to broaden the scope of the user support facilities. Development could also seek to include knowledge that is outside the scope of BRE digests. Noting the caveat in the conclusions of the literature search in concrete durability, it is technically possible to include knowledge-bases that would deepen the ability of the system to reason beyond the limits of the codes of practice.

A different direction for development of the DEX system might arise as a result of the Beta-testing of the system by BRE experts. The creation of the knowledge-base and hypertext links has already brought to light some minor problems with the digests themselves. Several of the authors cited in chapter 3 and 4 note that codes of practice are prone to logical

and other errors such as inconsistency, incompleteness, and ambiguity. One of the results of the DEX project therefore may be that the digests themselves can be improved by the process of implementing them as knowledge-bases, [Reboh, 1983].

References

- ACI Committee 201, 1977,
"Guide to Durable Concrete",
In Journal American Concrete Institute Vol. 74, No. 12, pp574-609
- ACI Committee 515, 1979,
"A Guide to the Use of Waterproofing, Dampproofing, Protective and
Decorative Barrier Systems for Concrete",
In Concrete International: Design and Construction, Vol. 1, No. 11,
pp41-81
- Adeli H., 1987,
"Knowledge-Based Expert Systems in Structural Engineering",
In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
techniques to civil and structural engineering, Edinburgh: Civil-Comp,
pp391-398
- Adeli, Hojjat, 1988,
"Expert System Shells",
In Adeli, Hojjat (ed), 1988, "Expert systems in construction and
structural engineering", London, Chapman and Hall, pp33-44
- Adeli, Hojjat, 1988,
"An Overview of Expert System in Civil Engineering",
In Adeli, Hojjat (ed), 1988, "Expert systems in construction and
structural engineering", London, Chapman and Hall, pp46-83
- AIAI, 1989,
"Directory of AI Related Engineering Projects",
Compiled by Terry Lydiard, KEG Note 10, Artificial Intelligence
Applications Institute, University of Edinburgh
- Aldridge J., Cerutti J., Draisin W. and Steuerwalt M., 1986,
"Expert Assistants for Design",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
the Applications of Artificial Intelligence in Engineering Problems",
University of Southampton, 2 Volumes, Southampton, Computational
Mechanics, pp445-458
- Allen R.H., 1986,
"Design Guidelines for Expert Systems",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
the Applications of Artificial Intelligence in Engineering Problems",
University of Southampton, 2 Volumes, Southampton, Computational
Mechanics, pp651-660

Allwood R.J., Stewart D.J. and Trimble E.G., 1987,
"Some Experiences from Evaluating Expert System Shell Programs and
some Potential Applications",
In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
techniques to civil and structural engineering, Edinburgh: Civil-Comp,
pp1-6

Allwood R.J., Shaw M.R. and Smith J.L., 1988,
"Building Dampness: Diagnosing the Causes. How the BREDAMP system
was Developed",
In Building Research and Practice, 1988 Vol. 16, No. 1, pp37-42

Alty J.L. and Coombs M.J., 1984,
"Expert Systems Concepts and Examples",
NCC Publications, Manchester

Balachandran M. and Gero J.S., 1987,
"A Knowledge-Based Graphical Interface for Structural Design",
In Sriram D. and Adey R.A. (eds), 1987, "Artificial intelligence in
engineering: tools and techniques", Southampton, Computational
Mechanics Publications, pp335-346

Barlow, Donald F., and Jackson, Peter J., 1988,
"The Release of Alkalis from Pulverised-Fuel Ashes and Ground
Granulated Blastfurnace Slags in the Presence of Portland Cements",
In Cement and Concrete Research, Vol. 18, No. 2, pp235-248

Berry, Dianne C. and Hart, Anna E, 1990,
"Evaluating Expert Systems",
In Expert Systems, November, Vol. 7, No. 4., pp199-208

Blackledge G.F., 1987,
"Concrete Practice",
Cement and Concrete Association, Slough

Blum F.L., 1986
"Computer Errors - Who's Responsible?",
In Lenocker, W. Tracy (ed), 1986, "Computing in civil engineering:",
proceedings of the fourth conference, Boston Marriott Hotel, Copley
Plaza, Boston, Massachusetts, October 27-31 1986, ASCE, New York, USA,
pp715-721

Born, Gary, and John, Robert, 1986
"Expert System Shells for Development and Delivery",
In Knowledge-Based Systems '86, Proceedings of the international
conference, London, July 1986, pp273-283

Bowley M.J., 1979,
"Analysis Of Sulphate-Bearing Soils in Which Concrete Is to be Placed",
Code of Practice CP2/79, Watford, Building Research Establishment

- Bramer M.A., 1986,
 "Expert Systems in Britain: Progress and Prospects",
 In British Computer Society Specialist Group on Expert Systems,
 "Research and development in expert systems III:", proceedings of
 Expert Systems 1986, the Sixth Annual Technical Conference, Brighton,
 15-18 December, 1986, edited by Bramer M.A., Cambridge University
 Press, pp1-12
- Breuker, Joost A., Wielinga Bob, J. and Hayward, Simon A., 1985,
 "Structuring of Knowledge Based Systems Development",
 In ESPRIT 1985, Brussels, status report of continuing work, 2 Volumes,
 Amsterdam, North-Holland, 1986, pp771-784
- Brewer F.D. and Gajski D.D., 1986,
 "An Expert System Paradigm for Design",
 In IEEE, 1986, 23rd Design Automation Conference Proceedings, IEEE
 Computer Society Press, Washington DC, USA, pp62-68
- Brewer, William E., 1987,
 "Design of Concrete Using Electronic Spreadsheets",
 In Stevens, David R., (ed) 1987, "Computer Applications in Structural
 Engineering", Proceedings of the Structures Congress, Orlando Florida,
 August 17-20 1987, ASCE New York, USA, pp445-460
- British Standards Institution, 1973
 "CP 102: October 1973, Protection of Buildings Against Water from the
 Ground",
 British Standards Institute, London
- British Standards Institution, 1975,
 "BS 1377: 1975, Methods of Test for Soil for Civil Engineering
 Purposes",
 British Standards Institute, London
- British Standards Institution, 1981,
 "BS 5328: 1981, Specifying Concrete, including Ready Mixed Concrete"
 British Standards Institute, London
- British Standards Institution, 1985,
 "BS 8110: Part 1: 1985, Structural Use of Concrete",
 British Standards Institute, London
- Brown D.C., 1985,
 "Failure Handling in a Design Expert System",
 In Computer Aided Design Vol. 17, No. 9, 1985, pp436-442
- Brown D.C. and Breau, Robert, 1986,
 "Types of Constraints in Routine Design Problem-Solving",
 In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
 the Applications of Artificial Intelligence in Engineering Problems",
 University of Southampton, 2 Volumes, Southampton, Computational
 Mechanics, pp383-390

Brown D.C. and Chandrasekaran B., 1985,
"Expert Systems for a Class of Mechanical Design Activity",
In Gero J.S. (ed), "Knowledge Engineering in Computer-Aided Design:
IFIP WG 5.2 Working Conference on Expert Systems in Computer-Aided
Design, Budapest, Hungary, 17-19 September, 1984", North-Holland,
Amsterdam, pp259-290

Brown D.C. and Chandrasekaran B., 1986,
"Knowledge and Control for a Mechanical Design Expert System",
In Computer July 1986, pp92-100

Brownston L., Farrell R., Kant E. and Martin N., 1985,
"Programming Expert Systems in OPS5",
Addison Wesley Publishing Company Inc, Reading, Massachusetts USA

Building Research Establishment, 1981,
"Concrete In Sulphate-Bearing Soils and Groundwaters",
Digest 250, Watford, Building Research Establishment

Building Research Establishment, 1982a,
"Durability of Steel In Concrete: Part 1 Mechanism of Protection and
Corrosion",
Digest 263, Watford, Building Research Establishment

Building Research Establishment, 1982b,
"Durability of Steel In Concrete: Part 2 Diagnosis and Assessment of
Corrosion-cracked Concrete",
Digest 264, Watford, Building Research Establishment

Building Research Establishment, 1982c,
Durability of Steel In Concrete: Part 3 The Repair of Reinforced
Concrete",
Digest 265, Watford, Building Research Establishment

Building Research Establishment, 1987a,
"Concrete Part 1: Materials",
Digest 325, Watford, Building Research Establishment

Building Research Establishment, 1987b,
"Concrete Part 2: Specification, Design and Quality Control",
Digest 326, Watford, Building Research Establishment

Building Research Establishment, 1988,
"Alkali Aggregate Reactions in Concrete",
Digest 330, Watford, Building Research Establishment

Bundy, Alan, 1987,
"How to Improve the Reliability of Expert Systems",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems IV:", proceedings of
Expert Systems 1987, the Seventh Annual Technical Conference,
Brighton, 14-17 December, 1987, edited by Moralee D.S., Cambridge
University Press, pp3-17

Burton A.M., Shadbolt N.R., Hedgecock A.P. and Rugg G., 1987,
"A Formal Evaluation of Knowledge Elicitation Techniques For Expert
Systems: Domain 1",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems IV:", proceedings of
Expert Systems 1987, the Seventh Annual Technical Conference,
Brighton, 14-17 December, 1987, edited by Moralee D.S., Cambridge
University Press, pp136-145

Canham, Ian, 1987,
"The Use of Blended Cements to Control Alkali Silica Reaction",
Ph.D. Thesis, Aston University, Birmingham

Canham, Ian, 1988,
"Pore Solution Chemistry and Alkali Silica Reaction"
Research Report, Aston University, Birmingham

Carando, Patricia, 1989,
"Shadow: Fusing Hypertext with AI",
In IEEE Expert 1989, Vol. 4, No. 2, pp65-78

Chandrasekaran B., 1985,
"Generic Tasks in Knowledge-Based Reasoning: Characterising and
Designing, Expert Systems at the 'Right' Level of Abstraction",
In 2nd Conference on Artificial Intelligence Applications. The Engineering
of Knowledge-Based Systems, December 11-12, 1985, edited by Weisbin
C.R., IEEE Computer Society, New York, USA, pp294-300

Chandrasekaran B., 1990,
"Design Problem Solving: A Task Analysis",
In AI Magazine, Winter 1990 Vol. 11, No. 4., pp59-71

Chatterji S., Thaulow N., Jensen A.D. and Christensen P., 1986,
"Mechanisms of Accelerating Effects of NaCl and Ca(OH)₂ on Alkali-Silica
Reaction"
In "Proceedings of the 7th International Conference on Alkali Aggregate
Reaction", Ottawa, 1986, pp115-119

Chehayeb F.S., Connor J.J. and Slater J.H., 1985,
"An Environment for Building Engineering Knowledge-Based Systems"
In Dym C.L. (ed), "Applications of Knowledge-Based Systems to
Engineering Analysis and Design", American Society of Mechanical
Engineers, Winter Meeting, Miami, November 17-22 1985, pp137-145

- Chiang T.Y-L. and Brown D.C., 1987,
 "DSPL Acquirer - A System for the Acquisition of Routine Design Knowledge",
 In Sriram D. and Adey R.A. (eds), 1987, "Artificial intelligence in engineering: tools and techniques", Southampton, Computational Mechanics Publications, pp95-109
- Chidley T.R.E., Elgy J., Marinari, S. and Pooley, M., 1987,
 "An Expert System for Irrigation Engineers",
 In Sriram, D. and Adey, R.A. (eds), 1987, "Knowledge based expert systems for engineering: classification, education and control", Southampton, Computational Mechanics, pp247-253
- Chung P.W.H. and Kumar B., 1987,
 "Knowledge Elicitation Methods: A Case-Study in Structural Design",
 In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence techniques to civil and structural engineering, Edinburgh: Civil-Comp, pp21-26
- Chung P.W.H., Chan N. and Filby I., 1988,
 "Bibliography of AI In Engineering",
 Artificial Intelligence Applications Institute, University of Edinburgh
- Clancey, William J., 1985,
 "Heuristic Classification",
 In Artificial Intelligence 27, 1985, pp289-350
- Clancey, William J., 1989,
 "Viewing Knowledge Bases as Qualitative Models",
 In IEEE Expert 1989 Vol. 4, No. 4, pp9-23
- Clark K.L. and McCabe F.G., 1982,
 "PROLOG: a language for implementing Expert Systems"
 In Machine Intelligence 10, 1982, "Intelligent Systems Practice and Perspectives", editors Hayes J.E., Michie D. and Pao Y-H., Ellis Horwood, Chichester, pp455-472
- Clark J.A. and Mac Randal D., 1991,
 "An Intelligent Front-End for Computer-aided Building Design",
 In Artificial Intelligence in Engineering, Vol. 6, No. 1, pp36-45
- Clifton, James R., Oltikar, Bhalachandra C. and Johnson, Steven K., 1985,
 "Development of Durcon: an Expert System for Durable Concrete Part 1",
 U.S. National Bureau of Standards, Gaithersburg, USA
- Clifton J.R., 1986,
 "Knowledge Based System for Durable Reinforced Concrete",
 In Proceedings Corrosion86 forum, 1986, National Association of Corrosion Engineers Conference, Houston, Texas 17-21 March 1986, NACE, Houston, USA, (paper 60)

Clifton J.R. and Oltikar B.C., 1987,
"Expert System for Selecting Concrete Constituents",
In Ginsburg, Schlomo (ed) 1987, "Computer Applications in Concrete
Technology", ACI SP98, Detroit, USA, pp1-24

Clifton, James, R. and Kaetzel, Lawrence, J., 1988,
"Expert Systems for Concrete Construction",
In Concrete International: Design and Construction, Vol. 10, No. 11,
pp19-24

Clocksinn W.F. and Mellish C.S., 1984,
"Programming in PROLOG",
Springer-Verlag, Berlin, Germany

Cohn A.G., 1985,
"Deep Knowledge Representation Techniques",
In British Computer Society Specialist Group on Expert Systems, "Expert
systems 85: proceedings of the Fifth Technical Conference", University
of Warwick, 17-19 December, 1985, edited by Merry M., Cambridge
University Press, pp299-306

Collins H.M., Green R.H. and Draper R.C., 1985,
"Where's the Expertise? Expert Systems as a Medium of Knowledge
Transfer",
In British Computer Society Specialist Group on Expert Systems, "Expert
systems 85: proceedings of the Fifth Technical Conference", University
of Warwick, 17-19 December, 1985, edited by Merry M., Cambridge
University Press, pp323-334

Collins R.J., 1986,
"Porous Sandstone Aggregates from NW England - A Technical Appraisal
of Prospects for Use In Concrete",
Watford, Building Research Establishment

Concrete Society, 1987,
"Alkali-Silica Reaction - Minimising the Risk Of Damage to Concrete.
Guidance Notes and Model Specification Clauses",
Technical Report No. 30, Concrete Society

Conklin, Jeff, 1987
"Hypertext: An Introduction and Survey"
In Computer, September 1987, pp17-44

Corby, Oliver, 1986,
"Blackboard Architectures in Computer Aided Engineering",
In Artificial Intelligence 1986, Vol. 1, No. 2, pp95-118

Coyne, R.D., 1988,
"Logic Models of Design",
Pitman Publishing, London

- Coyne R.D., 1990,
 "Design Reasoning Without Explanations",
 In AI Magazine Winter 1990, Vol. 11, No. 4, pp72-80
- Coyne R.D., Rosenman M.A., Radford A.D., Balachandran M. and Gero J.S.,
 1990,
 "Knowledge-Based Design Systems",
 Addison-Wesley Publishing Company, Reading, Massachusetts, USA
- Cusens A.R., 1984,
 "The Inter-Relationship between Research, Durability and Design Life",
 In Durability and Design Life of Buildings, 1984,
 London 26-27 November 1984, ICE, London, pp211-224
- Day K.W., 1984,
 "Computer Mix Design",
 In Concrete International: Design and Construction, Vol. 6, No. 9,
 pp26-31
- Deacon, Collin and Dewar, Joe, 1982,
 "Concrete Durability - Specifying more simply and surely by strength",
 In Concrete, February 1982, pp19-21
- Derrington P., 1987,
 "Management of the Design Process",
 In Sriram, D. and Adey, R.A. (eds), 1987, "Knowledge based expert
 systems in engineering: planning and design", Southampton,
 Computational Mechanics, pp19-34
- Diamond, Sidney, 1981,
 "Effects of two Danish Flyashes on Alkali Contents of Pore Solutions of
 Cement-Flyash Pastes",
 In Cement and Concrete Research Vol. 11, No. 3, pp383-394
- Dieng R., Corby O. and Haren P., 1987,
 "Explanatory Knowledge Tools for Expert Systems",
 In Sriram D. and Adey R.A. (eds), 1987, "Artificial intelligence in
 engineering: tools and techniques", Southampton, Computational
 Mechanics Publications, pp321-334
- Dilly R., Beizai V., and Vogt W., 1987,
 "Application of Computers and Spreadsheet Software to Assure Durable
 Concrete",
 In "The Katherine and Bryant Mather International Conference on
 Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA,
 pp555-575
- Dixon J.R. and Simmons M.K., 1983,
 "Computers that Design: Expert Systems for Mechanical Engineers",
 In Computers in Mechanical Design, November 1983, pp10-28

- Dixon J.R., Simmons M.K. and Cohen P.R., 1984,
 "An Architecture for Application of Artificial Intelligence to Design",
 In IEEE, 1984, 21st Design Automation Conference Proceedings, IEEE
 Computer Society Press, Washington DC, USA, pp634-640
- Dobie T.R., 1986,
 "Correlating Water-Soluble Alkalies to Total Alkalies in Cement -
 Considerations for Preventing Alkali-Silica Popouts on Slabs",
 In ASTM STP930, "Alkalies In Concrete", ASTM, Philadelphia, USA, 1986,
 pp46-57
- Dolar-Mantuani, Ludmila, 1983
 "Handbook of Concrete Aggregates",
 Noyes Publications, New Jersey, USA
- Duffy A., 1987,
 "Bibliography - Artificial Intelligence in Design",
 In Artificial Intelligence in Engineering, Vol. 2, No. 3, pp173-179
- Dunstan E.R., Jr., 1981
 "The Effect of Fly Ash on Concrete Alkali-Aggregate Reaction",
 In Cement, Concrete and Aggregates, Vol. 3, No. 2, Winter 1981,
 pp101-104
- Durand, Benoit, Berard, Jean and Soles, James A., 1986
 "Comparison of the Effectiveness of Four Mineral Admixtures to
 Counteract Alkali-Aggregate Reaction"
 In "Proceedings of the 7th International Conference on Alkali Aggregate
 Reaction", Ottawa, 1986, pp30-35
- Dym C.L., Henchey R.P. and Gonick S., 1988,
 "A Knowledge-based System for Automated Architectural Code Checking",
 In C.A.D. 1988 Vol. 20, No. 3, pp137-145
- Englemore R. & Morgan T. (eds), 1986,
 "Blackboard Systems",
 Addison Wesley Publishing Company Inc, Reading, Massachusetts USA
- Evans, Peter and Deehan, Geoff, 1990,
 "The Descent of Mind: the Nature and Purpose of Intelligence",
 Grafton Books, London
- Farbiarz, Josef, Carrasquillo, Ramon L. and Snow, Peter G., 1986
 "Alkali-Aggregate Reactions in Concrete containing Fly Ash"
 In "Proceedings of the 7th International Conference on Alkali Aggregate
 Reaction", Ottawa, 1986, pp55-59
- Fenves S.J., 1985,
 "A Framework for a Knowledge-Based Finite Element Analysis Assistant",
 In Dym C.L. (ed), "Applications of Knowledge-Based Systems to
 Engineering Analysis and Design", American Society of Mechanical
 Engineers, Winter Meeting, Miami, November 17-22 1985, pp1-7

Ferrante, Richard D., 1985,
"The Characteristic Error Approach to Conflict Resolution",
In IJCAI-85: proceedings of the Ninth International Joint Conference on
Artificial Intelligence, [Los Angeles] 2 Volumes, Morgan Kaufmann, Los
Altos, California, USA, pp331-334

Figg J.W., 1973
"Methods of Measuring the Air and Water Permeability of Concrete",
In Magazine of Concrete Research Vol. 25 No. 85, December 1973,
pp213-219

Finn G.A. and Reinschmidt K.F., 1986,
"Micro-computer-Based Engineering Expert Systems",
In Lenocker, W. Tracy (ed), 1986, "Computing in civil engineering:",
proceedings of the fourth conference, Boston Marriott Hotel, Copley
Plaza, Boston, Massachusetts, October 27-31 1986, ASCE, New York, USA,
pp812-827

Fischer, Gerhard and Rathke, Christian, 1988,
"Knowledge-based Spreadsheets",
In AAAI-88, 1988, 7th International Conference on Artificial Intelligence,
August 1988, Saint Paul, Minnesota, USA, pp802-807

Forsyth, Richard (ed), 1989,
"Machine Learning. Principles and Techniques"
Chapman and Hall Computing, London

French W.J., 1986
"A Review of Some Reactive Aggregates from the United Kingdom with
Reference to the Mechanism of Reaction and Deterioration",
In "Proceedings of the 7th International Conference on Alkali Aggregate
Reaction", Ottawa, 1986, pp226-230

Frohnsdorf G.J.C., Clifton J.R., Jennings H.M., Brown P.W. and Struble
L.J., 1988,
"Implications of Computer-Based Simulation Models, Expert Systems,
Data-Bases, and Networks for Advancing Cement Research",
In Ceramic Bulletin Vol. 67, No. 8 1988, pp1368-1371

Frohnsdorf G.J.C., 1990
Private Communication with the author

Fukuda S., 1986,
"Development of an Expert System for Design Support of Oil Storage
Tank",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
the Applications of Artificial Intelligence in Engineering Problems",
University of Southampton, 2 Volumes, Southampton, Computational
Mechanics, pp791-796

- Fukuda S., 1988,
"Codes and Rules and their roles as Constraints in Expert Systems for structural design",
In Adeli, Hojjat (ed), 1988, "Expert systems in construction and structural engineering", London, Chapman and Hall, pp309-323
- Gaines B. and Boose J. (eds), 1988,
"Knowledge Acquisition for Knowledge-Based Systems",
Academic Press Limited, London
- Gaze M. and Nixon P.J., 1983
"The Effect of Pfa upon Alkali-Aggregate Reaction"
In Magazine of Concrete Research Vol. 35, No. 123, June 1983, pp107-110
- Georgeff, Michael and Bonollo, Umberto, 1983,
"Procedural Expert Systems",
In IJCAI-83: proceedings of the Eighth International Joint Conference on Artificial Intelligence, [Karlsruhe] 2 Volumes, Morgan Kaufmann, Los Altos, California, USA, pp151-157
- Gero J.S., Radford A.D., Coyne R.D. and Akiner V.T., 1985,
"Knowledge-Based Computer-Aided Architectural Design",
In Gero J.S. (ed), "Knowledge Engineering in Computer-Aided Design: IFIP WG 5.2 Working Conference on Expert Systems in Computer-Aided Design, Budapest, Hungary, 17-19 September, 1984", North-Holland, Amsterdam, pp57-88
- Gero J.S. and Balachandran M., 1986,
"Knowledge and Design Decision Processes",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on the Applications of Artificial Intelligence in Engineering Problems", University of Southampton, 2 Volumes, Southampton, Computational Mechanics, pp343-354
- Gero J.S., 1988,
"A Basis for Knowledge Based Design in Engineering",
Notes from a Seminar at Herriot-Watt University May 18 1988
- Gero J.S., Maher M.L. and Zhang W., 1988,
"Chunking Structural Design Knowledge as Prototypes",
In Gero J.S. (ed), 1988, "Artificial intelligence in engineering: design", Amsterdam and Southampton, Elsevier co-published with Computational Mechanics Publications, pp3-21
- Gero J.S., 1990,
"Design Prototypes: A Knowledge Representation Schema for Design",
In AI Magazine Winter 1990 Vol. 11, No. 4, pp26-36
- Gero J.S. and Rosenman M.A., 1990,
"A Conceptual Framework for Knowledge-based Design Research at Sydney University's Design Computing Unit",
In Artificial Intelligence in Engineering 1990, Vol. 5, No. 2, pp65-77

Gerstenfield A., Gosling G. and Touretzky D., 1987,
"An Expert System for Managing Cooperating Expert Systems",
In Sriram D. and Adey R.A. (eds), 1987, "Artificial intelligence in
engineering: tools and techniques", Southampton, Computational
Mechanics Publications, pp245-258

Gilbert, G. Nigel, 1987,
"Question and Answer Types",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems IV:", proceedings of
Expert Systems 1987, the Seventh Annual Technical Conference,
Brighton, 14-17 December, 1987, edited by Moralee D.S., Cambridge
University Press, pp162-172

Gilmore, John F. and Pulaski, Kurt, 1985,
"A Survey of Expert System Tools",
In 2nd Conference on Artificial Intelligence Applications. The Engineering
of Knowledge-Based Systems, December 11-12, 1985, IEEE Computer
Society, New York, USA, pp498-502

Gjørsv O.E., 1983
"Durability of Concrete Containing Condensed Silica Fume",
In "The 1st Conference on Fly Ash, Silica Fume, Slag and Other Mineral
By-products in Concrete, Montebello, Canada", edited by Malhotra V.M., 2
Volumes., ACI publication SP-79, pp695-702

Glasser F.P. and Marr J., 1984
"The Effect of Mineral Additives on the Composition of Cement Pore
Fluids",
In British Ceramics Society, 1984, "Chemistry and Chemically Related
Properties of Cement", proceedings of the British Ceramic Society,
pp419-429

Glasser F.P., Luke K. and Angus M.J., 1988
"Modification of Cement Pore Fluid Compositions By Pozzolanic
Additives",
In Cement and Concrete Research Vol. 18, No. 2, pp165-178

Gordon J.E., 1978,
"Structures or Why Things Don't Fall Down",
Penguin Books, Harmondsworth, Middlesex, England

de Greef, Paul and Breuker, Joost, 1985,
"A Case Study in Structured Knowledge Acquisition",
In IJCAI-85: proceedings of the Ninth International Joint Conference on
Artificial Intelligence, [Los Angeles] 2 Volumes, Morgan Kaufmann, Los
Altos, California, USA, pp390-392

Guida G. and Tasso, Carlo, 1983,
"The Issue of Knowledge Acquisition in the Design of Rule-Based Expert Systems",
In IFAC/IFIP, 1983, Artificial intelligence: proceedings of the IFAC Symposium, Leningrad, USSR, 4-6 October 1983, edited by Ponomaryov V.M., Oxford, Published for the International Federation of Automatic Control by Pergamon, 1984, pp31-36

Gutt W. and Collins R.J., 1987
"Sea-dredged Aggregates in Concrete",
Information Paper 7/87, Watford, Building Research Establishment

Harmon, Paul, Maus, Rex and Morrissey, William, 1988,
"Expert Systems Tools and Applications",
John Wiley and Sons, New York, USA

Hart, Anna, 1985,
"Knowledge Elicitation: Issues and Methods",
In C.A.D. 1985 Vol. 17, No. 9, pp455-462

Hayes, Patrick J., 1983,
"The Second Naive Physics Manifesto",
Cognitive Science Technical Report URCS-10,
University of Rochester, New York, USA

Hayes-Roth F., Waterman D. and Lenat D., 1983,
"Building Expert Systems",
Addison Wesley, Publishing Company Inc, Reading, Massachusetts USA

Hickman, Frank, 1986,
"Knowledge Acquisition: the Key to Success for Commercial Expert Systems",
In Knowledge-Based Systems '86, Proceedings of the international conference, London, July 1986, pp205-214

Hillier, W.E. and Freeman, E.M., 1984,
"Harnessing the User's Experience",
In Computer-aided Engineering Journal April 1984, pp74-83

Hobbs D.W., 1986
"Deleterious Expansion of Concrete Due to Alkali-Silica Reaction: Influence of Pfa and Slag",
In Magazine of Concrete Research, Vol. 38, No. 137, December 1986, pp191-205

Hogan F.J. and Meusel J.W., 1981,
"Evaluation for Durability and Strength Development of Ground Granulated Blast Furnace Slag",
In Cement, Concrete and Aggregates, Vol. 3, No. 1, Summer 1981, pp40-52

Hogan F.J., 1985

"The Effect of Blast Furnace Slag Cement on Alkali Aggregate Reactivity: A Literature Review",

In Cement, Concrete and Aggregates, Vol. 7, No. 2, Winter 1985, pp100-107

Hover K.C., 1985,

"Computer Aided Concrete Mix Design",

In Topping B.H.V. (ed), 1985, 2nd International Conference on Civil and Structural Engineering Computing, London, Civil Comp 85, 2 Volumes, Edinburgh, Civil-Comp, pp233-238

Hover K.C., 1987,

"Computer Assistance for Concrete Construction",

In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence techniques to civil and structural engineering, Edinburgh: Civil-Comp, pp125-129

Howe, Adele, Cohen, Paul, Dixon, John and Simmons, Melvin, 1986, "DOMINIC: a Domain Independent Program for Mechanical Engineering Design",

In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on the Applications of Artificial Intelligence in Engineering Problems", University of Southampton, 2 Volumes, Southampton, Computational Mechanics, pp289-299

Hozayen, Hozayen and Haas, Ralph, 1990,

"Expert System Technology for the Evaluation of Pavement Materials",

In OECD, 1990, OECD Workshop on Knowledge-Based Expert Systems in Transportation, Espoo, Finland, 26-28 June 1990

Hu, David, 1987,

"Programmer's Reference Guide to Expert Systems",

Howards W. Sams and Company, Indianapolis, USA

Hudec, Peter P., 1983

"Aggregate Tests - Their Relationships and Significance",

In Durability of Building Materials, No. 1, 1982/1983, pp275-300

Hughes, Sheila, 1986,

"Question Classification in Rule-based Systems",

In British Computer Society Specialist Group on Expert Systems, "Research and development in expert systems III", proceedings of Expert Systems 1986, the Sixth Annual Technical Conference, Brighton, 15-18 December, 1986, edited by Bramer M.A., Cambridge University Press, pp123-137

Idorn, Gunnar M., and Roy, Della M., 1984
"Factors Affecting the Durability of Concrete and the Benefits of Using Blast-Furnace Slag Cement",
In Cement, Concrete and Aggregates, Vol. 6, No. 1, Summer 1984, pp3-10

Idorn G.M. and Roy D.M., 1986,
"Opportunities with Alkalies in Concrete Testing, Research, and Engineering Practice",
In ASTM STP930, "Alkalies In Concrete", ASTM, Philadelphia, USA, 1986, pp5-15

Jackson, Alan H., 1986,
"The Role of Tool Kits in System Development",
In Knowledge-Based Systems '86, Proceedings of the international conference, London, July 1986, pp261-271

St. Johanser, J.T. and Harbidge R.M., 1986
"Validating Expert Systems: Problems and Solutions in Practice",
In Knowledge-Based Systems '86, Proceedings of the international conference, London, July 1986, pp215-229

Kaetzel L.J. and Clifton J.R., 1986,
"Bulletin Board Systems for Feedback to the DURCON Expert Systems: A Description and Reference",
National Bureau of Standards Report 86/3332, Gaithersburg, USA

Kalay Y.E., Swerdloff L.M. and Harfmann A.C., 1987,
"A Knowledge-Based Computational Model of Design",
In Gero J.S. (ed), 1987, "Expert Systems in Computer-Aided Design: IFIP WG 5.2 Working Conference on Expert Systems in Computer-Aided Design, Sydney, February 1987, proceedings", Amsterdam, North-Holland, pp203-227

Klein, Mark and Lu, Stephen C.Y., 1989,
"Conflict Resolution in Cooperative Design",
In Artificial Intelligence in Engineering 1989, Vol. 4, No. 4, pp168-180

Kollek J.J., Varma S.P. and Zaris C., 1986,
"Measurement of OH⁻ Ion Concentrations of Pore Fluids and Expansion Due to Alkali-Silica Reaction in Composite Cement Mortars",
In "Proceedings of the 8th International Congress on the Chemistry of Cement", Rio de Janeiro, 1986, pp183-189

Koskela L., Hynynen R., Kallavuo M., Kahkönen K. and Salokivi J., 1986,
"Expert Systems in Construction: Initial Experiences",
In Proceeding of the International Joint Conference on CAD and Robotics in Architecture and Construction, Proceedings, 1986, Hermes, Paris, France, pp83-92

Koskela L., Hynynen R., Kahkönen K., Salokivi J. and Serén K-J., 1988,
"Expert Systems in Construction: Initial Experiences",
In Pham, Duc Truong (ed), 1988, "Expert systems in engineering", Berlin London, IFS, pp174-188

Kramer G.A., 1987,
"Incorporating Mathematical Knowledge into Design Models",
In Gero J.S. (ed), 1987, "Expert Systems in Computer-Aided Design: IFIP
WG 5.2 Working Conference on Expert Systems in Computer-Aided
Design, Sydney, February 1987, proceedings", Amsterdam, North-Holland,
pp229-265

Kriz, Jiri, 1987,
"Knowledge-based Expert Systems in Industry: Introduction",
In Kriz, Jiri (ed), 1987, "Knowledge-based Expert Systems in Industry",
Ellis Horwood, Chichester, pp11-15

Kuhn T.S., 1970,
"The Structure of Scientific Revolutions",
University of Chicago Press, Chicago, USA

Kumar B., Chung P.W.H. and Topping B.H.V., 1987a,
"Approaches To Fortran-Prolog Interfacing For An Expert System
Environment",
In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
techniques to civil and structural engineering, Edinburgh: Civil-Comp,
pp15-20

Kumar B., Chung P.W.H., Rae R.H. and Topping B.H.V., 1987b,
"A Knowledge-Based Approach to Structural Design",
In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
techniques to civil and structural engineering, Edinburgh: Civil-Comp,
pp79-92

Kumar B. and Topping B.H.V., 1988,
"Issues in the Development of a Knowledge-based System for the
Detailed Design of Structures",
In Gero J.S. (ed), 1988, "Artificial intelligence in engineering: design",
Amsterdam and Southampton, Elsevier co-published with Computational
Mechanics Publications, pp295-314

Kumar B., and Topping B.H.V., 1990,
"Non-Monotonic Reasoning for Structural Design",
In Civil Engineering Systems, Vol 7, No. 4, pp209-218

Kumar B., and Topping B.H.V., 1991,
"Knowledge-Based Processing for Structural Design",
In Proceedings of the Institute of Civil Engineers, Part 1, 90, April,
1991, pp421-446

Laufman, Steven C., DeVaney D. Michael and Whiting, Mark A., 1990,
"A Methodology for Evaluating Potential KBS Applications",
In IEEE Expert December 1990 Vol. 5, No. 6, pp43-61

Lenat, Douglas B., 1982,
"The Nature of Heuristics",
In Artificial Intelligence 19, 1982, pp189-249

Lenat, Doug, Prakash, Mayank and Shepherd, Mary, 1986,
"CYC: Using Common Sense Knowledge to Overcome Brittleness and
Knowledge Acquisition Bottlenecks",
In AI Magazine 6, 1986 pp65-85

Liebowitz, Jay, 1986,
"A Useful Approach for Evaluating Expert Systems",
In Expert Systems April 1986, Vol. 3, No. 2, pp86-96

Lippolt, Ben J., 1990
"Failures and Success of Expert Systems",
In OECD, 1990, OECD Workshop on Knowledge-Based Expert Systems in
Transportation, Espoo, Finland, 26-28 June 1990

Maher M.L. and Fenves S.J., 1985,
"HI-RISE: an Expert System for the Preliminary Structural Design of
High Rise Buildings",
In Gero J.S. (ed), "Knowledge Engineering in Computer-Aided Design:
IFIP WG 5.2 Working Conference on Expert Systems in Computer-Aided
Design, Budapest, Hungary, 17-19 September, 1984", North-Holland,
Amsterdam, pp125-140

Maher M.L., 1988,
"Expert Systems for Structural Design",
In Pham, Duc Truong (ed), 1988, "Expert systems in engineering", Berlin
London, IFS, pp147-161

Maher M.L., Fenves S.J. and Garrett J.H., 1988,
"Expert Systems for Structural Design",
In Adeli, Hojjat (ed), 1988, "Expert systems in construction and
structural engineering", London, Chapman and Hall, pp85-122

Maher M.L., 1990,
"Process Models for Design Synthesis",
In AI magazine Winter 1990, Vol. 11, No. 4, pp49-58

Mathews J.D., 1987a
"Pulverised-Fuel Ash - its use in Concrete: Part 1 Material Properties,
British Standards and Concrete Strength",
Information Paper 11/87, Watford, Building Research Establishment

Mathews J.D., 1987b
"Pulverised-Fuel Ash - its use in Concrete: Part 2 Influences on
Durability",
Information Paper 12/87, Watford, Building Research Establishment

McDermott, John, 1985,
"Extracting Knowledge from Expert Systems",
In IJCAI-83: proceedings of the Eighth International Joint Conference on
Artificial Intelligence, [Karlsruhe] 2 Volumes, Morgan Kaufmann, Los
Altos, California, USA, pp100-107

Miller A.J., 1985,
"DESIGN - A Prototype Expert System for Design Oriented Problem Solving",
In Australian Computer Journal, Vol. 17, No. 1, February 1985, pp20-26

Minsky, Marvin, 1987,
"The Society of Mind",
William Heinemann Ltd., London

Mittal, Sanjay, Dym, Clive L. and Morjaria, Mahesh, 1985,
"PRIDE: An Expert System for the Design of Paper Handling Systems",
In Dym C.L. (ed), "Applications of Knowledge-Based Systems to Engineering Analysis and Design", American Society of Mechanical Engineers, Winter Meeting, Miami, November 17-22 1985, pp99-116

Mittal, Sanjay, Dym, Clive L. and Morjaria, Mahesh, 1986,
"PRIDE: An Expert System for the Design of Paper Handling Systems",
In Computer July 1986, pp102-114

Muir, Robin, 1987,
"The Integrated User",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems IV:", proceedings of Expert Systems 1987, the Seventh Annual Technical Conference, Brighton 14-17, December, 1987, edited by Moralee D.S., Cambridge University Press, pp126-135

Muller, Robert C., 1986,
"New Directions in Expert Systems",
In Knowledge-Based Systems '86, Proceedings of the international conference, London, July 1986, pp325-333

Neches R., Swartout W.R. and Moore J., 1985,
"Explainable (and Maintainable) Expert Systems",
In IJCAI-85: proceedings of the Ninth International Joint Conference on Artificial Intelligence, [Los Angeles] 2 Volumes, Morgan Kaufmann, Los Altos, California, USA, pp382-389

Neveu, Bertrand and Breau, Robert, 1986,
"Smeci: An Expert System for Civil Engineering Design",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on the Applications of Artificial Intelligence in Engineering Problems", University of Southampton, 2 Volumes, Southampton, Computational Mechanics, pp317-326

Neville A.M., 1981,
"Properties of Concrete",
Longman Scientific and Technical (3rd edition) 1981

- Newman K., 1987,
 "Labcrete, Realcrete, and Hypocrete - Where we can Expect the Next Major, Durability Problems",
 In "The Katherine and Bryant Mather International Conference on Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA, pp1259-1283
- Nguyen T.A., Perkins W.A., Laffey T.J. and Pecora D., 1985,
 "Checking an Expert Systems Knowledge Base for Consistency and Completeness",
 In IJCAI-85: proceedings of the Ninth International Joint Conference on Artificial Intelligence, [Los Angeles] 2 Volumes, Morgan Kaufmann, Los Altos, California, USA, pp375-378
- Nixon P.J., Collins R.J. and Rayment P.L., 1979,
 "The Concentration of Alkalis by Moisture Migration In Concrete - A Factor Influencing Alkali Aggregate Reaction",
 In Cement and Concrete Research Vol. 9, No. 4, pp417-423
- Nixon P.J., Page C.L., Bollinghaus R., and Canham I., 1985,
 "The Effect of a Pfa with a High Total Alkali Content on Pore Solution Composition and Alkali Silica Reaction",
 In Magazine of Concrete Research, Vol. 38, No. 134, March 1986, pp30-35
- Nixon P.J., Canham I., Page C.L. and Bollinghaus R., 1986,
 "Sodium Chloride and Alkali-Aggregate Reaction",
 In "Proceedings of the 7th International Conference on Alkali Aggregate Reaction", Ottawa, 1986, pp110-114
- Nixon P.J. and Page C.L., 1987,
 "Pore Solution Chemistry and Alkali Aggregate Reaction",
 In "The Katherine and Bryant Mather International Conference on Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA, pp1833-1862
- Oberholster R.E. and Davies G., 1986,
 "The Effect of Mineral Admixtures on the Alkali Silica Expansion of Concrete Under Outdoor Exposure Conditions",
 In "Proceedings of the 7th International Conference on Alkali Aggregate Reaction", Ottawa, 1986, pp60-65
- O'Brien T., Cather R. and Figg J., 1987,
 "Concrete Durability: The Interface Between Research and Practice",
 In "The Katherine and Bryant Mather International Conference on Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA, pp255-264
- O'Neill, Margaret and Morris, Anne, 1989,
 "Expert Systems in the United Kingdom Evaluation of development methodologies",
 In Expert Systems April 1989 Vol. 6, No. 2, pp90-99

- Oxman Rivka, 1990,
 "Design Shells: A Formalism for Prototype Refinement in
 Knowledge-based Design Systems",
 In Artificial Intelligence in Engineering 1990, Vol. 5, No. 1, pp2-8
- Page C.L., 1986,
 "Corrosion and Service Life Prediction for Reinforced Concrete",
 In Magazine of Concrete Research Vol. 38, No. 137, December 1986, pp174
- Palmer, Richard N. and Mar, Brian W., 1988,
 "Expert Systems for Civil Engineering Applications",
 In Civil Engineering Systems Vol. 5, No. 4, December 1988, pp170-180
- Parrott L.J., 1986,
 "A Review of Carbonation in Reinforced Concrete",
 Cement and Concrete Association, Slough
- Partridge D. and Wilks Y., 1987,
 "Does AI Have a Methodology which is Different from Software
 Engineering?",
 In Artificial Intelligence Review (1987) 1, pp111-120
- Paterson A.C., 1984,
 "The Structural Engineer in Context",
 In The Structural Engineer, Vol. 62A, No. 11, November, 1984, pp335-342
- Pecora D., Zumsteg J.R. and Crossman F.W., 1985,
 "An Application of Expert Systems to Composite Structural Design and
 Analysis",
 In Dym C.L. (ed), "Applications of Knowledge-Based Systems to
 Engineering Analysis and Design", American Society of Mechanical
 Engineers, Winter Meeting, Miami, November 17-22 1985, pp135-147
- Pereira L.M. and Oliveira E., 1983
 "Prolog for Expert Systems: a Case Study"
 In IFAC/IFIP, 1983, Artificial intelligence: proceedings of the IFAC
 Symposium, Leningrad, USSR, 4-6 October 1983, edited by Ponomaryov
 V.M., Oxford, Published for the International Federation of Automatic
 Control by Pergamon, 1984, pp73-82
- Pomeroy D., 1987,
 "Concrete Durability: From Basic Research to Practical Reality",
 In "The Katherine and Bryant Mather International Conference on
 Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA,
 pp111-130
- Popovics S., 1987,
 "A Classification of the Deterioration of Concrete Based on Mechanism",
 In "The Katherine and Bryant Mather International Conference on
 Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA,
 pp131-142

- Powell M., 1989,
 "Spring Arrives for Artificial Intelligence",
 In Computer Weekly Jan 19 1989, pp25
- Preece, Alun D., 1990,
 "Towards a methodology for Evaluating Expert Systems",
 In Expert Systems, November 1990, Vol. 7, No. 4, pp215-223
- Quinlan J.R., 1983,
 "Consistency and Plausible Reasoning",
 In IJCAI-83: proceedings of the Eighth International Joint Conference on
 Artificial Intelligence, [Karlsruhe] 2 Volumes, Morgan Kaufmann, Los
 Altos, California, USA, pp137-144
- Quinlan J.R., 1986,
 "INFERNO: a Cautious Approach to Uncertain Inference",
 In Klahr, Phillip and Waterman, Donald, A. (eds), "Expert Systems
 Techniques, Tools and Applications", Addison Wesley Publishing, Reading,
 Massachusetts, USA, pp350-390
- Rada, Roy, 1990,
 "Expert Systems in the UK",
 In IEEE Expert 1990 Vol. 5, No. 4, pp12-17
- Rajkumar C., Kaur S. and Mullick A.K., 1987,
 "Distress Investigations and Computer Reasoning,"
 In 4th International Conference on Durability of Building Materials, and
 components, Singapore 4-6 November 1987, Pergamon Press, Oxford, 2
 Volumes, pp427-432
- Rasdorf W.J. and Parks L.M., 1987,
 "Natural Language Prototypes for Analysing Design Standards",
 In Sriram D. and Adey R.A. (eds), 1987, "Artificial intelligence in
 engineering: tools and techniques", Southampton, Computational
 Mechanics Publications, pp147-160
- Rasdorf W.J. and Wang T.E., 1987,
 "SPIKE: A Generic Design Standards Processing Expert System",
 In Sriram, D. and Adey, R.A. (eds), 1987, "Knowledge based expert
 systems in engineering: planning and design", Southampton,
 Computational Mechanics, pp241-257
- Rasdorf, William J. and Watson, Bruce R., 1988,
 "A Knowledge-based Approach to Engineering Information Retrieval and
 Management",
 In Adeli, Hojjat (ed), 1988, "Expert systems in construction and
 structural engineering", London, Chapman and Hall, pp267-289
- Rasdorf W.J. and Wang T.E., 1988,
 "Generic Design Standards Processing In an Expert System
 Environment",
 In Journal of Computing In Civil Engineering Vol. 2, No. 1 1988, pp68-87
 (Discussed In Vol. 3, No. 2, pp205-206)

- Reboh, René, 1983,
 "Extracting Useful Advice From Conflicting Expertise",
 In IJCAI-83: proceedings of the Eighth International Joint Conference on
 Artificial Intelligence, [Karlsruhe] 2 Volumes, Morgan Kaufmann, Los
 Altos, California, USA, pp145-150
- Rodway, Lloyd E., 1985,
 "Durability of Concrete",
 In Cement, Concrete and Aggregates Vol. 7, No. 1 summer 1985, pp43-48
- Roesner, Hannelore, 1988,
 "Expert Systems for Commercial Use",
 In Savory, Stuart, (ed), 1988, "Artificial Intelligence and Expert
 Systems", Ellis Horwood, Chichester, pp35-51
- Rosenman M.A. and Gero, J.S., 1985,
 "Design Codes as Expert Systems",
 In C.A.D. Vol. 17, No. 9 1985, pp399-409
- Rosenman M.A., Gero J.S., Hutchinson P.J. and Oxman R., 1986a,
 "Expert System Applications in Computer-Aided Design",
 In C.A.D. 1986 Vol. 18, No. 10, pp546-551
- Rosenman M.A., Gero J.S. and Oxman R., 1986b,
 "An Expert System for Design Codes and Design Rules",
 In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
 the Applications of Artificial Intelligence in Engineering Problems",
 University of Southampton, 2 Volumes, Southampton, Computational
 Mechanics, pp745-758
- Rosenman M.A., Gero J.S., Hutchinson P.J. and Oxman R., 1986c,
 "Expert System Applications in Computer-Aided Design",
 In Smith, Alison (ed), 1986, "CAD 86, London, Knowledge engineering and
 computer modelling in CAD:", proceedings of CAD '86 London 25
 September 1986, London, Butterworths, pp218-225
- Rosenman, Michael A., Balachandran, Bala M. and Gero, John S., 1989,
 "The Place of Expert Systems in Civil Engineering",
 In Civil Engineering Systems 1989 Vol. 6, Nos. 1-2, pp11-21
- Rowlinson, Steve, 1987,
 "Expert Systems Development - Problems in Practice",
 In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
 techniques to civil and structural engineering, Edinburgh: Civil-Comp,
 pp7-13
- Rubin, Tony, (ed), 1988,
 "User Interface Design for Computer Systems",
 Ellis Horwood, Chichester

- Rychener, Michael D., 1988,
 "Research in Expert Systems for Engineering Design",
 In Rychener, Michael D. (ed), 1988b, "Expert Systems for Engineering Design", Academic Press Inc, London, pp1-34
- Sacks, Russel and Buyukozturk, Oral, 1987,
 "Expert Interactive Design of R/C Columns Under Biaxial Bending",
 In Journal of Computing in Civil Engineering Vol. 1, No. 2 April 1987, pp69-81
- Sales M. and Topping B.H.V., 1985,
 "Legal and Ethical Issues for Civil and Structural Engineering Computer Users",
 In Topping B.H.V. (ed), 1985, 2nd International Conference on Civil and Structural Engineering Computing, London, Civil Comp 85, 2 Volumes, Edinburgh, Civil-Comp, pp11-22
- Samarin A., 1987,
 "Methodology of Modeling for Concrete Durability",
 In "The Katherine and Bryant Mather International Conference on Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA, pp1205-1225
- Saouma V.E., Doshi S.M. and Pace M., 1989,
 "Architecture of an expert-system-based code-compliance checker",
 In Engineering Applications of AI Vol. 2 March 1989, pp49-56
- Sauers R. and Walsh R., 1983,
 "On the Requirements of Future Expert Systems",
 In IJCAI-83: proceedings of the Eighth International Joint Conference on Artificial Intelligence, [Karlsruhe] 2 Volumes, Morgan Kaufmann, Los Altos, California, USA, pp110-115
- Savory, Stuart, 1986,
 "What is an Explanation?",
 In Knowledge-Based Systems '86, Proceedings of the international conference, London, July 1986, pp231-238
- Shafer, Dan, 1989,
 "Designing Intelligent Front Ends For Business Software",
 John Wiley and Sons, New York, USA
- Shapiro, Alen D., 1987,
 "Structured Induction In Expert Systems ",
 Turing Institute Press in association with Addison-Wesley, Wokingham, England
- Shaw M.R., 1989,
 "Expert Systems and the Construction Industry",
 Information Paper 4/89, Watford, Building Research Establishment

- Shaw M.R., 1990,
Private communication with the author
- Shirley D.E., 1985,
"Introduction to Concrete",
Cement and Concrete Association, Slough
- Simmons, M.K., 1984,
"Artificial Intelligence for Engineering Design",
In Computer-aided Engineering Journal, April 1984, pp75-83
- Skwara, Andrew, 1987,
"Computer Spreadsheet Applications in Structural Engineering Design",
In Stevens, David R., (ed) 1987, "Computer Applications in Structural Engineering", Proceedings of the Structures Congress, Orlando Florida, August 17-20 1987, ASCE New York, USA, pp412-423
- Slade, Stephen, 1991,
"Case-Based Reasoning: A Research Paradigm",
In AI Magazine, Vol. 12, No. 1, Spring 1991, pp42-55
- Sloman, A., 1984,
"Why We Need Many Knowledge Representation Formalisms",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems:", proceedings of the Fourth Technical Conference of the British Computer Society Specialist Group on Expert Systems: University of Warwick, 18-20 December 1984, edited by Bramer M.A., Cambridge University Press, pp163-183
- Smith R.L. and Raba Jr. C.F., 1986,
"Recent Developments in the Use of Fly Ash to Reduce Alkali-Aggregate Reaction",
In ASTM STP930, "Alkalis In Concrete", ASTM, Philadelphia, USA, 1986, pp58-68
- Smithers T., Conkie A., Doheny J., Logan B., Millington K., and Xi Tang M., 1990
"Design as Intelligent Behaviour: an AI in Design Research Programme"
In Artificial Intelligence in Engineering 1990, Vol. 5, No. 2, pp78-109
- Somerville G., 1984,
"The Interdependence of Research, Durability and Structural Design - Concrete",
In Durability and Design Life of Buildings, 1984,
London 26-27 November 1984, ICE, London, pp233-250
- Somerville G., 1986,
"The Design Life of Concrete Structures",
In The Structural Engineer Vol. 64, No. 2 February 1986, pp60-71

Sriram D., 1986,
"Destiny: A Model for Integrated Structural Design",
In Smith, Alison (ed), 1986, "CAD 86, London, Knowledge engineering and
computer modelling in CAD:", proceedings of CAD '86 London 25
September 1986, London, Butterworths, pp226-235

Sriram D. and Maher M.L., 1986,
"The Representation and Use of Constraints in Structural Design",
In Sriram D. and Adey R.A. (eds), 1986, "1st International Conference on
the Applications of Artificial Intelligence in Engineering Problems",
University of Southampton, 2 Volumes, Southampton, Computational
Mechanics, pp355-366

Sriram D., 1987a,
"All-Rise: A Case-study in Constraint-Based Design",
In Artificial Intelligence in Engineering 1987, Vol. 2, No. 4, pp186-203

Sriram D., 1987b,
"Knowledge-based approaches for Structural Design",
Southampton, Computational Mechanics

Sriram, Duvvuru, Stephenopoulos, George, Serrano, Robert and
Navinchandra, Dundee, 1989,
"Knowledge-Baed System Applications in Engineering Design: Research at
MIT",
In AI Magazine, Vol. 10, No. 3, Fall, 1989, pp79-96

Stahl F.I., Wright R.N., Fenves S.J. and Harris J.R. 1983,
"Expressing Standards for Computer-Aided Building Design",
In Computer-Aided Design, Vol 15, No 6., November 1983, pp329-334

Stark D. and Bhatti M.S.Y., 1986,
"Alkali-Silica Reactivity: Effect of Alkali in Aggregate on Expansion",
In ASTM STP930, "Alkalis In Concrete", ASTM, Philadelphia, USA, 1986,
pp16-30

Steels L., 1987,
"Second Generation Expert Systems",
In British Computer Society Specialist Group on Expert Systems,
"Research and development in expert systems III", proceedings of the
Sixth Technical Conference of the British Computer Society Specialist
Group on Expert Systems, Brighton, 15-18 December 1986, edited by
Bramer, M.A., Cambridge University Press, pp175-183

Steels L., 1990,
"Components of Expertise",
In AI Magazine 1990, Vol. 11, No. 2, pp28-49

Sterling L. and Shapiro E., 1986,
"The Art Of Prolog",
MIT Press, Cambridge, Massachusetts, USA

- Stewart, Alastair, 1986,
"BREXBAS - An Expert System",
In Building, 1986, Vol. 251, No. 7475, pp53
- Stone D. and Wilcox D.A., 1987,
"Intelligent Systems for the Formulation of Building Regulations",
In 4th International Symposium on Robotics and AI in Building and
Construction, Haifa Israel 23-25 June 1987, Technion, Haifa, 2 Volumes,
pp740-761
- Struble L. and Diamond S., 1986,
"Influence of Cement Alkali Distribution on Expansion Due to Alkali-Silica
Reaction",
In ASTM STP930, "Alkalis In Concrete", ASTM, Philadelphia, USA, 1986,
pp31-45
- Sturup V., Hooton R., Mukherjee P. and Charmichael T., 1987,
"Evaluation and Prediction of Concrete Durability - Ontario Hydro's,
Experience",
In "The Katherine and Bryant Mather International Conference on
Concrete Durability", Atlanta, 1987, Noyes Publications, New Jersey, USA,
pp1121-1154
- Swaffield, Gail and Knight, Brian, 1990,
"Applying Systems Analysis Techniques to Knowledge Engineering",
In Expert Systems May 1990 Vol. 7, No. 2, pp82-93
- Swamy R.N. and Al-Asali M.M., 1986,
"Influence of Alkali-Silica Reaction on the Engineering Properties of
Concrete",
In ASTM STP930, "Alkalis In Concrete", ASTM, Philadelphia, USA, 1986,
pp69-88
- Taylor, John, M., 1986,
"Expert Systems: Where Do We Go From Here?",
In Knowledge-Based Systems '86, Proceedings of the international
conference, London, July 1986, pp313-324
- Taylor H.F.W., 1987,
"A Method for Predicting Alkali Ion Concentrations in Cement Pore
Solutions",
In Advances in Cement Research Vol. 1, No. 1 October 1987, pp5-17
- Taylor N.K. and Corlett E.N., 1987,
"An Expert System Which Constrains Designs",
In Artificial Intelligence in Engineering 1987, Vol. 2, No. 2, pp72-75
- Taylor, William A., 1988;
"What Every Engineer should know about Artificial Intelligence",
The MIT Press, Cambridge, Massachusetts, USA

- Teychenné D.C., Franklin R.E. and Erntroy H.C., 1988,
"Design of Normal Concrete Mixes",
Building Research Establishment 1988
- Tenoutasse N. and Marion A.M., 1986,
"Influence of Fly Ash in Alkali-Aggregate Reaction",
In "Proceedings of the 7th International Conference on Alkali Aggregate
Reaction", Ottawa, 1986, pp44-48
- Thewalt, Chris and Moskowitz, David, 1990,
"Automated Text Generation For Building Standards",
In Journal of Computing in Civil Engineering, Vol. 4, No. 1 January 1990,
pp20-36
- Tong C., 1987a,
"AI in Engineering Design",
In Artificial Intelligence in Engineering Vol. 2, No. 3 1987, pp130-132
- Tong C., 1987b,
"Toward an Engineering Science of Knowledge-Based Design",
In Artificial Intelligence in Engineering Vol. 2, No. 3 1987, pp133-166
- Trimble, Geoffrey, 1988,
"Knowledge Acquisition for Expert Systems in Construction",
In Adeli, Hojjat (ed), 1988, "Expert systems in construction and
structural engineering", London, Chapman and Hall, pp297-308
- Turner, Mike, 1985,
"Expert Systems: A Management Guide",
PA Computers and Telecommunications, England
- Vadera, Sunil (ed), 1989,
"Expert System Applications",
Sigma Press, Wilmslow, United Kingdom
- Vedder, Richard G., 1989,
"PC-Based Expert Systems Shells: Some desirable and less desirable,
Characteristics",
In Expert Systems February 1989 Vol. 6, No. 1, pp28-42
- Vowler J., 1989,
"Putting knowledge into machines"
(interview with Ed Feigenbaum), In Computer Weekly Jan 12 1989,
pp24-25

Wager D.M., 1987,
"Can Expert Systems Help the Construction Industry?",
In Topping B.H.V. (ed), 1987, Application of Artificial Intelligence
techniques to civil and structural engineering, Edinburgh: Civil-Comp,
pp27-29

Wenzel T.H., 1987,
"Use of Spreadsheet Programs in Teaching Reinforced Concrete Design",
In Ginsburg, Schlomo (ed) 1987, "Computer Applications in Concrete
Technology", ACI SP98, Detroit, USA, pp149-157

Westerberg A., Grossman I., Talukdar S., Prinz F., Fenves S., and Maher
M.L., 1990
Application of AI in Design Research at Carnegie-Mellon University's
EDRC"
In Artificial Intelligence in Engineering 1990, Vol. 5, No. 2, pp110-124

Wilkins A.J. and Elgy J., 1991a,
"Application of Expert Systems Technology in the Design of Durable
Concrete",
In Magazine of Concrete Research, Vol. 43, No. 154, March 1991, pp65-69

Wilkins A.J. and Elgy J., 1991b,
"DEX: An Expert System for the Design of Durable Concrete",
In Civil Comp 91, proceedings of the Second International Conference on
the Application of Artificial Intelligence Techniques to Civil and
Structural Engineering, 3-5th September 1991, Oxford, (to be published)

Wilkins A.J. and Tillotson H.T., 1991,
"Enhancement of Highway Maintenance Management Systems Through
Optimisation of Critical Deterioration Levels",
In Civil Comp 91, proceedings of the Second International Conference on
the Application of Artificial Intelligence Techniques to Civil and
Structural Engineering, 3-5th September 1991, Oxford, (to be published)

Yamamoto C. and Makita M., 1986,
"Effect of Ground Granulated Blast Furnace Slag Admixture and
Granulated or Air-Cooled Blast Furnace Slag Aggregate on
Alkali-Aggregate Reactions and their Mechanisms",
In "Proceedings of the 7th International Conference on Alkali Aggregate
Reaction", Ottawa, 1986, pp49-54

Zumsteg J.R. and Flaggs D.L., 1985,
"Knowledge-Based Analysis and Design Systems for Aerospace
Structures",
In Dym C.L. (ed), "Applications of Knowledge-Based Systems to
Engineering Analysis and Design", American Society of Mechanical
Engineers, Winter Meeting, Miami, November 17-22 1985, pp62-80

Appendix A: Hydroxyl Ion Concentration Prediction Algorithm

Full Procedure covering both cases of mixes with or without cement replacement with pfa. Adapted from the algorithm given by Taylor, [1987]

0. Prompt for time in days (T in formulae to follow)

Majority of steps involve calculation of equation(5) with appropriate constants:

$$F = 1 - \exp[-k_2 * (T-k_3)^{k_1}]$$

1.1. Prompt for total weight of Na₂O = na2o_total

Prompt for weight of Na₂O present as sulphate = na2o_sulph
or use default value

$$\text{na2o_rest} = \text{na2o_total} - \text{na2o_sulph}$$

1.2. Ask if the distribution of remaining Na₂O is known

If so prompt for each in turn (and check they add up to 1.0!)

If not defaults (PHASE_fract) are given in Table 1¹.

¹ There is a slight simplification here in that it is not mathematically necessary to calculate the weight in each fraction by multiplying each fraction by na2o_rest as Taylor says. After simplification this multiplication can be performed once only at a later stage.

	Alite	Belite	Aluminate	Ferrite
Na ₂ O content (%)	0.17	0.27	1.05	0.06
K ₂ O content (%)	0.14	0.79	0.97	0.07
Na ₂ O fraction	0.44	0.17	0.36	0.03
K ₂ O fraction	0.29	0.41	0.27	0.03

Table 1: Assumed contents of Na₂O and K₂O in individual clinker phases and fractions of the total non-sulphate Na₂O and K₂O assumed to occur in those phases.

1.3. Apply equation(5) to each phase in turn using the constants k_1 , k_2 and k_3 given in Table 2 (along with the necessary modifications for the effect of pfa replacement) to calculate each PHASE_F.

Multiply each result by $(1 - (R / 100))$ where R is the percentage replacement of cement by pfa.

Calculate for each phase: $PHASE_na2o = PHASE_fract * PHASE_F$

1.4. Calculate $na2o_released = na2o_sulph +$

$$na2o_rest * (alite_na2o + \\ belite_na2o + \\ alluminate_na2o + \\ ferrite_na2o)$$

1.5. Repeat steps 1.1 to 1.4 for K₂O with appropriate defaults

Quantity represented by F	k ₁	k ₂	k ₃
Fractions hydrated: alite	0.25	0.70	0.90
belite	0.46	0.12	0
aluminate	0.28	0.77	0.90
ferrite	0.26	0.55	0.90
Bound Water			
(fraction of 31.6 g / 100 g cement)	0.25	0.69	0.90
Ca in C-S-H and AFm phases			
(fraction of 791.0 mmole / 100 g cement)	0.25	0.56	0.90

Table 2: OPC hydration: values of the constants assumed in the empirical Equation (5), for age T in days.

$$F = 1 - \exp[-k_2 * (T-k_3)^{k_1}]$$

2. If R > 0 Prompt for weight fraction of glass in pfa = G

 Calculate A = G * R where R is as above

 Calculate pfa_react = A * equation(5)

3. If R > 0 Prompt for total Na₂O & K₂O in pfa (= Nt)

 Ask if fractions present as sulphates are known (= Ns)

 If so prompt for them otherwise use default values

4. Calculate $pfa_na2o_released_g = ((Ns * R) + ((Nt - Ns) * A / G)) / 100$

$total_na2o_released_g = pfa_na2o_released_g + na2o_released_g$

Calculate $total_na2o_released_m = total_na2o_released_g * 1000 / 62^2$

5. If $R > 0$ Repeat 4. for K_2O .

6. Calculate volume of bound water: $V_b = V_{b1} + V_{b2}$

$V_{b1} = 31.6 * (1 - R/100) * equation(5)$

$V_{b2} = 0.17 * pfa_react$ (step 2)

Calculate volume of pore solution: $V = (100 * w/b) - V_b$

where w/b is the water/binder ratio

7. If $R > 0$ Calculate $P = (1 - R/100) * R^{0.09} * equation(5)$

Else Calculate $P = equation(5)^3$

2 As this is a figure in grammes it is necessary to convert it to a figure in mmoles:

divide by the molecular weight of Na_2O ($23 * 2 + 16 = 62$)
multiply by 1000

3 Intuitively, this should not be necessary. It should be possible to use the 'with pfa' formulae in cases without pfa substitution with all terms involving $R (= 0)$ simplifying to the formulae given for cases without substitution

8. Calculate concentration of Na⁺ in the pore solution.

$$C_{Na^+} = M_r / (V + (b * P) + (b' * A))$$

Where b = 31.0 & b' = 3.0

M_r is total_{na2o}_released

A is pfa_{react}

9. Repeat step 8. for K⁺ where b = 20.0 & b' = 3.3

10. Calculate CSO₄⁻⁻ = 0.06 * (CK⁺ + CNa⁺)²

$$\text{Calculate } COH^- = C_{Na^+} + C_{K^+} - (2 * CSO_4^{--})$$

COH⁻ is the concentration of Hydroxyl ion in the pore solution.

Appendix B: Reviews of Expert System Shells

1st Class, [Adeli, 1988] [Vedder, 1989] [Palmer & Mar, 1988]

Advisor-2, [Vadera, 1989]

APES, [Allwood et al, 1987]

ART, [Adeli, 1988] [Gilmore & Pulaski, 1985] [Vadera, 1989] [Jackson, 1986]

Duck, [Gilmore & Pulaski, 1985]

EMYCIN, [Adeli, 1988]

ENVISAGE, [Allwood et al, 1987] [Born & John, 1986] [Chidley et al 1987]

ES/P ADVISOR, [Allwood et al, 1987] [Koskela et al, 1986, 1988]

ESE, [Adeli, 1988]

EX-TRAN7, [Allwood et al, 1987]

EXPERT, [Adeli, 1988]

EXPERT-EASE, [Adeli, 1988]

EXSYS, [Palmer & Mar, 1988] [Vedder, 1989]

GURU, [Palmer & Mar, 1988] [Vedder, 1989] [Koskela et al, 1986, 1988]

INSIGHT 2, [Koskela et al, 1986, 1988] [Adeli, 1988] [Palmer & Mar, 1988]

K-craft, [Gilmore & Pulaski, 1985]

KAS, [Adeli, 1988]

KEE, [Adeli, 1988] [Gilmore & Pulaski, 1985]

KES, [Allwood et al, 1987] [Palmer & Mar, 1988] [Gilmore & Pulaski, 1985]

M.1., [Adeli, 1988] [Gilmore & Pulaski, 1985] [Saouma et al, 1989]

MicroExpert, [Shaw, 1989]

P.R.O. [Vadera, 1989]

Personnal Consultant Easy, [Vedder, 1989]

Personnal Consultant Plus, [Palmer & Mar, 1988] [Vedder, 1989]

ROSIE, [Adeli, 1988]

RuleMaster, [Gilmore & Pulaski, 1985] [Adeli, 1988]

S.1., [Adeli, 1988] [Gilmore & Pulaski, 1985]

SAGE, [Allwood et al, 1987] [Chidley et al 1987]

SAVOIR, [Allwood et al, 1987] [Shaw, 1989]

TESS, [Allwood et al, 1987]

The Deciding Factor, [Adeli, 1988]

TWAICE [Savory, 1986]

VP-EXPERT, [Palmer & Mar, 1988]

Xi, [Allwood et al, 1987]