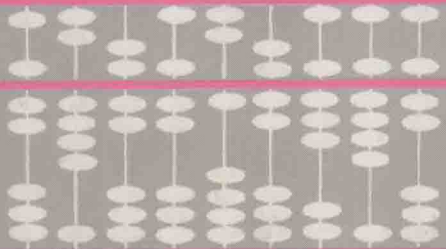
KNOWLEDGE SYSTEMS DESIGN

JOHN K. DEBENHAM

Advances in Computer Science Series



Richard P. Brent – Editor

Knowledge Systems Design

John K. Debenham

School of Computing Sciences
University of Technology, Sydney

PRENTICE HALL

New York London Toronto Sydney Tokyo

Printed and bound in Australia by
Impact Printing, Brunswick, Victoria

3 4 5 92 91 90

ISBN 0-13-517129-6 (hardback)
ISBN 0-13-516428-1 (paperback)

**National Library of Australia
Cataloguing-in-Publication Data**

Debenham, J.K. (John K.).
Knowledge systems design.

Includes index.
ISBN 0-13-516428-1.
1. Expert systems (Computer science). 2. Data base design. I. Title.
006.3'3

**Library of Congress
Cataloguing-in-Publication Data**

Debenham, J.K.
Knowledge systems design.

(Prentice Hall advances in computer science series)
Bibliography: p.
Includes index.
1. Expert systems (Computer science). 2. Data base design.
I. Titles. II. Series.
QA76.76.E95D43 1989 006.3'3 88-30655
ISBN 0-13-517129-6

Prentice Hall, Inc., Englewood Cliffs, New Jersey
Prentice Hall of Australia Pty Ltd, Sydney
Prentice Hall Canada, Inc., Toronto
Prentice Hall Hispanoamericana, SA, Mexico
Prentice Hall of India Private Ltd, New Delhi
Prentice Hall International, Inc., London
Prentice Hall of Japan, Inc., Tokyo
Prentice Hall of Southeast Asia Pty Ltd, Singapore
Editora Prentice Hall do Brasil Ltda, Rio de Janeiro

PRENTICE HALL

A division of Simon & Schuster

Knowledge Systems Design

PRENTICE HALL
ADVANCES IN COMPUTER SCIENCE SERIES
Editor: Richard P. Brent

Gough & Mohay	*Modula-2: A Second Course in Programming*
Hille	*Data Abstraction and Program Development using Pascal*
	Data Abstraction and Program Development using Modula-2
Rankin	*Computer Graphics Software Construction*
Seberry & Pieprzyk	*Cryptography: Introduction to Computer Science*

Contents

Preface

This is a book about design - the design of knowledge-based systems. It is written for those who have an interest in building expert systems and deductive database systems. It is written in particular for those who have an interest in building *maintainable* expert systems and *maintainable* deductive database systems. The text, therefore, will appeal to computer professionals, postgraduate students and senior undergraduate students.

The text contains a detailed description of a method for designing expert knowledge-based systems and deductive database systems. This method is complete and extends from initial knowledge acquisition to knowledge base implementation and knowledge base maintenance. It is claimed that the method presented is suitable for team work. This method represents and analyzes the knowledge in a way that is quite rigorous and yet is independent of any particular expert system shell or computer language.

The majority of expert systems constructed in large corporations during the mid- to late 1980s have not been fully integrated with the central computing resources, in particular with the corporate databases. There are two main reasons for this; first, the lack of suitable, reliable, integrated knowledge base management system software, and second, the lack of design methodologies which permit "knowledge" to be gathered, modeled, analyzed, normalized and implemented with the same degree of precision as is typically applied to "information". This book addresses the second issue and its text should be of prime interest to those large corporations with a commitment to building up a large, integrated, corporate knowledge base.

The work reported here began in 1982 with a joint research project with Mike McGrath of Telecom Australia. At that time we rebuilt the substantial Telecom Telephone Accounting System for the Sydney region as a deductive database using logic programming. This was a large database of some 250 mega-bytes of "information" and a substantial and complex set of rules or "knowledge". It is believed to be the first large, commercial database actually implemented in logic. This design exercise involved a very large number of design decisions made at the time on a more or less *ad hoc* basis. However, this experiment clearly defined the major problems which this text now attempts to solve. Since that time, research has concentrated on assembling a set of systematic techniques for solving the major problems encountered in that early experiment. This set of techniques is now complete and is reported herein.

A key feature of our method is the construction of a "normalized model" of the application. The normalized model consists of a formal part and an informal part. The formal part of the normalized model is called the "system model"; it contains all the information required by a programmer to implement the system. The informal part of the normalized model is called the "application model"; it consists of a description of the application in stylized natural language. Thus, the application model acts as both the

documentation for the system model and as the specification of the application with which a nontechnical domain expert can easily interact. The normalized model is in normal form in a fairly elaborate sense which is a direct extension of the well established normal forms for information. Our normal forms include, in particular, a set of technical normal forms for knowledge. The idea behind the normal forms is that each real "thing" in the application should be represented in one place, and in one place only, in the normalized model. This should greatly assist the knowledge base maintenance process.

The text is illustrated with examples expressed in logic programming. Logic programming was chosen because it is both very simple and widely understood when compared with other knowledge languages. We would like to stress that we are *not* necessarily promoting logic programming as an ideal language for implementing expert, knowledge-based systems or deductive databases. We would also like to stress that the ideas presented in this text are quite independent of logic programming; the general principles discussed and the examples given may easily be re-expressed in any other suitable formalism or general purpose expert system shell.

It is important to appreciate that the work reported here is not intended to be a complete account of expert systems or deductive databases. In fact no attempt is made to present material on expert systems or deductive database systems that is generally available in the current literature. For example, the reader whose principal interest is the design of expert systems is assumed to be familiar with (Waterman, 1986); the reader whose principal interest is the design of deductive database systems is assumed to be familiar with (Kerschberg, 1986). The reader is also assumed to be familiar with the design of databases, the managerial difficulties in maintaining databases, the elements of knowledge processing, and, preferably, the use of at least one expert system shell. For example, "machine learning", which is a powerful technique that may be applied to the generation of knowledge from hard data, is not discussed at all; see, for example, the work of Quinlan. Also, for example, "plausible inference", which lies at the heart of many expert systems shells, only receives a mention in passing (Horvitz, Bresse and Henrion, 1988). This is not then a suitable text for the beginner; it is designed for the educated, knowledge-processing specialist. It attempts to show such a specialist how to do a better and more systematic job. With the exception of Chapter 2, the material presented herein is not generally available elsewhere.

The approach to designing corporate knowledge bases presented here is deliberately compatible with established techniques for database design. This compatibility has been made possible by the way in which we regard "data", "information" and "knowledge" as strictly separate, but integrated, concepts (Chapter 3). In other words, existing techniques for information analysis are used to design the information component of a knowledge base. Thus, in a sense, we see knowledge base architecture as a direct extension of existing database architecture, and have acknowledged this relationship in the design technique presented here which may be seen as a direct extension of existing techniques for database design. This text then should also be of interest to those involved in the design of conventional databases in which the management of the "rules" is a complex issue.

The design method for knowledge systems (Chapter 4) discussed in this text is presented as though for hand computation. It should be stressed that this is for the sake of exposition only. For the effective application of the method to a large problem, a support environment, or "knowledge base design assistant" is essential. Such an environment

would be expected to assist with the knowledge acquisition (Chapter 6) and knowledge analysis (Chapter 7) phases by looking after all the "housekeeping" and by playing an active role in the normalization process (Chapter 5). The knowledge base engineering (Chapter 8) and knowledge base implementation (Chapter 9) phases should be fully automated. The knowledge base design assistant should also play a crucial role in knowledge base maintenance (Chapter 10). A restricted, prototype knowledge base design assistant was constructed in 1986 by Alan McNamarra as part of the Knowledge Engineering Work Bench project which was made possible by the generous support of the Australian Federal Department of Science. This prototype system has been used extensively for conducting experiments associated with the research reported herein.

In September 1988 construction commenced of a software package which supports and implements the knowledge systems design technique reported in this text. This software is being constructed by the Australian Commonwealth Scientific and Industrial Research Organisation (CSIRO) within their Division of Information Technology. This software functions as a complete Knowledge Analyst's Assistant, and is intended for serious, professional use. It is constructed so as to be independent of any particular expert systems shell or other implementation language. As well as enabling the knowledge analyst to design and maintain modules of knowledge, the software provides a prototyping facility. A full description of the operation of this software is given at the end of Chapter 10 in Section 10.8. In the first instance, enquiries should be directed to:

Knowledge Systems Design Project,
CSIRO,
Division of Information Technology,
PO Box 1599,
North Ryde, NSW, 2113,
Australia.

The software is scheduled to be available for distribution late in 1989.

We adopt two important conventions. First, new terms, as they are defined, will be presented in italics and a reference to each definition should be found in the index; italics are also used for emphasis. Second, when terms are used in the general text before they have been formally defined, in which case the reader is called upon to provide an intuitive meaning, the term will appear within quotation marks. Thus, if a technical term appears within quotation marks in the general text, then this means that the reader is not expected to know precisely what the term means, but from the context, should be able to sense approximately what the term means.

I would like to thank all those who have assisted in the development of this text. First, I would like to thank Professor Ross Quinlan of the University of Sydney who encouraged me to write the book. Second, I would like to thank the University of Technology, Sydney, for their generous study leave provisions during which the greater part of the writing was done. Third, I would like to thank the CSIRO's Division of Information Technology at North Ryde, Sydney, for welcoming me to their laboratory where most of the book was written. In particular, I would like to thank the chief of the Division who was initially Dr G. E. Thomas and subsequently Dr J. F. O'Callaghan for their hospitality. Fourth, I would like to thank Dr I. W. Montgomery, an exponent of

the Binary Relationship approach to information analysis, for his assistance with the information analysis. Last, I would like to thank Dr R. M. Colomb, principal research scientist at the Division, for painstakingly reading and rereading the manuscript as it developed and for his detailed comments; I would like to acknowledge his excellent suggestions, many of which which have been incorporated into the text.

John K. Debenham,
University of Technology, Sydney
November 1988.

1 Perspective

1.1 INTRODUCTION

In this chapter we place knowledge systems in an historical perspective. First, we discuss a major development in knowledge processing machinery, namely the Japanese Fifth Generation Computer Systems Project. Second, we compare and contrast traditional hardware and software technology with knowledge processing technology. Third, we look at the recurring patterns in the history of design techniques and predict the need that knowledge systems will have for rigorous design. And last, we discuss the relevance of "expert systems" to the evolving world of knowledge systems.

1.2 FIFTH GENERATION COMPUTER SYSTEMS

The Japanese Fifth Generation Computer Systems Project (FGCS) has been one important factor responsible for the rise in interest in Knowledge Systems. The FGCS project is, of course, not the only such development. However, it was the first to be publicized on a grand scale and it has been well reported so it is appropriate for us to begin by reviewing briefly some of the goals and intentions of this substantial project.

The Japanese Fifth Generation Computer Systems (FGCS) Project has been dubbed "The Second Computer Revolution"; others have described it as the *first* computer revolution (Feigenbaum and McCorduck, 1983) and the "Third Industrial Revolution" (Sinclair, 1984). What is this project? Why is it so special? What does it mean to the professions in general? What does it mean to the information industry in particular?

During 1981 the Japanese Government conducted a substantial survey which attempted to identify the computing requirements of Japan into the 1990s. This survey concluded with four main goals (Moto-Oka, 1982):

1. to increase productivity in low-productivity areas; key target areas being:
   - document processing;
   - office management;
   - decision making in management;
   - office automation;
2. to meet international competition and contribute toward international cooperation. An important consequence is "knowledge" being seen as a commodity which can be packaged and sold;

3. to assist in saving energy and resources. This includes the reduction of movement of people through the installation of sophisticated, distributed knowledge-based systems;
4. to cope with an aged society. This includes the effective education of an aging society; in particular, the provision of effective computer-assisted education for the professional at home and in the office throughout that professional's career.

It is clear that all four goals have significant implications for professional life. If the four goals noted above are achieved, then the main impact on professional life implied directly by those goals will be:

1. both a substantial change in the intellectual "objects" which professionals use within their own businesses, and a substantial change in the intellectual "objects" used by the whole business community, and with which professionals will have to interact;
2. the sort of knowledge which members of professions presently possess and from which these professionals presently derive an income will become increasingly available in a mechanized form;
3. both a substantial change in the way in which professionals interact with each other and a substantial change in the way in which professionals interact with their clients and their clients' businesses;
4. the way in which professionals acquire, maintain and propagate their professional knowledge will change.

Thus, the impact on professionals within the information industry will take two forms; first, the way in which they, as professionals, go about their business will change, and second, the computer systems with which they deal will be playing a significantly different role in their clients' lives than the majority of computer systems do today.

In October 1981 the Japanese Government announced the FGCS project, and on 14 April, 1982 the Institute for New Generation Computer Technology (ICOT) was launched in Tokyo. The aim of the FGCS project is to design and build computing equipment that will satisfy the four goals identified in the survey, and noted above. The plan is for the project to be complete by 1992. The extent to which this plan is being realized on schedule is not clear. However, whether the FGCS project meets its goals or not, it is our view that considerable progress is being made by this and other projects towards the construction of knowledge processing machines of some sort. It is our view that during the early 1990s knowledge processing hardware will be widely available, and will be sufficiently powerful to threaten the professional community with substantial change.

How will fifth generation computers differ from current generation computers? The differences between the first four generations of machines are principally in the hardware, that is, in the way in which they are constructed; the basic design remains much the same. The design philosophy of traditional computers is often referred to as "Von Neumann architecture", and is named after an early pioneer in computing. Four key features of machines based on Von Neumann architecture are:

1. They have a program controller which controls the sequential operation of the program (i.e. one instruction is executed at a time).

2. They have a memory which consists of a large number of discrete memory locations. Each location is primarily intended to store a number.
3. They have a processor which is capable of performing operations on the values stored in the memory. The operations are primarily arithmetic.
4. They have input and output devices which are based principally on typed characters.

The whole architectural concept of fifth generation computers will differ radically from the Von Neumann architectural philosophy; both the architectural concept itself and the way in which this concept is realized in VLSI (i.e. "computer chips") or in ULSI (i.e. Ultra Large Scale Integration, the computer chips of the near future) will represent major advances. Key features of the new architecture follow:

1. There will be a large number of processors which may compute, in some sense in parallel, more or less independently from one another.
2. The processors will hold their own data, and, in addition, will be tightly coupled with large database machines, which will be especially designed for that purpose.
3. Each processor will be designed to perform logical deduction as its basic computational step. In addition, the processors will have access to very high speed arithmetic and other special purpose, auxiliary processors.
4. Input and output facilities will include the mechanization of "intelligent interfaces" based on work in Artificial Intelligence. These will include optical character recognition and speech recognition, as well as spoken output.

In short, the machines will provide an architecture well suited to the mechanization of intelligence. A central design decision of the FGCS project was to adopt "logic" as the kernel programming language. We will discuss "logic" in the following chapter. If the FGCS project succeeds in meeting all of its goals, it will, without doubt, constitute a quantum leap in the development of computing machinery.

What do the Japanese hope to gain from the FGCS project? The FGCS project is a vital component in Japan's decision to become the first post-industrial society. In industrialized society the wealth of nations depends on natural resources, the accumulation of money and upon weaponry. In a post-industrial society the wealth of nations will depend on information, knowledge and intelligence.

> "Japanese planners view the computer industry as vital to their nation's economic future and have audaciously made it a national goal to become number one in this industry by the latter half of the 1990s. They aim not only to dominate the traditional (Von Neumann) forms of the computer industry (Kashiwagi, 1985) but to establish a 'knowledge industry' in which knowledge itself will be a saleable commodity like food and oil. Knowledge itself is to become the new wealth of nations." (Feigenbaum and McCorduck, 1983.)

1.3 HARDWARE AND SOFTWARE

The much publicized Japanese Fifth Generation Computer Systems Project has undoubtedly been partly responsible for the rapidly increasing interest in knowledge processing among the members of the computing profession. This project has presented

the profession with the promise of an architecture which differs radically from conventional hardware design and which is intended for knowledge processing. This has quite understandably generated considerable motivation for professionals to find out what "knowledge processing" is all about. It is, however important to remind ourselves that while these knowledge processing machines will no doubt bring dramatic increases in speed and decreases in cost for processing knowledge they will not possess an increase in capability over conventional architectures. That is, anything that can be done on a knowledge processing machine can be done on a conventional machine, possibly at considerably slower speed. Thus, whereas it is true to state that some knowledge processing applications which are unfeasible today will become feasible on these new architectures, it is also true to say that knowledge processing machines will not provide us with any theoretical increase in computing capability. In other words, knowledge processing machines will not transcend the capability of the classic Turing model.

We have seen that the impact of knowledge processing machines will be largely one of efficiency; in particular, the efficiency of processing "knowledge". Thus, we propose that the appropriate response to the advent of such machines is to ask, "How can we design application systems which take the greatest advantage of this new hardware?". Consideration of this question forms the greater part of the work reported here.

The Japanese Fifth Generation Computer Systems Project was also greatly responsible for the sudden increase in interest within the computing profession in the logic programming languages. After all, logic programming was announced as the kernel language of these new computers. Professionals in the information industry have been confronted with a language which has some extraordinary properties. This language admits a purely declarative semantics, and so contains no purely imperative statements such as the assignment statement. As we will see, a language which admits a declarative semantics is more suited to the representation of "knowledge" or "rules" than a conventional, imperative programming language. Logic programming is described as being a "very high level" language and yet it is to be, in effect, the machine code of these new computers. Thus, professionals have been faced with the prospect of a quantum leap in software technology of undeniably impressive dimensions.

There is no doubt that, in general, knowledge can be expressed far more simply in logic than in, say, the programming language BASIC. However, it is important to remind ourselves that the use of logic as an implementable formalism for the representation of knowledge will not, in itself, simplify the design process. In other words, just because we are using logic to implement a system, the whole business of constructing a "good" design will not be reduced to the application of a few simple rules. To draw an analogy with programming, one could argue that a programmer is likely to write better structured code when using the programming language Pascal than when using the programming language BASIC. However, it is quite possible to write equally *unstructured* code in Pascal as anything that can be represented in BASIC. That is, a programming language may reflect a design philosophy, but, if it is a general purpose language, it can't be expected to enforce the application of that philosophy.