

# **Designing Expert Systems**

---

A Guide to Selecting  
Implementation Techniques

**Paul J. Kline**

**Steven B. Dolins**



# **Designing Expert Systems**

---

## **A Guide to Selecting Implementation Techniques**

**Paul J. Kline**

**Steven B. Dollins**

Texas Instruments  
Dallas, Texas



**WILEY**

**JOHN WILEY & SONS**

**NEW YORK • CHICHESTER • BRISBANE • TORONTO • SINGAPORE**

Copyright © 1989 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by Section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

*Library of Congress Cataloging in Publication Data:*

Kline, Paul J.

Designing expert systems.

Bibliography: p.

Includes index.

1. Expert systems (Computer science) 2. System design. I. Dolins, Steven B. II. Title.

QA76.76.E95K54 1989 006.3'3 88-37869

ISBN 0-471-50484-X

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1



# Preface

---

There is currently a great deal of excitement about the prospects for building new types of computer programs called *expert systems*. While it has long been accepted that computer programs can automate computational and clerical tasks, only with the advent of expert systems has it become possible to automate certain tasks requiring expert knowledge. The expertise of skilled physicians, troubleshooters, designers, analysts, and equipment operators are now being tapped to build expert systems to perform tasks of economic and scientific importance.

The automation of these tasks typically confers a variety of benefits:

- Proliferation of scarce expertise
- Consistent and reliable application of expertise
- Documentation of scarce knowledge

The recent emergence of a variety of software products for expert-system construction has provided widespread access to many of the tools and techniques needed to build expert systems. What is much less widespread is knowledge of how to design expert systems.

This book was written to try to help remedy that problem by providing guidelines that relate problem features to expert-system designs. These guidelines assist expert-system builders in choosing the right expert-system design and the right artificial intelligence (AI) techniques to solve the problem at hand. Since expert-system problems can differ rather dramatically, the guidelines should be useful for designers building their first expert-system or their twenty-first.

Even the experienced expert-system builder who needs no help in developing effective designs for expert systems may find it valuable to have a published source of design guidelines to refer to in design reviews or customer briefings.

We developed the guidelines primarily to assist the practitioner, that is, the expert-system designer. However, the use of quotations from the published literature on expert systems to support the guidelines also makes the book valuable to students of artificial intelligence. This book brings together in one place a wealth of insight and discussion that would otherwise require an extensive study of the professional literature on expert systems.

This discussion of expert systems provides one assessment of the current state of the art in this particular area of artificial intelligence:

- The techniques that have been developed
- The problems those techniques solve
- The pitfalls that might be encountered in using the techniques

We feel fortunate to have been able to collect and record some of the first steps taken in the direction of changing the design of expert systems from an art into a science.

PAUL J. KLINE  
STEVEN B. DOLINS

*Dallas, Texas*  
*April 1989*

# **Acknowledgments**

The research for this book was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss AFB, New York 13441 under contract no. F30602-83-C-0123. This book incorporates a large amount of the material that appeared in the final report of that contract. The preparation of this book was made possible by Texas Instruments Incorporated and its management, who assisted us in numerous ways.

We would like to thank Dr. Northrup Fowler III of Rome Air Development Center for his valuable suggestions during the course of this research. We are very grateful to the expert-system builders who were kind enough to comment on earlier versions of the guidelines: Bruce Buchanan, Penny Nii, Ruven Brooks, John Kunz, Bruce Porter, Robert Drazovich, Frederick Hayes-Roth, Robert Neches, Tim Finin, and Jim Kornell. They are, of course, in no way responsible for any deficiencies of the current set of guidelines.

We would also like to thank the following individuals for their assistance: Byron Davies, Tom Ekberg, Jim Bender, Ken Hill, and Gary Honeycutt at Texas Instruments and Diane Cerra and Wanda Cuevas at John Wiley & Sons.

We are very grateful to the publishers and authors who allowed us to reprint selections from their publications in this book. Rather than acknowledging all the copyright holders here, their names accompany reprinted materials throughout the book.

PAUL J. KLINE  
STEVEN B. DOLINS

# Contents

---

List of Figures	xiii
List of Tables	xv
Preface	xvii
Acknowledgments	xix

1	Introduction	1
1.1	Organization of This Book / 4	
1.1.1	Example Guideline: Signal-to-Noise Ratio / 5	
1.1.2	Support for Example Guideline 1.1.1 / 6	
1.1.3	Example Pitfall: Model-Driven Reasoning / 8	
1.2	Locating the Relevant Guidelines / 8	
1.2.1	Using Problem Characteristics to Index Design Guidelines / 13	
1.2.2	Using Problem Characteristics to Index Implementation Techniques / 15	
1.3	Cautions in Using the Guidelines / 15	
1.4	Organization of Design Guidelines / 17	

Headings in italics denote Design Guidelines.

## **2 Connecting Evidence to Hypotheses** **21**

---

- 2.1 *Evidence: Black, White, Gray* / 21
  - 2.1.1 *Support for Example Guideline 2.1* / 22
- 2.2 *Visibility of Intermediate Steps* / 24
- 2.3 *Generalized Evidence* / 26
- 2.4 *Confirmation by Exclusion* / 28
- 2.5 *Levels of Detail* / 29
- 2.6 *Generalized Hypotheses* / 30
- 2.7 *Levels of Abstraction* / 32
- 2.8 *Single Focus or Multiple Possibilities* / 34
- 2.9 *Crucial Step in Search* / 35

## **3 The Arrival Time of Information** **37**

---

- 3.1 *Constraint Propagation or Forward-Chaining* / 38
  - 3.1.1 *Support for Example Guideline 3.1* / 39
- 3.2 *Anticipate-Assess-Resolve Uncertainty* / 40
- 3.3 *Island-Driving* / 45
- 3.4 *Extent of Planning Prior to Execution* / 47
- 3.5 *Monitoring Values of Measurements* / 49
- 3.6 *Change Invalidates Conclusions* / 53
- 3.7 *Creating New Relationships* / 55
- 3.8 *Falsifying Assumptions* / 56

## **4 The Expert System's Environment** **59**

---

- 4.1 *Cooperation from Source of Inputs* / 59
  - 4.1.1 *Support for Example Guideline 4.1* / 61
- 4.2 *User-System Partnership* / 62
- 4.3 *Incomplete Information* / 63
- 4.4 *Explain Situation or Explain Reasoning* / 66
- 4.5 *Alternative Explanations of Reasoning* / 69
- 4.6 *Order of Questions* / 73
- 4.7 *Initiative in Interaction* / 76
- 4.8 *Frequency of Device Changes* / 77



---

<b>5</b>	<b>Computational Efficiency</b>	<b>79</b>
5.1	<i>Choice of Next Action / 79</i>	
5.1.1	<i>Support for Example Guideline 5.1 / 81</i>	
5.2	<i>Choice of Next Information to Request / 83</i>	
5.3	<i>Selectivity in Requesting Information / 87</i>	
5.4	<i>Speed of Response to Inputs / 88</i>	
5.5	<i>Aggressive or Conservative Search / 89</i>	
<b>6</b>	<b>The Nature of Solutions</b>	<b>93</b>
6.1	<i>Multiple Faults / 94</i>	
6.1.1	<i>Support for Example Guideline 6.1 / 95</i>	
6.2	<i>Novel Faults or Predictable Faults / 98</i>	
6.3	<i>Single Hypothesis or Multiple Hypotheses / 99</i>	
6.4	<i>Prenumerated or Constructed Solutions / 101</i>	
6.5	<i>First Solution or All Solutions / 103</i>	
6.6	<i>Relationship between Objectives / 104</i>	
6.7	<i>Achieving Interacting Goals / 109</i>	
6.8	<i>Solution: Absolute-Relative-Explanation / 119</i>	
<b>7</b>	<b>Knowledge Representation</b>	<b>125</b>
7.1	<i>Degree of Structure in Knowledge / 125</i>	
7.1.1	<i>Support for Example Guideline 7.1 / 126</i>	
7.2	<i>Type of Knowledge / 130</i>	
7.3	<i>Instances Inherit Default Properties / 133</i>	
7.4	<i>Overlap in Factual Knowledge / 134</i>	
7.5	<i>Overlapping Capabilities / 135</i>	
7.6	<i>Networks of Knowledge / 137</i>	
7.7	<i>Organize Knowledge to Support Reasoning / 139</i>	
<b>8</b>	<b>Pitfalls of the Implementation Techniques</b>	<b>14</b>
8.1	<i>Bayes Rule / 143</i>	
8.1.1	<i>Subjective Bayesian Methods / 144</i>	
8.2	<i>Bidirectional Search / 145</i>	

- 8.3 Blackboards / 146
- 8.4 Causal Models / 146
- 8.5 Chronological Backtracking / 147
- 8.6 Confirmation by Exclusion / 148
- 8.7 Constraint Propagation / 148
  - 8.7.1 Multiple Views / 151
- 8.8 Control Rules / 152
- 8.9 Dempster-Shafer Uncertainty Management / 154
- 8.10 EMYCIN Certainty Factors / 154
- 8.11 EMYCIN Context Trees / 156
- 8.12 Forward-Chaining / 157
- 8.13 Frames / 158
- 8.14 Fuzzy-Set Theory / 158
- 8.15 Generate-and-Test / 159
- 8.16 Goal-Driven Reasoning / 160
- 8.17 Inheritance Mechanisms / 161
- 8.18 Intelligent Schedulers / 161
- 8.19 Least Commitment / 162
  - 8.19.1 Guessing / 163
- 8.20 Match / 163
- 8.21 Means-Ends Analysis / 163
- 8.22 Model-Driven Reasoning / 164
- 8.23 Opportunistic Search / 165
- 8.24 Partial Matching / 165
- 8.25 Resolution / 166
- 8.26 Rules / 167
- 8.27 Scoring Functions / 167
- 8.28 Semantic Networks / 168
- 8.29 Truth Maintenance Systems / 170

## **9**

### **Future Applications of Design Guidelines**

---

**173**

- 9.1 Initial Design Assistance / 174
- 9.2 Suboptimal Use of Techniques or Need for Redesign / 175
- 9.3 Response to User Queries / 176
- 9.4 Conclusion / 177

<b>A</b>	<b>Glossary</b>	<b>179</b>
----------	-----------------	------------

---

<b>B</b>	<b>References</b>	<b>201</b>
----------	-------------------	------------

---

	<b>Index</b>	<b>215</b>
--	--------------	------------

---

# **List of Figures**

---

- 1.1 It is easy to select the right tool for driving nails and screws. Expert-system development software is often referred to as *toolkits*, which suggests, incorrectly, that it is easy to choose the right AI tool to solve an expert-system design problem. 2
- 1.2 Selecting the right household chemical to remove rust stains and mustard stains is harder than the hammer-screwdriver problem in Figure 1.1. Choosing the right AI technique to solve an expert-system problem is more like this stain-removal problem. 3
- 1.3 Tables 1.2 and 1.3 can be used to find the design guidelines appropriate for solving a particular problem. Guideline numbers found in Table 1.3 are appropriate for a wide range of problems. The guidelines in Table 1.2 can be indexed by evidence strength and solution type. 14
- 1.4 Some of the AI techniques recommended in this book are appropriate for only certain combinations of evidence characteristics and solution characteristics. The location of techniques in this figure provides a rough indication of where they are appropriate. 16
- 4.1 Explanation produced by direct translation. Reprinted with permission from Swartout, W. *Explainable Expert Systems*. USC/Information Sciences Institute, Marina del Rey, CA, October 1983. 70
- 5.1 Breadth-First Search is more conservative than Depth-First Search. Breadth-First Search would be expected to have fewer very fast solutions and fewer very slow solutions. 90

- 6.1** Expert-system problems can be distinguished by the number of solutions that are expected. The three expert systems shown differ in the number of solutions they are prepared to handle. The ISIS system is an expert system that can cope with the possibility that there is no solution to the problem as originally stated. 94

# **List of Tables**

---

- 1.1 The problems solved by expert systems can be categorized by the strength of the available evidence (rows of the table) and the way solutions are developed (columns of the table). The problems solved by some of the expert systems referred to in this book are categorized according to these dimensions. 10
- 1.2 The numbers in each of the nine cells in this table are design guideline numbers. The design guidelines associated with cell  $(i, j)$  are the guidelines relevant to problems characterized by strength of evidence  $i$  and solution type  $j$ . 11
- 1.3 These guidelines are relevant to all nine cells in Table 1.2 because they are applicable to all evidence strengths and all solution types. 13

# 1

## Introduction

---

A large number of decisions must be made in designing an expert system. For example:

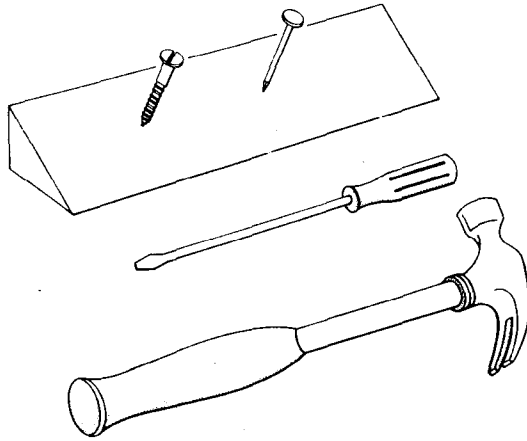
- What knowledge representation technique should be used?
- What problem-solving strategy should be employed?
- How should uncertainty be handled?

Making the right decisions greatly simplifies the development of an expert system and helps to ensure its lasting usefulness. The goal of this book is to help expert-system builders make intelligent design decisions.

Before beginning, it is worth considering the difficulty of making expert-system design decisions: How difficult is it to select the proper artificial intelligence (AI) methods for a particular task? A recent paper by John McDermott, the developer of some of the most successful expert systems ever built, provides an answer:

Although efforts, some successful, to develop expert systems (application systems that can perform knowledge-intensive tasks) have been going on now for almost 20 years, we are not yet very good at describing the variations in problem-solving methods that these systems use, nor do we have much of an understanding of how to characterize the methods in terms of features of the types of tasks for which they are appropriate [J. McDermott 1988, p. 225].

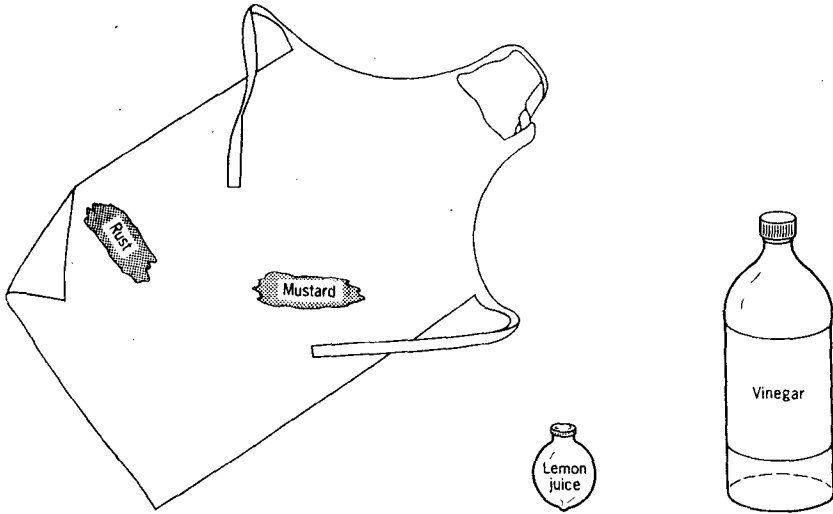




**Figure 1.1.** It is easy to select the right tool for driving nails and screws. Expert-system development software is often referred to as *toolkits*, which suggests, incorrectly, that it is easy to choose the right AI tool to solve an expert-system design problem.

Despite McDermott's suggestion that it can be difficult to make design decisions, it would be possible for the casual observer to get the impression that it is easy to select the appropriate AI technique to solve a particular problem. For example, the vendors of software systems that support the construction of expert systems often refer to their products as "toolkits." The use of the word "toolkit" to describe this software conjures up images of hammers, screwdrivers, saws, and similar hardware items. The implied suggestion is that selecting the right AI tool for an expert-system problem is roughly as difficult as deciding whether a hammer, screwdriver, or saw is the right tool for a particular repair job. As suggested by Figure 1.1, selecting the appropriate household tool is often easy. Even without experience with these two tools, it is easy to see that the hammer is the right tool for driving the nail and the screwdriver the right tool for driving the screw.

Unfortunately, the task facing the expert-system builder is considerably more difficult than this analogy to selecting real tools suggests. A better analogy is the problem of selecting the appropriate household chemicals to solve cleaning and stain-removal problems. Figure 1.2 offers two alternatives for removing stains, rust stains in one case and mustard stains in the other. *Hints from Heloise* [1980, pp. 380f.] provides the right answer: The lemon juice will remove the rust stains and the vinegar will remove the mustard stains. (In case the correct answer



**Figure 1.2.** Selecting the right household chemical to remove rust stains and mustard stains is harder than the hammer-screwdriver problem depicted in Figure 1.1. Choosing the right AI technique to solve an expert-system problem is more like this stain-removal problem.

to this problem is obvious, perhaps you can also explain why Heloise provides the stern warning “Never use ammonia” to try to remove a mustard stain.)

It is not easy to guess that lemon juice is effective for removing rust stains and vinegar for removing mustard stains, even with a lifetime of familiarity with these chemicals. The fact that there are many such “nonobvious” tools for cleaning has kept Heloise busy for many years churning out household hints.

One way to view this book is as a version of *Hints from Heloise* for the expert-system builder. Simply knowing about Rule interpreters, Frame systems, and Blackboard systems does not guarantee that it is easy to see which of these should be used for a particular expert-system problem—no more than knowing about vinegar and lemon juice guarantees that it is easy to see which of them should be used on the stains.

The examination of the design guidelines for building expert systems found in Chapters 2 through 7 of this book supports the argument that selecting the right AI technique to solve a problem is more like choosing the right stain remover than it is like choosing the right tool from a toolbox.