This is to certify that the

dissertation entitled

Candidate Evaluation:
A Task Specific Architecture
Using Multi-Attribute Utility Theory
With Applications in International Marketing

presented by

Michel Mitri

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Computer Science

Carl V. Page
Major professor

Date February 28, 1992

A
USING
WITH APPLI

in parti

De

CANDIDATE EVALUATION:
A TASK SPECIFIC ARCHITECTURE
USING MULTI-ATTRIBUTE UTILITY THEORY
WITH APPLICATIONS IN INTERNATIONAL MARKETING

By

Michel Mitri

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1992

A
USIN
WITH APPI

This thesis
epresentation f
alled Candidate
isciplines. Firs
ttribute utility
or the framework
nfluenced by rec
CSA) and gener
rchitecture is i
EVAL (for Candid
alled CEVED (for
n intelligent da
onstructs is
nternational m
valuation model

ABSTRACT

CANDIDATE EVALUATION:
A TASK SPECIFIC ARCHITECTURE
USING MULTI-ATTRIBUTE UTILITY THEORY
WITH APPLICATIONS IN INTERNATIONAL MARKETING

By

Michel Mitri

This thesis presents a knowledge acquisition and representation framework for evaluative reasoning tasks, called Candidate Evaluation. It draws from two main research disciplines. First, the decision theoretic model of multi-attribute utility theory (MAUT) provides a mathematical basis for the framework. Second, the knowledge representation is influenced by recent research in task-specific architectures (TSA) and generic tasks (GT). The Candidate Evaluation architecture is implemented in an expert system shell called CEVAL (for Candidate Evaluator), and a development environment called CEVED (for Candidate Evaluation Editor). In addition, an intelligent database combining MAUT with semantic network constructs is presented. The thesis also presents international marketing applications of the Candidate Evaluation model and the MAUT semantic network database.

: would like to
:lped make this
dvisor, Carl Pa
ork and for pa
Thanks also go
Stocklen, George
McCarthy, for the
ther Eva and my
upport, without
ossible. Finally
Make my life very

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank those who helped make this dissertation possible. Many thanks go to my advisor, Carl Page for his tireless efforts in overseeing my work and for patiently yet firmly prodding me to improve. Thanks also go to the other members of my committee: Jon Sticklen, George Stockman, S. Tamer Cavusgil, and William McCarthy, for their valuable insights and comments. I thank my mother Eva and my father Moufid for their unceasing love and support, without which this effort would not have been possible. Finally, I thank Cheryl, Brendan, and Joshua, who make my life very happy indeed.

1. Introduction.
1.1 The St
1.2 The Rep
1.3 The Dom
1.4 The Cha

2. Task Specific
2.1 "Task-I
2.1.1 H
2.1.2 H
2.1.3
2.1.4 E
2.2 Philosc
Archite
2.2.1 M
2.2.2 M
T
2.2.3 M
2.2.4 S
2.3 Philosop
2.3.1 S
2.3.2 M
U
2.3.3 M
2.3.4 G

2.4 Examples
Other Me
2.4.1 H
2
2.4.2 P
2
2
2
2.4.3 A
2.4.4 I
2
2
2.4.5 S
2
2
2.5 Other Ap
2.5.1

TABLE OF CONTENTS

1. Introduction..... 1
   1.1 The Study of Evaluation..... 1
   1.2 The Representation of Knowledge..... 4
   1.3 The Domain of International Marketing..... 7
   1.4 The Chapters of this Dissertation..... 8

2. Task Specific Architectures and Generic Tasks..... 11
   2.1 "Task-Independent" Knowledge Representations.... 11
      2.1.1 Rule-based systems..... 12
      2.1.2 Frame-based systems..... 13
      2.1.3 Logic-based systems..... 15
      2.1.4 Blackboard systems..... 15
   2.2 Philosophical Precedent to Task-Specific Architecture..... 18
      2.2.1 Newell's Knowledge Level..... 20
      2.2.2 Marr's Information Processing Task and Type-1/Type-2 theories..... 22
      2.2.3 Minsky's Society of Minds..... 25
      2.2.4 Stefik et al.'s Expert Task Breakdown.... 26
   2.3 Philosophy of the Generic Task Approach..... 28
      2.3.1 So What is a Generic Task, Anyway?..... 31
      2.3.2 MDX-MYCIN: Accomplishing MYCIN behavior using Generic Task Methods..... 38
      2.3.3 More About the Idea of Modular Specialists 40
      2.3.4 Generic Tasks and Knowledge Acquisition.. 42

   2.4 Examples of Generic Tasks, and Comparisons to Other Methods..... 43
      2.4.1 Hierarchical Classification..... 43
         2.4.1.1 Comparison to Pattern Recognition. 48
      2.4.2 Routine Design and Planning..... 52
         2.4.2.1 Overview of the OPM/BB1 Approach.. 56
         2.4.2.2 Comparisons of Design/Planning Methods..... 62
         2.4.2.3 Final Thoughts about Comparison between DSPL and OPM..... 65
      2.4.3 Abductive Assembly..... 67
      2.4.4 Functional Reasoning..... 71
         2.4.4.1 O.S.U.'s Functional Reasoning: Being explicit about purpose..... 73
         2.4.4.2 Davis' Model Based Reasoning Approach..... 76
      2.4.5 Structured Matching..... 77
         2.4.5.1 Samuel's Signature Tables..... 78
         2.4.5.2 Another Approach to Structured-Matching's IPT..... 80
   2.5 Other Approaches to Task-Specific Architectures. 81
      2.5.1 TSA work done byMcDermott and Colleagues at DEC and Carnegie Melon..... 81

2
2
2
2
2
2.5.2
2.6 Conclus
1. Bayesian Mode
3.1 The Baye
3.2 Subject
3.3 Bayesia
3.3.1
3.3.2
3.3.3
3.4 Bayesian
3.4.1
3.5 Combinin
3.6 Conclusi
4. Regression, Li
Theory in Deci
4.1 Multiple
4.2 Use of R
4.
4.
4.
4.3 Multiatt
4.3.1
4.3.2
4.3.3
4.4 Combinin
4.4.1
4.4.2
4.4.3
4.4.4
4.4.5
4.4.6
4.5 Multiple
Candidat
4.6 Conclusi

2.5.1.1 MOLE: A Tool for Cover-and-Differentiate..... 83
2.5.1.2 SALT: A Tool for Propose-and-Revise Systems..... 85
2.5.1.3 KNACK: A Tool for Sample-Based Report Generation..... 87
2.5.1.4 SIZZLE: A Tool for Sizing Systems. 88
2.5.1.5 A Possible Way to Test the Generic Task Hypothesis..... 89
2.5.2 The KADS Approach: TSA Research in Europe 91
2.6 Conclusions about TSAs..... 94

3. Bayesian Models in Decision Theory and AI..... 98
3.1 The Bayes Model..... 98
3.2 Subjective Expected Utility Theory (SEUT)..... 100
3.3 Bayesian Knowledge Representations in DT and AI. 103
3.3.1 Decision Trees and Influence Diagrams.... 103
3.3.2 Probabilistic Inference Networks (PIN)... 105
3.3.3 Comparing Decision Trees and Inference Networks.....109
3.4 Bayesian Studies of Human Decision-Making..... 111
3.4.1 Why People Deviate from Bayes..... 113
3.5 Combining Bayesian Decision Analysis with AI.... 116
3.6 Conclusion..... 123

4. Regression, Linear Models, and Multi-Attribute Utility Theory in Decision Theory and AI..... 125
4.1 Multiple Regression..... 126
4.2 Use of Regression in Decision Sciences..... 129
4.2.1 Correlational Paradigm..... 129
4.2.1.1 Brunswick Lens Model..... 130
4.2.1.2 Linear Judgement Policy Models.... 130
4.3 Multiattribute Utility Theory (MAUT)..... 134
4.3.1 MAUT Approaches to Non-Linearity..... 136
4.3.2 Non-Compensatory Decision Rules..... 139
4.3.3 Applications of MAUT: Multiattribute Utility Technology..... 142
4.4 Combining MAUT and Linear Models with AI..... 142
4.4.1 Samuel's Signature Tables Revisited..... 143
4.4.2 Berliner and Ackley's Hierarchical Weighted Scoring..... 144
4.4.3 Continuous vs. Discrete Representation... 145
4.4.4 Context in Evaluation..... 148
4.4.5 Explanation in Evaluation..... 150
4.4.6 Empirical Comparison of Truth Tables and Linear Models..... 151
4.5 Multiple Evaluators: Methods for Voting on Candidates.....151
4.6 Conclusions..... 157

5. The Candidate
5.1 Develop
Evaluat
5.2 Overall
Archite
5.3 The Can
CEVED a
5.3.1
5.3.2
5.4 A Gener
5.4.1
5.4.2
5.5 Candidat
5.6 Knowled
5.6.1
5.6.2
5.6.3
5.6.4
5.6.5
5.6.6
5.7 Validat
5.7.1
5.7.2
5.7.3
5.8 Conclus
CEVED/C
6. Issues in Int
6.1 Selecti
6.1.1
6.1.2
6.1.3
6.2 Select
6.2.1
6.2.2
6.2.3
6.2.4
6.3 Some C
Market

5. The Candidate Evaluation Architecture..... 159
5.1 Developmental Principles for a Candidate Evaluation TSA..... 160
5.2 Overall Description of the Candidate Evaluation Architecture.....162
5.3 The Candidate Evaluation Shell -- CEVED and CEVAL..... 165
5.3.1 Candidate Evaluation Editor (CEVED)..... 165
5.3.2 Candidate Evaluator (CEVAL)..... 172
5.4 A Generic Task Analysis of Candidate Evaluation. 177
5.4.1 Structured Matching and Candidate Evaluation.....177
5.4.2 Abductive Assembly and Candidate Evaluation.....180
5.5 Candidate Evaluation as Implementation of MAUT.. 181
5.6 Knowledge Acquisition for Candidate Evaluation.. 182
5.6.1 KA and MAUT Assessment Techniques..... 184
5.6.2 Identifying the Experts (or "Stakeholders")..... 185
5.6.3 Identifying and Structuring the Major Criteria (Dimensions)..... 186
5.6.4 Identifying and Scaling the Indicator Variables (Evaluative Questions)..... 187
5.6.5 Weight Assessment 189
5.6.6 Interpretation Assessment..... 192
5.7 Validation and Verification of CE Expert Systems 194
5.7.1 What Should be Measured?..... 195
5.7.2 How Should Measurement be Done?..... 197
5.7.3 Testing Methodology for CE Expert System. 199
5.8 Conclusions: Strengths and Weaknesses of CEVED/CEVAL.....201

6. Issues in International Marketing..... 204
6.1 Selection of Foreign Markets..... 205
6.1.1 Stages of Country Selection..... 205
6.1.2 Regression-based Model for Country Evaluation.....208
6.1.3 Providing Market Research Information and Evaluation.....208
6.2 Selection of Entry Modes..... 209
6.2.1 Factors Involved in Selection Entry Modes 211
6.2.2 Classification of Entry Modes..... 215
6.2.3 Three Models of Entry Mode Selection..... 220
6.2.3.1 Goodnow's GIMS..... 220
6.2.3.2 Casson's Model of Contractual Entry Mode Selection..... 222
6.2.3.3 Cavusgil's CORE..... 223
6.2.3.4 A Final Look at the Three Models.. 227
6.2.4 Use of Candidate Evaluation for Entry Mode Selection.....228
6.3 Some Operational Issues in International Marketing.....232

6.4 Conclus

7. The Country C
Database.....
7.1 Semanti
7.1.1
7.1.2
7.1.3
7.1.4
7.2 Semanti
7.3 A Semar
Consult
7.3.1
7.3.2
7.3.3
7.4 The Cour
7.5 Knowledg
Country
7.6 Conclusi

8. Conclusions of
8.1 Contribu
8.2 Future D
8.2.1 M
8.2.2
8.2.3
8.2.4
8.3 Final Co

APPENDIX A: Form
Eval

6.4 Conclusions..... 234

7. The Country Consultant: An Inferential-Evaluative Database.....235
7.1 Semantic Network Knowledge/Data Representations. 238
7.1.1 Quillian's Semantic Memory Model..... 239
7.1.2 Schank's Conceptual Dependency Theory.... 240
7.1.3 Wood's "What's in a Link"..... 242
7.1.4 Brachman's KLONE..... 244
7.2 Semantic Networks as Database Models..... 246
7.3 A Semantic Network View of the Country Consultant.....249
7.3.1 How the CC Infers Evaluations..... 251
7.3.2 Spreading Activation in CC..... 254
7.3.3 Inferring Judgements via Weighted Evidence Accumulation..... 256
7.4 The Country Consultant as a MAUT Model..... 259
7.5 Knowledge Acquisition and Validation for the Country Consultant..... 260
7.6 Conclusions..... 262

8. Conclusions of the Dissertation..... 263
8.1 Contributions of the Thesis..... 263
8.2 Future Directions..... 265
8.2.1 Multiple-evaluator Issues..... 265
8.2.2 Generalizations and Extensions of the Semantic Network MAUT Model..... 266
8.2.3 Linkage of CEVAL modules with Country Consultant and Each Other..... 268
8.2.4 Knowledge Acquisition and Representation Enhancements to CEVED..... 268
8.3 Final Conclusions..... 270

APPENDIX A: Formal Characterization of the Candidate Evaluation Architecture..... 272

Three main topics
Levels of abstrac

Precursors to Cha
Genericness and k
nowledge represe
Overview of five

Comparison of two
functions........

Structural compon
Sample dimension
Sample dimension
Hypothetical effe
in CEVAL........
Sample evaluative
Partial score pro
Relationship betw
recommendation fr

Factors influenci
A descriptive tax
A comparison of th

Structural compone
Partial view of Co
A sample judgement
Country Consultant
Scope of spreading
Consultant.......

List of Figures

1.1 Three main topics of the dissertation..... 3
1.2 Levels of abstraction for expert systems development. 6

2.1 Precursors to Chandrasekaran's Generic Task theory... 19
2.2 Genericness and knowledge-use level of various knowledge representation schemes..... 95
2.3 Overview of five generic tasks..... 96

4.1 Comparison of two AI methods for static evaluation functions..... 147

5.1 Structural components of CEVED/CEVAL system..... 164
5.2 Sample dimension entry screen in CEVED..... 169a
5.3 Sample dimension hierarchy..... 169b
5.4 Hypothetical effect of contextual weight adjustment in CEVAL..... 170
5.5 Sample evaluative question entry screen in CEVED..... 174
5.6 Partial score propagation in a CEVAL consultation.... 175a
5.7 Relationship between dimension-ratings and recommendation fragments..... 175b

6.1 Factors influencing the choice of entry modes..... 214
6.2 A descriptive taxonomy of entry modes..... 217
6.3 A comparison of three models for entry mode selection 226

7.1 Structural components of the Country Consultant..... 237
7.2 Partial view of Country Consultant's semantic network 250
7.3 A sample judgement entry screen in Country Consultant 253
7.4 Country Consultant's inference strategy entry screen 253
7.5 Scope of spreading activation in the Country Consultant..... 255

This disser

for evaluation a

system shell, an

marketing. The

empirical findi

theory, psycholo

call this archi

shows, the Candi

of this thesis,

decision-theoret

multi-attribute

AI and knowledge

Specific Archite

international ma

1.1 The Study of

Many of the

involve deciding

these kinds of pr

of each candidat

problems involve

and weaknesses i

performance. Both

evaluation. In s

CHAPTER 1

INTRODUCTION

This dissertation presents a problem-solving architecture for evaluation and selection, its implementation in an expert system shell, and its application to problems in international marketing. The architecture is based on theoretical and empirical findings from academic literature in decision theory, psychology, artificial intelligence, and marketing. I call this architecture *Candidate Evaluation*. As Figure 1.1 shows, the Candidate Evaluation architecture, and the topics of this thesis, are based upon work done in three areas. Decision-theoretic approaches to evaluation, particularly multi-attribute utility theory, have a major impact. Issues in AI and knowledge representation, including the theory of Task-Specific Architectures play a role. Finally, the domain of international marketing was a guiding force.

1.1 The Study of Evaluation

Many of the problems we encounter in our day-to-day lives involve deciding between a set of options, or *candidates*. In these kinds of problems, one needs to determine the worthiness of each candidate in order to select the best one. Other problems involve assessing an individual candidate's strengths and weaknesses in order to suggests ways of improving its performance. Both types of problems require the process of *evaluation*. In schools, for example, students are evaluated

for both remedi

There is a

in the social s

Evaluation is of

scores and/or qu

are usually repr

are relevant to

scoring process,

levels of the at

used in educatio

1975), software

1989), expert sy

host of other do

Indeed, eva

has grown an ent

the study of eva

as a branch of d

areas of psycholo

methods that pe

includes study

discussed above,

models. In their

linear models, h

nonlinear and no

non-compensatory

lexicographic inf

for both remedial purposes and for selection and ranking.

There is a wealth of research literature, both in AI and in the social sciences, dealing with evaluation methodology. Evaluation is often described in terms of establishing numeric scores and/or qualitative ratings for a candidate. Candidates are usually represented in terms of attributes (criteria) that are relevant to the evaluation. Many models involve a weighted scoring process, where the weights indicate the importance levels of the attributes being scored. Such models have been used in education (Beggs and Lewis 1975), marketing (Wright 1975), software validation (Gaschnig et al. 1983; O'Keefe 1989), expert system viability (Slagle and Wick 1988), and a host of other domains.

Indeed, evaluation is a such a ubiquitous task that there has grown an entire research discipline devoted entirely to the study of evaluation techniques. This discipline has grown as a branch of *decision theory*, and can be found in several areas of psychology and other social sciences. It explores the methods that people use for evaluation and selection. It includes study of *compensatory* decision rules like those discussed above, usually implemented as algebraic weighted models. In their simplest form, these are simple weighted linear models, but they can also be altered to deal with nonlinear and nonmonotonic data. Also studied is the use of *non-compensatory* decision rules, which are usually lexicographic inform. Studies have indicated that both forms

International
Marketing

Domain

The three n
knowledge
decision the
and the bus
These topic
problem so
Evaluation.

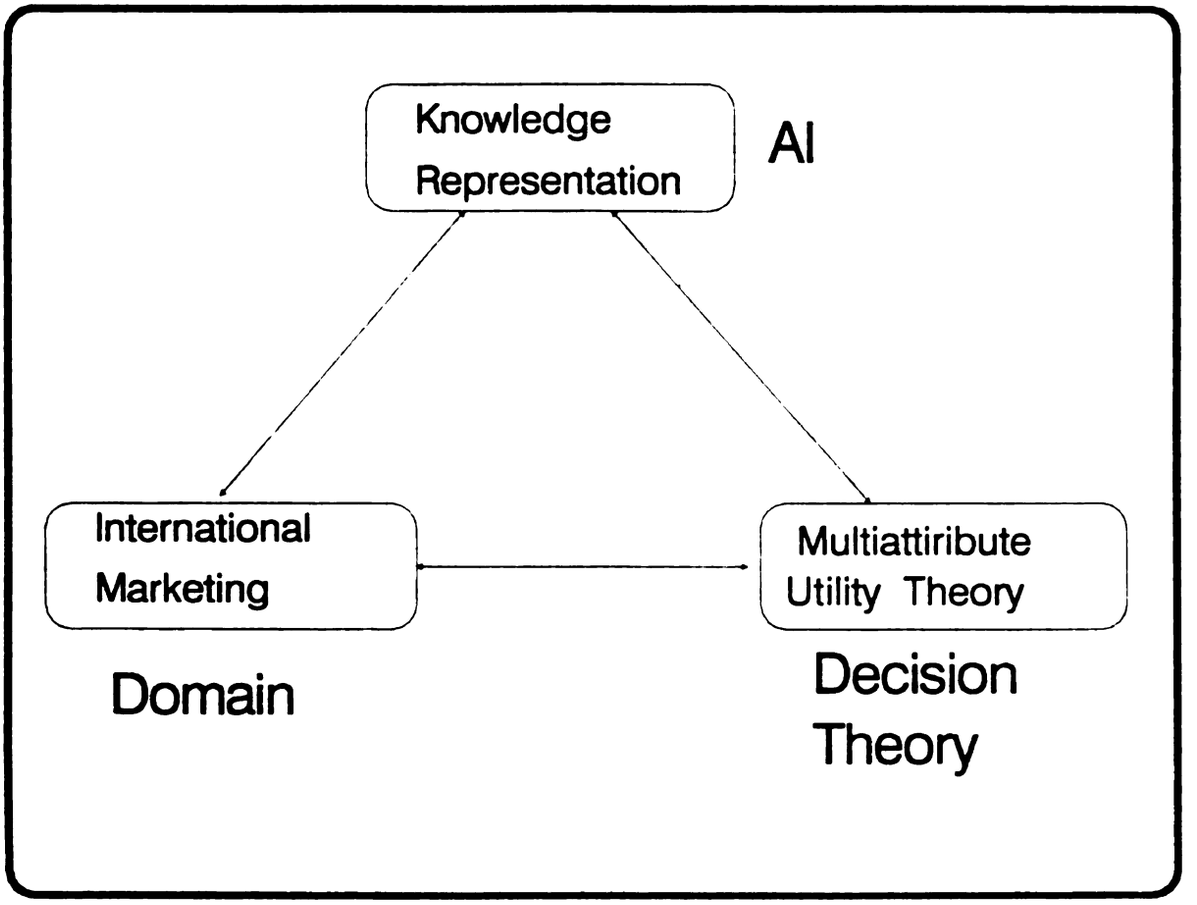


Figure 1.1

The three main topics of this dissertation deal with knowledge representation issues in artificial intelligence, decision theoretic methods for evaluation and selection, and the business domain of international marketing. These topics are combined into a task-specific problem solving architecture called Candidate Evaluation.

ay be used, d

te importance

1.2 The Represe

Knowledge

artificial i

representation s

representations

forward/backward

selection.

Although t

representation c

they still invol

Even with thes

translate the

imposed by the

usually a need

programming and

This KE, despit

no prior experi

significant le

questions of th

be established

which can be c

expert system

infamous "knowl

may be used, depending on the complexity of the problems and the importance of the decisions.

1.2 The Representation of Knowledge

Knowledge representation is a key research area in artificial intelligence. Typical "first-generation" representation schemes include rules, frames, and logic. These representations have corresponding inference regimes like forward/backward chaining, inheritance, and predicate/clause selection.

Although these regimes are improvements in knowledge representation compared to traditional programming languages, they still involve relatively low-level conceptual primitives. Even with these representations, there is much need to translate the knowledge of the expert into the structure imposed by the representation formalisms. Thus, there is usually a need for a knowledge engineer who is fluent in the programming and knowledge representation regimes involved. This KE, despite his or her AI competence, may have little or no prior experience in the domain field, thus necessitating a significant learning process in order to ask the pertinent questions of the expert. There is also a need for rapport to be established between knowledge engineer and domain expert, which can be complicated if the expert sees the idea of an expert system as a threat. All these factors lead to the infamous "knowledge acquisition bottleneck" (Hayes-Roth et al.

1983). A survey

the average cos

$240,000 (Barr

during the knowl

representation

result in signi

There have

bottleneck throu

automatic. One

develop language

primitive mirror

experts perform

1987). This app

(ISA), helps in

solved as well

requires less tr

or frame-based k

ISAs and more

illustrated in f

4.It has been

directly by dom

their knowledge

intermediary kn

this encoding.

1983). A survey conducted by SRI international indicated that the average cost for developing an expert system is around $260,000 (Barr et al. 1989). Much of this cost is absorbed during the knowledge acquisition process. Thus, any knowledge representation schemes that reduce the KA bottleneck will result in significant cost-savings.

There have been several attempts to alleviate this bottleneck through improved KA techniques, both manual and automatic. One method for speeding the KA process is to develop languages and expert systems shells whose conceptual primitive mirror the types of problem-solving techniques that experts perform (Boose 1989, Bylander and Chandrasekaran 1987). This approach, called "*Task-Specific Architecture*" (TSA), helps in analyzing the type of problem that is being solved as well as providing a representation framework that requires less translation than would be needed for rule-based or frame-based knowledge engineering. The distinction between TSAs and more traditional programming and AI methods is illustrated in figure 1.2, and discussed in detail in chapter 2.It has been my experience that TSAs can also be used directly by domain experts, so that these experts can encode their knowledge onto the computer without the need for an intermediary knowledge engineer or computer programmer to do this encoding.

Knowledg

TSA –

Rules, Fra

3 GL and

Asse

H

Levels of abstractio
paradigms are clos
tractional represen

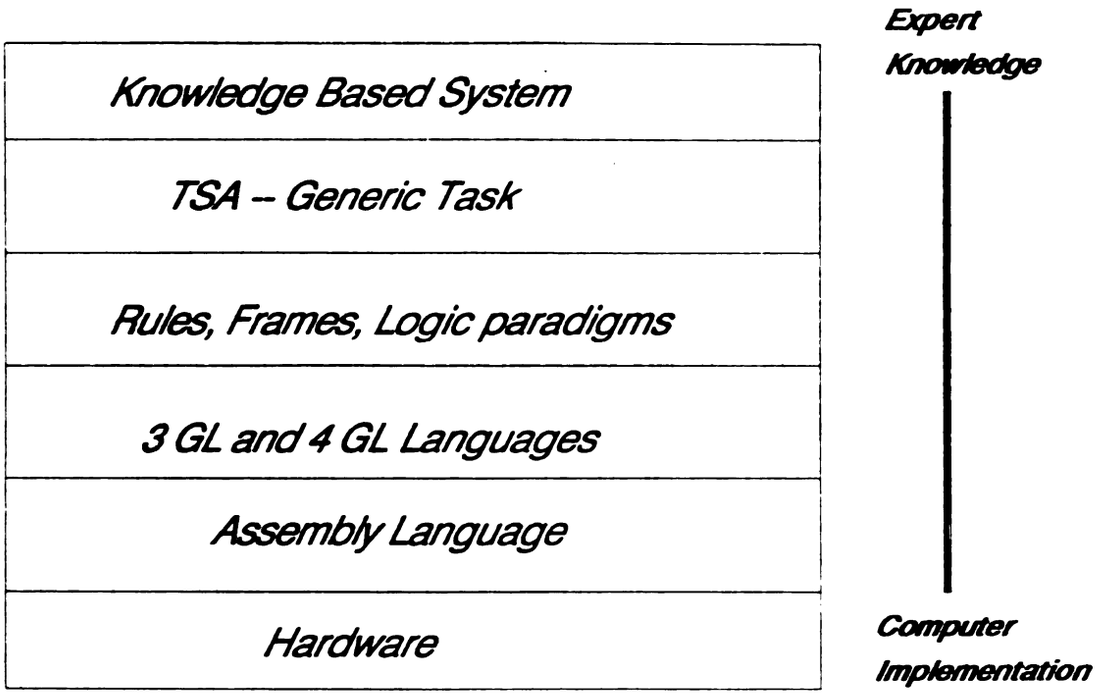


Figure 1.2

Levels of abstraction for expert systems development. The TSA and Generic Task paradigms are closer to the "knowledge use level" of representation than are traditional representational schemes such as rules, frames and logic.

1.3 The Domain

Recently,
national bound
enhancing globa
the European Co
is an increased
pertaining to t
Issues inv
reaching. A co
business abroad
commitment. Str
market to enter
to enter that m
etc.) and how t
Tactical decis
partnerships to
forwarders, eval
and foreign sul
pricing policies
To date, th
systems technolo
thesis explores
area, and discu
been applied in

1.3 The Domain of International Marketing

Recently, there has been an increased permeability of national boundaries, brought on by technological factors enhancing global communication and by trade agreements within the European Community, North America, and Asia. Thus, there is an increased need to disseminate knowledge and expertise pertaining to the domain of *international marketing*.

Issues involved in this arena are numerous and far-reaching. A company exploring the possibility of doing business abroad must decide whether it is ready for such a commitment. Strategic decisions must be made about which market to enter (i.e. what country or region) as well as how to enter that market (e.g. export, license, build a plant, etc.) and how to adapt the product or service accordingly. Tactical decisions must be made about the types of partnerships to set up, selecting distributors and freight forwarders, evaluating the performance of expatriate personnel and foreign subsidiaries, setting up legal contracts and pricing policies to fit the target market.

To date, there has been little effort in applying expert systems technology o the international marketing domain. This thesis explores some of the work that has been done in this area, and discusses how the evaluation task can be and has been applied in this domain.

1.4 The Chapter

This disse

(SSA) for reaso

evaluation (CE

for representi

levels, and n

includes a me

evaluation resu

on the evaluati

for use by non-

the ability to c

The application

area of interna

general enough

Following

in this disser

Chapter 2

Specific Archi

research done

(GT). This cha

SSAs and GTs,

representation

representation

similar proble

Chapter

theory and art

1.4 The Chapters of this Dissertation

This dissertation presents a task-specific architecture (TSA) for reasoning and knowledge representation in candidate evaluation (CE) tasks. The architecture includes primitives for representing candidates, their attributes, importance levels, and numeric/qualitative performance measures. It includes a mechanism for establishing and interpreting evaluation results, and for recommending actions to take based on the evaluation. The architecture is specifically designed for use by non-programming domain experts, and thus enhances the ability to quickly acquire and represent expert knowledge. The applications developed using this architecture are in the area of international marketing, although the architecture is general enough to be applied to a wide variety of domains.

Following is a brief synopsis of the remaining chapters in this dissertation:

Chapter 2 presents a detailed description of the Task-Specific Architecture (TSA) school of thought, focusing on research done by Chandrasekaran and others in Generic Tasks (GT). This chapter discusses the philosophical precursors to TSAs and GTs, describes the characteristics of these types of representations, and compares some specific GTs to other representation schemes that have been employed to deal with similar problem types.

Chapter 3 discusses Bayesian approaches to decision theory and artificial intelligence. Its purpose is to show how

OT and AI have

different purp

model for buil

making, Bayes r

Bayes model as

the reasons th

optimal) Bayes a

situations. Thi

expert system s

the Bayesian ap

Chapter 4

attribute regre

AI. Like chapt

experimental p

knowledge repr

discussions of

multiattribute

use of evaluat

with multi-att

Chapter

Candidate Eval

terminology. I

multi-attribut

compared with

acquisition, r

in the CE cont

DT and AI have been using common frameworks for often very different purposes. Topics discussed include the use of Bayes model for building psychological experiments in decision-making, Bayes model incorporated in expected utility theory, Bayes model as an AI knowledge representation formalism, and the reasons that most people deviate from the (supposedly optimal) Bayes approach when they make decisions in real-world situations. This chapter concludes with a discussion of an expert system shell developed by Langlotz (1989) that merges the Bayesian approach with expert systems techniques.

Chapter 4 shifts from the Bayesian approach to a multi-attribute regression model, and its use in decision theory an AI. Like chapter 3, there is discussion of the model as an experimental paradigm for psychological study as well as a knowledge representation scheme for AI systems. Included are discussions of compensatory and noncompensatory reasoning in multiattribute settings. Also included is a discussion of the use of *evaluation functions* in AI, which often attempt to deal with multi-attribute situations.

Chapter 5 presents a detailed description of the Candidate Evaluation architecture. CE is described in TSA/GT terminology. It is also discussed as an implementation of the multi-attribute utility models discussed in chapter 4, and compared with other AI evaluation methods. Issues of knowledge acquisition, representation, and validation are all discussed in the CE context.

Chapter

marketing, wit

using expert s

the best marke

entry is the

characteristics

distributors, f

subsidiaries an

Several theore

discussed.

Chapter 7

database of mar

indexed using a

uses some of t

Candidate Evalu

algebraic metho

database is ch

guesses about

inferencing is

activation in t

Chapter 8

contributions a

The appen

formal charac

architecture,

system, and c)

Chapter 6 explores some issues in international marketing, with an eye toward how these issues can be resolved using expert systems. Important issues include: selection of the best market (country) to enter; deciding which mode of entry is the best, based on the company and market characteristics; selection of partners (i.e. joint ventures, distributors, freight forwarders, etc.); evaluation of foreign subsidiaries and expatriate personnel; and product adaptation. Several theoretical models and empirical findings are discussed.

Chapter 7 presents the Country Consultant. This is a database of market research information that is catalogued and indexed using a *semantic network* structure. In addition, it uses some of the same evaluation mechanisms described for Candidate Evaluation, particularly the use of multi-attribute algebraic methods for arriving at evaluative inferences. The database is characterized by its ability to make educated guesses about information it does not explicitly know. This inferencing is done through a combination of spreading activation in the network and multi-attribute algebra.

Chapter 8 is a conclusion for the thesis, describing its contributions and suggesting areas of future research.

The appendices at the end of this thesis include: a) a formal characterization of the Candidate Evaluation architecture, b) the Joint Venture Partner Selection expert system, and c) sample output from the Country Consultant.

TASK SP

This chap
solving archit
some "task-ind
logic, and bla
philosophical
representations
"natural" for t
the generic t
colleagues at O
against other m
also review som
done by McDermo
Netherlands.

The TSA ap
of a Candidate
will be discuss
knowledge in a
chapter explo
representation
tools.

2.1 "Task-Inde

Before d
knowledge repr

CHAPTER 2

TASK SPECIFIC ARCHITECTURES AND GENERIC TASKS

This chapter presents a survey of task specific problem solving architectures (TSAs). I start with a brief review of some "task-independent" frameworks such as rules, frames, logic, and blackboard systems. Then I present some of the philosophical precedent for developing "higher-level" representations that capture knowledge in a form that is "natural" for the type of task being done. I review several of the *generic tasks* identified by Chandrasekaran and his colleagues at Ohio State University and compare some of these against other methods being used to solve similar problems. I also review some other streams in TSA research, including work done by McDermott et al. and by researchers in Belgium and the Netherlands.

The TSA approach serves as a motivation for development of a Candidate Evaluation problem solving architecture, which will be discussed in later chapters. The idea is to represent knowledge in a form that is tied to its intended use. This chapter explores the reasons for this "use-based" representation and its implementation in several task-specific tools.

2.1 "Task-Independent" Knowledge Representations

Before discussing the task-specific framework for knowledge representation, I will briefly discuss a few

general-purpos
developed to d
are techniques
expert system s
representation
systems, logic
these general-
describing the
specific TSA me

2.1.1 Rule-Base

One frequ
knowledge in t
pairs called r
knowledge bas
reasoning stra
and backward c

Via forwa
facts, then se
satisfied by t
part) produce
new facts, in
rules whose c
these rules ag
until a goal f
to process.

general-purpose knowledge representation schemes that were developed to deal with a wide variety of problem tasks. These are techniques that are widely implemented in commercial expert system shell today. I will discuss four main knowledge representation schemes: *rule-based* systems, *frame-based* systems, *logic* systems, and *blackboard* systems. Discussions of these general-purpose paradigms will set the stage for describing the motivation behind the TSA approach and some specific TSA methods.

2.1.1 Rule-Based Systems

One frequently-used knowledge-base paradigm involves knowledge in the form of lists of condition-action (IF-THEN) pairs called *rules* or *productions*. These systems involve a knowledge base of rules and an *inference engine* whose reasoning strategy involves deduction in the form of forward and backward chaining.

Via forward chaining, the inference engine starts with facts, then searches for rules whose conditions (IF-part) are satisfied by these facts. The actions of these rules (THEN-part) produce new facts. The inference engine then uses these new facts, in addition to the old ones, to search for further rules whose conditions are satisfied, and the THEN-part of these rules again produce new facts. This process continues until a goal fact is established, or there are no more rules to process.

Backward

ith this stra

ypothesis tha

HEN-parts ca

ypothesis. Fo

ypothesis, an

ypotheses are

end is reached

rules.

Backward c

purposes. One c

edical diagnos

ontrast, forw

onstruction an

R1) (McDermott

onfiguration.

2.1.2 Frame-Bas

Frame-base

to be represente

triggered acti

sometimes calle

databases. They

which are analo

unlike simple r

procedural actio

Backward chaining is the reverse of forward chaining. With this strategy, the inference engine starts with a goal or hypothesis that it seeks to prove. It searches for rules whose THEN-parts can establish the truth or falsehood of the hypothesis. For each of these rules, the IF-part becomes a new hypothesis, and new rules whose THEN-parts establish the new hypotheses are triggered. This process continues until a dead end is reached or the input facts satisfy the IF-parts of the rules.

Backward chaining systems are often used for diagnostic purposes. One of the first backward chaining rule base was a medical diagnostic system called MYCIN (Shortliffe 1976). By contrast, forward chaining systems are often used for construction and design. A prime example is XCON (also called R1) (McDermott 1982), an expert system for computer-hardware configuration.

2.1.2 Frame-Based Systems

Frame-based representation schemes focus on the *objects* to be represented, as opposed to focussing on the conditional-triggered actions of rule-based systems. The objects, sometimes called *frames*, are similar to record structures in databases. They typically contain attributes called *slots*, which are analogous to fields in a record structure. However, unlike simple record fields, the slots may actually trigger procedural actions, called *demons*, in order to determine their

values or in

have *default* v

data values.

Frame sys

called object-

the frame-slc

features. Firs

hierarchically

demons. Thus,

ancestor class

explicitly cod

their *instance*

messages to o

called *methods*

this allows for

methods which

the lower le

inheritance, a

code. That is,

take on object

same structure

slots). This i

of program cod

values or in upon receipt of a value. In addition, slots may have *default values*, which are used in the absence of explicit data values.

Frame systems have evolved into a programming technique called *object-oriented programming*. This paradigm combines the frame-slot-demon concept with two major additional features. First, frame types (called *classes*) can be arranged hierarchically, which allows for *inheritance* of slots and demons. Thus, a sub-class will inherit the slots of its ancestor classes, so that these slots do not need to be explicitly coded in by the programmer. Second, classes and their *instances* (i.e. - actual objects of the class type) pass messages to one another which trigger procedural actions called *methods*. Because these methods can also be inherited, this allows for higher-level classes to define general-purpose methods which can be used and *specialized* (i.e. modified) in the lower level classes. The use of class hierarchies, inheritance, and methods encourages encapsulation of program code. That is, in object-oriented programming, the actions to take on objects (the methods and demons) are defined in the same structure as the data descriptions of the objects (the slots). This increases modularity and allows for easier re-use of program code.

2.1.3 Logic-Ba

Logic-bas
predicate cal
frequently use
Propositional
connectives. L
IMPLIES. Sever
systems using
algorithm (Wan
1957). In fac
notions of fo
today's rule-b

Proposit
calculus, and
all". These,
operators, m
representatio

2.1.4 Blackb

The bl
methodology
problems. Si
is "made up
non-simple
of its part
and the law

2.1.3 Logic-Based Systems

Logic-based systems are based on propositional and predicate calculus. These mathematical representations are frequently used for theorem-proving and automatic deduction. Propositional calculus involves statements and their logical connectives. Logical connectives include AND, OR, NOT, and IMPLIES. Several algorithms have been developed to implement systems using propositional calculus, including Wang's algorithm (Wang 1960) and the Logic Theorist (Newell et al. 1957). In fact, it was Logic Theorist that introduced the notions of forward and backward chaining that are used in today's rule-based systems.

Propositional calculus is an extension of propositional calculus, and includes the *quantifiers* "there exists" and "for all". These, together with *resolution* and *unification* operators, make logic systems a valuable general-purpose representation for problem solving.

2.1.4 Blackboard Systems

The blackboard model is a generic problem-solving methodology designed to tackle complex, ill-structured problems. Simon (1969) described a *complex* system as one that is "made up of a large number of parts that interact in a non-simple way. In such systems,the whole is more than the sum of its parts, in the sense that] given the properties of parts and the laws of their interaction, it is not a trivial matter

to infer the

ill-structure

defined goals

from the init

Ill struc

because knowle

applied to the

the situation

path to the

results in pro

opportunistic

opportunistic

systems exhibit

The black

First is the bl

solution-state

objects in the

are linked tog

divided into

Some sys

planes, each

problem. For

includes a do

plane.

The seco

knowledge sou

to infer the properties of the whole." Newell (1969) said of *ill-structured* problems that they are "characterized by poorly defined goals and an absence of a predetermined decision path from the initial state to the goal."

Ill structured problems are often solvable, says Newell, because knowledge in the form of empirical associations can be applied to them. These knowledge fragments are triggered when the situation warrants, and there is no a priori reasoning path to the solution. This type of knowledge processing results in problem solving behavior that is incremental and opportunistic in nature, and it is precisely this incremental, opportunistic behavior that blackboard systems exhibit.

The blackboard model involves three main components. First is the blackboard itself, which is a database containing solution-state information. The blackboard is made up of objects in the solution state (often called *hypotheses*), which are linked together as the solution unfolds. A blackboard is divided into multiple levels of abstraction.

Some systems include several *blackboard panels*, or planes, each corresponding to different sub-portions of the problem. For example, BB1 (which will be discussed later) includes a domain knowledge plane and a control knowledge plane.

The second component of the blackboard model is the *knowledge sources*. A knowledge source is a specialist that

uses informat

(via its prec

the solution s

in the form c

other implemer

The thir

at its root,

loop, three ma

1) a sel
2) each
b
b
o
3) a con
s
c

The contr

the form of kr

the blackboa

heuristics pe

the domain.

task-specific

tailored to

blackboard

(eg-rules, f

the problem

approach the

uses information on the blackboard to judge its applicability (via its preconditions), then performs actions that modify the solution state on the blackboard. Knowledge sources can be in the form of rules, rule sets, procedures, or a number of other implementations.

The third component is the *control structure*, which is, at its root, a simple control loop. Each time through the loop, three main actions take place:

1) a selected knowledge source changes the blackboard
2) each of the knowledge sources looks at the blackboard to see if its preconditions have been met. If so, they are placed on an agenda, or schedule.
3) a control mechanism (central, or a knowledge source) selects a scheduled knowledge source based on control heuristics (eg - priority level).

The control loop is supplemented by control knowledge in the form of knowledge sources and sometimes a control area of the blackboard. Thus, a knowledge engineer can specify heuristics pertaining to the problem-solving method as well as the domain. Thus, although the blackboard framework is not task-specific (unlike DSPL), its control regime can be tailored to different kinds of tasks. Note also that the blackboard model does not presuppose any implementation (eg-rules, frames, etc.) but is a higher-level abstraction of the problem solving process, and therefore attempts to approach the knowledge level of problem formulation.

2.2 Philosoph

Approach

The noti

by a dissat

representatio

systems (rule

control regime

links, and pr

are general i

low a level of

formulating

abstraction i

software desi

tool should be

level (Newell

1986; Steels

to "think lik

Task-spe

expressing th

natural for t

less general

paradigms, b

more express

easier to imp

they apply.

In this

2.2 Philosophical Precedent to Task-Specific Architecture Approach

The notion of task-specific architectures was motivated by a dissatisfaction with the above-mentioned knowledge representation paradigms usually associated with expert systems (rules, frames, and logic) and their corresponding control regimes of forward and backward chaining, inheritance links, and predicate and clause selection. These frameworks are general in their applicability, but are expressed at too low a level of abstraction to make them useful and coherent in formulating complex problem solutions. Their level of abstraction is too close to the implementation level of software design, whereas a truly useful knowledge engineering tool should be expressed using constructs at the *knowledge-use level* (Newell 1981; Clancey 1985; Chandrasekaran 1983 and 1986; Steels 1990). In other words, the tool should be forced to "think like the expert", and not vice versa.

Task-specific architectures attempt to do this by expressing their representation constructs in terms that are natural for the type of problem to be solved. Thus, they are less general in scope than the rule-based or object-oriented paradigms, but make up for this lack of generality by being more expressive in their ontology and therefore making it easier to implement knowledge bases in the task areas to which they apply. Figure 2.1 illustrates this point.

In this section, I will discuss some of the historical

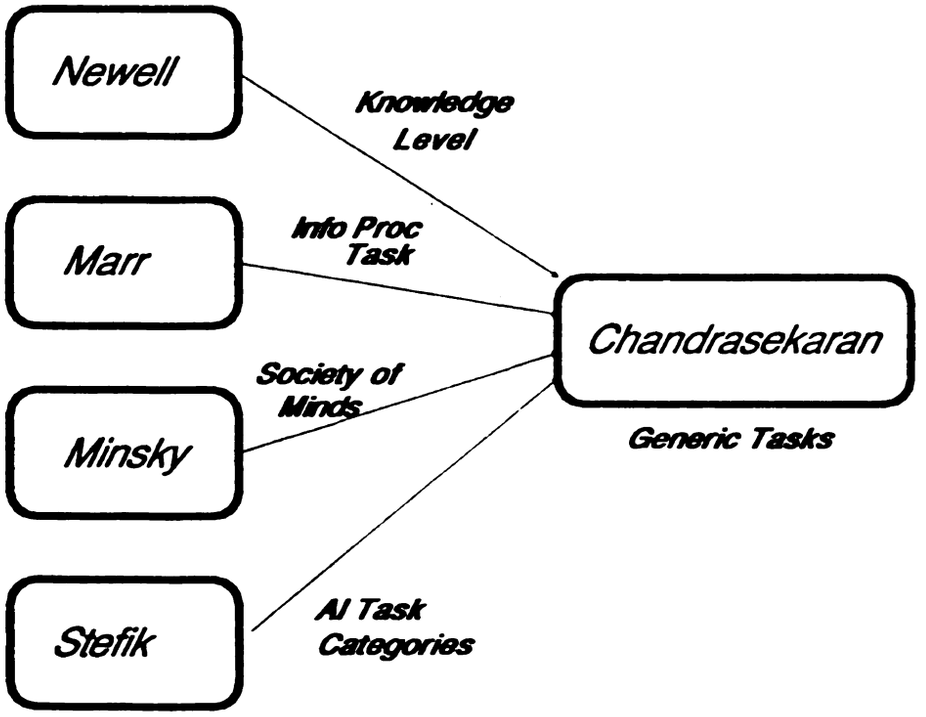


Figure 2.1

Chandrasekaran's Generic Task theory is influenced by a number of other theoretical and empirical work done by AI researchers.

precedent le

I will concer

Stefik.

2.2.1 Newell's

Newell (

excessive rese

for not spend

itself. He a

knowledge rep

more abstrac

representatio

Newell

engineers and

computer sys

highest: dev

The idea h

provides a r

system in q

the approp

representa

But a

even the s

knowledge

functiona

precedent leading to the task-specific architecture approach. I will concentrate on the work of Newell, Marr, Minsky, and Stefik.

2.2.1 Newell's Knowledge Level

Newell (1981) criticized the AI community for its excessive research emphasis on knowledge representation, and for not spending enough time exploring concepts of knowledge itself. He asserted that knowledge is not the same as knowledge representation. Knowledge should be expressed at a more abstract level than what had been used to describe representation paradigms.

Newell identified several levels of abstraction that engineers and scientists can use when describing and analyzing computer systems. These levels are, from lowest abstraction to highest: device, circuit, logic, register/transfer and symbol. The idea here is that each higher level of description provides a more abstract and less perfect approximation of the system in question. The symbol level, according to Newell, is the appropriate place to discuss issues of knowledge representation.

But another level is needed that is more abstract than even the symbol level. Newell called this the *knowledge level*. Knowledge level descriptions are descriptions of the functionality of the system, without concern for structural

details. Func
are useful be
system behavi
underlying p
computational
the world is
explicit struc
giant (infini
Since this i
system must b
generate only
hand.

Newell s
components,
level, compo
medium is kn
of rationali
action will
action.

Any sys
manner. Once
simple matt
description
system. This

details. Functional, non-structural descriptions of knowledge are useful because they allow prediction and understanding of system behavior without necessitating detailed descriptions of underlying processes. They are necessary because, whereas computational structure is finite and bounded, knowledge about the world is by nature unbounded. To describe knowledge with explicit structural form, said Newell, would be like having a giant (infinite) table containing all knowledge elements. Since this is computationally infeasible, an intelligent system must be able to generate knowledge dynamically, and to generate only that knowledge which is relevant for the task at hand.

Newell said that each level can be characterized by its components, medium, and behavior laws. For the knowledge level, components include goals, actions, and bodies. The medium is knowledge itself. The behavior law is the principle of rationality, which states that an agent's knowledge that an action will achieve a goal causes that agent to choose that action.

Any system, said Newell, should be describable in this manner. Once so described, he said, it would be a relatively simple matter to translate to a more detailed, structural description of the knowledge representation used by the system. This is a classic top-down approach to systems design.

2.2.2 Marr's

Theories

Marr (19

He discussed

defined as an

processing p

mathematics.

describe meth

Marr def

a particular

providing a

that methods

implementati

cases, knowl

terms befor

descriptions

Marr sa

called a

algorithmic

in that it

and gives a

The se

that cannot

processes a

Then, the i

In this case

2.2.2 Marr's Information Processing Task and Type-1/Type-2 Theories

Marr (1976) was concerned with similar issues as Newell. He discussed the idea of a problem-solving method, which he defined as an abstract account of how to solve an information processing problem. He likened methods to theorems in mathematics. To Marr, a major goal of AI is to identify and describe methods for solving various kinds of AI problems.

Marr defined a result in AI as involving first, isolating a particular information processing problem, and second, providing a statement of a method for solving it. Marr said that methods are not concerned with the details of algorithmic implementation. Note the similarity to Newell's ideas. In both cases, knowledge should be described in conceptual, abstract terms before being translated into detailed structural descriptions.

Marr saw two kinds of problem-solving types. The first, called a Type 1 Theory, provides an overall, global algorithmic solution to a problem. This is a "clean" theory, in that it explicitly provides all the steps in the process, and gives a holistic view of the method.

The second method, called Type 2 Theory, is for problems that cannot be solved via Type 1. In this method, sub-processes are defined along with the behaviors they produce. Then, the interactions between the subproblems are described. In this case, there is no overall view or understanding of how

the system w

understandin

communicate w

fall into thi

An examp

theories, in

vision syster

1) Jule

discriminable

first or seco

ws.

2) Marr's

identifying ar

Note the

formulation wh

task was to be

the other hand

performed a s

task. There w

"specialists",

communicating

Based on

following sugge

tackling an AI

1) What in
isolate
2) Can we
3) If not,

the system works to solve a problem. Rather, there is an understanding of how individual components work and how they communicate with each other. Most problems in AI, he said, fall into this camp.

An example of the difference between Type 1 and Type 2 theories, in the domain of texture discrimination in computer vision systems, is:

1) Julesz' (1975) theory that textured regions are discriminable if and only if there is a difference in the first or second order statistics of their intensity arrays.
vs.

2) Marr's solution to texture-vision discriminations by identifying and coding specific grouping processes.

Note the distinction here. Julesz described an overall formulation which characterizes how the texture-discrimination task was to be performed. This is a Type 1 theory. Marr, on the other hand, presented some specific procedures that each performed a sub-task of the overall texture-discrimination task. There was no overall algorithm, but a group of "specialists", each performing its assigned task, and each communicating its results to the others.

Based on this breakdown of theory types, Marr had the following suggestions for AI research. Important question in tackling an AI problem include:

1) What information processing problem has been isolated? (ie what is the task?)
2) Can we find a clean (Type 1) theory for solving it?
3) If not, can we describe a set-of-processes solution

(Typ

It is

abstract for

domains, whe

restricted i

descriptions

above three q

the issue of

discussing me

frame-based,

Type 2 theor

into sets of

very simple

representati

centralized

Marr a

larger than

for more a

solving ma

different a

relates to

Marr

particular

2½", but I

classifica

hierarchic

(Type 2)...how well will they work?

It is apparent that a Type 1 theory, being a more abstract formalism, is more likely to be generalizable across domains, whereas a Type 2 theory is likely to be more restricted in its applicability. Both theories involve descriptions of problem-solving methods. Note that of the above three questions, only question 1 is directly addressing the issue of task-classification. The other two questions are discussing method-classification. Note also that rule-based, frame-based, and blackboard formalisms are all examples of Type 2 theories. Each is a technique that divides knowledge into sets of semi-autonomous modules which are managed by a very simple control regime. In general, distributed knowledge representations can be considered Type 2 theories, whereas centralized methods are Type 1.

Marr also asserted that chunks of knowledge should be larger than what was happening at the time (another support for more abstraction in representation), and that problem solving may involve several simultaneous computations on different aspects of the problem. We will later see how this relates to Minsky's "society of minds" theory.

Marr said that "once a method is described for a particular problem type, it never has to be done again (p. 2)", but I don't agree with this. We will later see that the classification task can be attacked by many methods, including hierarchical classification, opportunistic reasoning (in

blackboards f

appropriatenes

type of class

nature of the p

the search for

one has been f

methods, and t

particular appl

2.2.3 Minsky's

Marvin Mi

high-level di

intelligence.

> "There ar
> of the mi
> detail.
> theory...
> cannot be

He warned

descriptions

particularly

data structur

different mec

are abandoned

to me. First,

wanted to desc

was higher th

blackboards for instance), or pattern recognition. The appropriateness of each method will depend on the particular type of classification that needs to be done, and on the nature of the problem space. Thus, it is not useful to abandon the search for problem solving methods for a given task once one has been found. It is better instead to have a toolbox of methods, and to be able to choose the one that best fits the particular application of the task at hand.

2.2.3 Minsky's Society of Minds

Marvin Minsky was another AI theorist who encouraged high-level discussions of knowledge representation and intelligence. In one article (1979), he said:

> "There are many real questions about overall organization of the mind that are not just problems of implementation detail. The details of an artificial intelligence theory...will miss the point if machines that use it cannot be made to think. (p. 428)."

He warned against being too quick to pin representational descriptions to mental facilities, saying "we must be particularly cautious about such questions as 'What sorts of data structures does memory use?' There is no single answer; different mechanisms succeed one another, some persist, some are abandoned or modified." This comment suggests two things to me. First, it suggests that Minsky, like Marr and Newell, wanted to describe intelligence at a level of abstraction that was higher than the level discussed in AI circles at the time.

Second, it su
may take many
stated above,
problem type,
there should b
tackling the s
agrees with th

Minsky th
product of a
communication
This idea is v
the notion of
the generic tas
theories, ment
societies were
(again like th
links may be ve
"subsocieties"
different subsc
agents is based
approach.

2.2.4 Stefik,

An often-
from Stefik et
mentioning, par

Second, it suggests that problem solutions for a given task may take many forms, and seems to contradict Marr's assertion, stated above, that once a method is found for a particular problem type, one need not look any further. On the contrary, there should be an effort to find and compare many methods for tackling the same task type. We will see that Steels (1990) agrees with this.

Minsky theorized that human intelligence is not the product of a single entity, but rather results from communication between many agents in a "society of minds". This idea is very similar to, and seems to have influenced, the notion of interacting specialists that we will see with the generic task theory. It is also similar to Marr's Type 2 theories, mentioned above. Minsky hypothesized that these societies were arranged in a generally hierarchical fashion (again like the generic task concept), where communication links may be very rich between agents which reside in the same "subsocieties", but are very sparse between agents that are in different subsocieties. The choice of control transfer between agents is based on local context, again like the generic task approach.

2.2.4 Stefik, et al.'s Expert Task Breakdown

An often-quoted categorization of problem tasks comes from Stefik et al. (1983). This task breakdown is worth mentioning, partly because it is so widely quoted, and partly

because it d

suggested by

following:

1) Diagnc
observ
2) Predic
situat
3) Interp
sensor
4) Design
5) Planni
6) Monito
vulnera
7) Debug:
8) Repair
prescri
9) Instruc
behavic
10) Contro
monito

It seems

categorization

doing. For exa

doing interpre

were doing dia

design and con

was the result

in the AI comm

based applicat

that are added

Note that

were often us

those tasks.

diagnosis invo

because it differs in many respects from the breakdown suggested by Chandrasekaran. The tasks identified are the following:

1) Diagnosis: infer system malfunctions from observable symptoms.
2) Prediction: infer likely consequences of given situations.
3) Interpretation: infer situation descriptors from sensor data.
4) Design: configure objects under constraints.
5) Planning: design of actions.
6) Monitoring: comparing observations to plan vulnerabilities.
7) Debug: prescribe remedies for a malfunction.
8) Repair: execute a plan to administer a prescribed remedy.
9) Instruction: diagnose, debug, and repair student behaviors.
10) Control: interpret, predict, repair, and monitor system behaviors.

It seems that this breakdown was developed as a categorization of what the expert systems of the time were doing. For example, many systems (Hearsay, HASP, etc.) were doing interpretation. Other systems (MYCIN, INTERNIST, etc.) were doing diagnosis. Still others, such as XCON, were doing design and construction. In other words, this task breakdown was the result of empirical observation of what was being done in the AI community. The implication is that, as knowledge based applications proliferate, there will be more task types that are added to this list.

Note that the different systems performing the same tasks were often using very different methods for accomplishing those tasks. For example, whereas MYCIN's approach to diagnosis involved rule-based backward chaining and certainty

propagation,

diseases that

it appears tha

without regar

contrast to Ch

of problem-sol

ways to solve

We will

qualify as "ge

the term. For

it can be solv

2.3 Philosophy

The gene

by Newell,

Chandrasekara

kinds of pro

knowledge-lev

In the early

at Ohio St

Intelligence

problem solv

Chandrasekar

"There

uses o

separat

propagation, INTERNIST used abductive reasoning to find diseases that could account for a given set of symptoms. Thus, it appears that Stefik's breakdown was truly a task breakdown, without regard to methodology. We will see that this in contrast to Chandrasekaran's generic tasks, a categorization of problem-solving methods which can be combined in various ways to solve many different tasks.

We will see also that some of Stefik's tasks do not qualify as "generic tasks" in Chandrasekaran's definition of the term. For example, diagnosis is not a generic task because it can be solved via a composite of generic task primitives.

2.3 Philosophy of the Generic Task Approach

The generic task approach follows from the issues raised by Newell, Marr, Minsky, and Stefik (cited previously). Chandrasekaran, and others, worked on enumerating different kinds of problem-solving methods, and describing them using knowledge-level constructs.

In the early-to-mid 1980s, Chandrasekaran and his colleagues at Ohio State University's Laboratory for Artificial Intelligence Research (LAIR) began to isolate various types of problem solving methods, much as Marr had suggested to do. Chandrasekaran's thesis was the following:

> "There exist different problem solving types, i.e. uses of knowledge, and corresponding to each is a separate substructure specializing in that type of

problem s

Chandras

types in the p

called MDX. TH

of appropriate

and consequenc

solving types

pertaining to

organized as

distributed

solving speci

itself, but a

that must be

Note se

knowledge o

about "soci

problems de

tasks and a

with Marr's

fact that

regimes in

attempting

and symbol

an interme

knowledge

problem-so

problem solving. (Chandrasekaran 1983,p9)."

Chandrasekaran identified four of these problem solving types in the process of developing a medical diagnostic system called MDX. These types included classification, recognition of appropriateness, intelligent data retrieval and inference, and consequence finding ("what will happen if"). These problem solving types all shared certain common characteristics pertaining to their structure and behavior. They are all organized as hierarchies of specialists, with the knowledge distributed among these specialists. Within each problem solving specialist resides knowledge not just about the domain itself, but also about the types of problem solving activity that must be performed relative to the task at hand.

Note several things about this. The idea of distributed knowledge of specialists is consistent with Minsky's ideas about "society of minds", and are in sync with the Type II problems described by Marr. The notion of identifying problem tasks and associated problem solving regimes is consistent with Marr's suggestions for AI research mentioned above. The fact that Chandrasekaran was describing problem solving regimes in a structural, yet abstract, way implies that he was attempting to bridge the gap between Newell's knowledge level and symbol level. Luc Steels (1990) coined a phrase for such an intermediate level between knowledge and symbol, called the *knowledge use level*. The fact that specialists include problem-solving knowledge as well as domain knowledge defies

conventional

engine" is s

"knowledge ba

Chandras

identify a fi

be combined i

problem task.

primitives of

implemented i

example, the

combination o

retrieval, an

its own (hie

regime, and l

of problem s

solving, in

The ide

together to

following qu

"You sh
possibl
creatin
may nev
will mi
tailore

The pro

were later

additional g

conventional thinking in expert systems that the "inference engine" is somehow separate from an independent of the "knowledge base".

Chandrasekaran's idea was that it should be possible to identify a finite number of problem-solving methods that can be combined in various ways to tackle almost any type of problem task. These methods would be considered the epistemic primitives of knowledge-based problem solving, and could be implemented in a series of expert system building tools. For example, the diagnostic task could be tackled via a combination of hierarchical classification, intelligent data retrieval, and consequence finding, each of which would have its own (hierarchical) substructure of specialists, control regime, and language. (Later, abduction was added to the list of problem solving methods employed in diagnostic problem solving, in Sticklen's MDX2).

The idea of combining different problem solving methods together to form an expert system is not a new one, as the following quote from Winston (1984) illustrates:

> "You should think of problem solving paradigms as possible ingredients, not as complete solutions. In creating particular problem solving systems, you may never use any paradigm by itself. Instead, you will mix them together, developing your own blends tailored to the problem domain you face." (p. 203).

The problem solving regimes identified by Chandrasekaran were later termed generic tasks. As time went on, some additional generic tasks were added to the list, including

abduction and

"task" can be

his colleagues

various tasks

themselves ar

emphasis is pl

to the problem

tasks. It is

called "generi

In partic

structure in

hierarchical,

minimum of

structure is

abductive ass

2.3.1 So What

Several

task and TSA

method become

an implementa

cross the lin

There are no

problem-solvi

"continuum o

are general

abduction and simple design. Note: I think the term generic "task" can be a bit misleading here. What Chandrasekaran and his colleagues are really proposing are methods of solving the various tasks that can be identified. Although the tasks themselves are described in the O.S.U. literature, most emphasis is placed on the representational issues pertaining to the problem solving approaches employed in tackling these tasks. It is these problem-solving methods that are often called "generic tasks".

In particular, much emphasis is placed on describing structure in these generic tasks, with a heavy emphasis on hierarchical, tree-like taxonomies of specialists and a minimum of tangledness in the tree. This hierarchical structure is not imposed on all generic tasks, however...the abductive assembly method did not use such a hierarchy.

2.3.1 So What is a Generic Task, Anyway?

Several questions arise from the reading of the generic task and TSA literature. At what point does a problem solving method become a "generic task"? At what point is it considered an implementation-level programming construct? When does it cross the line between task-specificity and domain-dependence? There are no clear dividing lines, but rather a continuum of problem-solving techniques that may be described as a "continuum of genericness". At one extreme of this continuum are general purpose languages and "first-generation"

knowledge-bas

the other ext

in the middle

architectures

architectures

area here...sc

For example, s

specialized th

Although

task-specific

identify some

regimes. Sti

criteria that

problem-solvi

applicability

granularity c

A proble

wide range of

strategy car

problems. Of

general purpo

construct wou

The pro

control regi

the knowledge

strategy can

knowledge-base constructs such as rules, frames, and logic. At the other extreme are domain-specific applications. Somewhere in the middle ranges of this continuum lie the task-specific architectures, and in the left portion of these task-specific architectures are the generic tasks. But there is much grey area here...some generic tasks are more "generic" than others. For example, structured matching is more ubiquitous and less specialized than routine design.

Although there is no clear distinction between task-specific and general-purpose languages, it is possible to identify some important characteristics of task-specific regimes. Sticklen, in his PhD dissertation, cited four criteria that can be used to identify whether a particular problem-solving strategy qualifies as a generic task: wide applicability, existence of a problem-solving template, proper granularity of problem solving, and task specificity.

A problem solving strategy must be applicable across a wide range of domains and problems encountered by humans. The strategy cannot be confined to, say, medical diagnostic problems. Of course, by this criterion alone, almost any general purpose programming language and any AI programming construct would qualify as a generic task.

The problem solving method should have an identifiable control regime and known set of primitives for representing the knowledge that this task embodies. This implies that the strategy can be implemented in the form of an expert system

shell (i.e.

procedures, e

one would enc

to the primit

This lea

be expressed a

for the task

such as rule-

being generic

too low a lev

of the task-s

engineers t

formalisms.

The fou

first three

"primitive"

other, alre

in the vie

itself. Ra

combination

abduction,

one that is

non-decompo

These

tasks. How

hard-and-f

shell (i.e. template), where instead of rules, objects, procedures, etc. as the knowledge representation constructs, one would encode the knowledge directly in terms that relate to the primitives of the task.

This leads to the third criterion, that these primitives be expressed at the proper level of granularity or abstraction for the task at hand. This is where general purpose methods such as rule-based or frame-based programming fall short of being generic tasks. These approaches represent knowledge at too low a level of abstraction to be natural representations of the task-specific knowledge and as a result force knowledge engineers to twist the knowledge into rule or frame formalisms.

The fourth criterion, one that has been implied by the first three, is that the problem solving strategy be "primitive", in the sense that it cannot be decomposed into other, already-defined generic tasks. This is why diagnosis, in the view of Chandrasekaran, is not a generic task in itself. Rather, it is a task that can be solved via a combination of generic tasks including classification, abduction, consequence finding, etc. A true generic task is one that is both high level and abstract on the one hand, and non-decomposable into other generic task on the other.

These criteria, are helpful in characterizing generic tasks. However, I don't think they will enable us to give a hard-and-fast judgement as to whether or not a problem-solving

method quali

The fi

identifiable

generation" A

reasoning, fr

search, and l

identifiable

chaining (ir

frame-based s

The thir

troublesome b

Frame-based

society of m

expressing kn

Mired in imp

it has come

and generic

language cor

The fo

Problem, fo

third crit

abstraction

lower leve

submethods

appears th

C.S.U. res

method qualifies as a generic task.

The first two criteria, wide applicability and identifiable control regimes, are satisfied by most "first generation" AI problem solving methods. Certainly rule-based reasoning, frame-based representation, heuristic state space search, and logic are widely applicable. They also all have identifiable control regimes: such as forward and backward chaining (in rule-based systems) or inheritance (in frame-based systems).

The third criterion, that of "proper granularity", is troublesome because this criterion will shift as time goes on. Frame-based reasoning, for example, grew out of Minsky's society of minds theory, and was thought to be a way of expressing knowledge at high levels and thereby avoid getting mired in implementation level details. Now, however, because it has come into wide use, it is judged by proponents of TSA and generic tasks as being too much like a programming language construct.

The fourth criterion, non-decomposability, is also a problem, for two reasons. First, it appears to contradict the third criterion. Expressing a PS strategy at a high abstraction level implies that it may be broken down into lower level subtasks and submethods. Why would these submethods not be considered the generic tasks? Second, it appears that some of the generic tasks identified by the O.S.U. researchers are, in fact, decomposable, or at least

partially so.

An exam

(1990) calls

subtasks, whi

Steels decom

First, genera

design impleme

implemented i

classificatio

sponsor-selec

best design p

some pre-enume

satisfy the c

example is i

granularity"

is unclear wh

best level o

of its subta

itself a gene

and satisfac

a generic t

design can

themselves

something m

way that di

There

partially so.

An example generic task is routine design, what Steels (1990) calls "construction". This task can be decomposed into subtasks, which themselves may be considered generic tasks. Steels decomposed the construction task into three parts. First, generate the partial solution. In DSPL, the routine design implementation shell developed by Brown (1987), this is implemented in a fashion very similar to hierarchical classification, using structured matching via a sponsor-selector approach to compare and choose from among the best design plans. Second, test the partial solution against some pre-enumerated constraints. Third, modify the solution to satisfy the constraints (what Brown calls "redesign"). This example is illustrative of the problems with the "proper granularity" and the "non-decomposability" criteria. First, it is unclear why the level of routine design is considered the best level of granularity for a generic task. After all, one of its subtasks is essentially hierarchical classification, itself a generic task. Why, for example, is constraint-testing and satisfaction not a proper grain size for qualification as a generic task? This brings up a second point. If routine design can indeed be decomposed into subtasks which are themselves generic tasks, does this not make routine design something more like a composite of generic tasks in the same way that diagnostic reasoning is considered a composite task?

There are other characteristics that help define a PS

method as

Chandrasekar

knowledge is

base of an e

engine (or p

paradigms. In

domain usuall

methods used

illustrates t

"Since

each tas

the repr

knowledg

to a pa

the cont

created

interpr

Another

generic tas

intelligence

meant to co

can perform

task is m

sacrifices

control re

(abstract)

Yet

tendency

specialis

the prob

method as a generic task. One characteristic of Chandrasekaran's version of generic tasks is that control knowledge is intertwined with domain knowledge. The knowledge base of an expert system is not divorced from the inference engine (or problem solver), as is the case with many other paradigms. In the view of Chandrasekaran, knowledge about a domain usually implies knowledge about the problem-solving methods used to attack that domain. The following quote illustrates this:

> "Since there is a control regime associated with each task, the problem solver can be implicit in the representation language.That is, as soon as knowledge is represented in the shell corresponding to a particular task, a problem solver that uses the control regime on the knowledge representation created for the domain can be created by the interpreter. (1986 p. 29)."

Another characteristic of the task-specificity of the generic task approach is that no longer is general intelligence the goal. Unlike rules and frames, which are meant to convey any type of intelligent activity, and which can perform any Turing-computable task, a TSA or a generic task is much more constrained in its applicability. It sacrifices generality in order to achieve explicitness of control representation, richness of ontology, and high-level (abstract) control and domain knowledge representation.

Yet another characteristic of the GT approach is the tendency to distribute knowledge among knowledge agents, or specialists, which each contain expertise in a limited area of the problem domain (and the problem-solving task), and which

communicate

hierarchical

unique to the

idea has been

of distribut

approach (Hay

each contain

via a commor

however, thre

and the black

and domain

specialist.

knowledge to

control know

sources, and

second diffe

sources are

one another

structured

third distil

limited to

blackboard

all knowled

blackboard

less struc

generic ta

communicate with each other along prescribed, usually hierarchical, channels. Of course, this characteristic is not unique to the GT school of thought. Minsky's society of minds idea has been adopted by many AI researchers. Another example of distributed knowledge among agents is the blackboard approach (Hayes-Roth 1985, Nii 1986), where knowledge sources each contain their own specialized expertise and communicate via a common data structure (the blackboard). There are, however, three important distinctions between the GT approach and the blackboard approach. First, with GT, control knowledge and domain knowledge may both reside within the same specialist. With blackboards, the convention is for domain knowledge to be separate from control knowledge. Sometimes the control knowledge is kept in special "control" knowledge sources, and sometimes it is kept in global routines. The second difference is that in blackboard systems, the knowledge sources are not linked hierarchically, but are independent of one another, whereas the GT approach always imposes a structured relationship between specialists. This leads to the third distinction: GT communication channels are almost always limited to parent-child links in the hierarchy, whereas the blackboard approach uses a common data source through which all knowledge sources can communicate. Thus, the behavior of blackboard applications tends to be more opportunistic and less structured than the behavior of applications using generic task methods. (There is one exception to this

imposition c

This excepti

detail later

Actuall

was addresse

explore the i

to the hier

approach woul

decompositior

communicatior

it in a caref

(Chandrasekar

camp as Minsk

"subsociety"

links, but wh

sparse. Al

Chandrasekar

the tools cs

2.3.2 MDX-MY

Task Methods

Much of

breakdown o

hierarchica

to accompli

imposition of hierarchical structure in the generic tasks. This exception is in abduction, which will be discussed in detail later).

Actually, this bias toward hierarchical representation was addressed by Chandrasekaran, who appeared willing to explore the idea of incorporating non-hierarchical components to the hierarchy-of-specialists structure. He said his approach would be to "...start by looking for hierarchical decompositions, and where there seems to be a need for communication outside of the hierarchical channels, to provide it in a carefully controlled fashion such as...blackboards." (Chandrasekaran 1983, p. 16). This idea puts him in the same camp as Minsky, for whom the communications channels within a "subsociety" would be rich and not limited to parent-child links, but whose communications between subsocieties would be sparse. Although this philosophy was espoused by Chandrasekaran, I see little evidence of its implementation in the tools CSRL, DSPL, HYPER, etc.

2.3.2 *MDX-MYCIN: Accomplishing MYCIN behavior using Generic Task Methods*

Much of the initial conceptualization of this taxonomic breakdown came out of their work on MDX, which used hierarchical classification and structured hypothesis matching to accomplish medical diagnostic tasks. Note that MDX was

designed for

(1985) comp

production-r

comparison wa

of the syster

provide mean

specifically

same expert k

the MDX forma

Operatic

metrics: per

positives",

literature.

was availabl

terms of hit

data was ava

and MDX had

was found to

available. H

problems wh

hierarchical

codes to be

was insuffic

In ter

significant

extensibilit

designed for the same general domain as MYCIN. Sticklen et al. (1985) compared the MDX (generic task) approach to MYCIN's production-rule-and-uncertainty-calculus method. This comparison was done for two criteria: operational performance of the systems and ease of knowledge engineering. In order to provide meaningful comparison, MDX was modified to apply specifically to infectious meningococcal diseases, using the same expert knowledge represented in MYCIN, but structured in the MDX format.

Operational system performance was measured using two metrics: percentage of "hits" and percentage of "false-positives", based on test data from cases in the medical literature. For cases where only initial, pre-screening data was available, MDX performed slightly better than MYCIN in terms of hits, but also had more false-positives. When full data was available, MDX and MYCIN both had perfect hit-rates, and MDX had less false-positives than MYCIN. Additionally, MDX was found to be more efficient in pruning when full data was available. However, its strict hierarchical structure caused problems when there was only partial data, because the hierarchical classification method did not allow for child nodes to be activated when knowledge used at the parent level was insufficient. MDX was later enhanced to allow for this.

In terms of knowledge engineering issues, MDX had significant advantages over MYCIN. MDX was superior in extensibility of the system. The hierarchical classification

structure ir

in MYCIN. A

reviewing th

modifying t

hierarchical

a new node or

This also mad

The comp

knowledge-lev

generic task

kinds perform

and refining

particularly

captures the

within its st

the specific

his colleague

2.3.3 More Ab

As menti

generic task

knowledge is

in a predomi

absolutely ne

been argued

implementatic

structure imposes a conceptual modularity that was not found in MYCIN. Adding a new disease hypothesis to MYCIN required reviewing the entire database of existing rules, and possibly modifying the clauses of those rules. By contrast, the hierarchical classification tree of MYCIN only required adding a new node or subtree at an appropriate spot in the hierarchy. This also made debugging easier in MDX than in MYCIN.

The comparison of MDX to MYCIN is illustrative of the knowledge-level approach to representation that underlies the generic task philosophy. Knowledge specialists of different kinds perform different tasks, such as testing of a hypothesis and refining the hypothesis. Modularity is the key here, and particularly a "conceptually valid" kind of modularity that captures the semantic content of the problem-solving methods within its structure. We now go to a description of some of the specific generic tasks identified by Chandrasekaran and his colleagues.

2.3.3 *More About the Idea of Modular Specialists*

As mentioned above, one characteristic of the way most generic tasks have been implemented is the notion that knowledge is distributed among specialists which are arranged in a predominantly hierarchical structure. This is not an absolutely necessary characteristic of generic tasks. It has been argued that the modularity characteristic is an implementation detail...not a knowledge-level consideration.

In fact, one

of such a "s

al. (1987) h

engineering

distribution

extensibilit

knowledge ba

Stickle

specialists a

analysis as

knowledge-lev

behavior of

predicting su

analysis does

relations of

knowledge-lev

level archite

solving agent

subagents, w

understandabl

descriptions

decomposition

a "simulatio

prediction of

merged the kn

AI theory des

In fact, one generic task, abductive assembly, is not composed of such a "society of minds" approach. However, Sticklen et al. (1987) has argued that it *is* important from a knowledge engineering point of view. Specifically, the hierarchical distribution of knowledge among specialists enhances the *extensibility*, *predictability*, and *debuggability* of the knowledge base.

Sticklen (1989) also argues that the hierarchy-of-specialists approach solves a problem with the knowledge-level analysis as defined by Newell. Specifically, although knowledge-level analysis is useful for explaining the observed behavior of a system or an expert, it is not good at *predicting* such behavior. This is because knowledge-level analysis does not discuss specific behaviors or cause-effect relations of the system. Sticklen proposed that Newell's knowledge-level hypothesis be supplemented with a *knowledge-level architecture hypothesis*, which allows for the problem-solving agent to be decomposed into a cooperative structure of subagents, where the behavior of the overall agent is understandable, and predictable, via knowledge-level descriptions of the subagents and their interactions. This decomposition enables one to view a knowledge based system as a "simulation" of a problem-solving agent, thus fostering prediction of problem-solving behavior. In this way, Sticklen merged the knowledge-level approach of Newell with the Type 2 AI theory described by Marr.

2.3.4 Generi

The de

implications

Bylander ar

interaction

"Repres
some pr
the pr
applied

The imp

process shoul

problem solvi

intimately re

that there

knowledge. T

generic task(

Once those

Chandrasekara

expressed in

generic task

I would

expressing t

high-abstrac

possible fo

without the

engineer or

the computer

2.3.4 Generic Tasks and Knowledge Acquisition

The desired characteristics of TSAs have significant implications for knowledge acquisition, as pointed out by Bylander and Chandrasekaran (1987). They discussed the *interaction problem* in knowledge representation, stating that:

> "Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the knowledge" (1987 p.232)

The implications are that the knowledge acquisition process should be guided by the language or vocabulary of the problem solving task at hand. The knowledge representation is intimately reflected by its intended use, with the implication that there is no such thing as "task-neutral" domain knowledge. Thus, an early goal of KA is to identify the generic task(s) that are appropriate for the problem at hand. Once those have been selected, said Bylander and Chandrasekaran, the interviewing process can be guided by, and expressed in terms of, the language constructs of the chosen generic tasks.

I would go a step further from this. In my view, expressing the architectural primitives of a TSA in terms of high-abstraction knowledge-use level constructs makes it possible for non-programmers to use a TSA shell directly without the need to go through an intermediary knowledge engineer or computer programmer for encoding knowledge onto the computer. In other words, the programming language of the

ISA (or the

acquisition

a user-frie

feel of a "

menus, graph

2.4 Exampl

Methods

In the f

tasks that

their implem

I will compa

GTs against

developed t

how the "t

differ from

of the

computation

implemented

2.4.1 Hiera

One of

group was

architectur

classificat

strategies

TSA (or the generic task) can itself serve as a knowledge acquisition tool, provided that the language is implemented in a user-friendly environment that does not have the look or feel of a "programming language" (i.e. - that it involves menus, graphical displays, and other user-friendly features).

2.4 Examples of Generic Tasks, and Comparisons to Other Methods

In the following sections, I describe some of the generic tasks that were identified by the O.S.U. researchers, and their implementations in specific GT languages. In so doing, I will compare the problem-solving methods embodied in these GTs against other knowledge-based systems that have been developed to solve similar problems. The idea here is to show how the "task-specific" nature of the GT representations differ from the more general-purpose representations in terms of the impact on representational expressiveness, computational efficiency, and generalizability of the implemented systems.

2.4.1 Hierarchical Classification

One of the earliest generic tasks defined by the O.S.U. group was *hierarchical classification*. This problem-solving architecture implements hypotheses as nodes in a classification hierarchy and uses top-down tree search control strategies for narrowing in on the most promising hypothesis

nodes. The C

Conceptual S

of develop

classificat

Based

researchers

normally go

intelligent

matching. CS

Note t

several pos

classificat

classificat

association

and distanc

includes s

approaches

classifica

In ke

CSRL's int

"...t
direc
exper
with
1986,

Also,

languages,

decomposed

nodes. The O.S.U. group developed a language called *CSRL* (for Conceptual Structures Representation Language) for the purpose of developing expert systems that do hierarchical classification. (Bylander and Smith, 1986).

Based on research in the MDX project, the O.S.U. researchers specified four main problem-solving tasks that normally go into solving diagnostic problems: classification, intelligent database access, abduction and hypothesis matching. CSRL is a language that tackles the first such task.

Note that hierarchical classification is just one of several possible methods for accomplishing the overall classification task. Steels (1990) identified six methods for classification: linear search, top-down refinement, association, differentiation, weighted evidence accumulation, and distance computation. The field of pattern recognition includes several statistical, syntactic, and parametric approaches to classification. O.S.U.'s hierarchical classification corresponds to Steel's top-down refinement.

In keeping with the overall generic task philosophy, CSRL's intent is:

> "...to allow the system implementor to more directly encode the knowledge acquired from domain experts, and to avoid much of the detail associated with general purpose languages (Bylander and Smith 1986, p.1)."

Also, similarly to most of the O.S.U. generic task languages, the domain and control knowledge in CSRL is decomposed into specialists, which "correspond to different

concepts ir

activity wi

These speci

and the onl

Each specia

specialists

whereas low

specific hyp

As desc

processing t

a case desc

classificati

This i

establish-r

done using

structured

the likeli

data. If

'establish

refinement

specialist

anchored

confidence

be suspen

refinement

Refin

concepts in the domain, and perform the decision-making activity within the concept" (Bylander and Smith 1986, p.2). These specialists are organized in a strict tree structure, and the only relationship between specialist is super-sub. Each specialist corresponds with a hypothesis. Higher-level specialists represent more general, abstract hypotheses, whereas lower-level specialists represent more detailed, specific hypotheses.

As described by Bylander and Smith, the information processing task of hierarchical classification is to identify a case description with a specific node in a predetermined classification hierarchy.

This is accomplished via a control strategy called *establish-refine*. Establishing a hypothesis (specialist) is done using "decision knowledge" (frequently implemented via structured matching) which returns a confidence value showing the likelihood that the given hypothesis matches the input data. If the confidence is high enough, then CSRL "establishes" the hypothesis, which qualifies it for further refinement. If the confidence level is too low, then the specialist is rejected, which eliminates the entire subtree anchored at that node from further consideration. If the confidence level is neither high nor low, the specialist may be suspended, and possibly reinvoked later for further refinement.

Refinement of a specialist involves invoking its

subspeciali

attempting t

available a

A spec

following ma

1) The
spec

2) The
know
As s
a st
stru

3) The
resp
supe
spec
know
send
whic
the

Note

superspecia

structure fa

and Smith:

"An in
requir
can o
this
preve
classi

and

"...we
but w
"knowl
in pro

subspecialists. This means expanding the tree node, and attempting to establish its children, using decision knowledge available at the child nodes.

A specialist is a knowledge structure that contains the following main components:

1) The *declare section* defines relationship to other specialists (super/sub).

2) The *KGS section* (knowledge group section) defines the knowledge used to establish or reject the specialist. As stated earlier, this is usually done in the form of a structure matching truth-table. See the section on structured matching for further details.

3) The *messages section* specifies how the specialist will respond to an establish-refine request from its superspecialist. A typical way to do this is for the specialist to first establish itself (using the KGS knowledge), then recursively call its subspecialists sending each an establish-refine request. The order in which subspecialists are called can be specified by the knowledge engineer in the messages section.

Note that each specialist can have only one superspecialist in CSRL. This makes the classification structure fairly rigid. This problem was addressed by Bylander and Smith:

> "An initial difficulty is that a CSRL hierarchy is required to be a tree structure, ie a specialist can only have one superspecialist. For medicine this appears to be overly restrictive, since it prevents implementation of alternative classifications of diseases.....(p. 4)"
>
> and
>
> "...we are not against tangled hierarchies per se, but we are against increasing complexity and "knowledge" without achieving a corresponding gain in problem solving ability. (p. 4)"

It app

implemented

classificat

structure t

implementor

representat

of the stat

Chandrasekar

needs of th

methods that

tasks. Agair

support this

An adva

classificato

knowledge at

Bylander and

"These
multi-l
levels,
and are
be any
summari
p.4)"

These

methods cite

or distance

hierarchical

approaches.

It appears, then, that hierarchical classification, as implemented in CSRL, may not be applicable to all kinds of classification tasks. Indeed, by forcing a rigid tree structure to the hierarchy, it is possible to force the system implementor to twist the knowledge into an unnatural representational structure. This seems to go against the grain of the stated goal of Generic Task philosophy (stated by Chandrasekaran) that the expert system shell should meet the needs of the task. This implies that there may be other methods that are more appropriate for certain classification tasks. Again, Steel's list of six classification methods support this notion.

An advantage of the hierarchical approach to representing classificatory knowledge is its ability to represent the knowledge at varying levels of abstraction, as stated by Bylander and Smith:

> "These constructs can be used to implement a multi-layer evaluation of a disease. At the lowest levels, rules test the values of database queries and are grouped into KGs. Following this, there can be any number of levels in which several KGs are summarized by another KG. (Bylander and Smith 1986, p.4)"

These advantages do not exist in some of the other methods cited by Steel, such as weighted evidence accumulation or distance computation. Thus, explanation may be easier using hierarchical classification than using other classification approaches.

2.4.1.1 Compa

Pattern

either statis

instances (us

predefined ca

each pattern.

same kind of

classificatio

classificatio

later.

A patte

classified. I

which contain

being represe

in one of two

A pattern mat

patterns and

pattern. Eac

value for fea

array (where

contains the

some proximi

Learnir

with a set o

the features

which the

2.4.1.1 Comparison to Pattern Recognition

Pattern recognition is a supervised learning method, either statistical or structural, whose goal is to group instances (usually called patterns) into one of a number of predefined categories based on the values of the features of each pattern. In this sense, it is a method for performing the same kind of information processing task as hierarchical classification. However, it differs from hierarchical classification in several respects, which will be described later.

A pattern is a representation of an object to be classified. It consists of a set of features, or attributes which contain information relevant to classifying the object being represented. Patterns are generally stored in a database in one of two forms: pattern matrices or proximity matrices. A pattern matrix is a matrix in which the rows represent the patterns and the columns represent the features of each pattern. Each cell *(i,j)* in the pattern matrix contains the value for feature *j* of pattern *i*. A proximity matrix is an *nXn* array (where *n* is the number of patterns) where cell *(i,j)* contains the "distance" between patterns *i* and *j*, according to some proximity metric.

Learning is facilitated by first presenting the system with a set of training patterns. The training patterns include the features mentioned above, and also include the *class* to which the pattern belongs. This is what makes pattern

recognition a

unsupervised

The inf

thusly: given

of potential

category or c

similarity be

described for

pattern-recog

techniques ca

methods. Most

methods, wh

nonparametric

One sta

theory. A Bay

posterior pro

class based

conditional

pattern is i

pick the clas

with the lowe

choice. Baye

classes are

real-world

approach all

probabilitie

recognition a supervised learning process, as opposed to the unsupervised learning of cluster analysis.

The information processing task here can be described thusly: given a pattern representing an individual, and a set of potential categories to which it may be assigned, find the category or class of patterns in which it best fits. Note the similarity between this task description and the task described for hierarchical classification. There are many pattern-recognition techniques for doing this; these techniques can be divided into statistical and structural methods. Most pattern recognition systems use statistical methods, which can be divided into parametric and nonparametric approaches.

One statistical method for PR uses Bayesian decision theory. A Bayesian decision rule is one that calculates the posterior probability that a pattern belongs in a particular class based on the prior probability of the class and the conditional probability of the pattern values given that the pattern is in the class. Bayes rule states that one should pick the class with the highest posterior probability combined with the lowest negative consequence (loss) for making a wrong choice. Bayes rule assumes that the prior probabilities of the classes are known in advance, which is often unrealistic in real-world problems. Some modifications to the Bayesian approach allow for parameter estimation, whereby the prior probabilities are estimated rather than known explicitly. The

Bayesian ap
parametric p
based on th
distribution
advance. Oth
maximum-likel
probabilities
that minimize

Another
approach, wh
pattern(s) th
is an exampl
define a den
algorithm co
each of the
several pos
distance, w
differences
distance, w
sup distanc
difference
distances a
pattern is
nearest tra
training p
"voting" w

Bayesian approach and its modifications are known as parametric pattern recognition techniques, because they are based on the assumption that the underlying probability distribution function of the set of classes is known in advance. Other parametric decision rules include minimax, maximum-likelihood, and Nehman-Pearson. All are based on known probabilities of the classes. Of these, Bayes rule is the one that minimizes the average risk.

Another PR technique is called the nearest-neighbor approach, where a pattern is grouped with the training pattern(s) that are "closest" to it in the pattern space. This is an example of nonparametric PR, and does not attempt to define a density function for the classes. Instead, a *k-NN* algorithm computes the "distance" between a test pattern and each of the training patterns. This distance can be based on several possible distance metrics, for example: *Euclidean distance*, which takes the square root of the sum of differences of all the features of two patterns; *Manhattan distance*, which simply sums the difference in features; and *sup* distance, which computes the distance as the maximum difference between two patterns for a single feature. Once the distances are computed, using one of these metrics, the test pattern is grouped into the same category shared by the *k* nearest training patterns. Of course, if not all the *k* closest training patterns are in the same class, some method for "voting" will take place, such as majority rule or weighting

the votes b

As men

that are p

classifica

their prob

a technic

classifica

knowledge

hierarchi

but rathe

an overa

sense,

hierarch

used ca

describ

classif

Finall

qualit

archi

to g

dist

know

how

the votes based on proximity to the test pattern.

As mentioned earlier, despite the similarity in the tasks that are performed by pattern recognition and hierarchical classification, there are many significant differences in their problem-solving methods. First, pattern recognition is a technique for *machine learning*, whereas hierarchical classification is a mechanism for representing pre-compiled knowledge. Second, pattern recognition does not use a hierarchical, modular representation of specialist hypotheses, but rather involves a flat file of patterns (or instances) and an overall algorithm for grouping these patterns. In this sense, pattern recognition is a Type 1 theory whereas hierarchical classification is a Type 2 theory. The algorithm used can be statistical or graph-theoretic, but it is described as a whole entity, whereas hierarchical classification tends to be more modular and distributed. Finally, pattern recognition does not involve verbal, qualitative explanation as an integral part of its architecture. It is solely a means to train the computer how to group objects into categories. Again, this is quite distinct from hierarchical classification, whose aim as a knowledge representation formalism is to explain to the user how and why a decision is made.

2.4.2 Rou

The
and part
will als
based sy
is calle
and was
the ongo

Bro
categor
design.
innovat
domain
involvi
are st
design
proble
desig
refine
struc
struc

2.4.2 Routine Design and Planning

The next task type I will discuss is object synthesis, and particularly the generic task called *routine design*. I will also discuss a language for development of knowledge based systems performing routine design tasks. This language is called *DSPL*, for "Design Specialists and Plans Language," and was created by Brown (1987) at O.S.U. in conjunction with the ongoing research in generic tasks.

Brown (1987) divided design activities into three categories, which he called Class 1, Class 2, and Class 3 design. Class 1 design requires uncommon creativity and innovation, and is characterized by lack of knowledge in both domain and problem-solving. Class 2 activity is more typical, involving knowledge of the domain, but problem-solving actions are still not known in advance. Class 3 design is routine design, and involves knowledge of both the domain and the problem-solving strategies. DSPL was developed to do routine design. Routine design is a top-down, "plan and successively refine" approach, and thus involves the same hierarchical structure that we have seen in hierarchical classification and structured matching. As Brown said,

> "Our view of routine design is that it is largely a top-down activity. By this we mean that the problem decomposition is performed in a top-down fashion to produce a hierarchy of design goals. It does not imply that the design decisions themselves are made strictly from the top down. In fact, it could be that they are made bottom-up in some situations, but such decisions are guided by the goals already established (Brown, p. 5)."

In c
built fro
structure
top-down
Fea
done man
differen
involve
plans ar
that an
experie
opposed
DS
configu
also
al.,19
and a
planni
requir
planni
domain
requi
very
are e
form

In other words, while the solution-space itself may be built from both top-down and bottom-up mechanisms, the control structure of the problem solver must act in a totally top-down, successive-refinement manner.

Features of a routine design task are: that it has been done many times before; that each time it is done requires different, but similar specifications; that all instances involve similar topologies; that an expert knows specific plans and knows about how to resolve failure situations; and that an expert has complete knowledge with respect to past experiences, with that knowledge being mostly compiled (as opposed to "deep"). .

DSPL was originally developed for design of configurations, such as mechanical devices. However, it has also been used to construct plans (Chandrasekaran et al.,1986). Design and planning are very similar activities , and a routine design tool can be applied for "routine planning" tasks. As with routine design, a theory of planning requires an ontology of the planning task (terms in which planning knowledge is encoded), a structural organization of domain knowledge, and an account of the control processes required to use the knowledge to produce a plan. These are all very similar to the requirements of routine design, and all are expressed as constructs in DSPL.

In DSPL, knowledge about a domain is organized in the form of active cooperating design *specialists* (Brown, p.17),

each of whic

The speciali

the children

sub-speciali

specialist's

message-pass

the root nod

attacking

sub-speciali

sub-speciali

recursively

results of

structural

earlier).

Specia

Here is whe

lies in DSP

of calls

constraint-

representa

instructio

knowledge

specialist

design dom

view that

totally s

each of which is responsible for a sub-portion of the design. The specialists are organized in a tree-like hierarchy, where the children of a specialist are themselves further refined sub-specialists attacking sub-portions of the parent specialist's task. Communication is possible (via message-passing) between parent and child specialists. Thus, the root node of a specialist tree represents a specialist attacking the entire design problem, and invokes sub-specialists to solve sub-portions of the design task. The sub-specialists in turn attack their sub-problems (perhaps by recursively calling sub-sub-specialists), then return the results of its attempt back to its parent. (Note the structural similarity between DSPL and CSRL, discussed earlier).

Specialists contain local design agents called *plans*. Here is where the actual domain-specific control knowledge lies in DSPL-generated systems. A plan consists of a sequence of calls to sub-specialists, calls to *tasks*, and *constraint-tests*. In this sense, a plan is a procedural-like representation of control knowledge as a sequence of instructions and/or "subroutine" calls. Note that this control knowledge is like a local "scheduler", imbedded in a specialist that itself is focusing on a subportion of the design domain. Thus, you see an example of Chandrasekaran's view that domain knowledge and control structure cannot be totally separate; knowledge of the domain involves knowledge

of the cont

plan in a D

messages sec

and conditi

controlling

specialist i

Each pl

current desi

evaluate the

is typically

Thus, the sp

KGS section

by the spec

sponsors. T

for each c

execution.

judged by t

any plan t

prioritiza

discovered

a manner s

Tasks

Their pur

modularit

make des

change th

of the control regime to use on that domain. In this view, a plan in a DSPL specialist serves a similar function as the messages section of a CSRL specialist...both control the order and conditions of calls to sub-specialists, as well as controlling the sequence of actions that occur within the specialist itself.

Each plan has an associated *sponsor*, which views the current design state (as stored on the design database) to evaluate the current appropriateness of the plan. The sponsor is typically implemented in the form of a structured matcher. Thus, the sponsor is performing an analogous function as the KGS section of a CSRL specialist. Plans themselves are chosen by the specialist's plan *selector*, based on input from the sponsors. The selector compares the results of the sponsors for each candidate plan, and chooses the "best" one for execution. It may look for "perfect" or "suitable" plans, as judged by the sponsors. Or, it can use a strategy of picking any plan that has not yet been tried. Or, it can impose a prioritization scheme, if more than one suitable plan has been discovered. Thus, the selector contains control knowledge in a manner similar to the messages section of a CSRL specialist.

Tasks are sequences of *steps* and possibly *constraints*. Their purpose as a DSPL construct appears to be to encourage modularity in design. The *steps* are the agents that actually make design decisions; that is, they are the agents that change the design state on the *design database*. Each step is

associated w

body of eac

values to b

the step r

include know

that is, sit

the design

suggestions

allow minor a

suggestions

that are acti

indications

tests on the

design is un

failure hand

and failure

DSPL a

varying leve

it is possi

refine it a

knowledge ba

2.4.2.1 Ove

OPM (f

Model that

[Hayes-Roth

associated with an attribute of the device being designed. The body of each step involves decisions to be made about (i.e. values to be placed in) that attribute, based on information the step retrieves from the design database. Steps also include knowledge about how to deal with *failure situations*, that is, situations in which constraints are not met during the design process. This comes in the form of *failure suggestions* and *redesigners*. Redesigners are constructs that allow minor adjustment of decisions made by the step. Failure suggestions are essentially output parameters from the step that are active if the step fails, and give the calling task indications on how to deal with that failure. *Constraints* are tests on the design state that, if failed, indicate that the design is unsatisfactory. This DSPL construct also includes failure handling capabilities in the form of failure messages and failure suggestions.

DSPL allows the design process to be accomplished at varying levels of abstraction and completeness. For example, it is possible to perform a "rough design" first, then later refine it according to the constraints defined in a DSPL knowledge base.

2.4.2.1 Overview of the OPM/BB1 approach.

OPM (for Opportunistic Planning Model) is a computer model that simulates human errand-planning behavior (Hayes-Roth and Hayes-Roth 1979). It is a blackboard-based

model desig

cognitive pr

BB1 (Ha

(Nii 1986),

engineers t

knowledge ir

representati

evolved from

OPM was

which had bee

(Erman et al.

(Nagao et al.

Planning, w

predeterminat

at achieving

refinement a

with intuiti

Here is whe

differs from

is trying to

complete kno

to rely on a

and BB1 sys

complete or

important pr

Rather, they

model designed to illustrate the opportunistic nature of the cognitive processes humans employ when constructing plans.

BB1 (Hayes-Roth 1984), which evolved partially from OPM (Nii 1986), is a blackboard-based shell that allows knowledge engineers to specify control knowledge as well as domain knowledge in a blackboard framework. The control knowledge representation of BB1 is similar to that of OPM and presumably evolved from it.

OPM was the first attempt to take the blackboard model, which had been traditionally applied to signal interpretation (Erman et al. 1980; Nii et al. 1982) and scene understanding (Nagao et al. 1980), and use it for a planning application. Planning, which Hayes-Roth (1979) defines as "the predetermination of a course of action aimed at achieving some goal (p. 275)," involves top-down successive refinement and bottom-up data-driven reasoning, interspersed with intuitive refocusing of attention as the plan unfolds. Here is where Hayes-Roth's view of planning (and design) differs from the routine design approach of Brown. Hayes-Roth is trying to tackle planning tasks without recourse to having complete knowledge of the planning process, and wishes instead to rely on an incremental approach to plan synthesis. The OPM and BB1 systems "typically forego efforts to predetermine complete or correct control procedures that anticipate all important problem solving situations (Hayes-Roth 1984, p.3)." Rather, they allow control knowledge to be

incomplete, and represented in the same blackboard-and-knowledge-source framework as domain knowledge. As Hayes-Roth puts it:

> "While not incompatible with successive-refinement models, our view is somewhat different. We share the assumption that planning processes operate on a two-dimensional planning space defined on time and abstraction dimensions. However, we assume that people's planning activity is largely opportunistic...For example, a decision about how to conduct initial planned activities might illuminate certain constraints on the planning of later activities and cause the planner to focus attention on that phase of the plan. Similarly, certain low-level refinements of a previous, abstract plan might suggest an alternative abstract plan to replace the original one. (Hayes-Roth 1979 p.276)."

Much of the evidence for these conclusions came as a result of protocol analysis of human subjects performing errand planning tasks. Hayes-Roth found that "...the planner does not plan strictly forward in time. Instead, he plans temporally-anchored sub-plans at arbitrary points on the time dimension and eventually concatenates the subplans (p.284)." As a result, when asked to explain their planning process, "planners will produce many coherent decision sequences, but some less coherent sequences as well (p.276)."

OPM and BB1 express this opportunism in their control knowledge by using a blackboard framework to represent that control knowledge. OPM has two blackboard planes for representing the control decisions of the planner, whereas BB1 has a single control blackboard. As they are similar in their structure, I will describe OPM's blackboard first, and point

out importa

blackboard

abstraction

plane, and

decisions o

and corresp

database i

hierarchical

procedures,

the system

abstraction.

The pl

attributes o

of actions t

directly to

The kno

the specific

OPM, this k

topology of

errands, and

Contro

planes. The

which establ

and point t

focus, which

focus attent

out important discrepancies with BB1 when necessary. OPM's blackboard is divided into five planes, the plan plane, plan abstraction plane, executive plane, meta-plan plane, and knowledge-base plane. The plan plane contains the decisions of actions the planner intends to take on the world, and corresponds to the design-state portion of the design database in Brown's DSPL system. The plan plane is hierarchically decomposed into four levels: outcomes, designs, procedures, and operations. Thus, the final design (plan) that the system produces is expressed at several levels of abstraction.

The plan abstraction plane characterizes desired attributes of potential plan decisions. It indicates the kinds of actions to take, and consists of four levels corresponding directly to the four levels of the plan plane.

The knowledge base plane contains domain knowledge about the specific problem the planner has to act on. In the case of OPM, this knowledge was of errands that needed to be done, topology of the geographic area in which to perform the errands, and possible routes through that area.

Control knowledge resides in the executive and metaplan planes. The executive plane has three levels: priorities, which establish principles for allocating cognitive resources and point to general areas of the blackboard to focus on; focus, which specifies where specifically on the blackboard to focus attention on; and schedule, which resolves remaining

conflicts a

and essenti

constructin

control kno

the form of

BB1 general

separate, wh

control kno

modified by

and referee.

director use

knowledge at

from the focu

level. Howev

blackboard

specialists

is in the k

is in the

of the cor

of the so

In princi

a blackbo

even to m

example, de

Plan *plane* m

the *focus* le

conflicts among executable specialists (knowledge sources), and essentially contains the agenda of actions to take in constructing the plan. Note that in contrast to DSPL's control knowledge, which consists of many local "schedules" in the form of plans and tasks, OPM's schedule is global. OPM and BB1 generally attempt to keep domain and control knowledge separate, which is different from the Ohio State approach. The control knowledge on the executive plane is generated and modified by special control knowledge sources called director and referee. These act in primarily a top-down manner. The director uses knowledge at the priority level to alter knowledge at the lower focus level. The referee uses input from the focus level to make decisions on the lowest schedule level. However, this top-down structure is not imposed by the blackboard model. There is no explicit hierarchy of the specialists themselves, unlike the DSPL model. The hierarchy is in the blackboard (data-base) only;.that is, the hierarchy is in the knowledge of the domain (the knowledge-base plane), of the control structure (executive and metaplan planes), and of the solution space (the plan and plan-abstraction planes). In principle, a specialist can use input from a lower level of a blackboard plane to make decisions on a higher level, or even to make decisions on a totally different plane. For example, decisions that are made on the outcome level of the plan plane may trigger knowledge sources that make changes to the focus level of the executive plane. This capacity for

specialists

the design

different

"opportunis

behave more

a strict top

As a re

tackle desig

about proble

In other word

tasks more co

DSPL deals

character of

The n

capacity fo

the solutio

locations,

OPM's and

at the c

explanati

problem s

including

heuristics

focus) and

current act

heuristics co

specialists that are triggered by data at certain points of the design and control space to make decisions at totally different locations gives the blackboard model an "opportunistic", "intuitive" flavor, and makes the system behave more like a committee of independent agents than like a strict top-down refinement of an abstract plan.

As a result, the blackboard model makes it possible to tackle design and planning tasks where complete knowledge about problem-solving sequences may not be known in advance. In other words, OPM seems to be trying to deal with planning tasks more complex than the Class 3 routine design tasks that DSPL deals with. This is part of the apples-and-oranges character of a comparison between OPM/BB1 and DSPL.

The non-hierarchical specialist organization, the capacity for specialists to take input from one location of the solution space and produce output at totally different locations, and the agenda-based control loop all contribute to OPM's and BB1's opportunistic behavior. This is done, however, at the cost of coherent decision paths, which can make explanation difficult in such a system. BB1 explains its problem solving actions in terms of a "dynamic control plan" including the current scheduling rule, the operative control heuristics, the current action (knowledge source or blackboard focus) and its priority, and a rating of how well the current action matched up with the operative control heuristics compared with other candidate actions. (Hayes-Roth

1984, p

An

terms c

control

applica

allevia

This is

constra

based u

formali

and de

opportu

2.4.2.2

As

DSPL ha

purpose

decisio

design

assigne

by the

the unf

plan pl

to att

parlance

sources.

1984, p. 4).

Another cost of the opportunism of blackboards is in terms of computational efficiency. Each time through the control loop, every knowledge source must be re-evaluated for applicability (although dividing the blackboard into planes alleviates this situation somewhat) (Hayes-Roth 1979, p.303). This is in contrast to the DSPL approach, which severely constrains the possible actions to take at each iteration, based upon the specialist hierarchy and the strict top-down formalism. Therefore, a routine design approach to planning and design will generally be more efficient than an opportunistic approach.

2.4.2.2 Comparisons of Design/Planning Methods

As mentioned earlier, many of the language constructs of DSPL have similar corresponding constructs serving similar purposes in OPM and BB1. For example, DSPL writes its design decisions to a portion of the design database containing the design state. These decisions are in terms of values that are assigned to attributes of the design state, and are generated by the steps. Similarly, in OPM, the design decisions (i.e. the unfolding plan that is being developed) are written to the plan plane of the blackboard, in the form of values assigned to attributes (often called hypotheses in blackboard parlance), and these values are generated by knowledge sources. Also, note that both DSPL and the blackboard approach

provide s
determina
example,
evaluate
approach
precondi
similari
part of
"rough"
plan ca
CPM, th
and the

Ho
far
the cor
For exa
of top
blackbo
agenda
knowled
specia
choose
situati
schedul
schedul
executa

provide specialists and action-taking modules with built-in determinants of their appropriateness for the situation. For example, in DSPL, each plan has an associated sponsor to evaluate that plan's appropriateness. In the blackboard approach, each knowledge source has a corresponding precondition portion to make this evaluation. Another similarity between the two approaches is that both have, as part of their reasoning processes, development of an interim "rough" design. For DSPL, there is an actual type of design plan called a rough design plan that accomplishes this. In OPM, the rough design is built on the plan abstraction plane, and the higher levels of the plan plane itself.

However, despite the similarities, the differences are far more pronounced. Probably the biggest difference is in the control and scheduling regime the two approaches employ. For example, DSPL's basic control architecture is in the form of top-down refinement of a solution tree, whereas the blackboard model uses a simple control loop updating a global agenda, or schedule. n DSPL, the domain-specific control knowledge is represented locally, within the domain specialists, where each specialist employs a plan selector to choose among competing plans whose appropriateness to the situation had been judged by their sponsors. In OPM, the scheduling is global. The executive plane contains the schedule (called a to-do set in BB1) which contains a list of executable knowledge sources. Higher levels of the executive

blackboard

generated

selection o

control kno

mostly in

accomplish

global to t

of control

A thi

there is n

DSPL's sp

problem-re

mirrors th

being des

until its

blackboar

need the

the hier

invocatio

of the bl

invoked

flavor of

A fourth

failure-

This is

blackboard (e.g. focus) contain control decisions previously generated by control knowledge sources, which guide in the selection of scheduled knowledge sources. Thus, whereas DSPL's control knowledge is local to the specialists and is expressed mostly in terms of actual sequences of actions to take to accomplish the design goals, OPM and BB1 control knowledge is global to the system as a whole and is expressed in the form of control heuristics for scheduling decisions.

A third difference is that in OPM and BB1, unlike DSPL, there is no explicit hierarchy of the specialists themselves. DSPL's specialist hierarchy ties in with its top-down problem-reduction approach. The specialist hierarchy usually mirrors the device hierarchy of the configuration (or plan) being designed. Thus, a given specialist cannot be invoked until its parent specialist has already been invoked. The blackboard model, with its agenda-based control loop, does not need the specialists to be arranged hierarchically to mirror the hierarchy of the domain and solution spaces. The invocation of a knowledge source depends solely on the state of the blackboard, not directly on which knowledge source was invoked previously. This is one way that the opportunistic flavor of the system's behavior can be expressed.

A fourth difference is that DSPL explicitly represents failure-handling mechanisms as constructs in the language. This is an example of how task-specificity allows an

expression

task at ha

solving met

as does BB1

constructs

2.4.2.3 *Fir*

The ab

and design

shortcoming

explicit pr

design step

simple desi

provide co

because of

class of de

the opportun

more difficu

often at the

decision pat

The que

the explicit

OSPL includi

the capacity

event-driven

two possible

expression of the knowledge in forms that are natural for the task at hand. The blackboard model, being a general problem solving method, expresses knowledge in a more generic fashion as does BB1, and OPM does not appear to have explicit planning constructs dealing with failure, despite its task-specificity.

2.4.2.3 Final Thoughts About Comparison Between DSPL and OPM

The above analysis of top-down vs opportunistic planning and design indicates that each approach has its merits and shortcomings. DSPL, with its capacity to represent rich and explicit problem-solving knowledge in the form of sequences of design steps and failure-handling strategies, can perform simple design tasks in a computationally efficient manner and provide coherent explanation of its reasoning. However, because of its tightly constrained control structure, the class of design tasks it can solve is limited. By contrast, the opportunistic flavor of OPM and BB1 enable them to tackle more difficult and "interesting" planning and design tasks, often at the expense of computational efficiency or coherent decision paths.

The question is, can we have it both ways? Can we merge the explicit, ontologically rich and task-specific nature of DSPL including its top-down refinement control structure, with the capacity to make intuitive leaps of reasoning via the event-driven control structure of blackboard models? I see two possible methods for doing this.

The f

into the

Hayes-Roth

could be t

model. Wit

would be

constraine

refinement

The

reasoning

example, i

not descer

incorporat

design sta

add to DSP

control, a

event-driv

structure

the abilit

to tackle

Which

human des

of "creat

for simp

"creativit

class 3 ta

The first is to incorporate top-down control heuristics into the blackboard framework. This was suggested by Hayes-Roth (1985 p.307), who said that the top-down model could be thought of as a special case of the opportunistic model. With this approach, the major thrust of the system would be opportunistic, but the opportunism would be constrained by control heuristic which enforce focusing via refinement.

The second approach is to incorporate opportunistic reasoning into the overall control regime of DSPL. For example, if specialists can call other specialists that are not descendants of themselves, and if the reasoning could be incorporate event-driven features such that changes to the design state trigger specialists into action, then this would add to DSPL's flexibility. By maintaining the overall top-down control, and only interspersing it with occasional event-driven reasoning, this would preserve most of the structure and computational efficiency of DSPL, while adding the ability for "creative thought" and therefore allowing DSPL to tackle more difficult design problems.

Whichever tactic one uses, it is clear that any theory of human design and planning must take into account the notion of "creativity" and "intuition". I believe this is even true for simple, "routine" tasks. Humans don't turn off their "creativity function" when faced with easy design tasks (i.e. Class 3 tasks in Brown's terminology). The fact that one has

extensive

design pro

make use o

example, i

that was v

use a top-

I would ta

choose to

This is a

use to ci

thereby

cognition

that it c

2.4.3 Abd

Abdu

for the d

For examp

following

Thus, th

Deductive

extensive knowledge of the problem-solving needed for a given design problem does not imply that this person would fail to make use of unexpected opportunities in the information. For example, if I had designed a simple computer program last week that was very similar to one I must create today, I would not use a top-down strategy to build today's program...rather, I would take advantage of my opportunity to be lazy, and would choose to make minor modifications to last week's program. This is an example of opportunistic reasoning that DSPL could use to circumvent its top-down plan refinement approach, and thereby come closer to representing a valid theory of cognition as well as expanding the class of design problems that it can address.

2.4.3 Abductive Assembly

Abduction reasoning is a search for hypotheses to *account for* the data of a particular case. It is not deductive logic. For example, in deductive logic, the *modes ponens* rule is the following:

Given:
if p then q
p
Conclude:
q

Thus, the evidence p logically implies the hypothesis q. Deductive logic proves q with certainty.

By co

In this ca
it cannot
be accoun
of reason
logic, bu
solving a
reasonin

Punc
method f
its imple
is a me
antibodi
on RED's
environm

The
a list o
PEIRCE
abductiv
the find
three re
should
maintain

By contrast, abductive reasoning looks like this:

Given:
if *p* then *q*
q
Conclude:
p

In this case, *p* is the hypothesis and *q* is the evidence. Here, it cannot be said that *q* logically implies *p*. Instead, *q* can be accounted for, or explained, by the hypothesis *p*. This type of reasoning does not have the logical certainty of deductive logic, but is nonetheless a useful heuristic for much problem-solving activity. A prime example is its use in diagnostic reasoning.

Punch et al. (1990,1991) described a problem-solving method for abduction in the framework of a generic task, and its implementation in two expert systems, *RED* and *PEIRCE*. RED is a medical diagnostic system that identifies red-blood antibodies in parient sera. PEIRCE is a shell, based largely on RED's abductive method, serving as an knowledge-engineering environment for representing abductive knowledge.

The particular method used is *abductive assembly*. Given a list of possible hypotheses (which, in the case of RED and PEIRCE are provided via hierarchical classification) the abductive assembler selects a subset of hypotheses to explain the findings. This *compound hypothesis* is generated based on three requirements: it should cover all the findings, it should use the most plausible hypotheses, and it should maintain consistency and compatibility between hypotheses.

RED's abd

al., is as

The fi

module is t

criteria: p

temporarily

one (starti

see if the

that covers

redundant a

on finding

essential

findings t

essential h

a good thin

PEIRCE

of RED's alg

RED's abductive assembly algorithm, as described by Punch et al., is as follows:

1) Select a finding that needs explaining.
2) Find the list of hypotheses (output from hierarchical classifier) that account for the finding.
3) Pick the most plausible one.
4) Try to integrate the chosen hypothesis with the existing compound hypothesis, based on compatibility constraints.
5) If compatible, add it to the compound hypothesis and mark the finding as explained.
6) If incompatible, either go back to step 3 (ie pick the next-most plausible hypothesis), or remove the incompatible hypothesis from the compound hypothesis and unmark its findings.
7) Update other findings from the integration.
8) Loop back to step 1. (pp. 8-9).

The final compound hypothesis generated by RED's assembly module is then critiqued (i.e. evaluated) based on two main criteria: *parsimony* and *essentiality*. Parsimony is enforced by temporarily removing hypotheses from the compound list one by one (starting with least plausible hypotheses), and testing to see if the compound hypothesis still provides an explanation that covers all the findings. If so, the removed hypothesis is redundant and unnecessary. The essentiality critique is based on finding which hypotheses in the compound are absolutely essential (i.e. there is no other way of explaining the findings than to use this hypothesis). An abundance of essential hypotheses in the compound hypothesis is considered a good thing in this evaluation.

PEIRCE's main contribution was in breaking the components of RED's algorithm into general-purpose building blocks and in

providing

execution

abduction

parsimony

sponsor-se

provide th

module has

table) to

then compar

most approp

ties, the s

In additio

PEIRCE's sp

execution

hierarchic

gathering

The

recommen

describe

fragmen

assembl

fragmen

list of

the *hier*

suppressio

recommenda

providing a flexible control strategy for scheduling the execution of these building blocks. The behaviors of RED's abduction algorithm, such as hypothesis integration and parsimony critique, were broken into behavior modules. A sponsor-selector mechanism, similar to DSPL's, was used to provide the control strategy. In particular, each behavior module has a sponsor (represented as a pattern-matching truth table) to evaluate and rate its appropriateness. A selector then compares the scores from each sponsor, and chooses the most appropriate behavior to execute next. In the case of ties, the selector chooses based on a-priori ordering lists. In addition to controlling abductive behavior modules, PEIRCE's sponsor-selector mechanism was used to control the execution of the various generic tasks themselves (e.g. hierarchical classification, abduction assembly, and data gathering).

There are some similarities between abduction and the recommendation-generating algorithm used in CEVAL, which is described in detail in chapter 5. CEVAL's "recommendation fragments" are like the "candidate hypotheses" of abductive assembly. CEVAL generates a list of all the recommendation fragments whose conditions were met. This is analogous to the list of hypotheses that the abductive assembler receives from the hierarchical classifier in RED of PEIRCE. CEVAL uses suppression links to maintain consistency in the recommendation, much as the abduction assembly's "integration"

phase. (A

handling)

implement

parsimony

context-sp

parsimony

2.4.4 *Funct*

Anoth

called *func*

to a kin

structure,

work in fu

had been d

systems, u

reasoning

work in t

functiona

this work

task in de

Deep

interest.

knowledge

complex kr

of thumb"

tasks, th

phase. (Actually, both are implementing a form of constraint-handling). CEVAL's suppression links are also used to implement "parsimony" requirements, much the same as RED's parsimony critique. In particular, suppression based on context-specificity and abstraction-level are motivated by parsimony considerations.

2.4.4 Functional Reasoning

Another generic task type developed at O.S.U.'s LAIR is called *functional reasoning* (Sticklen et al. 1989), and refers to a kind of qualitative simulation of the behavior, structure, and function of devices. The ideas behind O.S.U.'s work in functional reasoning evolved out of previous work that had been done in the area of "deep reasoning" systems. These systems, unlike compiled knowledge representations, operate by reasoning form "first principles". Since there has been much work in this area prior to the development of Sticklen's functional reasoning system, it is helpful to review some of this work before describing the functional reasoning generic task in detail.

Deep reasoning has long been a topic of research interest. Its motivation derives from the fact that compiled knowledge is often not robust enough to serve the needs of complex knowledge domains. Although experts usually use "rules of thumb" gained from years of experience in performing their tasks, they sometimes must resort to "first principles" when

confr

devel

expla

syste

deep

break

funct

typif

base

and

cons

is c

prov

doma

know

comp

in c

Used

atten

confronting tough problems. Another motivation for the development and use of deep reasoning is its application in explanation, an important component in any good knowledge base system.

A question may arise, what is the distinction between deep and compiled knowledge? Michie (1982) gives one possible breakdown. Compiled (or "low-road") reasoning involves a function like this:

SITUATION ----------> ACTION

This is called a *heuristic* machine representation, and typifies what we commonly know as expert-system knowledge bases.

Deep ("high-road") knowledge takes this form:

SITUATION & ACTION ----------> NEW SITUATION

and is a *causal* machine representation. Actually, this looks considerably like a finite-state automaton.

Another distinction between compiled and deep knowledge is cited by Sticklen (1987). A deep approach is one which provides a way to derive the *assumptions* under which its domain knowledge holds. In other words, it is not enough to know that *A* causes *B*. One must also know WHY *A* causes *B*.

Deep reasoning is computationally expensive compared to compiled knowledge, and runs the risk of getting bogged down in computability and complexity problems. Thus it should be used sparingly (Michie 1982). Typically, an expert system will attempt to solve a problem using a compiled knowledge base

firs

the

2.4.

abou

Semb

reas

reas

reas

of t

bein

seve

repr

limi

devi

repr

comp

comp

devi

eith

ment

devi

aspe

first...if this fails to produce a solution, it will resort to the deep reasoning component.

2.4.4.1 O.S.U. LAIR's Functional Reasoning...being explicit about purpose

Sticklen (1987), Chandrasekaran (1983), and Sembugamoorthy and Chandrasekaran (1986), describe a level of reasoning between surface compiled reasoning and the deep reasoning of qualitative simulation. This is called *functional reasoning*, and is distinguished by its explicit representation of the function, or purpose, of the device whose behavior is being simulated.

Sticklen's functional representation scheme is founded on several intuitive notions about how causal reasoning should be represented (Sticklen et al. 1989). First, there is a limitation to representations of physical *devices* instead of device properties or attributes. Second, device representations can be recursively decomposed into device components, so a functional representation must include this composition capability. Third, the major understanding in the device representation pertains to *functionality* more than to either structure or behavior, in contrast to Davis' model mentioned below. These all lead to a concern with representing devices in terms of the functionality of their components.

Sticklen's functional representation involves three main aspects of a device: its *structure*, *function*, and *behavior*.

The struc

its compo

providing

microscop

involves t

1) a state
2) a list
place.
3) a list

Thus,

of a dev

functiona

We also e

that fun

descript

represer

Th

the pu

descri

repres

infor

state

point

where

causes

Chandra

legitim

The structure is expressed as a breakdown of the device into its components, and their breakdown into subcomponents, thus providing a hierarchical structure from macroscopic to microscopic. The function of a device (and of each component) involves three facets:

1) a statement of the function, or purpose, of the device.
2) a list of preconditions necessary for the function to take place.
3) a list of behaviors by which the function is carried out.

Thus, we see that by explicitly representing the function of a device, we enable a higher-level description of functionality without having to go into details of behavior. We also explicitly state all underlying assumptions enabling that functionality. Finally, we enable a deeper-level description by indexing into detailed behavioral representation if necessary.

The behavioral aspect of a device's representation serves the purpose of showing state changes of a device, and describing how and why these changes take place. Behavior is represented as a tree whose nodes describe three types of information: *state-variable predicates*, *state-variable-change statements*, and *knowledge pointers*. If we ignore the knowledge pointers, this representation would be like a causal net, where links indicate that the state described by one node causes the state change at the next. As Sticklen (1987) and Chandrasekaran (1983) point out, however, this could not legitimately be called a "deep" representation, as it does not

explicit

states.

pointers.

state-var

another. H

other dev

(pointing

indication

From

we see wh

knowledge

successful

Second, t

explicitly

Now

functional

which it d

reasoning

finding,

"envisionm

The c

steps:

1) Specify
2) Determi
indexin
conditi
behavio
3) Constru
expandi
partial

explicitly represent the *reason* for the causal link between states. Thus, we see the justification for knowledge pointers. These are inserted in the graph between each pair of state-variable nodes, and serve to explain why one state cause another. Knowledge pointers can be decomposable (pointing to other device functions or behaviors) or non-decomposable (pointing to statements about world knowledge, definitions, or indications that the reasons for the causal link are unknown.

From the above description of functional representation, we see why it can legitimately be classified as "deep" knowledge representation. First, the assumptions underlying successfully carrying out a function are explicitly stated. Second, the reasons for causal links between states are explicitly stated via knowledge pointers.

Now that I've described the representation of the functional system, I will discuss the reasoning process by which it determines consequences of initial conditions. This reasoning process is called, appropriately enough, *consequence finding*, and can be compared with Kuipers' and DeKleer's "envisionment" algorithm.

The consequence finding algorithm involves the following steps:

1) Specify initial conditions and unavailable functions
2) Determine starting (invocable) functions and behaviors by indexing candidate functions of behaviors via starting conditions and filtering out inferior functions and behaviors.
3) Construct a partial state diagram (PSD) by recursively expanding all decomposable knowledge pointers until only partial state transitions are there. During this process,

assum
4) From
trave
5) Chang
itera
6) Go to

in that

constrai

behavio

satisfy

Also, a

branchi

each no

state,

variabl

in the

state

S

an aut

Compil

2.4.4.

A

devel

tackl

elect

Davis

assumptions are being accumulated.
4) From the PSD, determine composite device changes by traversing the PSD and changing variables.
5) Changes from 4 become initial conditions for the next iteration
6) Go to 2.

This reasoning approach is similar to envisionment in that it builds a state space. However, it does not use constraint propagation, but rather *indexes* into possible behaviors and functions based on initial conditions which satisfy the preconditions of these behaviors or functions. Also, as far as I can see, there is no nondeterministic branching in this method. Also, note the important fact that each node of the state diagram does not represent a full state, but a partial state, ie a change to a particular state variable. This is in contrast to envisionment, where each node in the state space represents a full
state change in the device.

Sticklen showed that functional reasoning can be used as an automated knowledge acquisition method in order to derive compiled rules based on qualitative, functional simulation.

2.4.4.1 Davis' Model Based Diagnosis Approach

A similar method was described by Davis (1984), who developed a system that reasons from first principles for tackling diagnostic problems in the domain of digital electronic circuits. Like the functional reasoning approach, Davis' method was to make explicit all assumptions underlying

the proper performance of the device in question, and enumerate all these assumptions. His troubleshooting activity can be described as a methodical enumeration and relaxation of underlying assumptions about the device (via a technique called *constraint suspension*), with subsequent consideration of the consequences of violating each of these assumptions, thus leading to generation of a list of candidate reasons for the device's failure. This is similar in many ways to the Sticklen's consequence-finding algorithm, mentioned above.

2.4.5 Structured Matching

Bylander et al. (1988) described a generic problem-solving method called *structured matching*, which is essentially a method for evaluating "goodness of fit", or to use Bylander's terminology, "recognition". The architecture of structured matching involves a hierarchy of "matchers", or truth tables, each mapping a limited set of parameter-value pairs onto a decision for that matcher. Parameters can be data about the world, or can be outputs from lower-level matchers. Each row of the truth table includes a conjunctive clause of the parameter-value pairs and the resulting output value that occurs if the clause is satisfied. The different rows of the truth table have a disjunctive relationship to each other, with each row containing a different alternative output value.

The inferencing action of structured matching is a goal driven top-down traversal of the matcher hierarchy. The goal,

at

mat

par

lev

con

mat

2.4

int

whi

inp

eva

fun

a c

mat

whi

dir

pos

pos

fea

and

pos

the

tha

ove

at each level, is to determine the output value of the matcher. This is done by examining the truth table. If parameter values in the truth table are determined by lower level matchers, then those matchers are examined. This process continues recursively until the value for the top-level matcher can be determined.

2.4.5.1 Samuel's Signature Tables

Structured matching is very similar to a method introduced by Samuel (1967), called *signature table analysis*, which involves a hierarchy of matchers which map patterns of input values onto some output score measuring a "goodness" evaluation. Samuel used this technique as a static evaluation function for evaluating potential checker board positions for a checker-playing computer program. In Samuel's method, each matcher is a *n*-dimensional array (called a *signature table*) which contains the scores for various board positions. Each dimension of the array represented a feature of the board position. An array's dimension would be divided into as many positions as there were possible values of the corresponding feature. Thus, if a feature could take on values -2, -1, 0, 1, and 2, the corresponding array dimension would have five positions. Each cell in the array contained a value which was the score corresponding with the combination of feature values that mapped onto that cell. In this way, a board position's overall score was specific to the *non-linear combination* of

feature

A

intera

space-

with a

would

this w

values

featu

level

board

a gre

signa

featu

was

featu

rang

of s

asse

a re

of p

to h

noti

line

sign

feat

features that the board position took on.

Although this technique was able to account for interaction between features, it introduced a significant space-complexity problem. For a large number of features, each with a large number of potential values, the size of the array would grow to a prohibitive expanse. Samuel's solution for this was two-fold. First, he restricted the number of possible values that a feature could take on. Second, he arranged his features into a hierarchy of signature tables. At the lowest level, the signature tables consisted of subsets of the board-position features themselves, with each feature having a greatly restricted range of possible values. Higher-level signature tables used lower-level tables as "composite features", and because the number of these lower-level tables was considerably smaller than the number of board position features themselves, the table-outputs could have a larger range of possible values. Thus, through the use of a hierarchy of signature tables, Samuel was able to arrive at accurate assessments of the worthiness of potential board positions in a reasonable amount of time. This greatly improved the quality of play of his checker program, and significantly contributed to his research in machine learning. Note, however, that any notion of "weighted scoring", the primary activity of the linear polynomial method, was totally abandoned in the signature table approach. This forced the space of potential feature values to be discrete, whereas the linear polynomial

appro

signa

task)

areas

appro

resea

class

analy

mult

2.4.

of i

mult

stru

refi

mech

In

attr

The

We w

attr

line

used

disc

approach allows for a continuous space.

Although Bylander was the first to generalize the signature table approach into a knowledge acquisition (generic task) tool, signature tables have been used widely in other areas of AI. For example, Page (1972, 1977) extended Samuel's approach to apply to pattern recognition tasks. In his research, signature tables were constructed in order to classify patterns in health screening and urban housing analysis domains. He found that such methods wresuperior to multiple regression for certain types of predictive tasks.

2.4.5.2 Another Approach to Structured-Matching's IPT

Both structured matching and signature tables are methods of implementing a sort of evaluation based on assessment of multiple attributes of an individual. In particular, structured matching is an integral part of CSRL's establish-refine operations and DSPL's and PEIRCE's sponsor-selector mechanism, whereby the most "promising" options are selected. In essence, structured matching is doing a form of multi-attribute utility assessment via a pattern-matching algorithm. The same is true of Samuel's signature tables. In Chapter 4, we will see that there are other possible approaches to multi-attribute utility measurement. In particular, hierarchical linear models (using weighted algebraic methods) have been used to perform the same sorts of tasks. Chapter 5 will discuss the Candidate Evaluation architecture, which uses a

hierarch

attribu

2.5 Oth

Th

philosc

there

In thi

the KA

2.5.1

Carne

gener

TSAs

McDe

acqu

narr

solv

cont

198

typ

Thu

McD

two

hierarchical quasi-linear model (HQLM) to perform multi-attribute utility assessment.

2.5 Other Approaches to Task-Specific Architectures

Thus far, I have concentrated mainly on the Generic Task philosophy in describing task-specific architectures. However, there has been significant work by other researchers in TSA. In this section, I will describe two other approaches to TSA, the KADS approach and McDermott's approach.

2.5.1 TSA work done by McDermott and Colleagues at DEC and Carnegie Melon

Just as Chandrasekaran's group at O.S.U. is doing work in generic tasks, McDermott and others are doing parallel work in TSAs at DEC and CMU (Marcus 1988). Like Chandrasekaran, McDermott's group is interested in facilitating the knowledge acquisition process by developing tools which solve rather narrowly-defined tasks using specific "role-limiting" problem solving methods (McDermott 1988). Role-limiting methods are contrasted with Newell's "weak methods" (Laird and Newell 1983) in that they are not generalizable across all task types, but rather are focussed on a particular task type. Thus, there is a similarity in the approaches taken by McDermott and Chandrasekaran.

However, there is also a subtle difference between the two research strategies, as described by Boose (1989).

Chan
gene
impl
McDe
deve
prob
othe
more

McDe
Chan
the
gene
the
some
expe
expe
repr
gene
whos
gene
the

in t
fact
gener

Chandrasekaran's team is primarily concerned with identifying *generic* problem solving methods and developing languages to implement these methods. In contrast, the approach of McDermott's group tends to focus on a *specific* problem, develop knowledge acquisition methods for solving that problem, and only later attempt to generalize the problem to other related problems. Thus, McDermott's tools tend to be more domain-oriented than Chandrasekaran's tools.

Another difference between the tools developed by McDermott and his colleagues and those developed by Chandrasekaran's team is the nature of the interaction between the tool and the expert. McDermott et al.'s tools are generally interactive knowledge acquisition tools that guide the expert via a question-and-answer interviewing process and sometimes allow for second-guessing and validation of the expert-generated knowledge. These KA tools often convert the expert-generated knowledge into more traditional representation paradigms, such as rules. By contrast, the generic task tools are generally "programming languages", whose constructs and primitives are task-specific. Thus, the generic task tools are not complete KA tools *per se*, unlike the tools generated by McDermott's group.

Despite these differences, there is considerable overlap in the methods they employ, as will be described below. This fact leads to implications that support the notion of TSAs in general. It also supports the idea that there are certain

proble

that c

tools

tools

2.5.1

heuri

solvi

diff

is a

The

the

uses

that

disc

in

(or

the

"sy

hie

nod

Pri

problem-solving methods that are used repeatedly by humans and that can be used to more easily generate expert systems.

The following sections will briefly describe some of the tools developed at DEC and CMU, comparing some of them to tools developed by OSU researchers.

2.5.1.1 MOLE: A Tool for Cover-and-Differentiate Systems

MOLE (Eshelman 1988) is a KA tool for developing heuristic classification expert systems using a problem-solving method called *cover-and-differentiate*. Cover-and-differentiate is analogous to abductive inference, in that it is a method whose purpose is to explain findings or symptoms. The PS method involves finding all hypotheses that account for the findings or symptoms of the case (this is cover). Then, it uses heuristics to select the "best" hypotheses out of those that cover the findings (differentiate). The reader can discern that this is essentially the same activity occurring in RED or PEIRCE described above.

MOLE's underlying representational structure is a network (or more accurately tangled hierarchy) of nodes. The nodes at the bottom level represent the possible "findings" or "symptoms" of a problem. Nodes at higher levels in the hierarchy represent hypotheses or compound hypotheses. Root nodes represent the ultimate or final explanations.

The covering activity is guided by the *exhaustivity principle*, which states that if an event has at least one

pote

leas

finc

heu

pat

the

cov

gui

dep

pro

suf

Bay

pr

an

im

in

de

ev

"f

Th

co

qu

ve

ar

potential explanation, the final diagnosis must include at least one of these potential explanations. This means that any finding that *can* be explained by the system *must* be explained.

The differentiating activity is guided by several heuristic principles. First, for a given finding, a single path leading to a top-level explanation is preferred. Second, there should be as few top-level explanations as possible to cover the findings. Third, the covering explanations should be guided by Bayesian principles of *independence* and *dependability*. Specifically, an explanation's "prior probability" and its "conditional probability" should be sufficiently high. (See chapter 3 for more discussion on Bayesian reasoning systems). Note that the first two principles are consistent with PEIRCE's *parsimony principle* and *essentiality principle*.

The third principle, dealing with Bayesian issues, is implemented using *event-qualifying* knowledge (for independence) and *connection-qualifying* knowledge (for dependability). Event-qualifying knowledge is represented by evidence nodes in the network, other than the actual "findings" nodes, that are linked to the hypotheses nodes. Thus, when a hypothesis node is activated (found from the covering activity based on the findings), any attached event-qualifying nodes are tested in order to obtain independent verification of the hypothesis node. Note that the effect of an event-qualifying node is global in the sense that it

affect

findin

hand,

evider

hypoth

connec

hypoth

abili

the h

other

value

2.5.1

syste

this

desic

for t

iden

revi

Proc

This

roug

thos

para

affects the validity of the hypothesis node regardless of what finding the hypothesis is supposed to explain. On the other hand, connection-qualifying knowledge is represented by evidence nodes that are associated with the link between a hypothesis node and a findings node. Thus, the effect of the connection-qualifying node is not to validate or discredit a hypothesis, but to validate or discredit that hypothesis' ability to explain the finding in question. In other words, the hypothesis may be true, and may be able to account for other findings, regardless of the connection-qualifying node's value.

2.5.1.2 SALT: A Tool for Propose-and-Revise Systems

SALT (Marcus 1988) is a KA tool for developing expert systems that construct, rather than select, a solution. In this sense it is similar to Brown's DSPL, handling a "routine design" task. The idea is to specify values and constraints for the *design parameters* of a particular design task. Marcus identifies the kind of task being done by SALT as "propose and revise".

There are three main types of knowledge in SALT. *Procedures* are used to propose values for design parameters. This can be done via calculations or database lookups, and is roughly equivalent to the *steps* of DSPL. *Constraints*, like those in DSPL, are used to identify, for a given design parameter, the nature of limits to its value. Finally, *Fixes*

propos

violat

of tas

Howev

diffe

the t

hiera

of th

cont

In s

cons

repr

revi

natu

gen

lan

use

exp

act

pro

in

no

to

kn

propose refinements to parameters whose proposed values are in violation of a constraint. Thus, a fix performs the same sort of task as the *failure suggestions* and *redesigns* of DSPL.

Thus we see some similarities between DSPL and SALT. However, there are also significant differences. One of these differences concerns the overall control and organization of the two representations. DSPL knowledge is organized as a hierarchy of specialists, where explicit *procedural* knowledge of the design process is sequentially represented in tasks. By contrast, SALT knowledge is organized in a dependency network. In SALT's network, nodes represent the design parameters, constraints, or inputs. Nodes are connected via directed links representing "contributes-to", "constrains", or "suggests revision of" relationships between the nodes.

Another difference between DSPL and SALT pertains to the nature of the knowledge acquisition process. Like other generic-task tools, DSPL is essentially a "programming language". It is a passive shell that the knowledge engineer uses to develop an expert system. By contrast, SALT is explicitly a knowledge acquisition tool, and takes a much more active role in the ES development process. For example, it prompts the user for input values, and checks for completeness in the knowledge base. It also checks to ensure that there are no cyclic dependencies in the network. Finally, like other tools developed by McDermott's group, SALT compiles the knowledge obtained from a domain expert into a rule-base. This

is c

(inc

the

syst

2.5.

invo

docu

proc

acqu

inve

repo

str

div

and

exp

inc

and

the

mod

rep

fit

rep

rep

is considerably different from any of the generic task tools (including DSPL), where knowledge is kept in the structure of the tool and *not* converted into "first-generation" expert-systems constructs.

2.5.1.3 KNACK: A Tool for Sample-Based Report Generation

KNACK (Klinker 1988) performs the "reporting task", which involves collecting data and presenting them in the form of a document. These can be technical documents, proposals, progress reports, etc. The method used by KNACK is called *acquire-and-present*.

The *acquire* portion of the problem solving method involves interacting with the domain expert to obtain sample reports, a domain model, and sample report-generation strategies. Sample reports are typed in by the expert, then divided into fragments by KNACK. Then KNACK obtains structural and functional descriptions of the domain model from the expert, via a graphical user interface. The domain model will include generalized concepts, their relations and attributes, and instantiated values for the concept attributes based on the sample report. Next, via the sample report and the domain model, KNACK interacts with the expert to generalize the report in order to make it possible to generate reports fitting the domain model. This involves creating a skeletal report (basically, an outline obtained through the sample-report fragmentation),then replacing the fixed report text

with ge

text wi

Finally

obtaini

reports

model.

called

obtain

Thus, t

the WR

2.5.1.

S

determ

types,

comput

disk s

sizing

proces

about

recomm

T

approa

A know

cases

quant

with generalized concept (by matching a word in the report text with an instantiated concept in the domain model). Finally, report-generating strategies are developed for obtaining information from the end-user in order to generate reports fitting the generalized report structure and domain model. The strategies are implemented in reporting systems called WRINGERS, which interact with end-users in order to obtain information in a coherent manner and generate reports. Thus, the *present* part of KNACK's PS-method is performed by the WRINGERs that KNACK generates.

2.5.1.4 SIZZLE: A Case-Based Tool for Sizing Systems

SIZZLE (Offut 1988) is a system for handling the task of determining the optimal size of resources to meet the needs, types, and quantities of users. It was first developed for computer system resource sizing (combining CPU power, RAM, disk space, etc.), but could be applied to other resource-sizing problems as well. Thus the "sizing" information processing task tackled by SIZZLE it to map the input facts about the types and quantities of resource users onto output recommendations regarding the size and quantity of resources.

The method used by SIZZLE is a case-based reasoning approach that Offut calls "extrapolate from a similar case". A knowledge base generated through SIZZLE contains a set of cases. Each case contains a description of the types and quantities of the users, together with the expert-supplied

soluti

SIZZLE

demanc

requir

T

find

requir

in the

in the

ident

the c

speci

resou

based

other

malpr

sizin

2.5.1

a fin

metho

that

requir

hypot

solution in terms of the suggested size of the resource(s). A SIZZLE knowledge base also contains an expert-supplied *user demand model*, which contains information about the resource requirements for each type of user.

Thus, the knowledge base uses case based reasoning to find the existing case that is most similar to the input requirements, based on matching the types/quantities of users in the database of cases against the types/quantities of users in the current situation. After the most similar case has been identified, the user demand model is used to extrapolate from the chosen case in order to adjust the solution to the specific requirements of the current situation.

Although SIZZLE was initially developed for computer-resource sizing problems, Offut claims that its simple case-based and extrapolation mechanism can be generalized to tackle other sizing problems such as electric-motor sizing, malpractice-suit settlement sizing, and automatic copier sizing.

2.5.1.5 A Possible Way to Test the Generic Task Hypothesis

The generic task hypothesis states that there should be a finite, manageably-sized number of general problem solving methods (implementable in knowledge-based languages or tools) that can serve as building blocks for solving any problem requiring an expert-systems solution. One way to test this hypothesis is to see if the generic task tools developed at

OSU c
McDer
some
SALT,
is st
preli
emplo
devel
SALT
cove
abdu

appa
beti
gen
the
ana
not
of
Li
ca
pa
ca

t

OSU can be successfully applied to the same problems that McDermott's tools are solving. One would think that either: some combination of OSU's generic tasks can handle the KNACK, SALT, SIZZLE, and MOLE tasks; or OSU's list of generic tasks is still incomplete and needs to be expanded. A preliminary analysis suggests that many of the PS methods employed by the above-mentioned tools are analogous to methods developed at OSU. The propose-and-refine strategy employed by SALT is similar to the routine-design method of DSPL. MOLE's cover-and-differentiate appears to be analogous to PEIRCE's abductive assembly.

However, there are some cases where the mapping is not as apparent. For example, it is difficult to see a clean mapping between KNACK's acquire-and-present strategy and one or more generic tasks. I see two possible reasons for this. First, there is no generic task that deals specifically with text analysis and generation. Second, the existing generic tasks do not include any sample-based (or case-based) mechanisms. Both of these are important components of KNACK's architecture. Likewise, it is difficult to see which generic task tool(s) can be applied to the sizing problem addressed by SIZZLE, partially because of the lack of case-based reasoning capabilities in current GT tools.

Thus, a preliminary comparison of the GT tools with the tools mentioned in this section indicates that OSU's list of generic tasks should be expanded to include such capabilities.

It a
enum
if n
simi
also

2.5.

Bre
Unli
esp
fra
rep
tas
to
can
dom
old
inf
phi

met
sys
bot
ana
vie

It also raises in my mind the possibility that an exhaustive enumeration of primitive problem-solving methods is a daunting if not impossible task. Here, I have found that, despite the similarity of some methods used by the two groups, there are also some tasks that are not handled by both groups.

2.5.2 The KADS Approach: TSA research in Europe

There is another research stream in TSAs described by Breuker and Wielinga (1989) at the University of Amsterdam. Unlike the generic task school of thought and unlike the ideas espoused by McDermott and his colleagues, the European framework does not believe that domain knowledge representation is guided by the role-limiting effects of the task or problem-solving type. Rather, domain knowledge is seen to be task-independent, and different task-specific PS methods can be successfully applied to the same task-independent domain knowledge representation. This is consistent with the old ideas of a separation between the knowledge base and the inference engine, and is a significant departure from the philosophies espoused by both Chandrasekaran and McDermott.

Breuker and Wielinga describe a knowledge acquisition methodology called KADS (for knowledge acquisition and design system). KADS is motivated by the notion that the KA bottleneck is not in knowledge elicitation, but in the *analysis* of acquired knowledge. They criticize the "mining" view of KA, which states that the main task of KA is to

extract the knowledge. Instead, they argue for a "modelling" view, whereby knowledge is transformed and abstracted prior to encoding. In this sense, their approach is consistent with the methods employed by Chandrasekaran and by McDermott.

However, there is a significant difference in the methods employed to represent domain knowledge. For both Chandrasekaran and McDermott, the domain knowledge representation is dependent on its intended use. For example, knowledge to be used for classification tasks will be represented differently than knowledge used for design. This is the *interaction hypothesis* (Bylander and Chandrasekaran 1987) mentioned earlier in this chapter. In other words, there is no such thing as "task-neutral" domain knowledge. The implication is that domain knowledge that had previously been used for one task must be reformulated and restructured for use in a different task. By contrast, the KADS approach assumes that domain knowledge can be, at some level, independent of its intended use. This idea of task-neutral domain models is an integral part of KADS, and is consistent with earlier notions of a separation from inference engine and knowledge bases.

The KADS methodology is supported by the tool KCML (for KADS Conceptual Modelling Language). KADS and KCML are based on a "layered" approach to problem solving. There are four layers, the domain layer, inference layer, task layer, and strategic layer. Note the similarity between the layered

approach in KADS and the approach used in OPM and BB1 described earlier in this chapter. The layers of KCML are analogous to the planes of the blackboard, with the higher layers representing a more abstracted view of the knowledge. The domain layer involves "generic facts and models (Breuker and Wielinka 1989, p.12) and an "axiomatic framework" that produces a very general purpose representation. The inference layer, roughly analogous to Clancey's (1985) heuristic classification description, where problem-solving processes are expressed in terms of abstraction, specification, matching, assembly, etc. The inference layer provides a higher-level description of domain-layer knowledge, where the language constructs include "match", "decompose, "abstract", etc. The task layer is a hierarchy of tasks and goals, which control the actions of the inference-layer. The strategic layer is responsible for monitoring, diagnosing, and planning. It controls the actions of the task layer.

Chandrasekaran's *generic tasks* chapter appear to bridge KADS' inference and task layers. In addition, Punch's research on TIPS (1989) deals with issues related to KADS' strategic layer. However, there is no generic-task equivalent to the domain layer in KADS. In fact, the interaction hypothesis states that such layer is impossible, or at the very least impractical, as a knowledge representation. This is where the philosophies of KADS and generic-tasks collide.

2.6

appr
some
proc
moti
sugg
high
appr
dist
that
fact

sch
sol
Cha
com
dev
and
and

2.6 Conclusions about TSAs

This chapter discussed the Task Specific Architecture approach to knowledge representation. It started by discussing some "first-generation" representation paradigms, then proceeded to a review of some philosophical arguments motivating a task-specific approach. Briefly, these arguments suggest that representation schemes should be expressed at a higher level of abstraction than the first-generation approaches provide. Additionally, knowledge should be distributed and modular, and should be represented in a way that is consistent with how it is to be used. These motivating factors lead to the TSA approach.

This chapter went into detail covering the generic task school of thought and enumerated several of the problem solving methods and expert system shells developed by Chandrasekaran and his O.S.U LAIR colleagues. The tools were compared against other systems and methods that have been developed to solve similar problems, as shown in figures 2.2 and 2.3. Generally, they were found to have greater structure and more explicit ontology than their

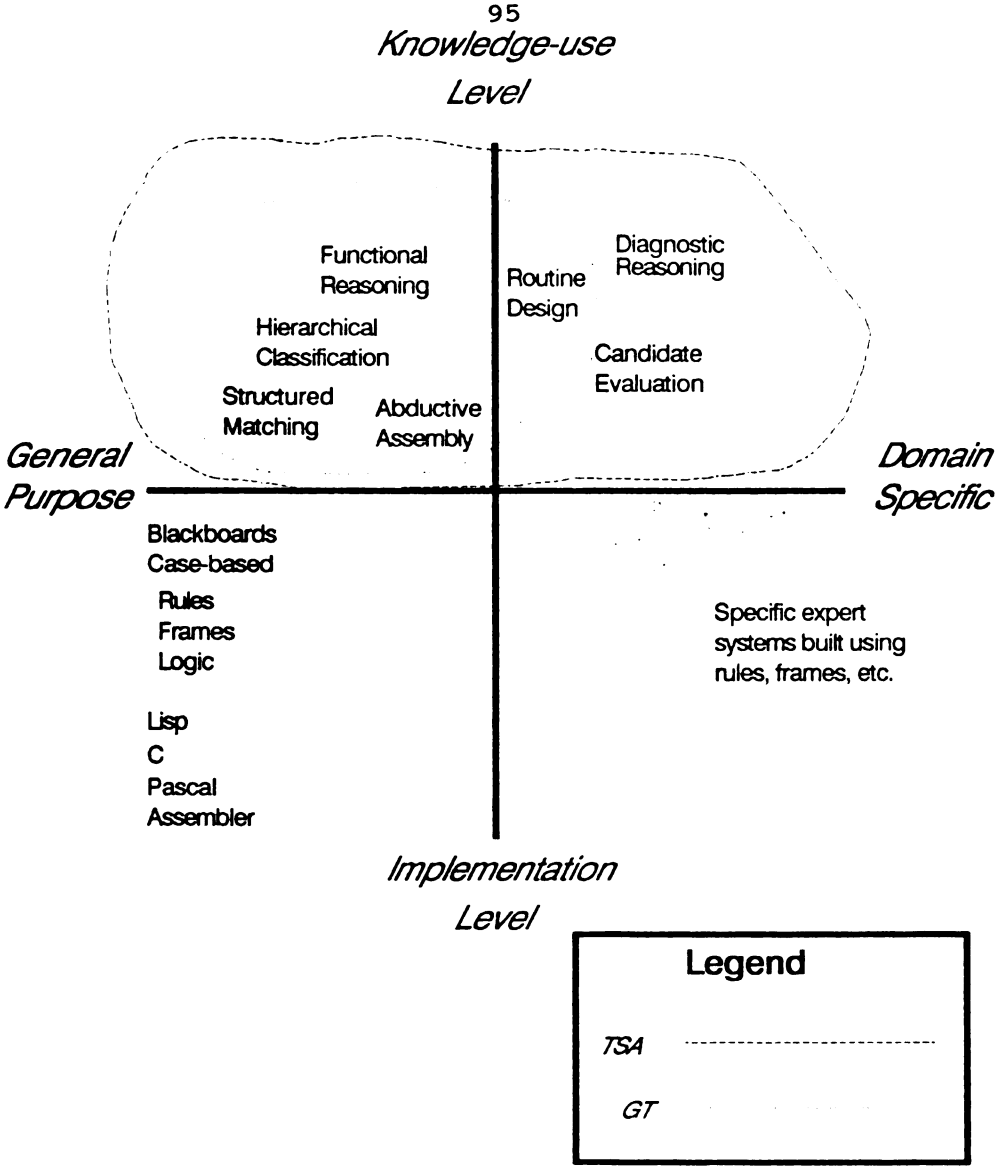


Figure 2.2

Different programming constructs and AI knowledge representation schemes differ in terms of their degree of "genericness" and in their level of representational abstraction. Task-specific architectures tend to be expressed at a higher "knowledge-level:, and can be either domain specific or general purpose. Generic tasks are also expressed at a knowledge level, but tend to be general-purpose in scope, applied to a wide variety of domains. Thus generic tasks can be thought of as a subset of task specific architectures.

IPT PS Theory KR Control Sample Alternative

	IPT Task	*PS Method*	*Theory Type*	*KR Primitives*	*Control Regime*	*Sample Applications*	*Alternative PS Methods/ Tools*
Hierarchical Classification *(CSRL)*	Given a particular entity, find the category or class to which it most likely belongs.	Top-down refinement	Type II Theory... Modular Specialists	Specialists, Knowledge Groups hierarchical taxonomy of classification nodes.	"Best-first" tree search guided by establish-refine heuristics	Medical Diagnosis	pattern recognition distance metrics weighted evidence accumulation
Abductive Assembly *(PEIRCE)*	Given a set of findings, find the best explanation for those findings.	Set-covering, cover-and-differentiate	Type I Theory... Overall Algorithm	Simple and compund hypotheses, compatability constraints	Control loop which iteratively adds compatable, non-redundant hypotheses to a list.	Medical Diagnosis	MOLE
Routine Design *(DSPL)*	Simple (class 3) design of artifacts and plans. Simple means little creativity, merely refinement.	Top-down classification, constraint-satisfaction, failure handling	Type II Theory... Modular Specialsists	Specialists, Plans, Tasks, Steps, Constraints,	Best-first tree search guided by sponsor-selector... redesign guided by constraint heuristics.	Design of mechanical devices (e.g. air cylenders) Design of mission plans.	OPM (Blackboard) SALT
Functional Reasoning *(FR)*	Consequence-finding.... Deep Reasoning	Simulation based on expansion of partial state diagrams.... state-transiitions	Type II Theory... Modular Specialsists	Devices, Components, Functions, Behaviors, State Variables	Starting from initial conditions and unavailable functions, recursive construction of partial state diagram to appropriate detail. Use of indexing.	Medical Diagnosis, Simulation of composite materials.	Davis' Model-Based Diagnosis
Structured Matching *(HYPER)*	Recognition, certainty-assessment, "evaluation" simple classification	Truth-table Pattern Matching	Type II Theory... modular specialists	Truth Tables (Knowledge Groups)	Top-down traversal of matcher hierarchy; recursive matching of parameter values until top-level matcher is known.	Used as an integral part of many other generic task shells	Signature tables, Hierarchical Linear Model (CEVED/CEVAL) * see Chapter 5

Figure 2.3

Five Generic Tasks, implemented in task-specific shells. . The task, problem-solving methods, representation primitives and control regimes are shown. Also shown are current applications and alternative methods and tools for performing the same tasks.

alternativ

generic ta

research. T

of the GT

manageable

all of the

true that d

"interactio

These issu

undoubtably

Nevert

offer as

knowledge

limiting pr

a problem w

that can

bottleneck

automate kn

the archit

this thesi

The r

solving in

theoretic

explanator

alternatives, although they were usually less flexible. The generic task tools were also compared against other TSA research. This comparison calls into question two assumptions of the GT philosophy. First, is it plausible that a manageable enumeration of problem solving methods can cover all of the tasks required by expert systems? Second, is it true that domain knowledge is tied to its use, the so-called "interaction hypothesis" (Bylander and Chandrasekaran 1987)? These issues are debatable, and future writers will undoubtably have much to say about them.

Nevertheless, it seems undeniable that TSAs have much to offer as knowledge representation tools. Representing knowledge at high levels of abstraction and using role-limiting primitives makes it possible to merge the analysis of a problem with the implementation of its solution in a manner that can significantly reduce the knowledge acquisition bottleneck. The use of these tools by domain experts helps to automate knowledge acquisition. This provides a motivation for the architecture that will be described in the remainder of this thesis.

The remainder of this thesis describes a TSA for problem solving in *evaluation* tasks. The method employs decision-theoretic judgement approaches combined with the verbal explanatory power of AI.

B

In th
probabilist
first desc
used in d
include d
(SEUT) and
diagram re
inference
makes exte
will comp
discussion
in underl
deviate fr
over whet
normative
system sh
AI-based

3.1 The B

The
a hypothe
is a func
in the ge
and the c

CHAPTER 3

BAYESIAN MODELS IN DECISION THEORY AND AI

In this chapter I will discuss the use of Bayesian probabilistic models in AI and in decision theory (DT). I will first describe the Bayes model. Then, I will show how it is used in decision theory and decision analysis. This will include discussion of subjective expected utility theory (SEUT) and its implementation in decision tree and influence-diagram representations. Next, I will discuss probabilistic inference networks (PIN), a knowledge representation that makes extensive use of the Bayes model and fuzzy logics. I will compare PINs against DT representations, including a discussion of similarities and differences in architecture and in underlying philosophy. Next, I will discuss how human's deviate from SEUT and Bayesian behavior, and review the debate over whether or not Bayesian reasoning should be considered normative (i.e. optimal). Finally, I will describe an expert system shell, called QXQ, which combines the SEUT model with AI-based explanation methods.

3.1 The Bayes model

The Bayes model asserts that the posterior probability of a hypothesis (i.e. its probability in light of the given data) is a function of its prior probability (i.e. its probability in the general case, or its probability if no data were given) and the conditional probability of the given data or evidence

(i.e. the

hypothesis

The Ba

conditional

odds-likel

where

The

ratio, ar

evidence

will see

The

sources

model.

This

decision

(i.e. the probability of the data's occurrence if the hypothesis were true). The equation expressing this is:

$$P(H_j \mid E) = \frac{P(E \mid H_j) \times P(H)}{\sum_{i-1}^{n} P(E \mid H_i) \times P(H_i)}$$

The Bayes model is often expressed in terms of prior, conditional, and posterior *odds* rather than probabilities. The odds-likelihood formulation is defined as:

$$O(H_i \mid E) = O(E \mid H_i) \times O(H_i)$$

where

$$O(E \mid H_i) = \frac{P(E \mid H_i)}{P(E \mid \neg H_i)}$$

The conditional odds $O(E|H)$ is called the *likelihood ratio*, and is a measure of how the presence or absence of evidence E impacts the likelihood of hypothesis H. Later we will see how AI and DT systems use this likelihood ratio.

The Bayesian approach to decision-making using several sources of information is represented as a multiplicative model.

This equation shows that, for Bayesian modelling of decision-making, all that is needed is a knowledge of the

prior odds

ratio indi

determinin

systems pr

shown late

conditional

values). Th

variables,

Of co

variables

Therefore,

a limitati

conditiona

Bayes mod

minimizes

3.2 Subjec

Becau

naturally

SEUT model

(1947), r

can be co

$$O(H|E_1,\ldots,E_n) = \prod_{i=1}^{n} O(E_i|H)\,O(H)$$

prior odds $O(H)$ and the likelihood ratio. Thus, the likelihood ratio indicates how important a particular data item is in determining the plausibility of the hypothesis. Both DT and AI systems provide this in their representations, as will be shown later. Note that this multiplicative model assumes conditional independence of the various data elements (the E_i values). The model can be extended to handle dependence of variables, as shown in the following equation:

$$P(H|E_1,E_2) = P(E_2|H,E_1) \times P(E_1|H) \times P(H)$$

Of course, as the number of conditionally dependent variables increase, the model quickly becomes unmanageable. Therefore, the independence assumption is often made. This is a limitation to the model. However, in circumstances where conditional independence of input data can be assumed, the Bayes model is considered optimal. This is because it minimizes the average error that can be made.

3.2 Subjective Expected Utility Theory (SEUT)

Because of its optimality, the Bayes model leads naturally to subjective expected utility theory (SEUT). The SEUT model, first developed by Von Neumann and Morgenstern (1947), rests on several assumptions. First, any two choices can be compared to each other, ie ranked. Second, the

probabilit

desirabili

should ac

particular

equation i

This

given actio

possible co

the probab

its utility

is the post

Thus we see

function t

SEUT

by Savage

which cons

actions th

possible c

a conseque

ranking of

the conse

occurrence

probabilit

probabilities of the occurrence of an outcome will affect the desirability of a choice. Third, in the ideal case, people should act to maximize the product of the utility of a particular outcome and its probability given the data. The equation illustrating this is:

$$E\mu(A_i) = \sum_{j=1}^{m} p(C_j|A_i) \times U(C_j)$$

This equation states that the expected utility for a given action A_i is equal to the sum of the products, for each possible consequence C_j that could result from the action, of the probability p of that consequence given the action times its utility (i.e. "goodness"). The probability function $p(C|A)$ is the posterior probability of consequence C given action A. Thus we see that the Bayesian model is combined with a utility function to form the core of SEUT.

SEUT was advocated as a normative decision-making model by Savage (1954). Savage described the notion of *small worlds*, which consist of a set of possible states S, a set of possible actions that a decision-maker can choose from F, and a set of possible consequences C. Each act f in F maps a state s onto a consequence c. Savage suggested that a person's preference ranking of the acts of F can be used to infer the utilities of the consequences as well as the probabilities of their occurrences. Thus, when explicit subjective expected probabilities and utilities are not available, SEUT enables

you to inf

human bei

These

three of w

complete,

actions po

the decisi

possible

Second, t

belief of

an outcor

not. The

postulat

one cons

preferer

third o

St

postula

Savage

Even t

should

many

Neverth

analyz

possib

how SE

you to infer them based on observations of the choices made by human beings.

These small worlds are governed by several postulates, three of which will be described here. First, there exists a complete, irreflexive and transitive ranking of the optional actions possible in the small world. SEUT thus assumes that the decision maker, if called upon to do so, can order all the possible actions, indicating a complete preference list. Second, the utility value of a consequence is independent of belief of its probability. In other words, the "goodness" of an outcome remains the same whether that outcome is likely or not. The third postulate, often called the *independence postulate* is a key assumption in the model. It states that if one consequence C_i is preferred over consequence C_j, then that preference remains in effect even if there is a chance for a third outcome C_k to occur.

Studies have shown that all three of the above SEUT postulates are regularly violated by human decision-makers. Savage acknowledged this, but claimed that SEUT is normative. Even though people don't follow SEUT in real life, they should. Other authors have disputed this claim, stating that many reasonable selection strategies violate SEUT. Nevertheless, SEUT is widely seen as a useful tool for analyzing a complex problem and selecting from among a list of possible problem-solving approaches. The next section describe how SEUT is implemented as a decision-aid model.

3.3 Bayesi

In th
representa
form of
representa
second rep
is the prob
each in tu

3.3.1 Decis

SEUT
trees or in
and of the
purpose of
analysis. H
used in deci
theoretic
Bayesian re

Influe
represent th
utility valu
Such graphs
analysis, b
translated i
for detailed
model cause-

3.3 Bayesian Knowledge Representations in DT and AI

In this section, I will describe two Bayesian knowledge representations. The first is an implementation of SEUT in the form of decision trees and influence diagrams. This representation is frequently used in decision analysis. The second representation, first presented by Duda et al. (1976), is the probabilistic inference network (PIN). I will describe each in turn, then compare their structures and uses.

3.3.1 Decision Trees and Influence Diagrams

SEUT is often represented in practice using *decision trees* or *influence diagrams*. The use of these representations and of the above-mentioned utility measurements for the purpose of aiding decision-making is often called *decision analysis*. Here, I will describe the representation schemes used in decision analysis. Later I will compare this decision-theoretic Bayesian representation against a commonly-used Bayesian representation used in AI.

Influence diagrams are graph structures whose nodes represent the various decision points, chance occurrences, and utility values of the decision-making process being modelled. Such graphs are usually used in the initial stages of decision analysis, because of their relative simplicity, then are translated into decision trees (which are more comprehensive) for detailed analysis. Decision trees and influence diagrams model cause-effect relationships as well as plan-refinement

strategie
choices t
point in
(thus imp
branches
consequenc
relationsh
terminates
that path
value can b
but its va
When utili
values are
probabiliti
way, utilit
When the ut
decision tr
present cir

The st
influence di
utilities a
likelihood c
and utiliti
chance-nodes
outcomes are
and are the

strategies. Branches emanating from *decision nodes* represent choices that can be made by the decision maker at a given point in time. These branches lead to other decision nodes (thus implementing plan refinement) or to *chance nodes*. The branches emanating from chance nodes represent possible consequences of the choice (thus implementing cause-effect relationships). A path in an influence graph or decision tree terminates at a node representing the utility of following that path of decisions and chance occurrences. This utility value can be calculated in many ways, depending on the domain, but its value is somehow determined from expert knowledge. When utilities at termination points are calculated, their values are propagated back through the path, attenuated by probabilities of the chance links that lead to them. In this way, utilities can be calculated for decision nodes as well. When the utilities at all decision nodes are calculated, the decision tree shows the "best" decisions to make in the present circumstances.

The structure and behavior of decision trees and influence diagrams obey the postulates set forth by SEUT. The utilities assigned to value nodes are independent of the likelihood of paths that lead to them. Indeed, probabilities and utilities are stored in completely different nodes: chance-nodes and value-nodes respectively. The possible outcomes are represented as the leaves of the decision tree, and are therefore complete and, because they have scores

(utilitie

Bayes rul

compared

calculate

and value-

Next,

using Baye

3.3.2 Prob

One A

probabilis

representa

These repr

medical

marketing)

chips). The

or near-o

decisions;

certainty

of inferen

1987).

Proba

involve noc

will call

links betwe

one would t

(utilities) assigned to them, are comparable and rankable. Via Bayes rule, decision nodes higher up in the tree can be compared and ranked also. Their derived utilities are calculated from the probabilities and utilities of the chance- and value-nodes in their subtrees.

Next, I will describe another knowledge representation using Bayes models.

3.3.2 Probabilistic Inference Networks (PIN)

One AI area making extensive use of the Bayesian model is *probabilistic inference network* (PIN) knowledge representations.

These representations have been used for diagnosis (both medical and machine), strategy formation (military, marketing), and design (software, civil engineering, VLSI chips). They are appropriate for domains that: require optimal or near-optimal decisions, and justifications for these decisions; involve information available at various levels of certainty and completeness; and have available general rules of inference that can be applied to the problem. (Tanimoto 1987).

Probabilistic inference nets are graph structures that involve nodes representing evidence (facts) and hypotheses. We will call such nodes E-nodes and H-nodes, respectively. The links between these nodes represent the reasoning steps that one would take in a problem-solving process (e.g. diagnosis),

in much t

based sys

An e

of E-node

It also i

various hy

during the

update the

nodes base

nodes, whi

H-nodes, a

be describ

In c

probabilit

values onto

the conditi

values for

probabilit

prior proba

probabilitie

probabilitie

through empi

when the dat

probabilitie

the table an

Early Ba

in much the same way that the if -- then structure of a rule based system works.

An expert system using PIN typically includes one layer of E-nodes, representing the data that is input to the system. It also includes one or more layers of H-nodes, representing various hypotheses and sub-hypotheses that the system computes during the problem-solving process. The Bayes rule is used to update the posterior probabilities of the first level of H-nodes based on the E-nodes. Probabilities for subsequent H-nodes, which are based on boolean combinations of early-level H-nodes, are usually determined using fuzzy logics, which will be described later.

In order to ascertain the first-level H-node probabilities, the expert system uses a table mapping E-node values onto H-node probabilities. Such a table will contain the conditional probabilities for each combination of E-node values for each first-level H-node (i.e. the joint probabilities). Also included in the representation is the prior probability for each first-level H-node. All these probabilities are provided by the expert as subjective probabilities, or are based on frequency statistics obtained through empirical studies in the domain being modelled. Thus, when the data become available to the system, the posterior probabilities of the first-level H-nodes are calculated using the table and the Bayes model.

Early Bayesian diagnostic systems used only these tabular

represent

expert le

These pro

a complete

of eviden

independen

use of oth

is to use

described

Once

via the

combined

at subs

equivale

the clau

truth ta

With pr

two pos

ab (Ta

clause

Frequ

representations. Although they frequently performed at or near expert level, there are problems with this representation. These problems stem from the axioms of probability requiring a complete, mutually exclusive hypothesis set, where each item of evidence presented to the system must be statistically independent of all others (Langlotz 1989). This motivated the use of other AI techniques and heuristics. One such heuristic is to use fuzzy logic to expand upon the Bayesian model, as described below.

Once the first-level H-node probabilities are determined via the probabilities tables mentioned above, they are combined using fuzzy logics to determine H-node probabilities at subsequent levels. Fuzzy logics are probabilistic equivalents of propositional calculus clauses. For example, the clause A OR B, in propositional calculus results in a truth table like this:

A	B	A OR B
T	T	T
T	F	T
F	T	T
F	F	F

With probabilistic values for A and B instead of T/F values, two possible fuzzy logics for A OR B are *max(a,b)* and *a+b-ab* (Tanimoto p. 248). Similar fuzzy logics exist for other clauses such as NOT, AND, IMPLY, and XOR.

Bayes rule is used for updating H-node probabilities. Frequently, the impact of one node's truth or falsehood on

another node's probability is expressed via the likelihood ratios $O(E|H)$ and $O(E|H')$ (see above for full description of likelihood ratios). These ratios are sometimes called the *sufficiency coefficient* and the *necessity coefficient*, respectively. The sufficiency coefficient $O(E|H)$ gives an indication of how strongly the presence of the evidence E implies the truth of hypothesis H. A high value (much greater than 1) indicates that E is sufficient evidence to conclude H. The necessity coefficient O(E|H') indicates how strongly the presence of E refutes hypothesis H'. A low value (<< 1) indicates that E is necessary to conclude H. Both coefficients are provided by the expert.

Often the evidence *E* being presented to a decision-maker is uncertain. Decision tree representations assume that the evidence is known and correct. PIN representations, however, are designed to deal with uncertain evidence. As mentioned above, the prior probabilities of the hypotheses and likelihood ratios of the evidence under the hypotheses in a PIN are typically provided by the expert. In addition, experts can also provide prior probabilities of each evidence item $P(E)$. This provides a sort of "default" value in the absence of observed data.

This introduces a problem of consistency in the representation. The system designer should make sure that P(H), P(E|H), and P(E) are not in conflict. If they are, the system may exhibit anomalous behavior. For example, suppose we

make an observation *E'*, and suppose P(E|E') = P(E). Then, to maintain consistency, it must also be true that P(H|E') = P(H). In other words, for a linear relationship between P(E|E') (the x-axis) and P(H|E') (the y-axis), there should be a point on the line at (P(E),P(H)). However, in practice, this is often not the case. In order to compensate for this, PINs use "interpolation functions" to enforce consistency. Duda et al. (1976) gives a good review of some common interpolation functions.

Like the decision-theoretic models of sequential decision-making mentioned above, PINs use a multiplicative model to update hypotheses based on multiple evidence. As mentioned above, this method assumes independence among the E-nodes. The multiplicative model is robust, however, in that it is unaffected by the order in which data is presented, unlike humans who are affected by primacy-recency factors.

3.3.3 Comparing Decision Trees to Inference Networks

Thus, we see that the decision tree model based on SEUT, and the probabilistic inference network model used as a knowledge representation in AI system design, both use the Bayesian model as their core component. However, the behaviors and structures of these two representations are quite different, and reflect different underlying philosophies and assumptions.

With decision trees and influence diagrams, the Bayesian

model is used to calculate utilities of the decision nodes based on the values assigned to the branches of chance-nodes and utility values calculated at outcome-nodes. The utilities are calculated based on the Bayes rule that a derived utility is the sum of products of the probabilities and utilities of each outcome. Thus, the decision tree acts in a data-driven manner. The branch to take from a decision node requires knowledge of the outcome utilities.

The semantics of decision trees uses notions of action and consequence. In this sense it is similar to game tree representations. In addition, the notion of utility gives decision trees and evaluative flavor, an indication of goodness or badness. By contrast, the PIN model uses notions of evidence and hypothesis, which have a more deductive or diagnostic flavor. With PINS, the Bayes model is used to represent certainty of hypotheses and impact of evidence.

With PINs, the Bayes model is supplemented with fuzzy logics for dealing with the various boolean combinations of facts and hypotheses. This makes it them flexible enough to handle conditional dependencies between variables, which as mentioned earlier, make the Bayes model unreliable or unwieldy. By contrast, decision trees and influence diagrams do not allow for conditional dependencies. In addition, decision trees distinguish between utility and probability. PIN models make no such distinction; everything is expressed in terms of probability.

Whereas decision trees assume that the set of possible solutions is complete and comparable, there is no such assumption underlying PINS.

3.4 Bayesian Studies of human decision-making

A frequently-used experimental paradigm in studies of human judgement uses a Bayesian model as a baseline against which to compare human performance. Typically, the subject is asked to estimate the posterior probability $P(H|E)$, where E is provided by the experimenter. With each new E_i, the subject must revise his or her estimate of the posterior probability.

As espoused by Savage, the Bayesian model is generally used as a normative model against which to compare subject's performance in decision-making under uncertainty. Typically, such studies try to identify the *reasons* for subjects' deviations from the Bayesian ideal. The experimenter will manipulate the data input to the subject, the manner in which the subject is required to respond, and perhaps the feedback given to the subject. The experimenter will then study the effects of these manipulations on the subject's decision-making process. Input manipulations may involve controlling the order in which data are presented, presenting the data in nominal vs probabilistic terms, or varying the diagnosticity of the data. Subject response mode manipulations include controlling whether subjects must give direct probability estimates or indirect estimates, requiring repeated vs

intermittent responses, and requiring categorical vs probabilistic responses. Feedback manipulations may involve payoff or non-payoff situations.

The experiments typically involve decision-making in an environment where the probability distributions are known mathematically or can be safely assumed. For example, one frequent experiment paradigm involves two bags of poker chips (call them A and B), each with a different percentage of red vs. blue chips. The subject is told the percentages of each chip in each bag. The subject then draws chips from one of the bag (s/he does not know which one) and give the posterior probability $P(H|E_i)$ that it is bag A, where E_i indicates the color that was drawn in draw i. The distribution of probabilities in such an experiment is binomial. Bayes rule would choose in a way that maximizes the posterior odds of bag A, given the chips that have been drawn. The odds and probabilities are known, via the binomial density function. Other experimental paradigms involve making decisions pertaining to human population samples. In this case, the probability distribution is usually normal.

As mentioned earlier, the assumption that the input data are conditionally independent is a limiting characteristic of the Bayes model, and attempts to extend Bayes to account for dependencies quickly renders the model unmanageable. Thus, psychological experimenters using the Bayes model will try to ensure that they are obeying the independence criteria by

using uncorrelated data. Interestingly, the choice of population can affect whether two data items are correlated or not. For example, in the general population, height and hair length are negatively correlated. This is because women generally have longer hair than men, and are generally shorter. However, within-gender, height and hair length are uncorrelated. Therefore, an experiment that asks the subject to decide, based on information about a person's height and hair length, the odds that the person is male, would be a valid use of Bayes model, because the hypothesis is within-gender, whereas the same experiment that asks the likelihood that the person has blue eyes would not, because the hypothesis is between-gender.

3.4.1 Why people deviate from Bayes

Experiments, such as the ones mentioned above, indicate that people regularly deviate from the Bayesian norm in their decision-making. These studies indicate that in general, people tend to be too *conservative* in the degree to which they will change their hypothesis when new data is presented. In other words, their posterior probability estimations tend to be closer to the prior hypotheses than would be warranted by the Bayesian model. This tendency to "err on the side of caution" can be attributed to several factors that have been discussed in the literature.

One reason for conservatism is the subject's lack of

understanding of the data generator itself. This phenomenon, called *misperception* in the literature, means that subjects usually don't understand the behaviors that can be expected of the common probability distributions such as binomial, multinomial, and normal distributions. Studies have suggested that this lack of understanding of the data generator causes people to be conservative in their hypothesis revisions. This conservatism is particularly acute when the data presented is very unlikely to occur, the "rare effects" phenomenon. Presumuably, knowing more about the distributions can help optimize the subjects' responses.

A second cause of conservatism is *misaggregation*, the subject's inability to correctly synthesize and combine multiple data elements in their posterior probability assessments, which can also explain the rare-effect phenomenon. Edwards et al. (1968) found that computer programs implementing Bayes model consistently performed superior to human subjects in tasks requiring aggregation of data elements. This lends support to the idea that expert systems could enhance human decision-making.

A third explanation for conservatism is the *response bias* of the subject. DeCharme (1970) found that subjects were unwilling to express extreme odds (e.g. greater than 10:1 or less than 1:10) in their responses, even in the face of very diagnostic data that strongly supported such extreme odds.

Tversky and Kahneman (1986) found that human judgement is

biased by the way a problem is presented ("framed") to the human. For example, he studied the effect of choice vs. matching in human decision-making. Choice involves selecting the "best" candidate among a set of candidates. Matching involves adjusting the value for a particular attribute of one option until it is deemed to be "as good as" the other option. Tversky found that when someone is asked to choose between a set of options (each with multiple attributes), that person's weighting of the attributes will be much different than when they are asked to match the options. In the choice task, the dominant (i.e. high-weighted) attributes have more impact than in the matching tasks. This reflects a bias in human thinking that goes against the axioms of probabilistic reasoning. Presumably, expert systems should be robust enough to avoid this bias.

Kuipers et al. (1988) compared protocol analysis of experts against formal decision analysis (decision tree) representations. He found significant differences. First, the sequence of events and decision structure is different from what is expected in dec tree. Experts used "top-down refinement" mixed with opportunism whereas decision tree representations use weighted averaging and "folding back" after determining utilities for all final outcomes. Second, concepts of likelihood are treated as categorical or ordinal values, not numeric. Even numeric expressions are expressions of categorical/ordinal info (e.g. thresholds or ranges).

Third, the final utilities of each choice tended to me more clear-cut to the experts than what would appear based on formal decision analysis.

It is interesting to note the distinction between the way a decision theorist would view these differences and the way an AI theorist would view them. As mentioned in the above section, the deviations from Bayesian reasoning are seen a flaws and biases from the viewpoint of the decision theorist. But AI theorists like Kuipers view such deviations as motivations to change the model of human knowledge from probabilistic to categorical in order to more closely represent human problem-solving processes.

3.5 Combining Bayesian Decision Analysis with AI

Langlotz (1989) developed an expert system shell that merges concepts from AI with concepts from decision theory, and brings the best of both worlds together into a system called *QXQ*, for "Qualitative Explanation of Quantitative Models." He explored potential contributions from AI and DT as they pertain to diagnostic, planning and explanation facilities for systems operating in the medical domain. In so doing, he presented an "axiomatic approach" to constructing such computer systems.

QXQ involves two modules: a modelling (development) environment, intended for the decision analyst for representation of quantitative decision models; and an advice

generator, or performance system, intended for use by the end user to consult with. Thus, QXQ facilitates both knowledge representation and design on the one hand, and knowledge use and performance on the other.

Langlotz defines *axiomatic* decision making framework as one that embodies an "explicit agreement between the decision making system and its users (Langlotz 1989, p.25)." For such a framework, a set of properties are defined, and the user and system must both behave in accordance with these properties. This impacts all aspects of the construction and performance of the expert system, including knowledge acquisition, reasoning and explanation. In addition to being axiomatic, the system should also be normative. This means that the axioms and properties defined must be intuitively appealing in terms of the constructs that experts use in the relevant tasks they perform. Note that this requirement for axiomatic, normative design principles is similar to the "knowledge-use level" requirements that motivated the task-specific and generic-task movements in AI, described earlier in this thesis.

A major theme running throughout Langlotz's thesis is that decision making involves two main components, diagnosis and planning. Diagnosis involves assessing the beliefs of the system, given what the system knows; it is a mapping from evidence to hypotheses. Planning is a mapping from hypotheses to actions, based on known goals of the system. Thus, the two activities are interrelated and interdependent. However,

although Langlotz discusses diagnosis as a major component of the decision making process, his QXQ system does not include a diagnostic component. Rather, QXQ is primarily concerned with ranking alternative plans and with providing explanations that justify those plan rankings.

Langlotz proposed a normative, axiomatic framework for each of the two major components of decision making, diagnosis and planning. He identified desirable properties for diagnostic reasoning. These include: clarity, completeness, orderability (i.e. ability to rank hypotheses according to degree of belief), thoroughness, and consistency. As we saw earlier, Bayesian decision theory satisfies all these properties. In fact, Langlotz claims that in terms of system performance, the Bayesian approach is superior to mathematical models and heuristic AI approaches for diagnostic tasks. Of course, other AI researchers and theorists may strongly disagree with this claim on the grounds that probabilistic approaches require too much prior knowledge about disease probabilities and are not good at explaining their reasoning. Langlotz addresses the first objection by saying that frequency statistics from medical records, probabilistic approximations from the medical literature, and expert judgements can all be used for probability assessment, but he also acknowledges that such probabilities are not exact, which weakens the Bayesian framework. Langlotz strongly agrees with the second argument (i.e. the explanatory power of AI

systems), and his QXQ system will show this.

Langlotz also lists the desired properties of planning systems. These include orderability, monotonicity (decision maker should pick the best outcome), decomposability (for dealing with uncertain scenarios), continuity, and substitutability. For these properties to hold, a system should be able to represent the concept of utility in its axiomatic framework. Thus, his approach to plan selection is strongly grounded in the SEUT decision-theoretic model described above.

Since the structure of a DT-based knowledge representation can involve either influence diagrams or decision trees, both are implemented in QXQ. As mentioned earlier, influence diagrams are acyclic directed graphs that involve three types of nodes: decision nodes represent the explicit choices made by a decision maker at a given point in the planning process; chance nodes represent probabilities that a given decision will result in a particular outcome; and value nodes represent the desirability (or utility) of a given outcome. QXQ represents these nodes as objects in an object-oriented framework, and each node contains certain slots and methods. An example for a slot for a value node is the "expected utility" slot. For a chance node, one slot is the "probability expression" slot. Both types of nodes include "situation" slots, which are verbal accounts of the situation for which the node is relevant, and which serves as an

explanation facility.

The second type of structure for DT-based representation is the decision tree. This is the structure that is actually used by the decision-making component of QXQ. It is often easier for the designer to represent knowledge in terms of influence diagrams, since this way one does not need to explicitly show all possible branches of a decision tree. However, in order to enhance the performance of a decision making system, QXQ will convert influence diagrams into decision trees.

Up to this point, I have described Langlotz's thesis and his QXQ system mostly in terms of decision theoretic constructs. He asserts that, in terms of system performance, decision theory is superior to heuristic (AI) approaches. However, such an approach requires several assumptions which are impossible for systems that must deal with incomplete or uncertain information. These assumptions include: that the diagnostic component of a system can be supplied with exact values of probabilities and utilities (orderability); that all conceivable plans can be evaluated (completeness); and that all possible consequences of every action can be considered (thoroughness). Such ideal performance is impossible in practice, so Langlotz proposes that heuristic approaches be introduced to supplement the decision-theoretic approaches.

In addition to compensating for the lack of completeness and certainty in knowledge that would be required for

decision-theoretic approaches to expert systems design, AI heuristic models also allow for intuitively appealing, qualitative explanations of their reasoning. In this sense, they are superior to mathematical models such as decision theory, whose explanations must be couched in terms of mathematical arguments. One of the major design considerations that went into the development of QXQ was the desire to provide qualitative explanations to support the mathematically-based decision framework. These qualitative explanations should not be mathematical in nature, but should be expressed in terms of the domain in question, for example, using medical terms and constructs.

In order to justify why QXQ makes a particular plan selection, it must identify how this decision differs from other possible decisions that could be made in the same situation. In order to do this, QXQ does *branch comparison*, which is "the process of analyzing the similarities and differences among decision branches to find elements of the tree that most contribute to the difference in expected utility (Langlotz 1989, p.208)." Before comparing the branches, however, the system must find branches that are "analogous" in the sense that their corresponding nodes pertain to the same situation. After the system finds two or more analogous branches, it does a quantitative comparison of the expected utilities that result from each branch. Note that expected utility can be assessed at any level in the tree, and

that sub-branches can be compared at various levels also. Thus, the comparison may be given at various levels of abstraction, which facilitates the explanation process. QXQ will continually decompose the branch comparison until it reaches the bottom levels of the tree, and plans will be ranked accordingly.

So far, I have just described how plans are ranked, without talking about how these plan rankings are translated into verbal explanations. Branch comparison has isolated the various differences of expected utilities among plan candidates. Now, the system must decide which of these differences are relevant for describing in qualitative terms to the end user. QXQ includes a control regime for providing explanations based on *rhetorical frameworks*, which are explanation strategies that provide a structure for the explanation that most effectively uses the available data to argue for the results of the decision model. All of these frameworks will result in an output whose form and content is a document that looks like a journal article. The different alternative rhetorical frameworks include: the *weighted-tradeoff method*, where phrases are in terms of "the advantages of choice *x* over choice *y* in terms of difference d_1 OUTWEIGHS the advantages of choice *y* over choice *x* in terms of difference d_2."; the *extreme-value method*, which identifies particularly sensitive variables (using sensitivity analysis) with extreme values and uses that as a justification for

making a particular plan decision; and the *pivotal-parameter method*, which is employed when the overall expected utilities for two plans are about the same and a particular sensitive variable's value is uncertain...in this case the system advises that more information be obtained before making a decision.

The choice of a rhetorical framework is based on the types of differences that are found during the branch comparison process, and these choices are made by rules embodied in control blocks. These control blocks can be developed by a knowledge engineer using QXQ's development environment. Thus, QXQ uses the nature of the differences between decision branches to decide the structure of the explanation.

3.6 Conclusion

We have seen that the combination of AI approaches and decision-theoretic approaches is both feasible and useful. The mathematical robustness of the Bayes decision rule and SEUT (implemented in QXQ as decision trees and influence diagrams) combines with the explanatory, heuristic power of AI systems (implemented in QXQ as branch comparisons and rhetorical frameworks) to provide a powerful decision support tool. As we will see later, my work with candidate evaluation and its implementation in the CEVED/CEVAL tools attempt to provide a similar mix of AI and decision theory. The difference is that

I concentrate my efforts on a different branch of DT, namely the "regression" approaches to decision theory, focussed primarily on *multi-attribute utility theory*. Thus, the next section of this thesis provides a detailed survey of regression-based DT models.

CHAPTER 4

REGRESSION, LINEAR MODELS AND MULTI-ATTRIBUTE UTILITY THEORY IN DECISION THEORY AND AI

The preceding chapter discusses the use of Bayesian statistics and subjective expected utility theory (SEUT) in decision theory and artificial intelligence. It closes with a description of Langlotz's QXQ program, an expert system shell designed to combine the SEUT model with qualitative explanatory facilities common to expert systems technology. In this chapter, I will present another stream in decision theory research. This stream of research involves regression approaches instead of Bayesian methods.

Although many authors, including Pearl (1977) advocate the use of Bayes model for evaluation modelling, many others subscribe to the regression approach. The regression approach to decision theory makes use of statistical techniques such as multiple regression, analysis of variance (ANOVA), and correlational approaches to model human decision-making processes. Like the Bayesian methods described previously, they are frequently used as normative models against which to compare actual human performance. As normative models, they can also be used in AI and expert systems as decision support aids. This chapter discusses regression models and their uses in decision sciences and AI.

The chapter serves as an introduction of the decision-theoretic foundation on which the Candidate Evaluation

architecture is based. Chapter 2 had discussed task-specific architectures, and identified several commonly used problem-solving methods such as classification, design, abduction, and matching. Candidate Evaluation is another task-specific problem solving method. It is based on the weighted linear model, and its variants, described in this chapter. In essence, this chapter describes multi-attribute utility theory (MAUT) in order to provide the theoretical basis for the tools CEVED (Candidate Evaluation Editor) and CEVAL (Candidate Evaluator), which will be described in chapter 5.

4.1 Multiple Regression

Multiple regression is a general statistical technique for analyzing the relationship between a set of independent variables (often called predictor variables) and a dependent variable (called a criterion variable). The multiple regression equation is a "prediction equation" whose form indicates how the predictor variables should be weighted and summed to obtain the best predictor of the criterion variable. The regression model, when applied against a set of sample cases whose values ar known for independent and dependent variables, will give the best weights for the independent variables. Graphically, this results in finding the best-fitting line (in the bivariate case) or hyperplane (in the multi-variate case) to fit the data given.

The bivariate regression model is:

$$Y' = A + BX$$

where Y' is the predicted value of the dependent variable, X is the independent variable, A is a constant representing the Y-intercept, and B is the *regression coefficient* which represents the slope of the regression line. Thus, B is an indicator of the expected change in Y given a unit change in X. It is B that is calculated when the regression model is presented with actual training samples.

The regression coefficient is calculated using the following formula:

$$B = \frac{\sum (X-\overline{X})(Y-\overline{Y})}{\sum (X-\overline{X})^2}$$

The numerator is the sum of the cross products of X and Y (SS_{xy}), and the denominator is the sum of squares of X (SS_x). The regression line represented by this coefficient is also called the "line of best fit" or least-squares line".

As you can see, the regression equation is a *linear model*. There is some debate about the adequacy of linear models for prediction and complex decision-making, especially in circumstances where the interactions of variables can have a significant impact. The simplicity of such a model makes it attractive. Nevertheless, depending on the actual linearity of the data, there may be significant error in the predictions

made by the regression coefficient. This error can be quantified via the sum of squared *residuals*. A residual is the difference between the predicted value of the dependent variable Y' and its actual value Y for any given sample. That is, $RES = Y - Y'$, as illustrated in figure XX, and the sum of squared residuals is expressed as SS_{res}. Thus, the prediction accuracy of the multiple regression model is expressed as:

$$r_{xy}^2 = \frac{SS_Y - SS_{res}}{SS_Y}$$

This prediction accuracy measurement expresses the degree to which the variability is Y is accounted for by the regression model, and its value lies within the interval (0,1). The closer r^2 is to 1, the more accurate the regression model is, and, by implication, the more linear the data are.

Thus far, I have described the regression formula for the bivariate case, that is, when there is one independent variable and one dependent variable. The multivariate version of the regression equation is:

$$Y' = A + \sum_{i-1}^{n} B_i \times X_i$$

In this case, the B_i are called "partial regression coefficients". Each B_i represents the expected change in Y if X_i is changed but X_1 remains constant. In other words, it reflects the impact that X_i has on Y independent of any

activity in the other predictor variables. This assumption of independence in inherent in the regression model, just as it is in the Bayes model.

4.2 Use of Regression in Decision Sciences

Regression models are used extensively in decision theory. They are used for modelling the interaction of decision makers and their universe. They are also used to model the policy-making behavior of judges. Slovic and Lichtenstein (1971) identified two major paradigms in regression-based decision models, the *correlational* paradigm and the ***ANOVA*** (analysis if variance) paradigm. Correlational approaches follow two main streams: the Brunswick lens model for analyzing the judge's performance in a real world setting; and the judgement-policy model, which simply establishes the relative importance of the various cues in the decision-making process. ANOVA approaches are used to deal with configural relationships between the cues, and are particularly useful in environments where the input variables exhibit some dependency relationships amongst themselves.

4.2.1 Correlational Paradigm

Correlational approaches correlate the information cues available to the judge against the judgements and/or decision that s/he makes, typically using the Pearson correlation coefficient:

$$r = \frac{\sum_{i-1}^{n} (X_i - \overline{X})(Y_i \overline{Y})}{\sqrt{\{[\sum_{i-1}^{n} (X_i - \overline{X})^2][\sum_{i-1}^{n} (Y_i - \overline{Y})^2\}}}$$

The correlation coefficient is a measure of the weight of importance of each cue presented to the judge. As with the models mentioned above, there is an assumption of independence between the cues.

4.2.1.1 Brunswick lens model

The Brunswick lens model, also called "probabilistic functionalism", focusses on the adaptive interrelationships between the organism and its environment. Studies involving this model are concerned with measuring how the judge uses and weights environmental cues and with learning the characteristics of the environment itself. In this sense, the lens model represents probabilistic interrelations between organismic and environmental components of the judgement process.

4.2.1.2 Linear Judgement Policy Models

Some researchers are interested solely in modelling the policy of the judge (i.e. the weights associated with each input cue) and are not concerned with the actual relationship between cue and result in the environment. This approach uses the simple linear model, with the idea that a judge's

ediction is simply a linear combination of the cues esented to him or her. There is no representation of the tual environmental correlation between cue and dependent riable. Thus, these models represent half of what the unswick Lens model represents. The mathematical description the linear model is shown below:

$$Y\text{-}\sum_{i\text{-}1}^{n} W_i X_i$$

re Y is the final judgement score of the criterion iable, X_i is the score of predictor variable and W_i is the ght of importance assigned to the predictor variable.

Dawes (1979) subdivided the overall linear model into two s, which he called *proper* and *improper* linear models. The erence between proper and improper models centers on how weights of predictor variables are determined. With proper ls, the weights are statistically generated, usually using dard regression analysis or discriminant function ysis, to optimally predict the criterion value of est. In contrast, the weights in improper models are mined using some nonoptimal method, usually via polling t judges (using Delphi studies, for example) to ascertain intuitions on the importance of each of the predictor oles.

Proper linear models, although optimal, are often ctical to implement in practice. Dawes (1979) cites two

reasons for this. First, in order to be statistically viable, the multiple regression method requires the ratio of observed data samples to predictor variables to be as high as 20 to 1. Second, the predictor and criterion variables of interest are often subjective in nature, not easily converted to objective quantitative measurement. For example, how does one give an objective measure for "motivation" or "self esteem" when evaluating potential job applicants? For these reasons, users of linear models often resort to suboptimal, improper methods for establishing weights of the predictor variables.

Linear models have proven to be very good predictors of human judgements, and have in fact been shown to be superior to human judgement performance in many domains. This fact has been documented by Slovic and Lichtenstein (1971) and by Dawes (1988). They described a process of building a linear model of the judge's policy (in terms of the weighted outcome criteria) and then using that model in place of a judge for decision-making. They called this process *bootstrapping*. Under circumstances when the predictor variables have conditionally monotone relationships with criterion variables, these models have been shown to work very well. As a result, they have become ubiquitous in the decision sciences literature. For example, Dawes and Corrigan (1974) successfully used linear models for evaluating graduate applicants at the University of Oregon. Wiggens and Kohen (1971) used them to predict GPAs of first-year graduate students, and found their performance to

superior to actual experts. Likewise, Goldberg (1970) monstrated that linear models of judge's decision policies tperformed 26 of 29 clinical psychologists in diagnosing uroses or psychoses of patients based on data from Minnesota ltiphasic Personality Inventory (MMPI) profiles.

Why do linear models work so much better than actual perts? Dawes suggests that this is because humans are nerally not good at integrating information from various urces in order to come up with valid decisions. In his rds:

> "People are good at picking out the right predictor variables and at coding them in such a way that they have a conditionally monotone relationship with the criterion. People are bad at integrating information from diverse and incomparable sources. Proper linear models are good at such integration when the predictions have a conditionally monotone relationship with the criterion. (1979, p.574)"

though the above paragraph pertains to proper linear models, wes argued that improper models are almost as good. In the udies that he conducted, and in other studies he cites, the proper models work as almost as well as proper models. In s words:

> "...[improper] linear models are robust over deviations from optimal weighting. In other words, the bootstrapping findings, at least in these studies, has simply been a reaffirmation of the earlier findings that proper linear models are superior to human judgements -- the weights derived from the judges' behavior being sufficiently close to the optimal weights that the outputs of the models are highly similar. (1979, p.577)"

Thus, the efficacy of this bootstrapping process suggests at linear models may work well as expert system

hodologies, provided that an underlying explanation
ility, so important for viable expert systems performance,
be incorporated into the overall model.

Multiattribute Utility Theory (MAUT)

The efficacy of the linear model approach to judgement
evaluation is further supported by a systematic theory
taining to evaluation, judgement, and preference models
led multiattribute utility theory (MAUT). Von Winterfeldt
Fischer (1975) described various multiattribute utility
dels, which they defined as "a class of psychological
asurement models and scaling procedures which can be applied
the evaluation of alternatives which have multiple value
levant attributes (Von Winterfeldt and Fischer 1975, p.47)."
types of models, and the classes of problems they are
signed for, varied along three dimension: first, whether or
the choice alternatives have multiple attributes; second,
ether or not the input data is uncertain (thus requiring
obabilistic assessments); and third, whether or not the
oice is time-variable (i.e. do temporal considerations
fect the utility of a choice alternative).

Two key assumptions pervade throughout most of these MAUT
dels: independence of attributes and transitivity of
eference. (Note that these assumptions are similar to those
scussed for SEUT). There are varying levels of independence,
problems which exhibit stronger attribute independence

sually allow for more structured, decomposable, and useful AUT preference models. For example, multiattribute choice tuations where attribute values are certain and time-variant can exhibit four relevant levels of independence: no ndependence at all, 1-WCUI (weakly-conditional utility ndependence), *n*-WCUI, and joint-independence. Thus, there are our possible MAUT models to choose from.

WCUI refers to situations where the preference based on lues of one attribute is independent of constant values in e other attributes. For example, when buying a car, all her things being equal, you would want the cheaper car in l cases. Thus, one may consider price to be a WCUI tribute. However, consider two other attributes, size-of-car d existence-of-power-steering. Given two cars with all tributes being equal except size, where both cars have power eering, one would probably choose the larger car. However, the same case where both cars do not have power steering, e may very well prefer the smaller car, because it would be sier to handle. Therefore, in this case, size of car would t be a WCUI attribute with respect to existence of power eering (Von Winterfeldt and Fischer 1975, p59).

1-WCUI refers to problems in which it is possible to find single attribute that is WCUI to all others. *n*-WCUI refers problems in which *all* attributes are WCUI to all others. oint independence refers to cases where each *pair* of tributes is WCUI to all others.

For problems where no independence can be established, he MAUT preference model is forced to be very general and lgebraically non-decomposable. At the other extreme, where oint independence is established, it is a feasible and sound ractice to use a highly structured model called *conjoint easurement*, which is typically represented in an additive ormat, which states that vector *X* is preferred over vector *Y* f and only if:

$$\sum_{i-1}^{n} f_i(x_i) \geq \sum_{i-1}^{n} f_i(y_i)$$

Vectors *X* and *Y* represent two alternatives from which to oose, and each element of a vector represents an attribute consider when making the choice. As you can see, this model a generalization of the standard weighted-averaging linear del espoused by Dawes and others. With Dawes' model, the f_i nctions are simply weight-multipliers indicating importance vels.

3.1 MAUT Approaches to Non-Linearity

One of the problems with the standard linear model proach to MAUT, which is the method espoused by Dawes and ers, is that it does not deal with *nonlinear* combinations input data. Slovic and Lichtenstein (1971) discussed two ects where non-linearity could affect the judgement cess.

The first aspect of nonlinearity deals with *curvilinear* nonmonotonic) relationships between input and output ariables. An example of this is the relationship between the peed of driving a car and the likelihood of reaching a estination in time. In general, the faster the car goes, the ooner you will reach your destination, *up to a point*. After at, there is an increased likelihood of getting into an cident or being stopped by the police, which will certainly ow your progress toward the destination. Such curvilinear lationships can be expressed by using exponential terms in e policy equation, which changes the standard linear model a nonlinear one. Note that this is still consistent with e general conjoint measurement model described above. The sumption of independence between input variables is still intained, although there is no assumption of monotonic lationship between input and output variables.

The second type of nonliearity refers to what Slovic and chtenstein (1971) call *configurality*. This refers to the ssibility that an interpretation or weighting of one input riable may be influenced by the value of another input riable. An example of this effect is in stock market alysis. Suppose you are evaluating whether or not to invest a company's stock. You have two input variables: the rrent strength and activity of the company's stock and the neral stock market conditions. If the company has a strong ting, this is a positive sign. However, it is far more

meaningful to have a strong stock in a bearish market than to have a strong stock in a bullish market. In a bullish market, everyone looks good. In a bearish market, only the truly viable companies will continue to have consistent strength. Thus, the weight assigned to a company's current trading strength will be affected by the general market conditions. This introduces a nonlinear dependency into the MAUT framework, which cannot be dealt with by standard linear models, or even by curvilinear conjoint measurement models.

Decision analysts often deal with such configural dependencies through the use of *analysis of variance* (ANOVA) techniques. ANOVA is a statistical method, using non-metric categorical input variables, which measures the effect that the categorical value(s) of the input variable(s) have on the metric value of the output variable. For a one-way ANOVA paradigm (involving a single input variable), the formula is expressed as:

$$SS_y = SS_{\underline{between}} + SS_{\underline{within}}.$$

Here, Y is the output variable and SS_y reflects the total variance of the Y. $SS_{\underline{between}}$ indicates the degree to which this input-variable variance is due to the different categories of the input variables, and $SS_{\underline{within}}$ indicates the impact of other, non-categorical factors on this variance. The ANOVA paradigm can be extended to multi-attribute cases via MANOVA (multivariate ANOVA), which uses several categorical input variables. This model involves a tabular representation which

cross-correlates the effect of categories of the input variable on the output variable.

4.3.2 Non-Compensatory Decision Rules

The multiattribute algebraic methods mentioned above fall into the class of *compensatory* decision rules, because of the fact that performance of the judged entity in one attribute may compensate for or override the effects of its performance along other attributes. This is an inherent aspect of weighted additive or multiplicative models such as the ones discussed so far.

Decision analysts sometimes use *non-compensatory*, non-algebraic, methods for dealing with nonlinear issues affecting multiattribute decision-making. Non-compensatory decision rules deal with "quick reject" or "quick accept" situations. These can be expressed in a number of ways.

One non-compensatory heuristic frequently cited is called the *conjunctive* rule of decision making (Wright 1974, Fischer 1975). This is a "satisficing" technique that establishes the minimum acceptable values for each attribute, and then judges the alternatives on each of these attributes. An alternative's whose performance falls below the minimum threshold for any attribute will be instantly rejected. The conjunctive rule is thus an efficient mechanism for quickly eliminating obviously bad candidates.

A second noncompensatory heuristic is called the

disjunctive rule. This rule is concerned with finding attributes for which a candidate excels. It sets a upper threshold for each attribute, and candidates scoring above that threshold for any single attribute will be accepted, even if they do poorly in other attributes.

A third type of non-compensatory rule, called *lexicographic* compares candidates according to the most important attributes first. Those candidates that dominate the others according to this attribute are selected for further analysis. They are then compared according to the second most important criterion, and the most dominant candidates remaining are then passed on to the next selection stage. This process continues sequentially until there is only one candidate remaining or all the attributes have been analyzed. Thus, the lexicographic rule can be thought of as a phased, sequential version of the disjunctive rule described above.

Likewise, Tversky (1972) suggested another non-compensatory rule called *Elimination By Aspects* (EBA), which is similar to the lexicographic rule, except that it is conjunctive in nature. Using this model, decision makers compare candidates according to the most important attribute, as in the lexicographic model. the difference is that, like the conjunctive rule, a lowest-bound threshold is established for this attribute, and candidates falling below this threshold are eliminated. This process is continued for the next-most important attribute, then the third-most important,

etc. until there is only one candidate left or the candidates have been examined for all attributes. Thus, the EBA model combines the sequential, phased flavor of the lexicographic rule with the "process of elimination" flavor of the conjunctive rule. Tversky found that the EBA rule was a good descriptive model of human decision making in a wide variety of choice situations.

When do subjects use compensatory rules and when do they use non-compensatory rules? Empirical studies in decision-making indicate that subjects tend to use non-compensatory conjunctive and EBA rules early on in the selection process in order to weed out the poor candidates and later use compensatory weighting methods and disjunctive rules on the reduced set of alternatives (Slovic et al. 1977) . Secondly, they tend to use noncompensatory rules in particularly complex situations involving incomplete data and time constraints. The key is to use the simpler-to-implement non-compensatory strategies for as long as they are useful, and then use more detailed compensatory analyses when dealing with alternatives whose appeal are similar. Third, subjects tend to use compensatory rules when their task is to rate an individual candidate, and tend to use non-compensatory quick-elimination methods when their task is to choose from or rank-order a large number of candidates.

.3 Applications of MAUT: Multiattribute Utility Technology

Edwards and Newman (1982) developed a systematic hodology for social program evaluation, based on the axioms MAUT. They called this method Multiattribute Utility chnology. Their approach is intended to apply the eoretical constructs of MAUT into a decision aid tool. From ne perspective of the knowledge engineer, it provides a basis albeit in a manual, non-computerized form) for structuring a task-specific architecture for multi-attribute evaluation problem-solving. Chapter 5 of this thesis describes Edwards and Newman's method for knowledge acquisition of MAUT-related tasks. This method is particularly useful for the Candidate Evaluation architecture described in chapter 5, because it describes how to obtain and weight the evaluation criteria needed for performing evaluation tasks.

4.4 Combining MAUT and Linear Models with AI

Just as Langlotz sought to combine the decision-analysis strengths of SEUT with the explanatory power of AI in his QXQ system (as described in chapter 3), I am combining MAUT models with explanation and non-linear factors in a task-specific problem solving architecture called *Candidate Evaluation*, implemented in an expert system shell called *CEVED/CEVAL*. The architecture and shell will be described in chapter 5, but now I will discuss previous applications of linear models and MAUT in artificial intelligence.

The idea of using linear weighted models in AI evaluation ks is not new, and was employed in early *evaluation ctions* for state-space search problems and game-playing stems as far back as the 1950s. Two users of this technique r game-playing systems were Samuel (1959), who used in a ecker playing system, and Berliner (1977), who used it in a ackgammon playing system. In each case, the *linear polynomial* as used to evaluate the worthiness of potential board positions, and was used in conjunction with game-tree search strategies such as *minimax* and *alpha-beta pruning*. Samuel in particular was interested in studying how a system learned which features to select for the evaluation function and how to weight each of these features.

Both researchers found that a straight linear polynomial method to evaluation had a significant drawback. The context of the evaluation was not accounted for, and therefore decisions made based on this evaluation function could be in error. This finding would seem to contradict the claims made by Dawes and other proponents of the linear model.

4.4.1 Samuel's Signature Tables Revisited

For Samuel, the problem is that the linear polynomial method treats individual features as if they are independent of each other, and thus does not account for interactions between features. Samuel found that this greatly inhibited the quality of learning in his checker playing system. Thus, he

introduced *signature table* method, described in chapter 2 (and generalized into the *structured matching* generic task), which results in a board position's overall score being specific to the *non-linear combination* of features of the board position being evaluated. As mentioned in chapter 2, this technique introduces a significant space-complexity problem, which is alleviated somewhat by restricting the number of features to evaluate and by arranging the features into a hierarchy of signature tables. Note that any notion of "weighted scoring", the primary activity of the linear polynomial method, is totally abandoned in the signature table approach. This forces the space of potential feature values to be discrete, whereas the linear polynomial approach allows for a continuous space.

1.4.2 Berliner and Ackley's Hierarchical Weighted Scoring

Another hierarchical static-evaluation technique was eveloped by Berliner and Ackley (1982). They had done revious work on linear polynomial evaluation functions Berliner 1977) and found similar problems as those perienced by Samuel. Their domain, like Samuel's, was me-playing, where the game was backgammon.

Berliner's earlier program, BKG, used a straight linear ynomial function to rate board positions. All board itions (i.e. states in the state-space) were treated ntically, where the same linear polynomial function was to evaluate each one. Berliner soon discovered similar

roblems as those found by Samuel, namely that the linear olynomial was too rigid to account for the *context* of the oard position. Samuel had described the context in terms of he interrelationships between features. Berliner expressed ontext by partitioning the space of board positions into *tate-classes*, which will be discussed later.

Like Samuel, Berliner and Ackley moved from a straight inear polynomial function to a hierarchical representation of he evaluation features, in a new program called QBKG. owever, they differed from Samuel by not abandoning the inear polynomial method entirely. Instead, *primitive features* ould have weights and scores associated with them, and the eighted scores would be propagated through the hierarchy in rder to obtain scores for higher-level aggregate features called *concepts* in B&A's terminology). In effect, they aintained the ability to deal with a continuous space of ossible feature values, while Samuel's method forced that pace to be discrete.

The following sections describe specific differences etween Samuel's signature table approach and Berliner's SNAC ethod for multi-attribute evaluation. These differences are ummarized in Figure 4.1.

.4.3 *Continuous vs. Discrete Representation*

Thus, a major difference between the two approaches to valuation is one of discrete (Samuel) vs. continuous

(Berliner) representations of the evaluation space. Berliner and Ackley (1982) criticized discrete representations in evaluation functions, stating that they were *fragile* and suffered from the *boundary problem*.

By fragility, they meant that erroneous or noisy data could dramatically skew the results. In their words:

> "In a discrete medical diagnostic system using production rules, for example, an erroneous result on a test could prevent the system from ever making an accurate diagnosis, because the knowledge relating to the actual disease is not used, due to the non-satisfaction of the condition portions of the relevant productions. (1982 p.214)".

They said continuous representations alleviate the fragility problem by ensuring that all relevant factors are taken into account by including them in an overall scoring process.

Actually, continuousness of representation *per se* is not what alleviates the fragility problem. Rather, it is the use of a compensatory scoring mechanism that dampens the effect of erroneous or incomplete input data. This would be true in discrete representations as well.

By the boundary problem, B&A were referring to the tendency for systems with large grain sizes to make erroneous decisions when a feature's actual value (as it would appear in a continuous domain) lies in a grey area at the boundary between two possible discrete values, and is mapped arbitrarily onto one of those discrete values. The idea here is that fine granularity is less likely to produce errors than

	Signature Tables	*SNAC*
Knowledge Primitives	n-dimensional array representing mapping from specific pattern of input values onto a score.	weighted algebraic summation (linear model)
Evaluation Space	must be discrete, since each input value is represented as a cell in the array.	can be discrete or continous.
Organization	hierarchical...lower level signature tables send their scores up the hierarchy to be input values for higher level signature tablses.	hierarchical...lower level concept scores are weighted and summed to arrive at higher-level concept scores.
Context Representation	Context is represented as explicit pattern of parameter-values.	Context is represented by state-classes, which affect weights of importance.
Explanation Methods	Discrete representation allows for explicit description of specific combination of factors leading to score.	Continuous representation requires "discretizing" after scoring has taken place.
Original Application	Evaluating board positions for checker-playing program	Evaluating board positions for backgammon playing program.
Associated Generic Task	Structured Matching	Hierarchical Linear Model Component of Candidate Evaluation

Figure 4.1

Comparison of two AI methods used for static evaluation functions.

coarse granularity. This can be a significant problem with structured matching, which imposes a small limit on the number of allowable values that a parameter may take on. In the interest of computational tractability, structured matching forces coarse granularity, thus becoming vulnerable to boundary errors.

However, there are two problems with continuous representation based on scoring methods that make qualitative, discrete representations more attractive. The first is that scoring methods cannot account for interactions between variables in the way that conjunctive-clause production rules can. The second is that explanation is much easier with discrete representations.

Berliner and Ackley's system does not appear to account for specific interactions between variables the way that Samuel's signature tables can. However, by arranging their scoring into a feature hierarchy they were able to provide explanations at various degrees of abstraction for their system. This is similar to Page's (1977) functional explanation of signature table inference.

4.4 Context in evaluation

Another distinction between the Samuel's and Berliner's approaches involves the *context* of the evaluations they perform. Although both Berliner and Samuel express context in terms of abstraction levels via the hierarchical arrangement

of their knowledge groups (signature tables and scorers), they differ in other expressions of the context of the game.

For Samuel, the context is represented specifically by the interactions of the variables at each signature table in the hierarchy, as described previously. Contextual factors and evaluative factors are treated uniformly, all are represented as variables in the system.

In contrast, Berliner's method, which does not explicitly show interactions of variables, represents context by dividing possible board positions into *state classes* (Berliner 1979), which are categories describing overall aspects of the game. For example, one state class contains all "endgame" board positions, where the pieces for each player have already passed each other and the two sides are racing to take their pieces off the board. This is in contrast to another state class, "engaged game" where the opposing pieces may still land on one another. Context in this sense plays an important role in determining the *weightings* for the various features of a board position. For example, in an endgame context, it is no longer important to avoid having a piece stand alone, since no opposing piece can land on the lone piece. This would be an important consideration if the opposing players were "engaged". This dynamic, context-based weight adjustment introduces a non-linear flavor to their SNAC derivation of the linear polynomial evaluation function. Figure 2 illustrates this notion.

4.4.5 Explanation of Evaluation

As mentioned above, discrete knowledge representations tend to be better than continuous representations at facilitating the explanatory power of knowledge based systems. When dealing with overall scores, it is difficult to identify just what is wrong with the candidate being evaluated, or exactly why one candidate is better than another.

Thus, it would appear that the signature table approach is superior to SNAC in this regard. With QBKG, Berliner and Ackley wanted to maintain the performance advantages of the continuous representation, but keep the explanatory power of the discrete representations. They saw two main tasks in this regard:

1) Isolate relevant knowledge (pertaining to a query for explanation) from irrelevant knowledge.
2) Decide when quantitative changes should be viewed as qualitative changes.

They handled the first task via their hierarchical rrangement of features. Thus, explanations at lower levels ended to be narrow, and detailed, while explanations at igher levels were broad and unfocused. They handled the econd task by partitioning the differences in scores between ndidates (for any given feature) into "contexts", which may phrased as "about the same", "somewhat larger", "much rger", etc. In other words, they "discretized" the aluation space *after* all the scoring had taken place,

thereby avoiding the pitfall of losing valuable information due to arbitrary early classification. Once discretized, the qualitative differences in scores could be displayed as explanations for choosing one candidate over another.

4.4.6 Empirical Comparison of Truth Tables and Linear Models

Based partially on Dawes' research, Chung (1987, 1989) empirically compared the linear model approach against a rule-based or truth-table approach in terms of inductive knowledge acquisition methods for classificatory problem types. His study indicated that the relative performance of systems using these approaches differed based on the type of classification problem they were applied against. Specifically, he found that tasks involving conditional monotonicity are good linear candidates, whereas those violating conditional monotonicity are better rule-based or truth-table candidates. This finding is consistent with the findings of Dawes and Corrigan (1974).

4.5 Multiple Evaluators: Methods for Voting on Candidates

The previous discussions assume that the multi-attribute evaluation is being done by a single evaluator. However, in many real-world situations, evaluation is a team effort, jointly accomplished through several stakeholders who often have dissimilar priorities. Political elections are a prime example of this phenomonon. As pointed out by Edwards and

Newman (1982), the types of differences between evaluators can be fairly minor (involving differences in weights or score-assignments) or can be quite extensive (involving major differences in the selection of attributes upon which to base an evaluation).

Edwards and Newman pointed out that, so long as the various evaluators use a common set of attributes, differences between them can be accounted for by weight adjustment. However, with different evaluators using different sets of attributes, there is no way of effectively comparing their evaluations. This issue is not unlike the issue of *feature selection* in pattern recognition.

The issue of multi-evaluator weight assessment has been addressed in the decision science literature, and some researchers have proposed techniques for resolving evaluator differences. For example, Sheridan and Sicherman (1977) suggested a method of *electronic anonymous voting*, whereby each voter would rank-list his or her preferred candidates (which were represented by various values for two attributes), and based on the preference rankings, attribute weights would be established according to the axioms of utility theory.

Rank voting methods can also be used for feature election, particularly with respect to deciding the relative mportance of each of the features (Edwards and Newman 1982). or example, the list of "candidates" being ranked may be *tributes* as well as candidates being evaluated based on the

attributes. Chapter 5 gives more discussion about using attribute-ranking methods for weight assessment in MAUT-based evaluation tasks.

Voting methods can take many forms. Saari and Newenhizen (1985) identified several voting techniques. For example, in *plurality voting*, only the first-place alternative (or candidate) is chosen from each voter (evaluator). In *bullet voting*, a voter is given two votes, and is allowed to either: 1) cast one vote for his top ranked candidate, 2) cast one vote each for the top two ranked candidates, or 3) cast both votes for the top ranked candidate. In *approval voting*, a voter casts a vote for all candidates that he or she approves of; thus the voter has a total of *n* votes he can choose to cast (where *n* is the total number of candidates).

Each of these voting methods have been shown to be nontransitive (Saari 1985), and therefore suboptimal as candidate ranking methods. In fact, Arrow (1963) proved that no standard non-dictatorial voting method is perfectly transitive. In other words, all voting methods can result in a situation where the preference between any two candidates may change depending on whether other candidates from the total candidate pool are present in the ranking. Unfortunately, when this is the case, it is difficult to establish reliable attribute weights for an MAUT model via rank voting.

Thus, if attribute weights are to be obtained via rank

voting, then one should choose a voting method that minimizes the inconsistencies described above. Saari (1985) showed that the best non-dicatatorial voting method is the *Borda Count*. In this method, voters rank their preferred candidates. If there are n candidates, the Borda Count tallies $n-j$ points for a voter's jth-place candidate.

Intuitively, it makes sense that Borda Count will be superior to the other three voting methods. The Borda Count method captures more information than either plurality voting (in which each voter gives one point to his top candidate and zero points to all other candidates), approval voting (in which each voter gives one point to his top k candidates and zero points the remaining $n-k$), and bullet voting (in which each voter gives, at most, two points to a single candidate or one point to the top two candidates, and zero to the remainder). In all three of these voting methods, the information provided by each voter divides the candidate pool into at most two discriminable subsets (candidates that received a vote and those that did not), whereas for Borda Count, the information provided by a voter divides the candidate pool into n discriminable subsets (where n is the number of candidates) since each voter gives a different number of points to each candidate in the ranking. In essence, the Borda Count is the only representation that provides information about the total candidate ranking; all other voting methods assume voter indifference about lower-ranking

candidates.

A voting method can be described as a vector $\underline{W}^n$ = $\langle w1,w2,\ldots,wn \rangle$, where w_j points are tallied for a voter's jth-place candidate. For example, plurality voting is represented by the vector $\langle 1,0,0,\ldots,0 \rangle$. Borda Count voting is defined as $\langle n,n-1,n-2,\ldots 0 \rangle$. Saari's proof of the superiority of Borda Count is based on consideration of the set of all the possible final ordinal rankings of all subsets (of at least two candidates) from the total candidate pool that could result from a group of voters using a particular voting method. Call this set R_w. The total number of non-transitive outcomes from R_w is $|R_w| - n!$, where n is the number of candidates. Now, let R_B be the set of all possible final rankings using the Borda Count voting method. Saari proved that R_B is a proper subset of R_w for any $W <> B$. From this, it is a simple extension to show that $|R_B|-n!$ represents the smallest number of non-transitive outcomes, therefore making Borda Count the optimal voting method. For a full proof that Borda Count minimizes intransitive outcomes, see (Saari 1985).

Borda Count has other attractive properties not found in other voting methods. For example, with Borda Count, it is impossible for a Condorcet winner (i.e. a candidate that wins every pair-wise election) to be ranked last in a full ordinal ranking. Likewise, no Condorcet loser can be ranked first in a full ranking via Borda Count.

Another issue when dealing with multi-evaluator voting

for attribute weight assessment is the agenda-control of the voting procedure. The above discussion of voting methods assumed that all voters would be voting on all the candidates concurrently. However, in many organizations, this is not the case (Hammond 1986). For example, when doing binary comparisons between legislative option, the agenda can completely determine the outcome. Likewise, the organizational structure of bureaucracies can also act as an agenda, in the sense that the set of options and conflicts becomes smaller as the decision-making makes its way up the organizational hierarchy. Thus, the structure of an organization, like the structure of an agenda, can greatly influence the outcome of a vote.

Multi-evaluator issues also bring up the notion of istributed problem solving. Evaluation tasks can be istributed among multiple agents, whether via parallel ocessors or network nodes. The structure of a dimension erarchy makes it a simple matter to distributing the scoring subtrees to multiple agents. Alternatively, different nts could evaluate different candidates, so candidates d be evaluated concurrently. Various AI formalisms, uding blackboards, could be used to facilitate multiple- evaluation.

4.6 Conclusions

We have seen that multiattribute evaluation models based on statistical methods such as regression, correlation, and ANOVA are common frameworks in both decision sciences and artificial intelligence. Despite the limitations imposed by assumptions of independence, these models have proven to be useful for performing many tasks involving evaluation. In addition, their compensatory nature makes them less susceptible to input-error effects than non-compensatory discrete methods such as rules and truth tables. Finally, I showed that the coefficients of an MAUT model can be determined in a multiple evaluator situation through the use of voting methods, preferably a method that minimizes the possibility of non-transitive rankings such as Borda Count.

Thus, there is motivation to develop a problem-solving architecture combining MAUT principles with AI knowledge representation facilities. This would fill the same niche regarding MAUT that Langlotz's QXQ fills regarding SEUT, as discussed in chapter 3. In addition, since MAUT is usually used for evaluative reasoning, a resulting knowledge representation could be properly characterized as a task-specific architecture (as described in Chapter 2).

In the next chapter, I will discuss just such an architecture, called Candidate Evaluation. I describe it as a task-specific architecture, and compare it with other TSAs. In other chapters, I discuss its use in international

marketing, and describe another knowledge representation for use in "evaluative databases".

CHAPTER 5

THE CANDIDATE EVALUATION ARCHITECTURE

The previous chapters described some of the literature background of decision-theoretic and AI approaches to evaluation and selection problems. Particular attention was devoted to multi-attribute utility theory (MAUT) and its implementation in algebraic linear scoring models for compensatory decision-making as well as non-compensatory decision rules such as conjunctive, disjunctive and lexicographic models. Also discussed was a description and motivation for the notion of generic tasks (GT) and task-specific architectures (TSA). The purpose of this chapter is to marry the two disciplines (MAUT and TSA) in order to arrive at a general-purpose problem solving architecture for dealing with evaluation tasks. I call this architecture *Candidate Evaluation*.

Combining these two disciplines produces several advantages. First, MAUT is a tried-and-true, mathematically valid technique for evaluation and selection, as I showed in chapter 4. Second, the explanatory power of expert systems provides a verbal, qualitative contribution that can supplement MAUT's mathematical model, and thereby produce answers that users are comfortable with. Third, the use of TSAs provide a knowledge representation whose constructs are natural for the task at hand, thereby reducing the need to "twist the knowledge" into rules, frames, or other, more

primitive, representation paradigms. The motivation for this is to provide environments for *non-programming domain experts* to easily encode evaluative knowledge that can be used in expert systems, thereby reducing the problems encountered with the infamous "knowledge acquisition bottleneck".

This chapter presents a detailed description of the Candidate Evaluation architecture, its structure, behavior, and use. First, I present the developmental principles behind the architecture. Then, I show how the architecture satisfies these principles. Then, I describe the structure and behavior of the architecture, as it is implemented in the expert system shell *CEVED/CEVAL*. I analyze Candidate Evaluation from the perspective of the Generic Task point of view, and then from the perspective of MAUT. Finally, I discuss how knowledge acquisition and expert-systems validation would be done using this architecture.

5.1 Developmental Principles for a Candidate Evaluation TSA

The philosophy guiding the development of the candidate evaluation architecture is based on the following principles:

The architecture should allow for all types of evaluative decision making, including both compensatory and non-compensatory evaluation, using both discrete and continuous representations of the evaluation space. The architecture must allow for quick-reject or quick-accept decisions (non-compensatory decision rules) as well as

thorough assessment of strengths and weaknesses in the candidate being evaluated (compensatory decision rules). It should also allow the evaluation process to be sensitive to non-evaluative, contextual, factors in the environment.

The architecture should adhere to the task-specific architecture (TSA) school of thought, and as much to the generic task framework as possible. In particular, its conceptual primitives should be natural for representing evaluative knowledge, and the problem-solving method it embodies should be applicable over a wide range of problem domains.

The architecture should allow for a rich explanatory facility, taking into account non-linear combinations of features, and expressing various levels of abstraction. The evaluation and explanation process should be easy for the novice to interact with, and should provide effective, valid evaluations and recommendations.

The architecture should be implementable in an expert system shell. This shell should be directly usable by *non-programming domain experts*, who will use the framework of the architecture to encode their knowledge into performing evaluative expert systems. There should be no need for an intermediary AI programmer to encode evaluative knowledge. The knowledge acquisition bottleneck should thus be eased somewhat.

5.2 Overall Description of the Candidate Evaluation Architecture

The Candidate Evaluation architecture meets these goals in the following ways:

It incorporates all the evaluation methods mentioned above. For compensatory decision-making, it utilizes a linear model (MAUT) approach that, like Berliner's SNAC, provides for dynamic weight adjustment based on context. Like Samuel's signature table approach, it also takes into account the interaction of variables, at least at an abstract level, by mapping combinations of feature ratings onto recommendations. It also provides for quick-reject (conjunctive or EBA decision rule) or quick-accept (disjunctive or lexicographic decision rule) decisions by allowing the developer to set threshold levels for any single feature or composite feature.

The architecture adheres to the TSA and GT philosophies in the following ways. First, its semantic structure and conceptual primitives are at a level of abstraction that is meaningful for evaluative tasks. The primitives include: a hierarchy of composite features (*dimensions of evaluation*); a set of *evaluative questions* (equivalent to QBKG's primitive features); a set of *contextual questions* for dynamic weight adjustment (serving the same purpose as SNAC's state-classes); and a set of *recommendation fragments* which, like Samuel's signature tables, account for interactions of feature ratings. These will be discussed in detail later. Second, in keeping

with generic task requirements, the architecture is applicable across a wide variety of domains. Many problems requiring evaluative reasoning are solvable using this architecture.

The architecture has a rich explanation facility intertwined with its conceptual structure. Like QBKG, the hierarchy of features allows for explanation at various levels of abstraction. Also like QBKG, there is a post-evaluation mapping from the continuous space (score) onto a discrete space (feature rating), which allows for qualitative expressions of the evaluation. In addition, because recommendations are tied to combinations of feature ratings, the system can provide explanations for interactions of variables, like Samuel's signature tables and Bylander's structured matching. Finally, textual explanations can be tied to specific questions or dimensions.

The architecture has been implemented in an expert system shell, including a development environment (CEVED) and a run-time module (CEVAL). These will later be discussed in detail. The shell is very easy to use, and has been used by graduate students and domain experts in international marketing. None of the users had prior computer programming or AI experience, yet they were able to quickly learn to use the tool and have developed a dozen international-marketing related expert systems using the tool.

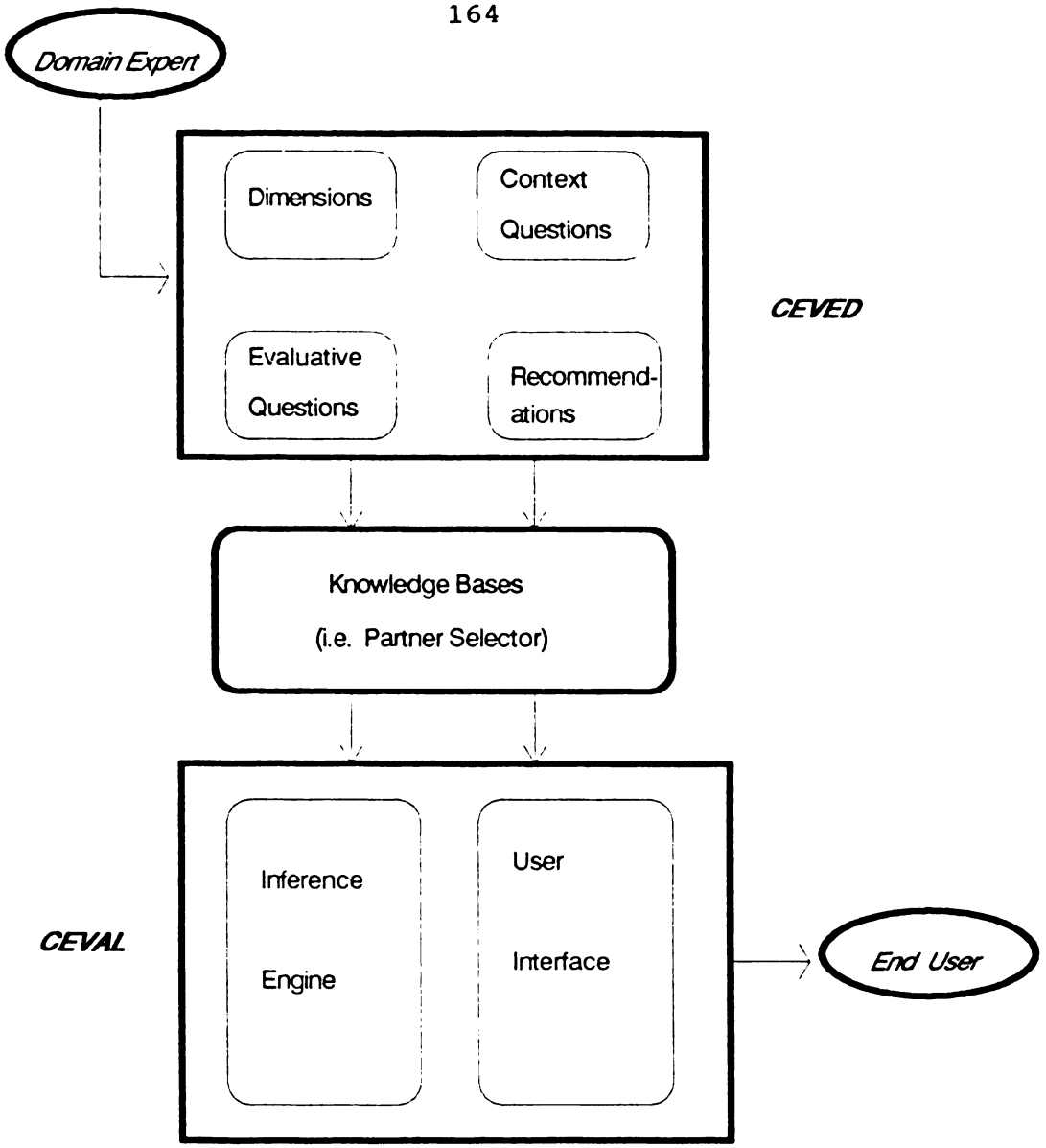


Figure 5.1

Structural components and flow of knowledge in the CEVED/CEVAL system. Domain expert creates knowledge bases using CEVED. End user is led through consultation via CEVAL.

5.3 The Candidate Evaluation Shell -- CEVED AND CEVAL

The shell for implementing candidate evaluation is described below. It involves two components. The Candidate EValuation EDitor (*CEVED*) is the development module. The Candidate EVALuator (*CEVAL*) is the run-time module.

5.3.1 Candidate EValuation EDitor (CEVED)

CEVED is a development environment intended to make it easy for a non-programmer to represent evaluation-type knowledge. There are four main types of objects that can be represented via CEVED: *dimensions of evaluation, contextual questions, evaluative questions*, and *recommendation fragments* (see figure 5.1). In keeping with the requirements for user-friendly development environment, and to avoid the feel of a "programming language", CEVED requires only two types of input from the user: menu choice and text processing. The text-processing facility of CEVED is for typing in explanations and recommendations, and includes many text-editing features found in standard word processors, including cut-and-paste, text file import and export, etc.

Dimensions of Evaluation CEVED allows the developer to define a hierarchy of abstracted candidate features, called *dimensions*, which serve as the baseline for evaluating the candidate. A dimension is made up of the following attributes: (see Figure 5.2)

1) The dimension's *name*
2) Its *parent* dimension
3) a list of qualitative *ratings* and corresponding *threshold scores*. A rating is a verbal evaluative description of the dimension, for example, "excellent", "fair", "poor". A threshold score is a minimum quantitative score for which a given rating holds.
4) The dimension's *weight* of importance...ie the degree to which the dimension contributes to the overall score of its parent.
5) An optional *explanation* message. The developer can write a comment here that will help the end-user understand what the dimension is measuring.
6) Optional *threshold messages*. These messages are tied to a threshold score for the dimension. For example, the developer can define a *reject message* that would appear if the final score for a particular dimension falls below a specified threshold.

Dimensions are related to each other in a parent-child relationship (via the parent attribute), producing a tree-structured hierarchy. A parent dimension's overall score is based on a linear weighted sum of the scores of its children dimensions. Figure 5.3 shows a sample dimension hierarchy for an International Joint Venture Partner Evaluation expert system, called PARTNER.

Contextual Questions are multiple-choice questions designed to establish the weights of importance of various features (dimensions) based on the context. They serve essentially the same purpose as Berliner's state classes. The contextual questions contain the following attributes:

1) The *dimensions* (features) to which the question pertains.
2) The question's *text*.
3) A list of *multiple-choice answers*, and a corresponding list of *weight-adjustment values*. Each weight-adjustment value specifies the direction and degree to which the chosen answer will change each

associated dimension's weight (via multiplication).
4) An optional *explanation message*.

During a consultation, the answers that a user gives for contextual questions will determine the final weights that each dimension takes on. This process is illustrated in figure 5.4. Contextual questions will be asked before evaluative questions.

Evaluative Questions CEVED allows the developer to define multiple-choice evaluative questions which will be presented to the end user during a consultation. Questions are grouped into question sets. Each question set is associated with a lowest-level dimension (i.e. a leaf node in the dimension hierarchy).

An evaluative question contains the following attributes: (see Figure 5.5)

1) The question *text*.
2) The question's *weight* of importance. This identifies the degree to which this question contributes to the overall score of its question set.
3) A list of *answers* and their corresponding *scores*. The answers will be presented to the end user as a menu to select from during a consultation. Depending on the answer chosen, its corresponding score will be assigned to that question.
4) Optional *threshold* and *explanation messages*. These are similar to the messages defined above for dimensions.

During a consultation, The overall score of a question set is a linear-weighted sum of the questions' weights and their scores based on user answers during a consultation. This score provides the rating for the question set (leaf-node dimension), and is propagated upward to contribute to the scores of the dimension's ancestors in the dimension

hierarchy.

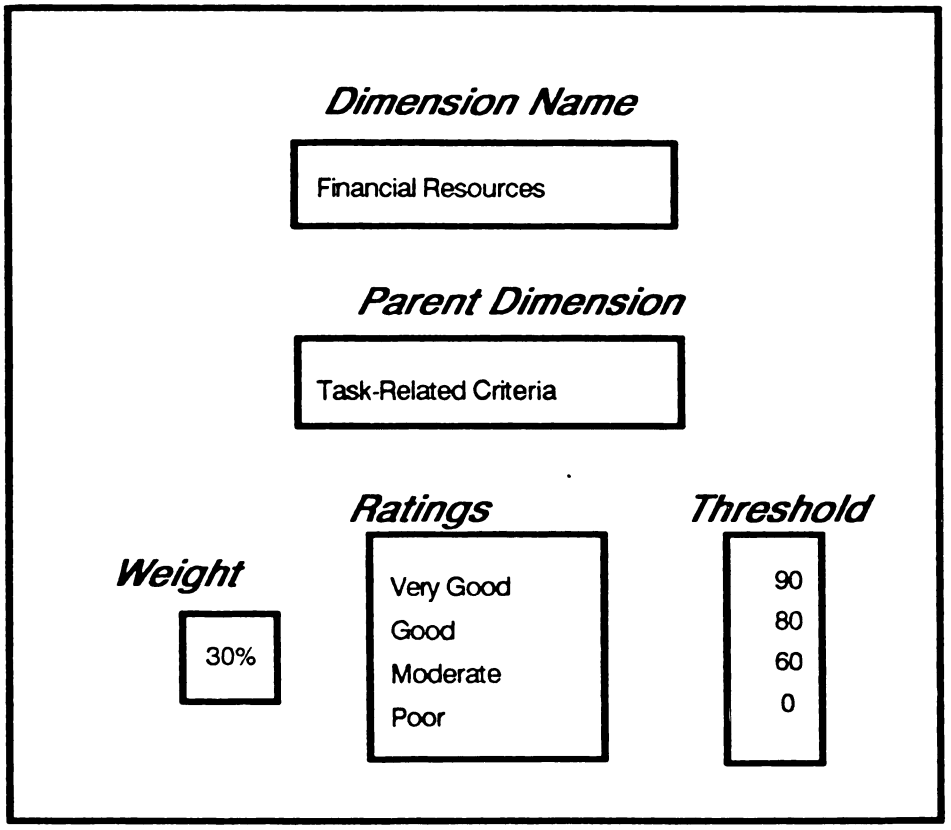


Figure 5.2

A sample DIMENSION ENTRY screen in CEVED. Developer uses this screen to enter dimension's name, parent, importance level (weight), ratings, and threshold scores.

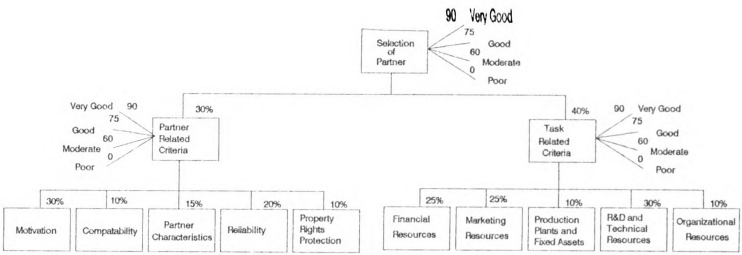


Figure 5.3

Example dimension hierarchy for International Partner Selector module.
(Some ratings and thresholds are shown)

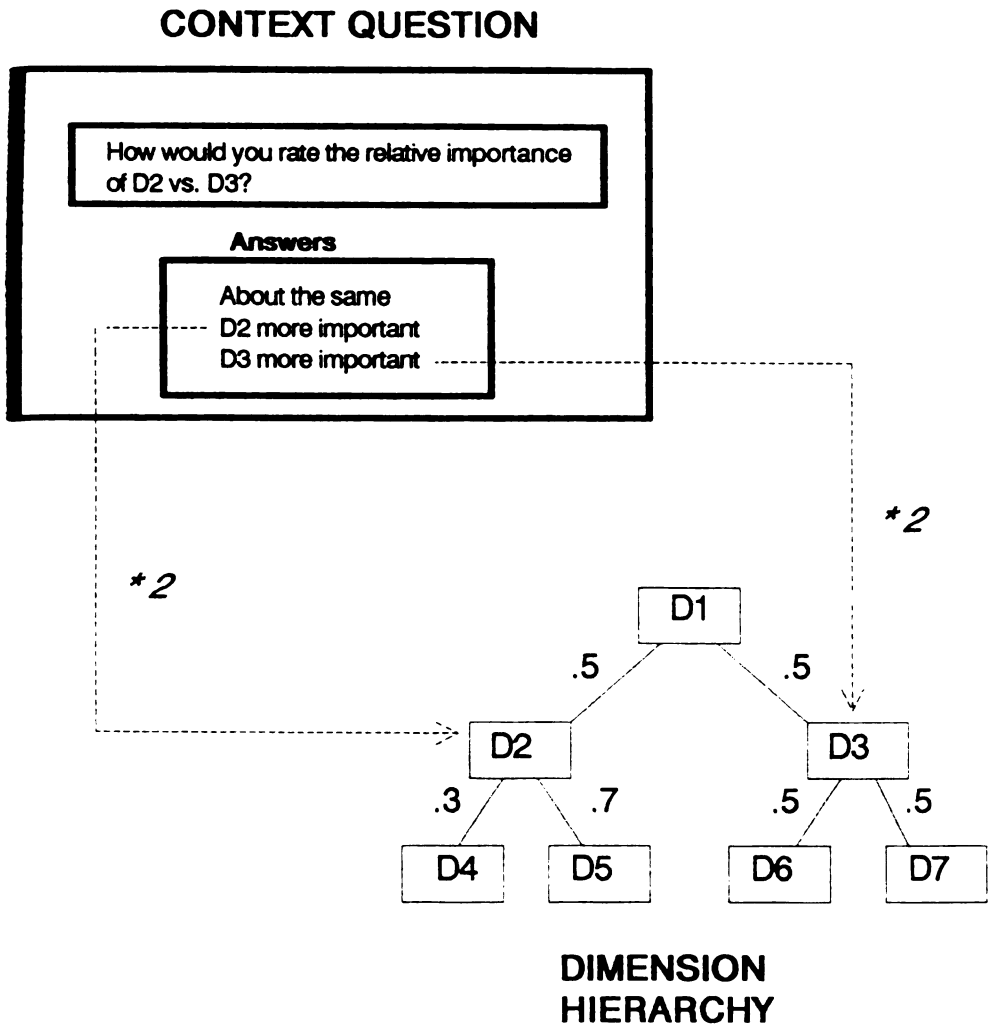


Figure 5-4

A hypothetical contextual effect on the default weighting of the dimension hierarchy. If answer 1 is chosen, there is no effect. If answer 2 is chosen, the default weight of D2 is doubled. Then, the weights of D2 and D3 are normalized so that their total is 1.0. This results in a final weight of .67 for D2 and .33 for D3. Answer 3 produces the opposite effect as answer 2. In this way, context questions allow for dynamic weight alterations during the course of a CEVAL consultation.

Recommendation Fragments CEVED allows the developer to define *recommendation fragments* and a *recommendation presentation strategy*. Recommendation fragments are linked to (and triggered by) combinations of dimension ratings. The recommendation presentation strategy controls the order in which recommendation fragments appear, and allows some recommendation fragments to suppress others.

A recommendation fragment includes the following attributes:

1) The recommendation *heading*. This is a one-line description.
2) The recommendation fragment's *presentation conditions*. The condition is made up of a list of dimensions and their desired ratings. A recommendation fragment will be presented only if the corresponding dimension's have the desired ratings.
3) The recommendation fragment's *text*.
4) The recommendation fragment's local *presentation strategy*. This includes a list of all other recommendation fragments that this recommendation fragment suppresses, or prevents from appearing. It also includes a list of all recommendation fragments that it is suppressed by. This way, the developer of a candidate-evaluation knowledge base can prevent redundant or conflicting recommendation fragments from appearing together.

The global *recommendation presentation strategy* involves a recommendation *ordering strategy* and a recommendation *suppression strategy*. The ordering strategy allows the developer to describe, in general terms, the order in which recommendation fragments should be presented to the end user during a consultation. For example, the developer can specify that more abstract fragments come before less abstract ones, or that fragments tied to more important dimensions should

appear before fragments tied to less important ones. The suppression strategy allows the developer to define, in general terms, which types of recommendation fragments will prevail if two or more redundant or conflicting fragments have satisfied their presentation conditions.

5.3.2 Candidate EVALuator (CEVAL)

CEVAL is the run-time inference engine that executes the knowledge bases developed via CEVED. It presents the questions to the users, inputs their answers, scores and rates the dimensions of the dimension hierarchy based on these answers, and presents a final recommendation to the user based on the dimension ratings and the recommendation presentation strategy.

CEVAL's inference behavior can be described as a weighted depth-first traversal of the dimension hierarchy. First, contextual questions are asked in order to determine the weights of the various dimensions in the hierarchy. If the resulting weight of any dimension is zero, that dimension and its subtree are pruned from the search, thereby reducing the number of questions asked. Then the depth-first traversal takes place, where the most "important" (i.e. high weight) dimensions are explored first. When a leaf-node dimension (an evaluative question-set) is reached, the evaluative questions in that set are presented to the user, and the user's answers are input. Then, CEVAL determines the score of the question

set via a linear-weighted sum of the questions' weights and answers' scores, and propagates the score up the tree to determine minimum and maximum possible scores for each ancestor of the question set dimension (see Figure 5.6). If any dimension's (or question's) score falls below (or above) its quick-reject (or quick-accept) threshold, a message appears recommending to terminate the evaluation and make a reject (accept) decision immediately. This is how CEVAL implements non-compensatory evaluative reasoning.

After propagating the score up the tree, CEVAL attempts to establish the qualitative rating for the ancestor dimensions of the question set. If any ratings can be determined (i.e. if the minimum and maximum scores for a dimension are within a range that corresponds to a single rating for that dimension), CEVAL triggers any recommendation fragments tied to those dimension ratings. Each triggered recommendation fragment checks its presentation conditions, and if they are satisfied, the recommendation fragment is added to a recommendation agenda list. The ordering and suppression strategies are then used to order and prune the recommendation list.

After the recommendation fragments have been processed, CEVAL resumes the traversal of the tree. It continues to do this until either all the questions have been asked, or enough questions have been asked to qualitatively rate all the dimensions deemed relevant by the end user. Then, the

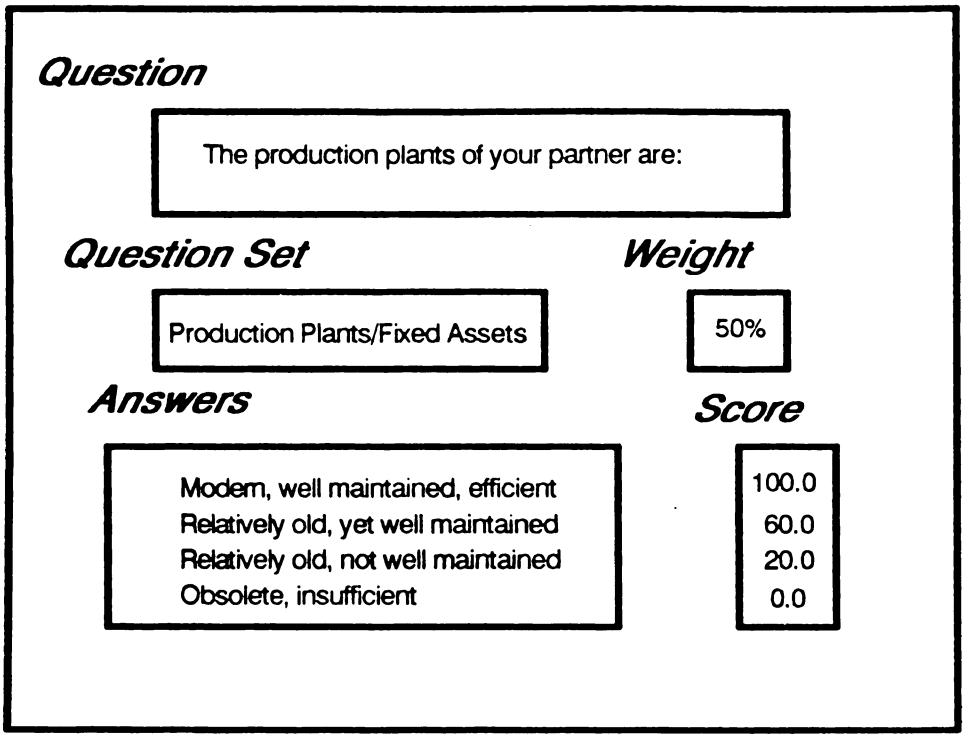


Figure 5.5

A sample EVALUATIVE QUESTION ENTRY screen in CEVED. The developer uses this screen to enter the question's text, weight, possible answers and corresponding scores, and the dimension (question set) to which the question belongs.

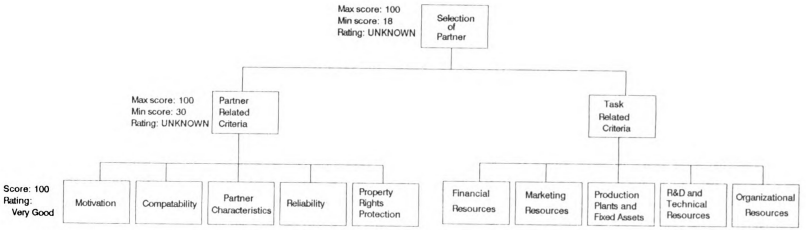


Figure 5.6

A possible scenario of score propagation partway through a CEVAL consultation. Here the user has just completed the Motivation portion, scoring 100%. Rating is known for Motivation, and max- and min- scores propagate up the tree. Ratings are still unknown at higher levels in the hierarchy.

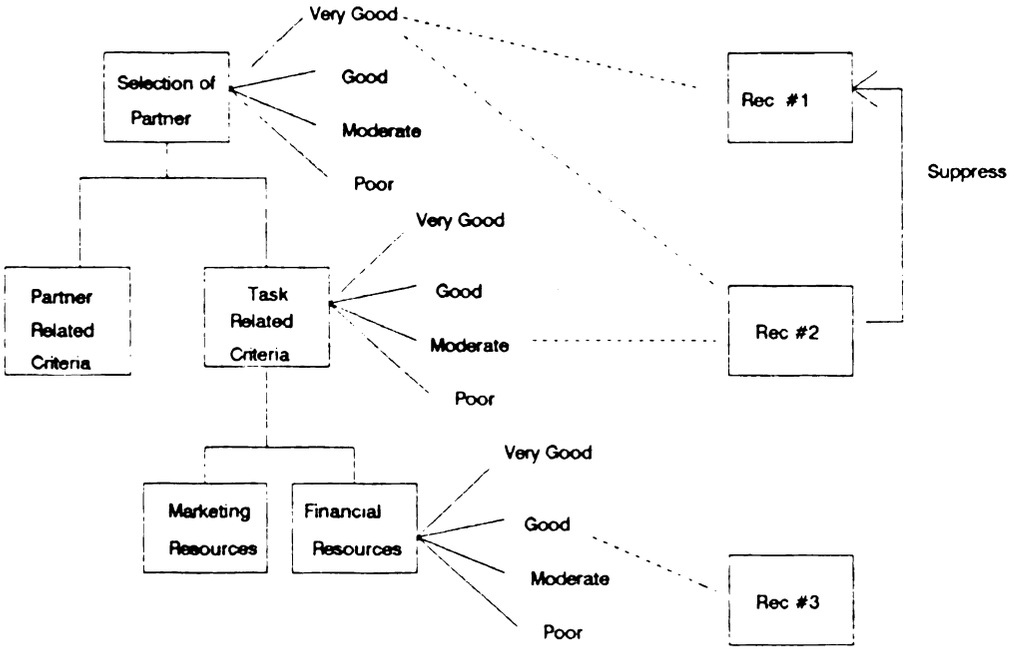


Figure 5.7

Dimension-ratings and recommendation fragments are linked via trigger/condition links (dashed lines). Recommendation fragments may be linked to each other via suppression links. Suppression links help prevent redundant and/or conflicting recommendation fragments from appearing in the same recommendation.

recommendation fragments that have remained on the recommendation list are presented to the user based on the ordering and suppression strategies. Thus, the overall recommendation is a combination of the recommendation fragments whose conditions have been met and which have not been suppressed.

As an example, see Figure 5.7. Assume that the user obtained Very Good for Selection of Partner, Moderate for Task Related Criteria, and Good for Financial Resources. In this case, recommendation fragments 1, 2, and 3 all satisfy their conditions. However, because fragment 2 suppresses fragment 1, this will cause fragment 1 to be deleted from the final recommendation fragment list. Also, if the ordering strategy indicates that highly abstract recommendation fragments appear after more detailed recommendations, this will cause recommendation fragment 3 to be displayed before recommendation fragment 2.

CEVAL allows the user to specify whether or not s/he wants detailed explanations to appear during the question-and-answer process, and to determine whether s/he wants the recommendations to appear after each score propagation or only at the end of the entire consultation. In addition, the user can save a consultation for re-running at a future time, and can save recommendations that result from these consultations.

5.4 A Generic Task Analysis of Candidate Evaluation

Candidate Evaluation, as implemented in CEVED/CEVAL, is clearly a TSA. But is it a *generic task*, in Chandrasekaran's meaning of the word? I believe the answer is no, primarily because its structure and reasoning process can be decomposed into at least two other generic tasks, structured matching and abduction.

5.4.1 Structured Matching and Candidate Evaluation

Actually, there are significant differences between the dimension hierarchy of Candidate Evaluation and the structured-matching/signature-table hierarchies described in chapter 2. The main difference is that each node in the structured-matcher is a truth table mapping specific patterns of input values onto a specific score (discrete, non-linear representation), whereas each node in the Candidate Evaluation dimension hierarchy implements a weighted linear additive averaging process. Nevertheless, there is significant overlap in the type of task performed by these two representations. The purpose of both representations is to evaluate. Both representations are represented in a tree structure. In both representations, scores are propagated up the tree.

This again bring to question, what is a generic task? Is it defined by the broad purpose that is being served? Or does it also embody the method in which the purpose is achieved? If the former, than the hierarchical linear model of CEVAL and

the truth-table hierarchy of structured matching are merely alternative implementations of the same generic task. If the latter, than the hierarchical linear model is indeed a new generic task, something distinct from structured matching.

In either case, the difference between the hierarchical linear model and the hierarchical truth table are very significant with respect to their implications for knowledge acquisition. Specifically, the hierarchical linear model is better for acquiring and representing compensatory decision rules whereas the truth-table representation is better for noncompensatory decisionmaking.

Compensatory decision-making can be represented very naturally using the hierarchical linear models (such as Berliner's SNAC method) because weights of importance for various attributes can be explicitly represented in the SNAC architecture. Also, modifying the compensatory evaluative knowledge of the system is easy with a SNAC-like approach. If an expert or knowledge engineer decides to change the importance level of a particular feature, all s/he has to do is change that feature's weight value. By contrast, the signature-table approach (and structured matching) is awkward for representing the compensatory rule. The relative importance of each feature is not explicitly represented, but rather is implied by the combination of feature values in a pattern of the truth-table. This makes it difficult for the observer to ascertain which attributes are important and which

are not. Also, changing the importance level of a single feature may require making changes to several patterns in a truth table, creating a maintenance nightmare for knowledge engineers dealing with large knowledge bases.

The discrete nature of signature tables and structured matchers makes it easy and natural to represent non-compensatory evaluative knowledge. Structured matching in particular allows "quick reject" or "quick accept" parameters to be evaluated early on, and thereby cut down on unnecessary computing. By contrast, linear polynomial methods, including SNAC, are not very good at representing or reasoning via non-compensatory decision-rules. A continuous representation is unnecessary for such "threshold" issues. Additionally, the compensatory nature of weighted scoring makes it difficult to minimize the number of variables that need to be resolved in order to make a decision.

Thus, we see that linear models, or variants thereof, are generally better at representing compensatory evaluative knowledge whereas truth tables are better for handling non-compensatory evaluation. The hierarchical linear model of CEVAL is suited for compensatory but not noncompensatory decisionmaking. However, noncompensatory decisionmaking is facilitated through other CEVAL mechanisms like reject/accept thresholds. Additionally, the recommendation generation process can be used to deal with nonlinear combinations of findings.

5.4.2 Abductive Assembly and Candidate Evaluation

The recommendation processing of CEVAL is very much like *abductive assembly*. It makes sense that this would be the case, because the entire purpose of the recommendation fragments are to *explain* the findings of the dimension hierarchy. Abduction is first and foremost a means of explanation, not a deductive process. A comparison of the CEVAL's recommendation generating process with PEIRCE's abductive assembly algorithm (described in chapter 2) shows that both perform the same broad steps:

1) Find the "hypotheses" that are consistent with the findings.

2) Prune out inconsistent or redundant "hypotheses."

Step 1 embodies the same "set-covering" behavior that you see in all abductive systems. With RED and PEIRCE, this is done sequentially, based on a request to account for a specific finding. Steps 1-3 of PEIRCE's algorithm shows this. In CEVAL, recommendation fragments are triggered and added to a list if their preconditions match the dimension-rating pairs in the dimension hierarchy. This can be done if the user wants to explain a particular finding (dimension-rating), or if the user wants an overall interpretation performed.

Step 2 is accomplished in RED and PEIRCE by testing the matching hypotheses (i.e. those that account for desired finding) against the compatibility constraints. In CEVAL this

is accomplished through the use of the suppression strategy, where certain recommendation fragments can suppress other ones.

The terminology of abductive assembly has analogies in CEVAL. "Hypotheses" are the same as recommendation fragments. A "compound hypothesis" is like a full recommendation in CEVAL.

5.5 Candidate Evaluation as an Implementation of MAUT

It is obvious that CE draws much of its conceptual structure from the MAUT model. The idea of the algebraic weighted model, where different terms reflect different attributes of the option being judged, comes right out of the MAUT literature. In addition, the notion that context plays a role in determining the weight of an attribute is consistent with the general conjoint measurement approach, in which the coefficients for the attribute terms are described as functions and not as constant numeric values, such as what you'd find in a straight linear model. In this regard, CE provides the sort of structure required for compensatory decision-making. Because of the capacity for dynamic, contextual weight adjustment, CE also deals with the non-linear MAUT issues cited by Slovic and Lichtenstein (1971), discussed in chapter 4. Context-based weight adjustment allows for conditionally non-monotone (curvilinear) relationship between input values and output values of the weighted sum

model, as well as some forms of configural relationships (assuming that context questions and evaluative questions are both considered to be input values).

Non-compensatory decision-making is also implemented in CE through the use of quick reject (i.e. conjunctive) and quick accept (i.e. disjunctive) threshold messages. CEVAL can be instructed to attempt a quick reject (non-compensatory) evaluation prior to performing an exhaustive, compensatory analysis. Thus, the Candidate Evaluation architecture is consistent with both the compensatory and non-compensatory aspects of MAUT theory.

5.6 Knowledge Acquisition for Candidate Evaluation

Knowledge acquisition is the most important, difficult, and frustrating aspect of expert systems development. As I argued in chapter 2, this process can be greatly simplified by the use of task-specific architectures such as Candidate Evaluation. This is because a TSA provides a knowledge-use level "language" for implementation and imposes a structure that forms as a blueprint for the knowledge acquisition process. In this sense the CEVED/CEVAL tool, like other generic-task and TSA tools (e.g. CSRL, DSPL, HYPER, etc.) can be thought of as knowledge acquisition aids as well as expert system shells.

The idea of using automated and semi-automated techniques for knowledge acquisition is not new, and has been well

documented throughout the history of expert systems. There have been several manual KA strategies that have led different types of computerized KA tools. Some strategies result in implementations of very general KA interviewing technique. One example of this is AQUINAS (Boose and Bradshaw, 1987), which implements psychological interviewing methods such as multidimensional scaling (Butler and Corter, 1986), and repertory grid analysis and personal construct theory (Kelly 1955). Other KA strategies involve developing domain-specific tools for obtaining expert knowledge. Examples of this include OPAL (Musen et al. 1987) for gathering cancer-therapy knowledge and STUDENT (Gale 1987) for statistical analysis consultation. A third class of KA strategies is to develop languages and knowledge structures for describing and defining ask-specific but domain-independent problem solving methods. This is the strategy used by Chandrasekaran and his colleagues, and it is this strategy that CEVAL/CEVED fits into.

Of course, the first step in the knowledge acquisition process is to identify the particular task type that is appropriate for the problem at hand. It would not do to use CEVED/CEVAL for representing design knowledge, just as it would be inappropriate to use DSPL for evaluation and assessment. However, once a problem has been identified as one for which Candidate Evaluation is an appropriate problem-solving methodology, the knowledge acquisition process can be

structured accordingly.

5.6.1 KA and MAUT Assessment Techniques

Because the Candidate Evaluation task is grounded so firmly in the tradition of MAUT and linear models, it is useful to review the methods that decision analysts use to acquire and fit knowledge to the MAUT format. These assessment procedures are used for both descriptive (empirical) and normative (advisory) purposes.

Edwards and Newman (1982) identified the major assessment tasks in the MAUT evaluation process as:

1) Identifying the entities being evaluated
2) Identifying the *stakeholders* (i.e. the evaluators) This is equivalent to identifying the "experts" in an expert systems design process.
3) Elicit from the stakeholders the important attributes (dimensions), and organize them into a hierarchy. Edwards and Newman call this a *value tree*. In CEVED/CEVAL terminology it is called a dimension hierarchy.
4) Assess for each stakeholder the relative importance of the attributes. (get the weights).
5) Ascertain how well each evaluated entity performs in the various dimensions at the lowest levels of the value tree. Edwards and Newman calls these lowest level nodes "location measures". They correspond to the evaluative questions of the Candidate Evaluation architecture.
6) Aggregate location measures with measures of importance. This is equivalent to the "score propagation" phase of the CEVAL inference process.
7) Perform sensitivity analysis by varying the weights and location measures.

From the perspective of expert systems design, it is really steps 2 through 4 that are knowledge acquisition steps. Steps 1, 5, and 6 are actually performance steps of a CEVAL

expert system. Step 7 may be considered an expert-system validation step. In this sense, the term "MAUT assessment" really refers to knowledge acquisition, knowledge representation, *and* actual performance of the system.

Because CEVED/CEVAL also includes explanation and recommendation facilities, which are not part of the MAUT model, there are additional aspects to KA for CEVED/CEVAL that would not be accounted for in Edwards and Newman's MAUT assessment procedures. Therefore, the following sections, which describe knowledge acquisition for Candidate Evaluation, will include both MAUT-based processes and non-MAUT aspects.

5.6.2 Identifying the Experts (or "Stakeholders")

Of course, identifying and enlisting help from the domain experts is a key aspect of the KA process. The idea of *domain expert* (from the AI terminology) is related to but somewhat distinct from the idea of *stakeholder* (in Edwards and Newman's MAU Technology). Domain experts are people whose knowledge and experience are being captured into the expert system. In many cases, these people will not be the final users of the system. Indeed, it is often the case that the domain expert is or will be unavailable for consultation (for example, s/he may be retiring from the job), and this unavailability is the prime motivator for capturing the knowledge into an expert system. In this sense, the domain expert may not have a stake in the final outcome of the system, other than personal satisfaction.

By contrast, a stakeholder is an actual decision-maker or at least someone who will be directly affected by the decisions made (Edwards and Newman 1982, pp. 33-34). These people have much more of a stake in the outcome of the decisions being generated by an expert system. They may also be domain experts, in that they know which criteria are important in the decision-making process. (Note: in the following sections I will use the terms "expert", "stakeholder", and "evaluator" interchangeably. This is not to imply that all stakeholders or evaluators are necessarily experts in the domain in question. Instead, this interchangeable use of terminology is done to draw analogies between the activities described in Edwards and Newman's MAUT methodology and the activities done in many knowledge acquisition techniques).

5.6.3 Identifying and Structuring the Major Criteria (Dimensions)

Edwards and Newman suggest the following guidelines in establishing the dimension hierarchy (or value tree):

From each stakeholder, obtain an exhaustive list of all the criteria (or attributes) that they think are important in the evaluation and decision process. It is probable that there will be some overlap in the criteria given by the various stakeholders, but there will also be considerable deviations.

From these lists, it is often the case that some

attributes are actually important for the evaluation process and others are merely topics of interest. For the MAUT and CE models, it is important to separate the attributes that are critical for evaluation from the non-evaluative attributes.

Often, the distinction between attributes given by various stakeholders reflect differences in terminology, and not meaningful semantic differences. Thus, it is useful to explicitly define each attribute and to standardize terminology in order to avoid "distinctions without a difference."

Group the attributes into common categories, and then group the categories into super-categories, etc. The idea here is to place similar concepts together so that explanations that are generated by the expert system can be made at abstract, summary levels, as well as more detailed level.

5.6.4 Identifying and Scaling the Indicator Variables (Evaluative Questions)

Edwards and Newman's term for indicator variable is *location measure*. They define location measure as "an assessment of how desirable an option is with respect to a particular twig or bottom node of a value tree (1982, p.65)." Thus, location measure is an expression of the utility of a particular candidate for a particular low-level attribute.

Edwards and Newman distinguish between two types of location measures. The first is an arithmetic transformation

of objective measures and the second is an arithmetic transformation of impressionistic judgements. CEVAL's architecture allows for both types of transformations, with the caveat that both the objective measure and the impressionistic judgement must be from a discrete and finite list of possible values. Currently, CEVAL has no mechanism for providing a continuous function mapping input values to scores or location measures. Therefore, if an input value comes from a continuous space, that space must be partitioned by the expert into ranges before a location measure is established. The location measure must be a discrete mapping from an input value to a score between 0 and 100. In this sense, the utility assessment done by CEVAL is incomplete with respect to MAUT, which allows for utilities to be calculated via continuous functions as well as discrete mappings.

In addition to identification of location measures (evaluative questions) and assignment of numeric scores based on input values, Candidate Evaluation requires the expert to establish threshold values for score-to-rating transformations. This is a requirement that goes beyond the standard MAUT model, because it involves assigning qualitative interpretations of the quantitative results. The expert must be careful about assigning verbal ratings to scores, since this assignment may lead to the boundary problem identified by Berliner and Ackley (1982) and discussed in chapter 4.

Page (1977) suggested a technique to establish thresholds

("cutpoints" in his terminology) for transforming a continuous input space to a discrete output space. He was using signature tables as a pattern recognition heuristic. The use of a signature table representation requires discrete variables, as discussed in chapter 4 of this thesis. However, many of the actual data types for input variables in his system were continuous in nature. Page's approach was to develop a computer program that would choose threshold values that maximize correct-prediction rates and maximize discriminability of candidates. This method requires a significant number of training samples to be input to the system, and is a "machine-learning" approach to threshold determination. Actual training samples are often not available when developing expert systems, so experts are forced to make a best guess at establishing threshold values. However, hypothetical training samples could be provided via Monte Carlo analysis. Such a facility could be provided to enhance the knowledge acquisition capabilities of CEVED. Chapter 8 discusses this as a potential future area of research.

5.6.5 Weight Assessment (default weights and contextual factors)

Assessing the weights of importance of each of the criteria (dimensions) and indicator variables (evaluative questions) is one of the most important aspects of the knowledge acquisition process. One possibility is to use

standard regression analysis with many training samples to arrive at these weights statistically. However, as discussed earlier, such "proper" linear models are often impossible to obtain because of the subjective nature of the criteria in question, the paucity of samples, and the possibility of colinearity of the criteria and indicator variables. Thus, the weights usually must be obtained from expert knowledge.

MAUT analysts have identified several methods of obtaining such weights. Edwards and Newman (1982) for example, described three such methods:

- using equal or unit weights
- determining weights from ranking
- ratio weighting

At first glance, the use of equal or unit weights would seem ridiculous. Surely not all attributes are equally important in most multiple-criteria evaluation situations. However Dawes (1979) pointed out that even unit weighting of variables can often produce "adequate" predictive results. In addition, as Edwards and Newman (1982, p.53) say, assigning unit weights is the simplest way to go, especially if there are multiple experts (or stakeholders) with widely variant opinions about the relative importance of the different evaluation criteria.

Nevertheless, in order to approach optimality of the evaluation process, some sort of differential weighting is called for. One way to do this is to ask the expert (or

stakeholder) to *rank* the criteria from most important to least important. This ranking could include assignment of weights to the rank-ordered criteria, which may be ordinal (simple ranking) or ratio (ratio ranking).

If there are multiple stakeholders involved, this brings into play the issues of voting method discussed in chapter 4. Note that assigning weights to rank-ordered criteria is analogous to assigning points to rank-ordered alternatives in a Borda Count voting method. Establishment of weights via ranking of criteria will be fairly complicated in multiple-expert situations, but could be facilitated through the use of these voting methods. Delphi studies are another way of establishing dimensions weights from multiple experts. Thus, in terms of the Candidate Evaluation architecture, Borda Count and Delphi studies could be incorporated as a multiple-expert knowledge acquisition technique. For each higher-level dimension, the experts could be asked to rank-list its subdimensions in order of perceived importance. Borda Count vote tallies could then be used to ascertain final weights. By combining the Borda Count method with ratio ranking, the KA process could also establish wide or narrow differences in importance levels. The same method could also be incorporated to establish weights of evaluative questions for a lowest-level dimension (a question set).

CEVED does not currently implement any of these techniques for weight assessment. It is merely a passive

recipient of expert-supplied weights in the sense that it asks the expert to assign weights to the various dimensions (or evaluative questions) and displays the list of dimension-weights so the expert can verify the distribution. In this regard, it is more similar to the generic task tools of Chandrasekaran's group than to the more active knowledge acquisition tools developed at other centers. As I discuss in chapter 8, the use of rank voting methods, delphi studies, and linear regression models could enhance the knowledge acquisition facilities in CEVED.

5.6.6 Interpretation Assessment (Recommendation Fragments)

This aspect of CE diverges from the strict MAUT decision-analysis model, because it pertains to explanation and interpretation of the final results. Here, we leave the realm of decision theory and enter the realm of expert systems and AI, particularly with respect to explanation of reasoning.

As described earlier, the final verbal interpretation of a candidate's evaluation is constructed from a set of *recommendation fragments*, each of which is a body of text that will be displayed if its associated conditions are met.

Also mentioned is the fact the maximum possible number of recommendation fragments is exponential with respect to the number of dimensions or attributes in a CEVAL model. Thus, it is important to use common sense when deciding just which combinations of dimension-ratings are relevant for the

particular problem at hand. Recommendation fragments should only be generated if they are important for the evaluation process.

A good guideline in this process is to create a recommendation fragment for each individual dimension-rating. Each of these initial recommendation fragments will have a single condition in its "if" clause. Thus, the system will be guaranteed to present a statement for each factor of the evaluation process. It also ensures computational tractability, since the number of these initial recommendation fragments will be equal to the product of the number of dimensions times the number of ratings, which is of polynomial complexity.

After creating these initial recommendation fragments, the next step is to identify the *relevant* nonlinear combinations of dimension-ratings and create recommendation fragments to deal with these nonlinear combinations. The key here is to keep the number of "combination" recommendation fragments to a manageable level. This avoids the computational complexity problems that could arise if recommendation fragments are created for all possible combinations of dimension-ratings.

Keep in mind that the purpose of the recommendation fragments is to provide an explanation, a verbal interpretation, of the evaluation results. This interpretation should be comprehensive enough to reflect all important

components of the findings. However, it should not be so exhaustive as to result in overly redundant, lengthy, and cumbersome explanations.

5.7 Validation and Verification of CE Expert Systems

This section presents some of the issues that should be considered when testing and validating the expert systems that are generated via the CEVED/CEVAL process. In addition, it suggests a multiple-criteria methodology for such testing, using a weighted-scoring method similar to that used by CORE and CEVAL applications themselves.

Gaschnig et al. (1983) identified four principles for evaluation of expert systems:

> 1) Complex objects or processes cannot be evaluated by a single criterion or number.
> 2) The larger the number of distinct criteria evaluated, the more information will be available on which to base an overall evaluation.
> 3) People will disagree on the relative significance of various criteria, according to their respective interests.
> 4) Anything can be measured experimentally as long as exactly how to take the measurements is clearly defined.

The implications of these four principles are: that a multi-criteria approach is appropriate for validating expert systems (based on 1 and 2); that a validation method should include a flexible weighting scheme (based on 3); and that a formal, systematic method should be developed (based on 4). Interestingly enough, these implications by themselves validate the candidate evaluation architecture as a general

evaluative method, which is based precisely on multiple-criteria, flexible-weighting measurement.

5.7.1 What should be measured?

Gaschnig et al. also identified several characteristics of an expert system that the validation/evaluation process should measure. Below, I discuss these characteristics as they relate to the Candidate Evaluation architecture.

5.7.1.1 The quality of the decision and advice.

In an ideal world, this would involve measuring correctness against an objective standard. However, for most expert systems domains, particularly in international marketing, the output of the expert system is qualitative and judgmental. It is difficult to establish correctness in an absolute sense. Thus, Gaschnig argued that the decision/advice of the ES should be measured against decisions that human experts would give based on the same information input.

For CEVAL expert systems, the decisions and advice takes on two forms:

1) A list of scores (0 - 100) and qualitative ratings (e.g. excellent, fair, poor, etc.) of the various dimensions (i.e. aggregate features) of the candidates.

2) A verbal, essay-like, assessment of the candidate's evaluation, in the form of recommendations, identifying strengths and weaknesses of the candidate and suggesting a

course of action to take.

5.7.1.2 The correctness of the reasoning techniques.

Because of the subjective nature of the advice generated by the expert systems, it is essential that its reasoning methods and problem-solving behavior be validated as well as the output. In CEVAL's case, the reasoning method includes the following characteristics:

1) The structure and content of the dimension hierarchy. Are all the relevant features for evaluation included? Are they arranged correctly?

2) The quality of the evaluative and contextual questions. Are the right questions being asked. Is there any redundancy? Are any important questions missing? Does each question have a complete list of multiple-choice answers?

3) The weighting scheme. This is probably the most important factor. Is the importance of the various dimensions and/or questions being assessed correctly?

4) The scoring scheme. Are the correct scores being assigned to each question?

5) The appropriateness of using a hierarchical, weighted scoring technique in the first place. In other words, is CEVAL the correct tool to use for the application?

5.7.1.3 The quality of the human-computer interaction.

Here, we are concerned with issues such as the

explanatory power of the system and the specific wording of the questions and recommendations. We are also concerned here with the ease of use of the CEVAL program itself.

5.7.1.4 The efficiency of the system.

How much of a time commitment is required of the user? Are irrelevant and unnecessary questions being skipped? Is the system taking up too much disk space or CPU time?

This is not as important of a criteria as the first three. Most CEVAL expert systems do not require massive amounts of user time commitment...they all involve 30-60 questions that the user must answer.

5.7.1.5 The effectiveness of the system.

This involve issues about the results of going through the expert systems. In other words, what tangible benefits did the users actually gain? In what ways did the expert system improve the users' understanding of the problem and/or the decisions that the users made? How much money did the expert system save? Such questions will usually require long-term study and may not be feasible for alpha testing.

5.7.2 How should measurement be done?

Gaschnig et al. gave four suggestions about evaluation methodology that are relevant for Candidate Evaluation:

1) Compare the results of the ES against the results of

human experts given the same input. The method suggested is to give a number of questionnaire results (i.e. questions and the answers given to those questions) to a number of experts and obtain *their* results. Experts would be asked to provide these results in the form of scores/ratings for dimensions and verbal assessment/evaluation. Then, the experts' results to the would be compared to results obtained from the expert system.

Note that obtaining results from actual experts would be useful for more than just comparison against the ES results. Expert results could also be used to validate weighting schemes. When a sufficient number of test samples are generated, a linear regression analysis can be performed, based on the final scores that the experts produce, to infer what weights should be.

2) Use blinding techniques to avoid bias in the evaluation. The researchers who do the comparison of results obtained from (1) should not know whether they are looking at the expert system results or the expert results.

3) Use a sequential process of validation. In other words, one study should validate the results of the system. Another, separate study should test the reasoning process. Still a third should measure the human-computer interaction. This way, the validators will know precisely what is the source of inadequacy when and if we find flaws in the system.

4) Use sensitivity analysis. The robustness of a system

requires that small changes in user input and/or weighting should not cause massive changes in output. The researchers of MYCIN used this technique to validate their certainty factors. Sensitivity analysis could be implemented by doing a monte-carlo study. This study would not require actual users...instead sample cases could be randomly generated.

5.7.3 Testing Methodology for Candidate Evaluation Expert System

This section presents a task breakdown describing the method currently being used for validating the Candidate Evaluation expert system modules. This general methodology attacks three main aspects of the expert systems: its semantic content and verbiage, its validity in dealing with actual test cases, and its ease of use.

5.7.3.1 Review of semantic content of expert systems.

Here, the expert reviews the semantic content of the knowledge base, and suggests refinements. Specific attention is focussed on the following:

a) Assessment of verbiage (qualitative information). Each evaluative question, including the question, its answers, and its explanation is reviewed. Then, each dimension's explanation, and finally, each recommendation fragment is reviewed.

b) Assessment of weights, scores, rating thresholds

(quantitative information).

5.7.3.2 Experiment based on Sample Cases

The structure of the experiment involves the following steps:

a) Create several sample sessions ("hypothetical candidates"). Each one is a series of answers to the questions. There should be some "excellent", some "good", some "poor", etc. Some should be strong in some areas and weak in others.

b) Present the questionnaires and answers for a sample candidate to each expert. Have the experts score and rate the candidate on each dimension, including an overall score and rating. Scores are between 0 and 100. Ratings must be using the rating terminology used in the CEVAL application.

c) Using a linear regression statistical method (via SPSS or SAS), get the weights of each of the questions. These empirically derived weights will be compared against the weights in the expert systems module itself. This is done to verify the weights assigned in the expert system. In the absence of a sufficient number of cases, a delphi study would replace the linear regression analysis.

d) Compare scores and ratings given by experts against scores and ratings given by expert system, using a similarity measure (e.g. percent difference in score, ordinal distance in rating). This will be done to verify the scores and rating

thresholds assigned in the expert system.

e) Using audiotaped interview (or perhaps a written form), ask expert to give a verbal interpretation of results, and recommendation about the candidate being evaluated. In this interview (or written form), ask the expert which factors influenced each of the interpretations given. This will be compared with, and be used to validate, the verbal recommendations given by the expert system.

f) Ask the expert to rank-order the sample cases (candidates) in order of preference. This rank order can be compared with the ranking established by CEVAL. If multiple experts are used, the group's final rank order can be established via Borda Count voting, as discussed in chapter 4.

5.8 Conclusions: Strengths and Weaknesses of CEVED/CEVAL

As mentioned earlier, one strength of the CEVED/CEVAL shell is its ease of use for non-programming domain experts. At Michigan State University's Center for International Business Education and Research (CIBER), we have found that development of expert systems is greatly facilitated by the use of this and other TSA-oriented shells. Our use of TSAs speed up knowledge acquisition and expert systems development because the domain expert is directly involved in encoding his or her knowledge on the computer. Figure 5.1 illustrates how the domain expert interacts with CEVED to encode his or her knowledge.

Another strength is that explanation in CEVAL is expressed in terms of the evaluation task, making it easier for the end-user to comprehend. When a user asks why a particular recommendation is given, the system responds by indicating the score/rating of the dimension(s) that resulted in the recommendation. The user can then get further information about the subdimensions and/or questions that led to that score/rating. Also, the user is shown how important the various dimensions and questions are, and how these importance levels were obtained. Thus, the structure of the candidate evaluation architecture causes explanations that are expressed in terms of evaluative reasoning, rather than in terms of rule-tracing as you find in general-purpose shells.

The above two strengths are due to the task-specific nature of the shell. However, task-specificity also leads to lack of flexibility. Obviously, not all tasks are evaluative in nature. CEVED/CEVAL cannot handle non-evaluative tasks. Other shells would be needed.

The reader may notice that the imposition of multiple-choice answers causes the system to be noncontinuous. In fact, the boundary problem cited by Berliner and Ackley (see above) is not solved using this tool. However, the fragility problem *is* solved because of the use of a weighted scoring scheme. In addition, despite the lack of "true continuity", there are two characteristics of the CE architecture that give it a "pseudo-continuous" flavor. First, the mapping of answers to scores

allows for ratio representation, not merely nominal or ordinal. Second, contextual weight adjustment significantly increases the number of possible points in the evaluation space.

Another weakness in the current CEVED tool is its passivity as a knowledge acquisition devise. As stated earlier, CEVED in its current form is merely a shell on which to create evaluative knowledge bases. It does not flag inconsistencies entered by the developer, nor does it implement any of the multiple-expert voting techniques described in chapter 4. For CEVED to be considered a true knowledge acquisition tool, it should be extended to incorporate some of these capabilities.

The next chapter deals with application of the Candidate Evaluation architecture to problems in the international marketing domain.

CHAPTER 6

ISSUES IN INTERNATIONAL MARKETING

In recent years, there has been much research and some development in the area of expert systems for marketing applications (Rangaswami et al. 1987). The specific applications have included areas such as contract negotiation (Rangaswami et al. 1989) and export-readiness assessment (Cavusgil and Nason 1990). Most marketing expert systems have not dealt specifically with *international* and *global* aspects of marketing, although this trend is starting to change.

There are many decisions a manager must make when dealing with internationalization of his or her marketing operations. Major strategic issues involve "where to market" and "how to market". The "where to market" issue involves selecting the best countries and/or regions to concentrate on. The "how to market" issue involves selecting the best mode of entry; that is, whether to export, license, franchise, set up a foreign manufacturing facility, or a myriad of other options. Often the "where" and "how" issues are dependent on each other. For example, if exporting, one needs to be cautious of countries with high tariff levels. Conversely, if one wants to market in a high-tariff country, the entry mode chosen should usually be something other than export.

In addition to these broad strategic issues, there are many day-to-day operational decisions that international marketers must face, such as: selection of distribution

channels; evaluation and selection of distributors, freight forwarders, or joint venture partners; evaluation of expatriate personnel and foreign subsidiaries; adapting products to meet foreign demand; and construction of legal agreements. All of these decisions can be aided through the use of expert systems.

This chapter takes a closer look at some of the issues described above.

6.1 Selection of Foreign Markets

Managers who wish to develop a comprehensive plan for foreign market entry face the question "Where do we want to go?" The real issue here is for a company to assess the market potential of candidate countries in terms of the company's product or service, the company's desired mode of entry, and the political, economic, commercial and cultural factors in the country itself.

6.1.1 Stages of Country Selection

Cavusgil (1985) suggested a three-stage, sequential process of country selection, outlined below:

1) Preliminary screening
2) Industry market potential analysis
3) Company sales potential analysis

Stage one involves assessment of the physical, political, economic, and cultural environment. *Physical/demographic*

factors include population size and distribution, climate, availability of natural resources, and physical distribution and communications networks. *Political* factors include system of government, ideology, political stability, government involvement in trade affairs, and government-imposed restrictions such as tariffs and non-tariff barriers. *Economic* factors include GNP, overall level of development, currency issues, inflation, unemployment, per-capita income, and balance of payments. *Cultural* issues include literacy and education levels, existence of a middle class, language, religion, and ethnicity. All of these factors can be considered *macro-indicators*, in that they are not industry-specific but rather involve the overall market climate in the country. Countries that perform poorly in these criteria should be disqualified, particularly if the company is rather new in the globalization process. Companies with extensive international experience may still want to consider such countries if they are willing to take a risk.

Stage 2 involves an industry-specific analysis of market access, product potential and local distribution and production issues. *Market access* issues include further analysis of tariff and non-tariff barriers such as standards, quotas, documentation and import regulations, as well as legal issues involving intellectual property protection, investment, employment, and repatriation. *Product potential* issues include customer demand, attitudes toward foreign-origin products,

competition, exposure to the product. *Distribution and production* issues include availability of intermediaries (distributors, agents, etc.) transportation facilities, and manpower availability. Thus, stage 2 involves issues that are specific to the company's particular industry. Such analysis is difficult and time-consuming, which is why many countries should have been weeded out in stage 1.

Stage 3 involves a detailed company profitability analysis. Issues here include sales volume forecasting, landed cost analysis, internal distribution costs, and pricing. This is a very intensive process, and should be applied only to a very few potential countries. As Cavusgil (1985) notes, "...Much of the information needed for the first and second stage of opportunity analysis can be gathered through desk research....In contrast, estimating company sales and profitability often requires field research. (p.31)". Thus, it is important to have weeded out the less promising markets earlier on in the screening process. Cavusgil's statement supports the idea of using expert systems technology to aid in steps 1 and 2 of market selection. Desk research usually involves gathering information from government, industry, and academic publications, sifting through and sorting the data, and using the data to evaluate potential target markets according to the macro- and micro-indicators mentioned above. Chapter 7 of this thesis describes a computer program using database and AI technologies for aiding in performing these

data-collection and evaluation tasks.

6.1.2 Regression-based Model for Country Evaluation

Root (1982) discussed similar stages and criteria in the selection of foreign markets as those used by Cavusgil. He also espoused the notion of using a weighted-averaging scoring model as accept/reject decision rules in the screening process. As an example, for assessing industry market potential for television sets, he suggested the following. First, identify population-based predictor variables such as literacy level, urban population density, per-capita income, standard-of-living index, number of households, etc. Then, use regression analysis of historical sales of television sets in order to obtain the coefficients (weights) for each of these variables. Finally, evaluate the potential market with the weights obtained through the regression analysis. Note that this is a prime example of the *proper linear models* espoused by Dawes and described in chapter 4 of this thesis. This use of a weighted averaging scoring process based on a regression formula lends credence to the potential usefulness of the Candidate Evaluation architecture or a similar MAUT-based approach for target-market evaluation and selection.

6.1.3 Providing Market Research Information and Evaluation

Much market research information can be found in databases, government documents, and industry publications.

For example, the U.S. Department of Commerce publishes annual reports, called Country Market Plans, on 60 countries. In these reports, they assess the economic, political, and commercial environment, including many of the issues described above. Industry-specific information can be obtained via D.O.C.'s Industry Sector Analysis reports as well as industry publications from Dun and Bradstreet, Price Waterhouse, and other firms. Much of this information has been electronically captured on databases such as D.O.C.'s National Trade Data Bank (NTDB) and Intellitrade Corp.'s Intellibanc.

However, to date there has been little effort to systematically catalog the information according to the features described by Cavusgil, nor to develop databases that give explicit *evaluations* of a country's performance in the various features via MAUT methods as Root suggests. It is precisely this sort of information-structure and judgement facility that is needed if country selection is to be automated in a decision support tool. In Chapter 7 I will discuss a database that combines a semantic network indexing scheme with a MAUT evaluation methodology to provide judgements and evaluations about countries based on information found in the publications mentioned above.

6.2 Selection of Entry Modes

In this section, I identify some major issues faced by managers and researchers as they attempt to answer the

question "How should we enter the target market?". In addition, I will propose a computational framework for answering this question, and compare it to other computational approaches that have been used for this and other decision-related problems. Thus, my discussion of the market-entry issue will be from the perspective of a knowledge engineer who is interested in representing the "how to enter" question in a computerized, expert-system model.

Two major issues in research of entry-mode selection involve classification of the factors involved in selecting entry modes and classification of the entry modes themselves. These issues are particularly pertinent in the context of expert systems development because, as we will see, the factors form the input to the expert system, and the chosen modes form the output. Thus, the way that we represent the factors and the modes will have a significant impact on the way we design the expert system, and on the way that the system performs its selection task.

In the topics that follow, I will first discuss the factors that go into selecting entry modes. Then I will discuss different methods for classifying the entry modes themselves. Following this, I will discuss the implications these classification schemes have for the type of knowledge representation most appropriate for expert systems development in this domain. Then I will compare and contrast some existing models from the marketing literature in terms of their

knowledge representation frameworks. Finally, I will propose a computational method for answering the "how to enter" question and suggest other ways that computer science can contribute toward solving this problem.

6.2.1 Factors Involved in Selecting Entry Modes

Much research has been devoted to identifying and categorizing the factors that go into selecting modes of entry into target markets. Goodnow (1985) summarized several theories pertaining to this issue. Such theories include: the theory of comparative costs and relative factor proportions (Ohlin 1983); theories based on value-added-chain considerations (Kogut 1984, Porter 1985); industrial structure theories (Knickerbocker 1973); desire-for-control theories (Rugman 1979); and theories pertaining to political, economic, and cultural factors in the target markets (Goodnow and Hansz 1972).

These theories typically concentrate on a single factor and attempt to explain how that factor influences the decision-making processes of managers as they explore their market entry options. However, a useful expert system for entry-mode selection must take many factors into account before making a recommendation. It must be able to identify the relative importance of each of the factors, based on the needs of the company and the circumstances of the market. It must also be able to represent how the factors interact with

one another and how they may compensate for each other as circumstances change. In more recent years, Goodnow (1985), Cavusgil (1981), and others have explored combinations of factors in terms of their influence on the choice of entry modes. These "eclectic" theories are in an early stage, and there needs to be more empirical research done to test them. However, they form the basis for the models that will be discussed in this paper.

Factors influencing entry have been classified by several authors. Root (1982) and Goodnow (1985) both divided these factors into two main categories, internal and external. Internal factors are those features of the company and its product that can influence the choice of entry mode. These include: characteristics of the product itself (bulk and weight, ease of use, price, service requirements, etc.); characteristics of the corporate strengths and competitiveness of the organization (corporate size, management experience, financial flexibility, etc.); and characteristics of the corporate policies and desires of the organization (level of commitment, desired payback period, willingness to take risks, degree of control desired).

External factors influencing choice of entry mode include: factors in the target market (political, social, economic, and cultural environment, market opportunity and demand for the product, government policies regarding foreign entry, physical and distribution infrastructure, etc.); and

factors in the home country (government policies toward export, market saturation of the product, demand at home, etc.).

The factors involved can be classified in a tree-structured hierarchy (see figure 6.1). Note that this hierarchy enables the expert system to represent factors at various levels of abstraction. This has important consequences for the explanatory power of the system. An expert system should be able to explain its reasoning at a general level, and if required, at more detailed levels. Hierarchical representations of the input factors provide a framework for doing this.

Two major issues involving the factors are: first, what are the relative importance levels of each of the factors as they pertain to the choice of entry mode; and second, how do the factors interact with each other in influencing the choice of entry mode? It appears that more research has been devoted to the first question than to the second, although my study of

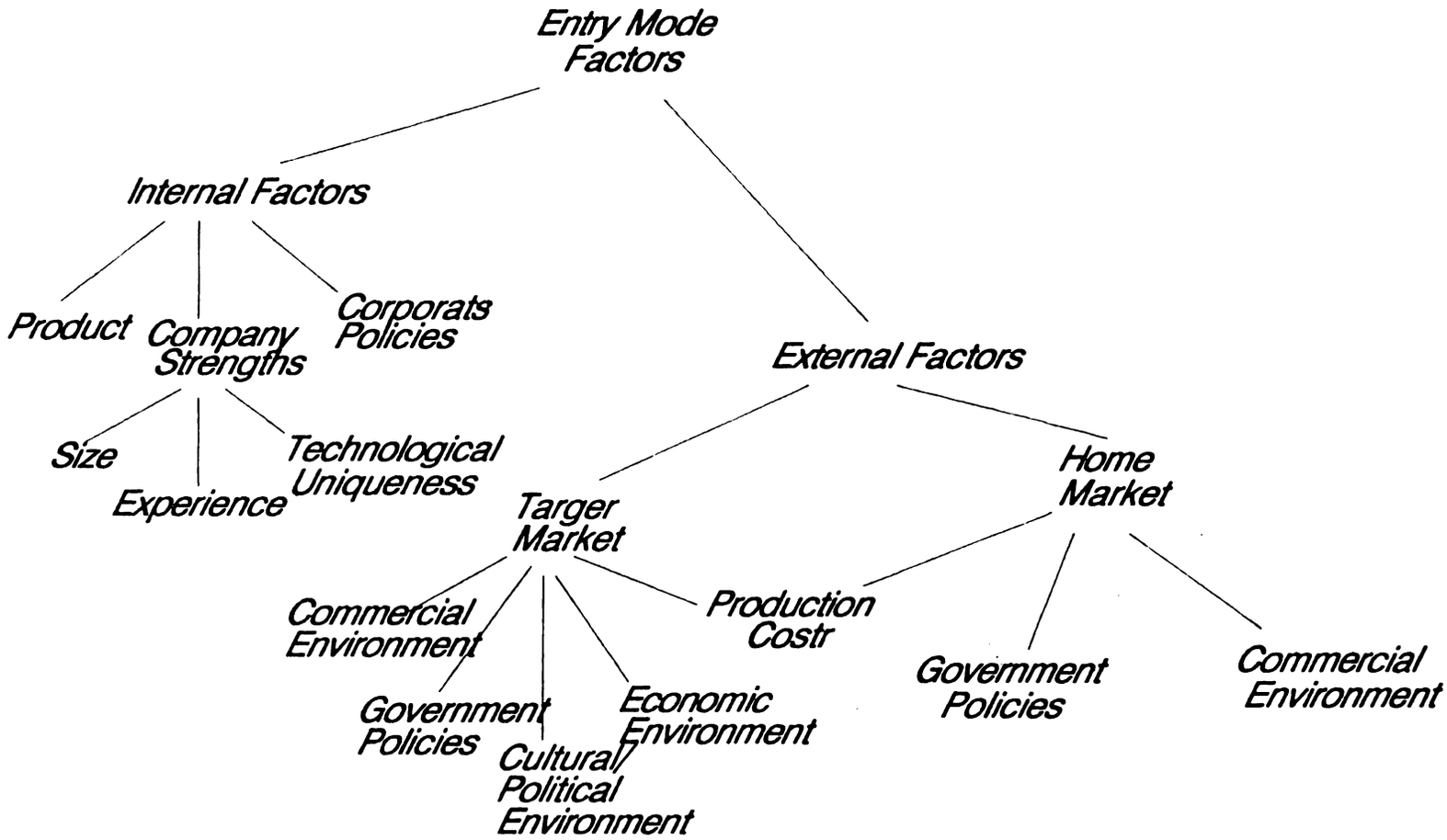


Figure 6.1

Factors influencing the choice of Entry Mode.

the literature is certainly incomplete in this regard. Note that which issue we focus on will have a profound impact on how we represent the knowledge that goes into the expert system; more about this later.

6.2.2 Classification of Entry Modes

Throughout the literature, there seems to be two main ways of characterizing and classifying the various modes of entry that a company can use in international marketing. The first classification scheme is a descriptive taxonomic approach, which divides entry modes into three main categories: export modes, contractual modes, and foreign direct investment modes (Root 1982, p.7). The second approach uses a continuum of modes, usually based on the degree of commitment, control, and/or risk involved in utilizing each of these modes (Goodnow 1985). Closely related to the second approach is one in which the entry modes used are associated with the "stage of internationalization" that a company is in (Root 1982). As a company becomes more experienced in the internationalization process, it will be more willing to devote resources to that effort and to take the risks needed for successful market entry.

The descriptive taxonomic approach is based on characteristics of the modes themselves (see figure 6.2). For example, the top-level tier of the hierarchy consists of exports, contractual modes, and investment modes. Export modes

all involve a home-based production process coupled with some form of marketing effort in the target country. The marketing effort may be done by the company itself (branch or subsidiary), by a distributor or agent in the foreign market, or by a trading house or agent in the host country.

Contractual modes all involve non-equity associations for transfer of technology, knowledge, or other intangible benefits of the company. This can involve licensing, franchising, manufacturing contracts, management contracts, etc. Foreign direct investment involves some form of direct ownership of a production process that would take place in the target country.

The continuum of commitment approach to entry-mode classification is based on the level of effort and resource commitment required, and control retained, when implementing a given entry mode. For example, modes such as indirect export and licensing require little effort, but also exact a cost of losing control over the process. Modes such as subsidiary-based export or wholly-owned manufacturingfacilities involve much effort and commitment, but also allow the company to retain control over the process.

The method used for classifying entry modes is an important issue in developing decision support and expert systems to help managers select from among the possible entry modes available to them. The type of classification scheme has

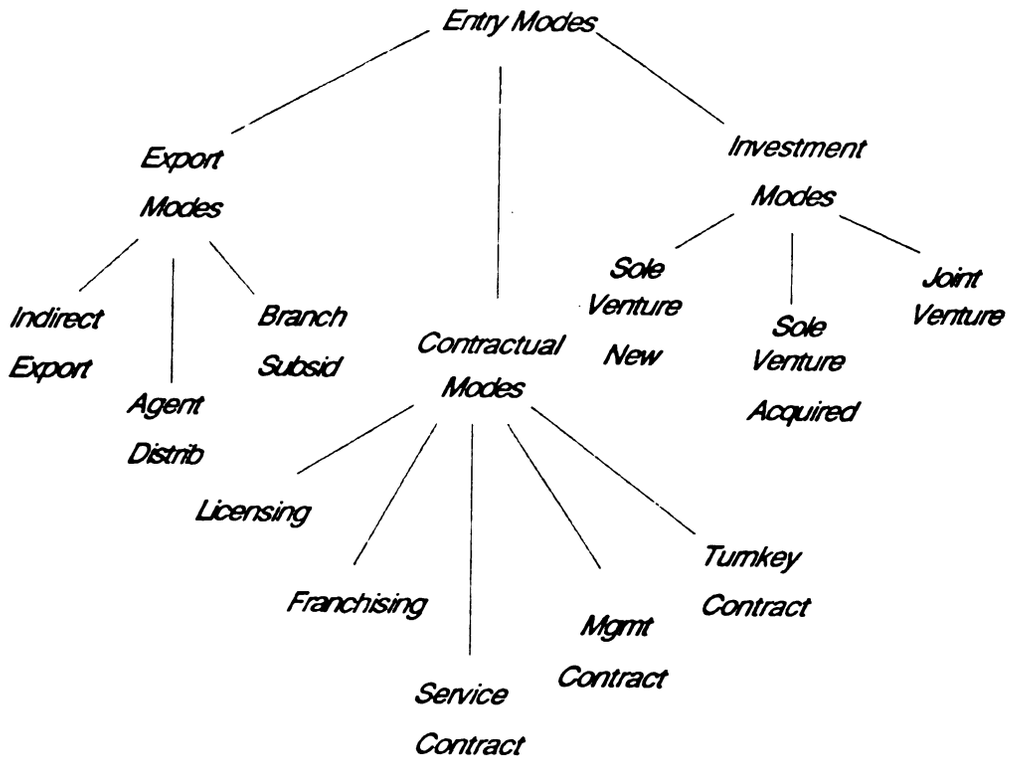


Figure 6.2

A descriptive taxonomy of Entry Modes
(from Root)

a direct impact on how the knowledge is acquired, represented, and used.

The taxonomic approach is useful as a descriptive framework. However, attempts to implement the descriptive taxonomy as a decision tree for an expert system will present problems. The taxonomic hierarchy may not be the best-suited representation for making entry-mode selection decisions. There are several reasons for this.

The first problem is one of fuzzy classification. For example, is a joint venture a contractual mode or an investment mode? Root classifies it as an investment mode. However, Casson (1987) describes it as a contractual mode. Obviously, it includes characteristics of both modes, and therefore does not fit cleanly in a particular spot of the taxonomy. Another example of fuzzy classification is a foreign subsidiary that assembles intermediate products which were produced in and imported from the home country. Would this be considered a foreign direct investment mode or simply an extension of the export process?

A second problem with using the descriptive taxonomy for a decision tree is that many important factors influencing a company's entry decisions span across this hierarchy. For example, there are high-commitment and low-commitment export modes. Likewise, there are high-commitment and low-commitment contractual modes. Therefore, a factor like commitment level is not one that could quickly rule out a branch of the

hierarchy. A major purpose of a hierarchical classification (decision tree) approach to selection is to be able to quickly rule out entire branches of the tree based on early, important questions. This speeds up the decision process. However, it has been our experience that quick "rule-out" factors that help prune a decision tree based on the descriptive taxonomy are hard to come by.

Closely related to the second problem is the fact that entry-mode choice factors often compensate for one another. For example, high tariff rates may appear to rule out export modes at an early stage in a decision-tree selection approach. However, if the product is in great demand and other costs are low, export may still be feasible despite the high tariff levels. After all, Japan has no problem selling Toyotas in the U.S.

Perhaps for these reasons, most models of entry-mode selection in the academic literature, and the few software products that have been developed to aid in choosing entry modes, tend to focus on the "continuum of commitment" classification of entry modes rather than on the descriptive taxonomy classification. We will see that this "continuum of commitment" approach is consistent with the weighted linear models discussed in chapter 4, and particularly with the Candidate Evaluation architecture described in Chapter 5.

6.2.3 Three Models of Entry Mode Selection

Below, I will discuss three models for selecting entry modes that have appeared in the academic international marketing literature. These three models include: Goodnow's Gauge for International Market Strategies (GIMS), Cavusgil's Company Readiness to Export (CORE), and Casson's model for selecting the best contractual arrangement.

6.2.3.1 Goodnow's GIMS

Goodnow's GIMS approach, implemented as a computer program written in BASIC, is based on a "continuum of commitment" classification scheme. It presents a questionnaire that assesses internal corporate factors and external market factors. The internal factors include corporate policy, competitiveness, financial strengths, and product characteristics. The external factors include domestic and foreign government policies, comparative host country costs, market opportunity, and the political, cultural and economic environment of the host country. Based on an overall score that results from the questionnaire, GIMS suggests modes that range from no entry or cash-in-advance-only at one extreme to wholly-owned subsidiary at the other. These modes are arranged in order of the degree of commitment and resources required to maintain them. GIMS will suggest that high-commitment modes are inappropriate for weak companies facing unpromising market conditions. For strong companies entering promising markets,

GIMS suggests that high commitment modes are feasible, but also that other, less costly modes, are acceptable as well. In essence, GIMS suggests to a firm that it has a wider latitude of entry strategies as it gains strength in the home and target markets. This is consistent with the "stages of internationalization" models described by Root and Cavusgil.

In addition to the overall recommendation based on total score, GIMS identifies specific variables which imply the inappropriateness of certain specific modes of entry. For example, if the user indicates that s/he wants a high degree of control over the distribution process, GIMS will flag this variable to imply that licensing and exporting may be unsuitable entry modes.

Thus, we see two main mechanisms operating in the GIMS program. First is a linear-weighted sum (ala MAUT) which results in an overall score indicating the strength of the organization, product, and environment in terms of appropriateness for market entry. Second is a flagging of specific individual variables in terms of their impact on the appropriateness of he alternative entry modes, dealing with non-compensatory issues. Note that there is a direct representation of the relative importance levels of the variables, expressed as user-provided weights. Note also that, despite the fact that individual variables are flagged to indicate unsuitability of specific modes of entry (which may imply a certain rule-like quality to the program), there is no

explicit representation of the interaction between variables in terms of their impact on their mode-selection effects. Thus, GIMS does not account for configural effects in entry mode selection.

6.2.3.2 Casson's Model of Contractual Entry Mode Selection

Casson (1987) suggested a theoretical model for choosing between alternative contractual arrangements via a weighted scoring technique that calculates scores for each possible contractual mode based on yes/no values for relevant factors. The possible contractual modes include: greenfield (i.e. starting from scratch), merger, joint venture, industrial cooperation, subcontracting, sales franchising, and licensing. There are eighteen input factors, which break down into four major categories: nature of the advantage, nature of the firm, nature of the industry, and nature of the home vs. target countries.

Like GIMS, Casson's model involves a weighted sum scoring method. However, Casson's method differs significantly from GIMS in the following respect. GIMS score is merely a measure of strength of the organization, product, and environment. The GIMS score indicates the degree to which the company can dive into the international marketing waters, so to speak, and the recommendation output from GIMS suggests a wider scope of potential entry modes as

the score increases. As I mentioned earlier, GIMS is

essentially using a continuum of commitment classification of entry modes, and the GIMS score indicates where a company lies on that continuum. The implication is that the company can use any mode that falls at or below the company's position on that continuum.

In contrast, Casson's method explicitly discriminates between entry modes by scoring each mode individually. Thus, whereas GIMS gives a single score, Casson gives eight individual scores, one for each contractual mode. The advantage of this approach is that the various modes can be directly compared to one another in order to pick the best one for a given situation. In this sense, Casson's model is similar to Berliner and Ackley's method of scoring different board positions based on feature values of the current game. Casson's entry modes are equivalent to Berliner and Ackley's board choices.

6.2.3.3 Cavusgil's CORE

Cavusgil's CORE (Company Readiness to Export) program (Cavusgil and Nason, 1990) is geared toward evaluating a firm and its product in terms of their suitability for internationalization. It is not an entry-mode selection method per se, although the output recommendations do give some indication as to which modes may be feasible based on the company's final evaluation.

Like GIMS, CORE uses a liner-weighted sum approach to

evaluate the company's strength in terms of international marketing factors. Also like GIMS, CORE uses specific variables to flag specific outputs for recommendation. However, CORE differs from GIMS in several important respects.

First, CORE's weighted scoring is broken down into the individual factor categories: business background, motivations, management commitment, product strengths, and market-specific strengths. Thus, unlike GIMS, whose final score is an undifferentiated accumulation of the overall strength of the company and its product, CORE offers the ability to identify those factors for which the company is strong and those for which the company is weak. This differentiation allows the output of CORE to be more specific to the particular situation that the company faces, and thus is a more tailored, intelligent output. Note that, in this respect, CORE attacks the problem of context in a manner very similar Edwards and Newman's value tree and to Berliner and Ackley's hierarchy of linear scorers described in chapter 4.

Second, unlike both GIMS and Casson's method, CORE actually accounts for interactions between the variables, at least at a high level of abstraction (i.e. CORE accounts for configural effect). The nine possible final recommendations are based on combinations of product and organizational ratings. Thus, if the company is strong organizationally but weak in terms of its product line, the final recommendation can take this into account. There is no comparable mechanism

in the GIMS program. This is another example of CORE's ability to tailor the final recommendation to the situation of the user. Thus, CORE is imposing a sort of rule-based approach to account for interactions between high-level variables, and is thus tackling the same issue of interdependence that Samuel's checker-playing program addressed.

Third, CORE does not flag its variables with the intention of identifying unsuitable entry modes, as does GIMS. Instead, it flags the variables to identify particular strengths and weaknesses of the company. This is a reflection of the difference in focus between CORE and GIMS...CORE is a readiness evaluator, whereas GIMS is a mode-selector.

CORE's method is one that combines a scoring system with a matching (rule-like) mechanism. Thus, it is the only method of the three that explicitly represents both the relative importance of the variables and the interactions between the variables. Thus, CORE attempts to incorporate the advantages of scoring and rule-based approaches to expert systems development, although at a fairly rudimentary level.

CORE's overall approach was generalized into the Candidate Evaluation architecture as implemented in CEVED/CEVAL. In other words, CORE is the seed from which CEVAL was sprung, in the same sense that MYCIN begat EMYCIN and MDX begat CSRL. CEVAL's evaluative questions are essentially identical in structure to those of CORE. The ides of weighted dimensions is a generalization of CORE's hierarchy of

	Goodnow's GIMS	Casson's Model	Cavusgil's CORE
Scoring Scope	Overall score only	Overall score only	Both Overall and Subsection scores
Object being Scored	The Company	The potential entry mode	The company and product
Variable Flagging	Yes, to identify unpromising entry modes	NONE	Yes, to identify specific strengths and weaknesses
Interactions of Variables	NONE	NONE	Yes, at higher levels in the feature hierarchy.
Hierarchical Representation	NONE	NONE	Yes, to facilitate comprehensiveness of explanations.

Figure 6.3

A comparison of three models for entry mode selection.

evaluation features. Like CEVAL, CORE has paragraphs that display conditioned on combinations of dimension-ratings.

6.2.3.4 A Final Look at the Three Models

Figure 6.3 illustrates the differences and similarities between the three models described in the previous sections. As we can see, CORE and GIMS differ from Casson's method in two main respects. First, the object being scored in both CORE and GIMS is the company itself whereas in Casson's method the scored objects are the alternative entry modes. Second, in CORE and GIMS, there is a flagging of individual variables which enables a customized output to occur. CORE differs from GIMS and Casson's approach in that it explicitly represents the interactions between variables at a high level, and that the scores accumulated by CORE are broken down by subcategories of the company/product/market factors.

I propose that a useful model of entry-mode selection should incorporate the strengths of all three approaches. Namely it should be like Casson's approach in terms of scoring and rating the various entry modes. It should be like GIMS in terms of flagging specific reasons that particular entry modes may be desirable or unattractive. It should be like CORE in terms of allowing differentiated scoring and explicit representation of variable interaction. A model that uses all these strengths will provide a customized expert system for entry mode selection.

6.2.4 Use of Candidate Evaluation for Entry Mode Selection

The framework required for a successful Entry Mode Selection expert system can be partially met via the Candidate Evaluation architecture as described in chapter 5. Specifically, the representation of a hierarchy of factors shown in figure 6.1 is representable as a dimension hierarchy in CEVAL, and questions are assigned to the lowest level features. An MAUT representation can be generated by associating weights to the dimensions and questions. An overall assessment of a company's strengths and weaknesses can be generated through the use of recommendation fragments and dimension-ratings, providing a significant improvement to CORE's method.

However, the actual selection of entry modes is not easily represented in the Candidate Evaluation architecture as described in chapter 5. What type of CEVAL object can be used to represent an entry mode? Despite the capability to have a taxonomic representation of entry modes, it is clear that such a tree is not the same as a dimension hierarchy. Rather, the entry mode hierarchy is more appropriately matched to the type of classification hierarchies represented in CSRL. However, as mentioned earlier, the descriptive taxonomy is not useful for selection purposes, largely because of the compensatory nature of the selection problem. In addition, the number of possible entry modes is sufficiently small (around a dozen) that a hierarchical representation may be unnecessary. Thus, there

should be a way of classification which does not rely on the hierarchical representation and which can support the compensatory evaluation needs of the entry-mode selection problem.

If not dimensions, then might entry modes be considered "candidates"? This would address the compensatory nature of the problem-solving task, and does not rely on a taxonomy. However, upon closer inspection, it is clear that the real *candidate* is the company, product, and market being evaluated, not the entry modes themselves. The choice of entry mode is made after (and as a result of) evaluating the company, product, and market.

Thus, a new type of object is needed for representing entry modes, and in general, for representing mutually-exclusive choices to make based on an evaluation. This object should essentially be an instance in a classification representation (like a leaf level node of a CSRL tree). However, the representation should allow a compensatory mechanism for selection, like the general MAUT model and particularly like Casson's model described earlier. Recall that Steels (1990) had identified *six* classification methods, one of which is a "weighted evidence accumulation" approach. It is this approach to classification that seems most applicable to entry-mode selection, due to its ability to deal with compensatory decision-making.

As a result of the need for a classification facility in

developing an entry-mode selection expert system, CEVAL's architecture has been expanded to include two new object types, called 1) *Plan Types* and 2) *Plans*. A plan type is a class of plans. "Entry Mode" is an example of plan type. A plan is an instance of a particular plan type. Export, joint venture, and licensing are all examples of plans. Each plan in a plan type is linked to the dimension hierarchy via *degree of support* links, by which a particular dimension-rating pair makes a weighted contribution to the plan's overall score. The architecture allows weights to be specific for the plan. For example, tariff levels are an important consideration when dealing with export modes, but are less important when dealing with licensing or direct investment. CEVAL's architecture allows the developer to adjust the importance of tariff level for each plan.

From the above discussion, we see support for Langlotz's assertion, discussed in chapter 3 of this thesis, that decision making involves two major components, *diagnosis* and *planning*. Prior to introducing plan types and plans into CEVAL's structure, Candidate Evaluation was mainly doing diagnosis in the sense that it was assessing a candidate's strengths and weaknesses. Although the text of the recommendation fragments could be worded to suggest plans of action, even the recommendations are abductive explanations of the findings (i.e. part of the diagnostic process). Thus, CEVAL as an architecture was incomplete with respect to

decision-making based on the evaluation. With the introduction of plan types and plans, CEVAL can suggest one of a number of action options based on these evaluation results. Thus, CEVAL can be used for solving the entry-mode selection problem.

It is interesting to consider the chronological sequence of events leading to development of the entry-mode module. First was an attempt to use CEVAL, with its dimensions, evaluative questions, context questions, and recommendation fragments. This led to discovery of significant obstacles, which in turn led to adding new CEVAL features to overcome these obstacles. This incremental addition of new features to CEVAL based on needs of the domain problem may disqualify CEVAL as a generic task implementation. CEVAL's development has been significantly influenced by the specific needs of the international marketing domain. A generic task should, ideally, be domain independent. This is one reason that I do not claim CEVAL to be an implementation of a generic task. However, CEVAL's plan types and plans *are* consistent with the primary TSA criterion requiring knowledge-use level primitives. In addition, the MAUT nature of CEVAL's reasoning method for plan selection (using degree-of-support links) is consistent with the compensatory reasoning philosophy underlying Candidate Evaluation as a whole. Thus, in my view, adding plans and plan types to CEVAL does not violate any principles of task-specific knowledge representations.

Using CEVAL to model the entry-mode selection problem is

an improvement over the three models mentioned previously. In effect, CEVAL contains the combined strengths of Goodnow's GIMS, Cavusgil's CORE, and Casson's contractual-mode selection model (as shown in Figure 6.3).

6.3 Some Operational Issues in International Marketing

Several other day-to-day issues must be addressed by the business manager who wants to market his or her products or services abroad. Some of these issues are discussed below.

A key operational issue is *selection of a foreign distributor*. This is of primary importance to exporters of products, since the distributor will frequently be their primary representative in the target market. Major criteria for distributor evaluation and selection include financial and company strengths, commitment to the relationship, marketing skills, degree of familiarity with the product, and other facilitating factors (Yeoh 1991)

A similar issue involves selection of *freight forwarders*. These are companies whose service involves shipping the product from the exporter's home base to the target market. Forwarders may be responsible for shipment, custom clearance, warehousing, and/or insurance. Thus, it is important to find well-qualified forwarders. Some important criteria include knowledge of customs procedures in target markets, specialization in the exporter's product line, physical facilities such as warehouses, financial strength, and

reputation (Ozsomer 1991).

For companies interested in going beyond mere export, to long term joint-venture arrangements, there is the important issue of selecting and evaluating potential *joint-venture partners*. Such an evaluation is based on two major criteria, partner-related and task-related characteristics. Partner-related characteristics include motivation, reliability, commitment, respect for property-rights protection, and other company-related factors. Task-related criteria involve the potential partner's financial strengths, research-and-development resources, marketing abilities, production plants and fixed assets, and organizational resources (Subieta 1991).

Any company that intends to commit significant resources into internationalization will sooner or later need a representative in the target market. Thus, they will need to evaluate and select *expatriate personnel* based on job-related skills, corporate fit, and country fit. Job-related skills include managerial, marketing, and communications. Corporate fit is particularly important with expatriate personnel because of the distance and resulting lack of supervision. Country fit involves the employee's level of comfort with the target market's culture as well as the specific network of relationships that the employee has cultivated in the target country (Whitney 1991).

The above four international marketing tasks all involve evaluation and selection of candidates. The type of candidate

differs, but the Candidate Evaluation method can be applied to all these tasks.

6.4 Conclusion

This chapter discussed several prominent issues dealing with international marketing. Primary among these issues include selection of target markets and selection of entry modes (the "where" and "how" of internationalization). Additional operational issues involve evaluation and selection of distributors, freight forwarders, joint venture partners, and expatriate personnel. In this chapter, I argued that the Candidate Evaluation architecture, or some other representations involving MAUT and linear-weighted algebraic models, can be applied to solve these types of problems. The reason for this is the evaluative nature of the tasks and the compensatory nature of the decisions being made. My argument is supported by theories espoused by Cavusgil and Root, and by models developed by Goodnow, Cavusgil, and Casson.

Chapter 7 describes a database using MAUT which aids in evaluating target markets based on market research information that can be found in government, industry, and academic publications.

CHAPTER 7

THE COUNTRY CONSULTANT:

AN INFERENTIAL-EVALUATIVE DATABASE

In previous chapters, I described the MAUT approach to evaluation problem-solving, and presented a problem-solving architecture called Candidate Evaluation which implements a combination of MAUT principles with AI explanatory techniques into an expert system shell called CEVED/CEVAL. I also showed some applications of this architecture and shell in international marketing.

In this chapter, I describe a database which makes further use of the MAUT evaluation approach. Specifically, I describe a method that combines MAUT with semantic networks to produce an *inferential evaluative database*. This database, called the Country Consultant, is a domain-specific repository of market-research information, designed to be used by international marketing professionals.

As mentioned in chapter 6, there is a significant motivation for providing databases of market information for various countries throughout the world. Some databases have been developed to contain such information, but these typically store raw statistical data or collections of articles and/or government documents. The Country Consultant is unlike most others in that it does not contain statistical or demographic data in a raw form, but instead contains *judgements* and *guidelines* pertaining to various aspects of the

countries in question, and geared toward specific industries and entry modes. In other words, the database contains processed information in the form of qualitative, judgmental knowledge, and catalogued according to specific markets, industries and entry modes. The ultimate purpose of this knowledge is to aid the end-user to make intelligent decisions pertaining to selection of the best countries to enter for marketing their products and/or services.

In addition to serving as a repository of processed, judgmental knowledge, the Country Consultant has the facility to respond intelligently to queries given by the user. If it cannot find a judgement or guideline that specifically meets the user's query, it can infer a likely value for that judgement or guideline by searching the database for conceptually similar judgements or guidelines. Thus, even if the database is incomplete (as it almost certainly will be), it can still give reasonable answers to queries for which it may not contain explicit data.

As mentioned in chapter 6,information on the demographic, political, economic, cultural and legal environments, as well as information on market entry conditions and on the market structure (in aggregate or disaggregate form) are pointed out in the literature as the principal information requirements in country selection. However, the proposed frameworks tend not to be comprehensive in defining information requirements for evaluating the market structure. Furthermore, the empirically

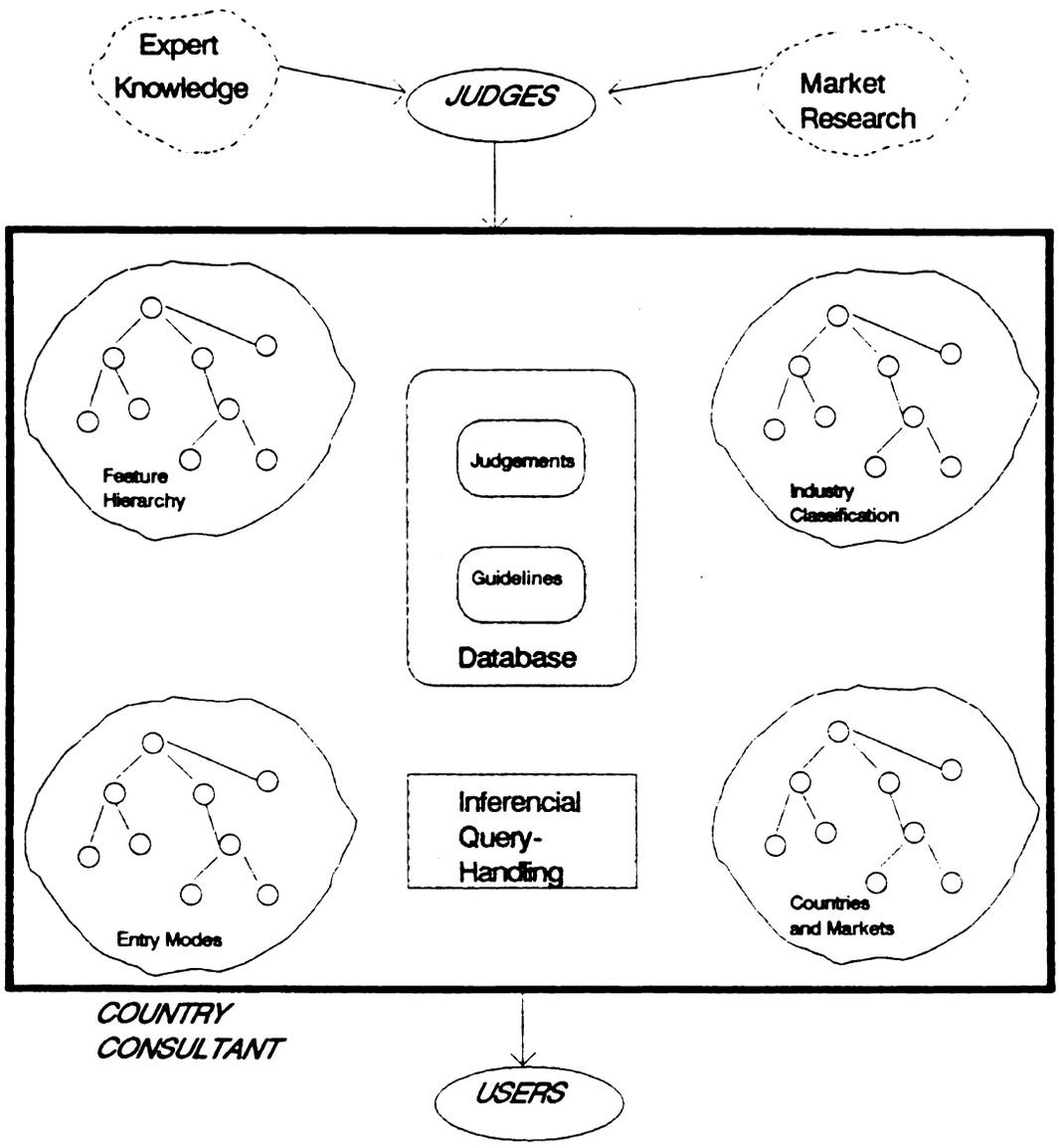


Figure 7.1

Structural components and information flow of Country Consultant. Judge enters information based on expert knowledge and market research findings. User queries the system for information, which triggers inferencing process. Semantic networks aid in knowledge organization and inferencing.

tested frameworks do not incorporate all of the information categories outlined above or their levels of aggregation differ.

All this implies that there is a need for information to be available at many levels of abstraction, and pertaining to many features of the market(s) being evaluated. While raw data is useful in obtaining this information, actual decisions are based on processed, qualitative, and judgmental information. Thus, there is a strong motivation for making such information available in software form, via an indexing mechanism that makes it easily accessible. Also, since such information is incomplete, there is motivation for use of AI techniques for allowing unavailable information to be inferred based on available information.

This chapter describes the Country Consultant, analyzing its structure in terms a semantic network model. It also describes the Country Consultant's evaluative inferencing mechanism from the perspective of semantic networks (spreading activation) and MAUT (attribute weight assessment). The overall framework for the Country Consultant is shown in Figure 7.1.

7.1 Semantic Network Knowledge and Data Representations

A semantic network is a form of knowledge representation based on a graph structure of nodes and links. The nodes usually represent objects in the world and the links

represent relationships between these objects. Semantic networks are useful knowledge representations for two main reasons. First, they provide explicit representations of the semantics, or meaning, of the terms in the knowledge substrate by showing relationships between these terms. Second, they allow for inferences to be made about knowledge that may not be explicitly entered, via a mechanism called *spreading activation*. Spreading activation is a process where the "attention" or "focus" of the computer travels from one node to another via the links which connect them. This fosters a kind of reasoning by association, where associations are the links in the network. Thus, the semantic network formalism is sometimes called an *associational* knowledge representation.

Semantic networks were first developed as models of human memory and natural language representations. Neither of these topics are within the scope of this paper. However, there has been work on semantic networks in database design, which is directly related to our work with the CC. Below, I will present some of the important work that has been done in semantic networks, then describe the network structure of the CC.

7.1.1 Quillian's Semantic Memory Model

Quillian's (1967) pioneering work in semantic network representations was primarily geared toward modelling long-term memory structures, particularly as it pertains to

sentence understanding. The nodes and links of his network were organized into *planes*, which were used to define concepts. A plane consists of two kinds of nodes. For each plane, there is a single *type node*, which identifies the concept which the plane is defining. Also, there are a number of *token nodes*, which identify other concepts that are related to, or subsumed by, the plane's concept. A token node points to another plane in the network, whose type node is identical to the token node pointing to it. Thus, token nodes serve as reference pointers to the conceptual structures (planes) that define the concepts that they "tokenize".

The nodes of a plane are related via associational links. Link types include superclass-subclass relations, modifiers, disjunctive and conjunctive clauses, and subject-object relations.

Quillian's work also introduced the notion of *spreading activation*, whereby the intersection of two concepts could be found in order to identify how the concepts are related to each other.

7.1.2 Schank's Conceptual Dependency Theory

Roger Schank's (1974) work with semantic networks was primarily concerned with applying the semantic network formalism to problems of natural language understanding. He hypothesized that all linguistic concepts can be grouped into six categories: real world objects, real world actions,

attributes of objects, attributes of actions, time, and locations. Thus, for Schank, all nodes of the semantic network fall into one of these categories.

Action nodes form the core of Schank's conceptual dependency representation. Schank identified twelve such nodes, and claimed that any verb could be mapped onto one of these primitive actions. The action nodes are: ATRANS (abstract transfer), PTRANS (physical location transfer), PROPEL, MOVE, GRASP, INGEST, EXPEL, MTRANS (mental information transfer), CONC (conceptualization), MBUILD, ATTEND, and SPEAK.

The links and link structures (called *cases*) of Schank's network include the following types: relations between actor and action; relations between actor and object; causal dependence links; and relations between donor, recipient, action and object. Links could also have modifiers indicating, among other things, past or future tense.

The main purpose of Schank's conceptual dependency networks was to provide inferencing power to systems attempting to understand and respond to natural language statements. Schank drew a sharp distinction between inferencing and logical deduction. He said that inferencing is more of a "reflex response", and may not be logically valid or true. For example, syllogisms (e.g. A implies B, B, therefore A) are not logically valid, but may be inferentially useful. Thus, Schank used the spreading activation capabilities of

semantic nets to perform inferences of various types. These inferencing mechanisms were forms of default reasoning, dealing with assumptions that can be made in the absence of contradictory information. Schank listed twelve inference types: including linguistic inference, action inference, trans-enable inference, result inference, object-affect inference, belief-pattern inference, instrumental inference, property inference, sequential inference, causality inference, backward inference, and intention inference.

7.1.3 Wood's "What's in a Link"

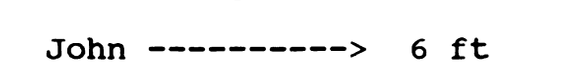
As can be seen by comparing Schank's and Quillian's models, the node-and-link formalism can be used in several ways and for several purposes. Thus, the idea of semantic network is not a rigid, standardized formalism as is, for example, boolean logic. Rather, it is a general model that may by modified to suit the needs of the user. This is useful, but at the same time may introduce ambiguities about the meaning of the term "semantic network".

Woods (1975) critiqued the various semantic network architectures that had been developed by the mid-1970s for their lack of firm semantic structure. His complaint was that the term "semantic network" was being used to describe several, often widely differing, node-and-link representations of so-called semantic knowledge. He was concerned that not enough emphasis was being placed on the meaning of the

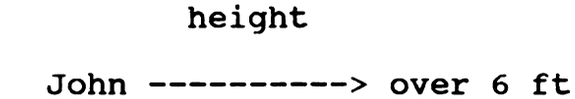
notation used in the semantic networks. In his words:

> "When one devises a semantic network notation, it is necessary not only to specify the types of nodes and links that can be used and the rules for their possible combinations (the syntax of the network notation), but also to specify the import of the various types of links and structures -- what is meant by them (the semantics of the network notation)." (p. 225).

For example, he cited several examples of links used in semantic networks that may imply that a link's purpose is essentially to represent attributes of an object. One example:



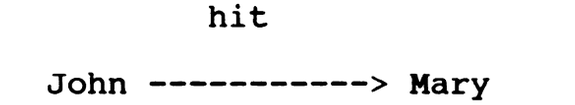
implies that the height link is an attribute link. However, consider the following:



In this case, the height link is a pointer to a predicate.

This also brings into question the semantics of a node. Is a node a value of an attribute? Or is it a predicate?

Links can also involve non-attributive relations between nodes. For example:



indicates a action-relationship between a subject (John) and an object (Mary).

Thus, Woods wanted more emphasis placed on the meaning of the node and link notations themselves, and not just the

concepts that the nodes and links are representing.

7.1.4 Brachman's KLONE

Brachman (1979) was concerned with the "level" of knowledge being represented by semantic networks. He discussed four levels of semantic network representations, each of which has its own types of primitive representational constructs. The lowest level, called the *implementational level*, treats semantic networks simply as data structures, and its primitives are atoms (nodes) and pointers (links). The next level is the *logical level*, whose primitives include propositions, predicates, and logical operators. Next comes the *conceptual level*, whose primitives are semantic relationships (cases), and primitive objects and actions. Finally comes the *linguistic level*, with primitives including words, expressions, and arbitrary concepts.

Brachman proposed a fifth level, to fit between the logical and conceptual levels, which he called the *epistemological level*. This level would involve primitives such as concept types, conceptual subpieces, inheritance and structuring relations. Epistemological formalisms would be neutral in regard to actual semantic relationships, unlike conceptual level representations. In Brachman's words: "It is the job of the epistemological formalism to provide case-defining facilities -- not particular cases. (p.206)"

For example, consider Schank's case types (links). These

were explicitly defined as actor-action relationships, actor-object relationships, causal dependency relationships, etc. Likewise, Quillian's cases included subject-object relations as well as logical connectives such as conjunction and disjunction. By contrast, Brachman suggested that the epistemological level provides the capability to create specific conceptual models using a generic semantic network "shell". The shell in question is called KLONE.

Brachman was presented a comprehensive survey of semantic network architectures as they existed around 1980. He showed that there was no standard that defines the semantic network model, rather, there was an ad hoc collection of several different models all sharing the node-and-link formalism of semantic networks, but all expressed using different levels of primitives in their representations. These early semantic networks were generally used to represent psychological models or linguistic structures. Brachman's analysis of these early systems showed that their primitives were expressed at different levels of

abstraction. At the lowest level were simple implementational primitives, mere nodes and links with no substantive knowledge-structure claims. At a higher level were semantic nets made up of logic primitives, where links represented logical relationships such as AND, SUBSET, etc.

Next were the conceptual models, where nodes and links represented conceptual entities and their relationships.

Brachman classified Schank's conceptual dependency model as fitting into this category. Finally, Brachman listed a linguistic level of semantic network primitives. Brachman suggested that there should be another level, between the logical and conceptual levels, that he called the epistemological level, and introduced KL-ONE as a language for representing semantic networks at this level.

7.2 Semantic Networks as Database Models

The idea of using a semantic network for representing database structure is not new, and has been employed for making databases more intelligent. Roussopoulos and Mylopoulos (1975) were among the first to experiment with semantic-network data models. One frequent complaint about traditional data representation formalisms (e.g. hierarchical, network, and relational models) is that they lack a coherent framework for representing the semantics of the data contained within the database. Although some work has been done in describing semantics via functional dependencies, Roussopoulos and Mylopoulos argued that this does not capture all semantic information about a database. Rather, they argue for a semantic network formalism, which they used to represent the semantic structure of the database. This semantic structure would then be converted into *relational* schema.

Their semantic net model is a graph representation using four types of nodes (concepts, events, characteristics, and

values). Nodes are linked together via edges that pertain to concepts such as sub-type, part-of, and definition-of. Large chunks of the nodes and edges of the semantic network are called scenarios, and it is through the use of these scenarios that inferences and predictions about the data can be made, even if the data is incomplete (i.e. the nodes of the semantic net are only partially instantiated).

The nodes and edges of their semantic net form a natural correspondence to, and can be converted easily into, relational schema such as concept relations, part relations, event relations, and characteristic relations. Thus, operations on the database should be comparable to those employed by the relational model.

Cohen and Kjeldsen's GRANT expert system (1987) uses a semantic network representation for the purpose of *indexing* into a database of research agencies. GRANT's approach differs from Roussopoulos and Mylopoulos's model in that the semantic network representation is not intended to be converted into a relational model. Rather, the links of the semantic network are used to provide a rich indexing scheme into the database, and thus foster the ability to do limited inferencing of the data by means of a constrained form of spreading activation.

The database itself consists of records (frames) pertaining to research agencies whose fields (slots) contain information about those agencies. The nodes of the GRANT's semantic network represent concepts pertaining to various

research interests that one or more agency may support. The concepts may be very specific (e.g. specialized sorts of heart disease such as mitral valve prolapse) or more general (e.g. medical issues in general). Nodes are connected via links that represent superclass-subclass hierarchies, cause-effect relationships, part-of relationships and many more (48 link types in all).

Cohen and Kjeldsen hypothesized (and showed empirically) that the spreading activation capability of semantic network database indexing would result in a higher "hit rate" (i.e. discovery of viable research agencies) for database queries than would a simple keyword search. This is because keyword search restricts the search to those words explicitly entered via the query, whereas spreading activation allows search to include words and concepts that are "related" to the explicit query words. However, spreading activation increases the "false-positive rate" (i.e. discovery of research agencies that are not viable for the stated query) for the same reason.

Thus, Cohen and Kjeldsen used several methods of constraining the spreading activation. One simple but weak method is to limit the distance of the spread to only four links. A second method is to stop the spread once a node with large "fan-out" (i.e. one connected to many other nodes) is reached. The third, and most sophisticated method, is to use heuristics to describe the "kinds" of paths that can be searched in the network. Such *path-endorsement* heuristics

describe what kinds of links can be combined together to form a traversable path.

Cohen and Kjeldsen found that the use of these constraints helped reduce the false-positive rate while still maintaining a significantly better hit rate than straight keyword searches of the database. Thus, the use of semantic network database indexing schemes appear to be a viable option, particularly if inferencing is required.

7.3 A Semantic Network View of the Country Consultant (CC)

As mentioned earlier, the CC's indexing scheme is based on four major conceptual groups (called *concept types*). The concept types are: *market feature*, *industry*, *mode of operation*, and *market*. Each concept type can be thought of as a miniature semantic network. For example, market feature consists of around sixty feature concepts (e.g. tariff level, commercial environment, political stability, economic growth, non-tariff barriers, etc.) which are represented as nodes in the network. The nodes are related to each other via links. Currently, the only kind of links in our system are *parent-child* links, representing the classic IS-A relation. for example, *tariff level* is a subordinate (child) feature of *regulations*.

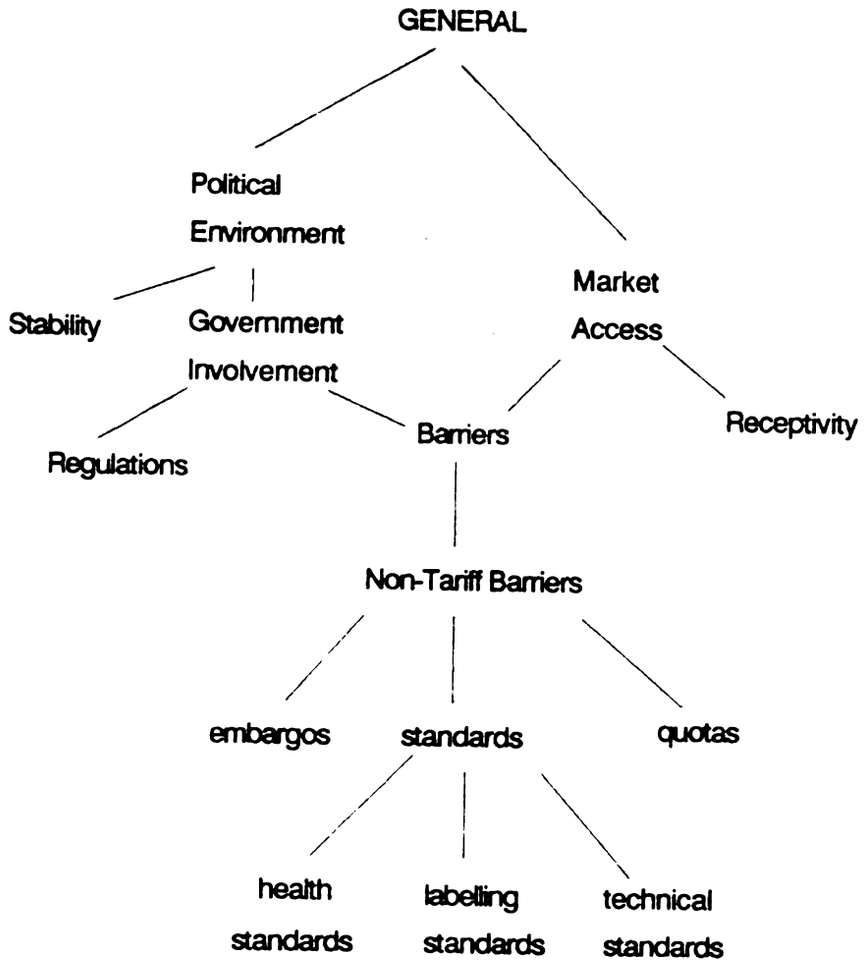


Figure 7.2

Partial view of Country Consultant's semantic network for MARKET-FEATURE concept type.

Figure 7.2 shows a partial semantic net view of the feature concepts. Similar relationships exist for mode-of-operations concepts and for industry concepts.

7.3.1 How the CC Infers Evaluations

The evaluative nature of the CC is expressed in the judgement records. These records, which are entered by experts (hereafter called judges), contain judgements pertaining to specific concept combinations (market feature, entry mode, industry, and market). For example, a judge may enter a judgement record stating that the commercial environment (feature) for exporting (mode) drugs and pharmaceuticals (industry) to Austria (market) is good. The judge can indicate his or her confidence in that judgement (between 0 and 1). The judge can also indicate the direction (improving, getting worse, etc.) and confidence in the direction. Finally, the judge can enter comments justifying the judgement entered. Figure 7.3 shows a sample Judgement Entry Screen.

Obviously, a database with a large numbers of industry classes, markets, features and entry modes will have a very large possible number of judgements. Currently the breakdown of conceptual primitives in the database is like this:

57 market features
98 industry categories
11 entry modes
39 markets

This results in 2,396,394 possible judgements in the

database, and we anticipate that this number will grow as more concepts are added to the network. Of course, it is not feasible for experts to enter all of these judgements. This is especially true because we are forcing judgements to be well-researched, based on text found in international marketing reports, such as U.S. Commerce Department's Country Market Plans (CMPs). Therefore, the CC should be able to infer what a judgement should be upon request, even if thatjudgement has not been explicitly entered by an expert, based on the explicit judgements that are "conceptually close" to it.

Additionally, with such a large database, it is important to maintain the integrity of the content of the database. Judgements should be consistent with each other. Thus, the CC should be able to "second-guess" judges. Inferring what a judgement should be based on conceptually close judgements would help in this regard.

The CC does this inferencing by combining ideas from two areas in AI:

1) the spreading-activation inference structure common to semantic networks.
2) the use of weighted evidence accumulation common in probabilistic inference networks and some rule-based systems. The linear model common to MAUT is used to

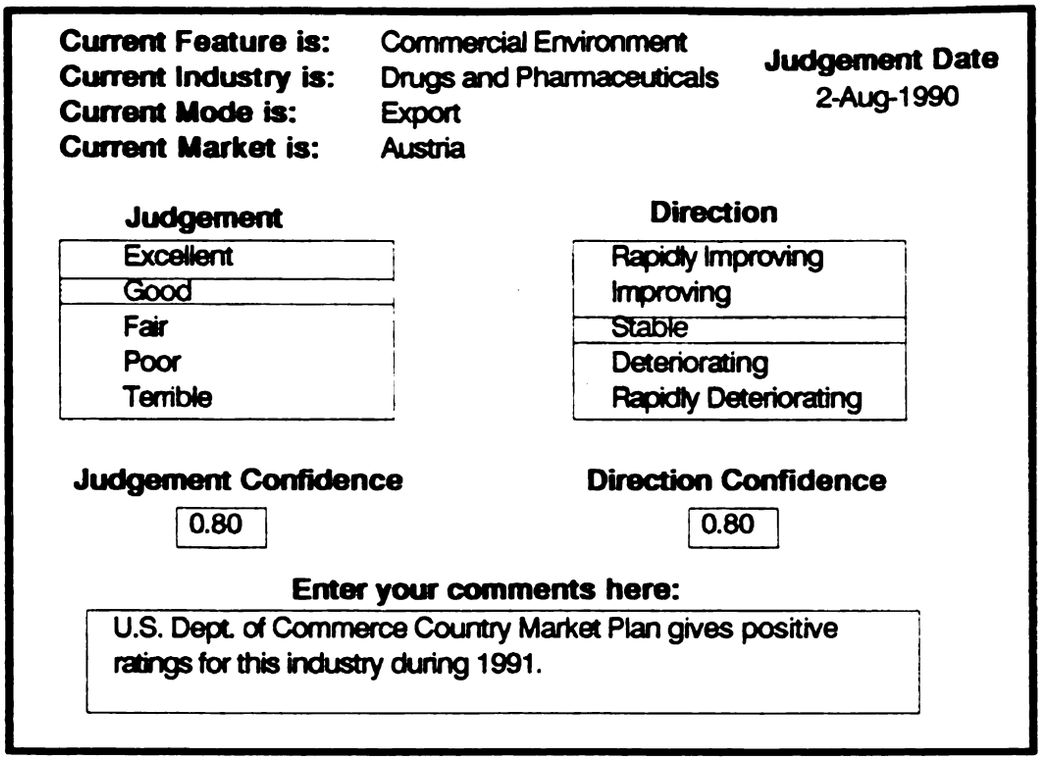


Figure 7.3
A Sample Judgement Entry Screen.

Please select your inference strategy.

Concept Type	*From Link*	*Steps*	*Att*	*To Link*	*Steps*	*Att*
FEATURE	Parent	2	0.9	Child	2	0.9
INDUSTRY	Parent	2	0.9	Child	2	0.9
MODE	Parent	1	0.9	Child	1	0.9

Figure 7.4
Country Consultant's Inference Strategy Entry Screen.

assess utility of any combination of concepts (market, market feature, industry, and entry mode).

7.3.2 Spreading Activation in CC

The user can request CC to infer the value of a particular unknown judgement (pertaining to a specific industry class, market, market feature, and entry mode). CC responds to this request by performing a constrained spreading activation of each concept type, anchoring at the queried concept in the type. The spreading activation is constrained via a default inference strategy established by the knowledge engineer or by an inference strategy selected by the user. The inference strategy sets a limit for how far along each link type to search, as shown in the Inference Strategy Entry screen of figure 7.4. Additionally, the inference strategy specifies an "attenuation factor", defining the degree to which the *conceptual distance* from the located judgement to the queried judgement along various links will diminish the influence that the located judgement has on the inferred (queried) judgement. As shown in figure 7.4, the attenuation factor for all links is set to 0.9. This means that for each step away from the concept being inferred, the influence is multiplied by 0.9. Thus, for one step away the influence is 90%, for two steps 81%, for three steps 72%, etc.

Figure 7.5 shows an example where the inference strategy specifies a limit of two steps along the parent link and two

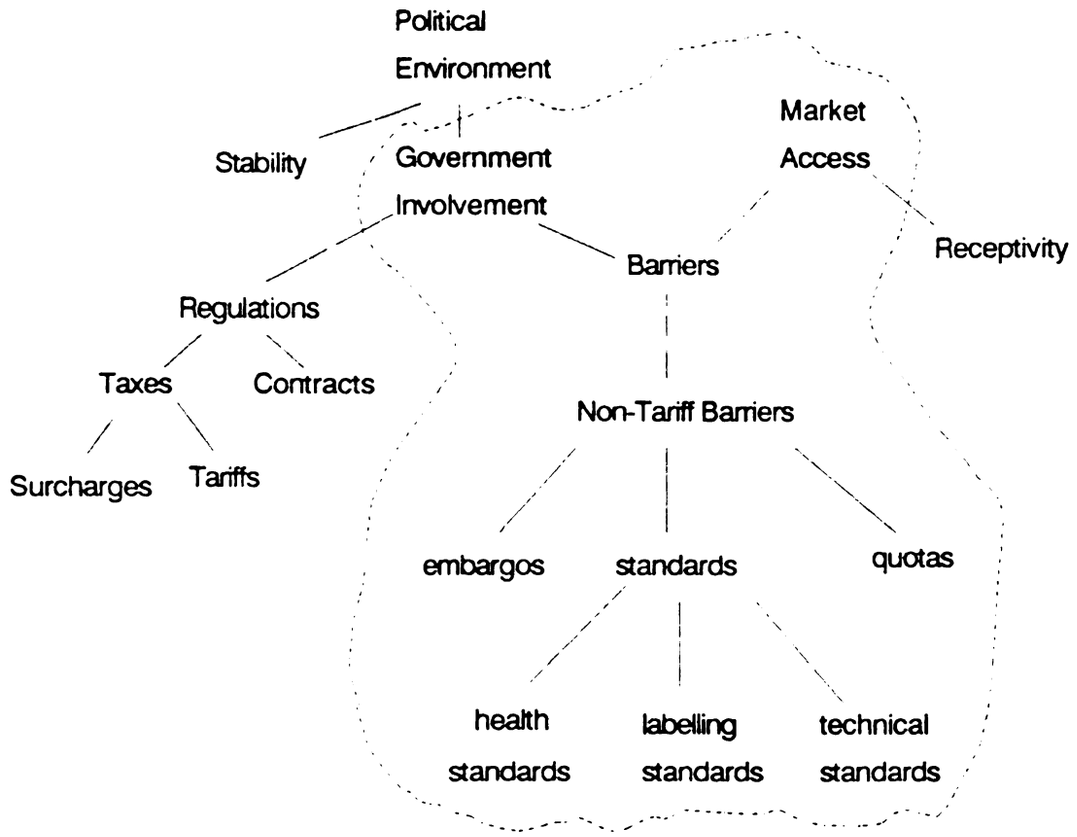


Figure 7.5

The "scope" of a spreading activation of MARKET FEATURE concepts, centering on NON-TARIFF BARRIERS, with parent- and child- link search steps limited to 2.. Concepts within the scope will be used in inference process.

steps along the child link for the MARKET FEATURE concept type, anchored at the concept Non-Tariff Barriers. You can see that with this constraint, CC is limited to looking at six feature concepts. By placing similar constraints of the other concept types, one can reduce the scope of the search considerably, as shown in figure 7.5.

Based on the number-of-steps constraint in the inference strategy,the CC will enumerate all possible combinations of FEATURE, INDUSTRY, and MODE concepts for a given market. Then it will search the database searching for judgements that pertain to any of these combinations. The judgements that are found will all be used to infer the queried judgement.

7.3.3 Inferring Judgements via Weighted Evidence Accumulation

As mentioned earlier, CC uses techniques from MAUT to infer a judgement for a given concept combination based on related judgements found during the spreading activation process. This section describes how weights are assigned to each located judgement in order to arrive at the final inferred judgement. Note that the standard weighted linear model is used in the Country Consultant, similarly to its use in CEVAL. However, unlike CEVAL, weights are not assigned explicitly by the knowledge engineer or domain expert, but rather are calculated by CC.

Once the relevant judgements have been located via the spreading activation process, the system must decide the

degree to which each judgement found will contribute to the judgement being inferred. This decision is based on the following two principles:

1) Judgements that are "conceptually close" the inferred judgement have more influence than those that are "conceptually far".
2) Judgements with higher confidence levels have more influence than those with low confidence levels.

Thus, the *weight* of influence that a located judgement exerts on the inferred judgement is based on a combination of these two principles, as expressed in the following equations:

Concept Attenuation Factor (*CAF*):

$$CAF\text{-}ATT^{STEPS}$$

where *ATT* is the attenuation factor for the given concept link type based on the inference strategy

and *STEPS* is the number of steps (links) between the found judgement and the inferred judgement along that link path

Judgement Attenuation Factor (*JAF*):

$$JAF = \prod_{i}^{n} CAF_i$$

where *n* is the number of concept types (currently 4)

Judgement Weight of Judgement *i* (JW_i):

$$JW_i = \frac{JAF_i \times JC_i}{\sum_{j}^{n} JAF_j \times JC_j}$$

where *n* is the total number of judgements found via the spreading activation process

and JC_i is the confidence level assigned to judgement *i*.

Inferred Judgement Score (*IJS*):

$$IJS = \frac{\sum_{i}^{n} JW_i \times JS_i}{n}$$

where *n* is the total number of judgements found via the spreading activation process

and JS_i is the judgement score for judgement *i*.

Once the judgement score is inferred, that score is mapped

onto a rating by comparison with threshold scores. For example, a minimum score of 90 results in an Excellent rating.

7.4 The Country Consultant as a MAUT Model

The reader will note that, like the Candidate Evaluation architecture described previously, the Country Consultant makes use of a weighted additive model to ascertain evaluative information. The main difference is that, unlike the CE model, the parent-child links between conceptual nodes are not used for propagating scores from lower-level nodes to higher-level nodes. Rather, the links are used to facilitate spreading activation, which in turn provides all possible judgements that could influence the final evaluation, subject to the constraints imposed by the inference strategy. Nevertheless, once these judgements have been obtained, they become the terms of a weighted linear model. Thus, the evaluation process is an implementation of *compensatory* MAUT decision rules, where the MAUT "attributes" are actually judgements. Also, like the CE model, the final score resulting from the additive model is translated, via a comparison with threshold values, into a verbal rating (excellent, good, fair, poor, terrible).

To my knowledge, this is the first time a semantic network representation has been combined with an additive MAUT judgement model to provide a database representation facilitating evaluative reasoning. Although this particular implementation is designed for international marketing, it is

my belief that the "MAUT semantic network" approach can be used in other domains as well. As I will discuss in chapter 8, a possible avenue of future research is to generalize the semantic-network/MAUT method into a domain-independent problem-solving architecture.

7.5 Knowledge Acquisition and Validation for the Country Consultant

In the Country Consultant, knowledge acquisition takes two forms. First is the development of the semantic network. This includes identification of the nodes (concepts) as well as their links (relationships). As mentioned earlier, nodes are of four types: markets, market features, entry modes, and industry classifications. Of these, the industry classifications and the markets are fairly straightforward. Currently, CC includes approximately 26 countries (markets) and approximately 100 industry classifications. The industry classifications roughly correspond to U.S. Department of Commerce classifications. the entry mode classification is based on the descriptive taxonomy shown in chapter 6 of this thesis. Likewise, the market features are based on research conducted by Cavusgil, Root, and others, also discussed in chapter 6. Development of the semantic network also involves assigning an inference strategy to each concept in the network. For example, the inference strategy for the feature Intellectual Property Protection was set to allow a search of

one step down the child link and zero steps up the parent link, with attenuation factor of 0.9 for each child-link step. Thus, each time the Country Consultant attempts to infer a judgement or guideline for intellectual property protection, the features that will be included in the search are IPP itself and each of its "children" (copyrights, trademarks, patent protection, and royalties).

The second form of knowledge acquisition is the day-to-day market research and entry of information into the proper concept combination. This is essentially a data entry task, performed by judges who scan the academic, government, and industry literature and enter appropriate information into the Country Consultant via edit screens, as shown in figure 7.3. The important element in this type of knowledge acquisition is to appropriately classify each entry in terms of its market, industry, entry mode, and feature. In addition, if the entry is a judgement, proper judgement and directions must be entered, as well as appropriate confidence levels. Because of the subjective nature of these judgements, it is important to continuously validate the information that has been entered. This issue, while an important one, has not been sufficiently addressed in my current research. It would be a pertinent issue to explore in future research, as discussed in chapter 8.

7.6 Conclusions

This chapter presents an *inferential-evaluative database*, called the Country Consultant, which makes use of MAUT methods and a semantic network indexing scheme. The database is used to store market research information for the international marketing domain.

The Country Consultant is currently being used at International Business Centers and Michigan State University to facilitate education and counselling in international marketing. It is being used both as an educational tool (Bhargava et al. 1991) and as an aid for international business counselling. As of this writing, approximately twenty small business executives and over one hundred graduate students have used Country Consultant in some way. It has two main advantages over other market-research databases. First, entries into CC are catalogued according to market, feature, entry mode, and industry, so that a user can query to obtain information specifically geared to answering a particular question. Second, through the use of MAUT and spreading activation, the system is able to infer judgements for which it has no explicit records. This introduces AI capabilities to CC, making it more than just a database.

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

The preceding chapters explored two major areas of research, and combined their findings into a new expert-system method for solving certain kinds of problems. The task-specific approach to knowledge representation in artificial intelligence and multi-attribute utility approaches to decision theory were combined to inspire development of a problem solving architecture for candidate evaluation. Below is listed the contributions of this thesis, and some suggestions for future research.

8.1 Contributions of the Thesis

This research makes a number of contributions:

First, it helps bridge the gap between AI and decision sciences, particularly in terms of multi-attribute utility theory. To this end, the thesis presents a review of parallel research being done in both fields, including decision theorists like Tversky, Dawes, and Slovic, as well as AI researchers such as Samuel, Berliner, and Langlotz. In addition, the thesis characterizes a HQLM (hierarchical quasi-linear model), describing it from an AI and DT perspective. An architecture is described which combines compensatory and non-compensatory reasoning methods into a single representation, and provides a facility for qualitative explanations of quantitative reasoning techniques. In this sense, the research

makes a similar sort of contribution as that made by Langlotz's QXQ system, specifically to provide qualitative insight into the quantitative reasoning processes inherent in decision theory.

Second, the research describes Candidate Evaluation as a new task-specific architecture. The thesis presents a comprehensive survey of and comparison between different TSA approaches, including Chandrasekaran's generic task approach and the knowledge acquisition methods developed by McDermott and colleagues. In addition, the thesis provides a detailed description of the Candidate Evaluation architecture and its implementation in the CEVED/CEVAL shell.

Third, the research explores another potential combination of MAUT and AI. A MAUT semantic network model is introduced for developing an evaluative-inference database, the Country Consultant. The thesis provides a description of the database, together with a review of the semantic network model in AI and in database representations.

Fourth, the thesis contributes to the business research community by applying AI and decision-theoretic techniques to international marketing problems. Specifically, the MAUT semantic network model is used to develop a database of market research information. The Candidate Evaluation architecture, through the tools CEVED and CEVAL, are used to develop expert systems for tasks such as entry mode selection, distributor/agent evaluation, freight forwarder evaluation,

and joint venture partner selection.

Fifth, the research contributes to knowledge acquisition by facilitating the use of TSA shells by non-technical domain experts. This is made possible through the knowledge-use level of the Candidate Evaluation language, and its specific focus on a single problem-solving method. CEVED was developed as an authoring tool to be used by College of Business faculty and students, most of whom have little or no AI and computer science background. The fact that successful applications have been developed using CEVED is testimony to the validity of the Candidate Evaluation architecture. In addition, several articles have been submitted to and/or published in academic literature, both in AI (Mitri 1990, Mitri 1991) and in marketing (Mitri et al. 1991a, Mitri et al. 1991b).

8.2 Future directions:

The research described in this thesis serves as a stepping stone for future research in AI, decision theory, and knowledge acquisition. Some of the issues pertinent to future research include the following:

8.2.1 Multiple-evaluator issues

The Candidate Evaluation architecture does not currently address the issue of multiple evaluators, an issue discussed in chapter 4. However, our experiences working with multiple experts establishing and validating dimension weights has

shown us how important it is to reconcile differences in weightings among the experts. Thus, a promising and necessary avenue for future research concerns developing computational methods for deriving weights based on candidate rankings, delphi studies, and various voting methods. One possibility would be to include in CEVED a facility for Borda Count voting among experts. Thus, the knowledge acquisition facility in CEVED would be enhanced by the introduction of an optimal voting method for ranking sample candidates and/or dimensions and thereby obtaining consistent dimension weights. In addition, CEVED or CEVAL could include Borda Count voting for ranking of candidates in the validation process.

8.2.2 Generalizations and Extensions of Semantic-Net MAUT model

Currently, the MAUT semantic net database is implemented in the form of the Country Consultant, a domain-specific database for international marketing. However, the model is generalizable across a wide variety of domains. Like the Candidate Evaluation architecture, it tends to be task-specific, and is best suited for tasks requiring associative, weighted multi-attribute reasoning techniques. Thus, a useful next step would be to develop a general-purpose MAUT semantic network shell that could be used to model many evaluative inferential databases.

Another enhancement to the MAUT semantic network would be

to refine the inference strategy. The current inference strategy implements a "weak" heuristic in the sense that it provides a default search constraint and attenuation factor for each concept of a particular concept type, but is not specific to a particular situation. The default inference strategy can be supplemented with concept-specific inference strategies, which provide specific search scopes and attenuations for specific combinations of concepts. Such "strong" heuristics would add intelligence to the database search.

One need that became clear with the development of the Country Consultant is the need for automated validation techniques. Factors pointing to this need include the potential size of such a database, the volatility of market research data and its inevitable change over time, and the complexity of the semantic network structure. Clearly the validation and verification process will be unwieldy if left to manual means alone. Thus, future research should concentrate on automating the validation process with respect to internal and external consistency of judgements and guidelines. Several issues are pertinent in validation research. First, how should judgements be distributed across the Excellent-to-Terrible spectrum? Are we looking for a normal curve? Second, how is "inferential consistency" (i.e. the consistency between an actual judgement and an inferred judgement for the same concept combination) to be measured?

Third, exploration should be done pertaining to the use of AI text analysis methods for verifying the judgements and/or concept-combination assignments for text entered into the system.

8.2.3 Linkage of CEVAL modules with Country Consultant and Each Other

Future development of CEVAL and Country Consultant will involve providing linkage between the tools. This is necessary because many decisions made in CEVAL modules will require assessment of target market characteristics. Such a linkage between TSA tools is not unusual. For example, CSRL currently has database hooks to obtain information from intelligent databases via a GT tool called IDABLE.

In the future, CEVAL modules can be linked together. The evaluation results coming from one module may become the contextual factors of a second module. In addition, plans in one CEVAL module may trigger the running of another CEVAL module. In this way, strategic control of large knowledge bases involving several modules can be implemented.

8.2.4 Knowledge Acquisition and Representation Enhancements to CEVED

Currently, the CEVED tool is closer in functionality to the generic task languages (CSRL, DSPL, HYPER, etc.) than to

knowledge acquisition tools like SALT, MOLE, and KNACK. In other words, CEVED is more or less a blank slate, an authoring tool, a programming language. It does not contain knowledge acquisition techniques such as interviewing, or doing consistency checks. Future research should concentrate on making true KA contributions to the MAUT technology via CEVED. Such enhancements may be facilitated through a number of modifications of CEVED. For example, CEVED should include the capability for Monte Carlo analysis. Monte Carlo analysis will provide the ability to produce "training samples" to the system, which could then be used to computationally derive threshold values for ratings, as suggested by Page (1977) and discussed in chapter 5 of this thesis. The implementation of delphi studies, Borda Count voting, and linear regression would also enhance the KA facility.

Graphic representations and browsers should be included to provide appealing visual information. Finally there should be the ability to do consistency checks in the knowledge base, such as verifying that combinations of preconditions for a recommendation fragment can actually occur given the current weight and rating-threshold circumstances. More exploration of MAUT and evaluation research will provide insight into further refinements that can be made to the Candidate Evaluation architecture and the CEVED tool.

8.3 Final Conclusions

The work described in this thesis is a result of research done in three main academic disciplines. First, research in artificial intelligence, and particularly in the task-specific architecture (TSA) approach to knowledge acquisition and representation for expert systems, served as a motivator and guide for the development of a shell to facilitate encoding of domain expertise. Second, research in decision theory, especially dealing with multi-attribute utility theory (MAUT), provided an architectural framework for encoding of evaluative reasoning tasks. Third, research and practical experience in international marketing provided a domain in which to apply and test the problem-solving architecture.

From the research, literature review, and software development accomplished for this thesis, it is clear that MAUT and TSA can be combined for generating an environment that facilitates knowledge acquisition for evaluative tasks and allows non-programming domain experts to play a more central role in expert systems development than is possible with conventional expert system shells. This supports many of the claims made by proponents of TSA and generic tasks (such as Chandrasekaran), and is consistent with the findings of researchers, such as Langlotz, who combine AI and decision-theoretic techniques. In addition, it is clear that there is much potential for this MAUT-TSA combination in the business world, and particularly in internatinonal marketing.

The CEVED and CEVAL tools have been and will continue to be used for this research and development effort. Enhancements described in the preceding section will be implemented to improve their effectiveness. In addition, there will be continued effort to apply the Candidate Evaluation technique to domains outside of international marketing in order to demonstrate the general utility of the model.

APPENDIX A

FORMAL CHARACTERIZATION OF THE CANDIDATE EVALUATION ARCHITECTURE

APPENDIX A

FORMAL CHARACTERIZATION OF THE CANDIDATE EVALUATION ARCHITECTURE

Space complexity

The following is an analysis, from a set-theoretic viewpoint, of the space complexity of the candidate evaluation architecture.

D = the set of all dimensions:

$$D=\{d_1, d_2, \ldots d_{n_d}\}$$

where n_d is the total number of dimensions in the knowledge base.

R = the set of all ratings:

$$R=\{r_1, r_2, \ldots, r_{n_r}\}$$

where n_r is the total number of ratings in the knowledge base.

Because ratings are tied to dimensions, another characterization of R is useful:

$$R=\{R_1 \cup R_2 \cup \ldots \cup R_{n_d}\}$$

where R_i is the set of ratings associated with dimension D_i, such that:

$$R_i \subseteq R$$

DR = the set of all possible dimension-rating pairs

$$DR = \{dR_1, dR_2, \ldots, dR_{n_d}\}$$

where dR_i is the set of dimension-rating pairs for the dimension di, such that:

$$dR_i = \{(d_i, r_{i_1}), (d_i, r_{i_2}), \ldots, (d_i, r_{i_{n_{R_i}}})\}$$

where n_{Ri} is the number of ratings associated with the dimension d_i.

It follows that:

$$DR \subseteq D \times R$$

and:

$$|DR| = \sum_i^{n_d} |R_i| \leq |D| \times |R|$$

Thus, the upper bound for the size of all possible dimension-rating pairs is the product of the number of dimensions and the number of ratings in the knowledge base.

Now, let RF be the set of all recommendation fragments in the knowledge base:

$$RF = \{rf_1, rf_2, \ldots, rf_{n_{rf}}\}$$

where n_{rf} is the total number of recommendation fragments in the knowlege base. Each recommendation fragment can be associated with at most one rating for each dimension. That is, a recommendation fragment may be tied to several dimensions, but at most to one rating for each of those dimensions. Thus, the maximum number of conditions associated with a recommendation fragment rf_i is n_d.

Consider a subset of D consisting of $\{d_{i1}, d_{i2}, \ldots, d_{ik}\}$. Call this set D_j. Then we can define a subset of RF, called RF_{Dj}, which is the set of all possible recommendation fragments that are associated with all the dimensions in D_j. The size of RF_{Dj} is bounded by the number of ratings for each dimension in D_j. In particular:

$$|RF_{D_j}| \leq |R_{i_1}| \times |R_{i_2}| \times \ldots \times |R_{i_k}|$$

In practice, not all these recommendation fragments will be plausible, since there may be inconsistencies between ratings of the different dimensions in D_j. For example, if d_{i1} is a parent of d_{i2} then, depending on the maximum and minimum possible weights of d_{i2}, the chosen rating for d_{i1} may in fact restrict the possible ratings for d_{i2}. It may be impossible for d_{i1} to be rated as excellent and d_{i2} to be rated as horrible. Thus, the above equation serves as an upper bounds for the size of RF_{Dj}, which is probably going to be much larger than its actual allowable size. (Actually, such a "consistency

constraint" is not currently implemented in the Candidate Evaluation shell; there is no way to prevent the developer from creating a recommendation fragment whose conditions are inconsistent with each other. However, that recommendation fragment will never appear to the end user during a consultation, since its conditions are impossible).

Now, the set of all D_i is the power set 2^R. Thus, the upper bound for the size of RF can be characterized as:

$$|RF| \leq \sum_{j}^{|2^R|} |RF_{D_j}|$$

From here, we can define a recommendation rec_i thusly:

$$rec_i = \{rf_{i_1}, rf_{i_2}, \ldots, rf_{i_k}\}$$

In other words, a recommendation is a set of recommendation fragments. Thus, we can define the set of all possible recommendations as the power set of all possible recommendation fragments:

$$REC = 2^{RF}$$

Of course, there is also a "consistency constraint" on the number of possible recommendations. Such a constraint is based on the following two principles:

1) A recommendation cannot contain two recommendation

fragments whose conditions are inconsistent with one another. In other words, two recommendation fragments whose conditions are associated with the same dimension but involve different ratings for that dimension will never appear together in a recommendation.

2) A recommendation cannot contain two recommendation fragments if one fragment suppresses the other.

These two constraints can significantly reduce the number of possible recommendations for a given knowledge base, although that number will still be exponential with respect to the number of dimensions. This is not a space complexity issue, however, since recommendations are not stored explicitly but are constructed from recommendation fragments based on the results of a consultation.

Time complexity

There are three basic steps in the candidate evaluation process. These are:

1) Dimension weight adjustment based on context questions.
2) Dimension scoring and score propagation based on evaluative questions.
3) Recommendation triggering and presentation based on dimension ratings.

Weight Adjustment

Adjusting a dimension's weight involves two main steps:

First, make the weight change based on the answer to a context question. This is $O(1)$.

Second, normalize the weights of the affected dimensions and all its siblings so that they add up to 100% while maintaining their new ratio to each other. The formula for doing this for each sibling dimension is:

$$\frac{Wt_{D_i}}{\sum_j^n Wt_{D_j}} \times 100$$

where n_s is the total number of siblings. Thus, there must first be a summation of all weights, which is $O(n_s)$. Then, the normalization must take place for each sibling, also $O(n_s)$. Since this is a step-by-step process (i.e. first sum the weights, then normalize each dimension weight), the total complexity for adjusting a dimensions weight is $O(n_s)$. If we make a worst-case assumption that the dimension hierarchy consists of one root node and $n_D - 1$ siblings under the root (where n_D is the number of dimensions in the knowledge base), then this implies $O(n_D)$ complexity for dimension weight adjustment and normalization of a single dimension.

Now, the answer to a context question may impact more than one dimension. In a worst case scenario (which, in practice should never occur), a context question's answer may affect all dimensions. Thus, the complexity for dimension

weig

ques

ques

comp

cons

only

case

Sco

set

ass

a m

to

nur

th

su

le

al

on

th

se

al

weight adjustment based on the answer to a single context question is $O(n_d^2)$. If we let n_c be the number of context questions in the knowledge base, then the worst case complexity for dimension weight adjustment for an entire consultation is $O(n_c n_d^2)$.

In practice, a context question will be associated with only a few dimensions, maybe two or three. Thus, the average case time complexity for weight adjustment should be $O(n_c n_d)$.

Score Propagation

The time to score a leaf node dimension (i.e. a "question set") is $O(n_{qi})$ where n_{qi} is the number of evaluative questions associated with the dimension i. This is because it is simply a matter of adding n_{qi} weighted scores together. Thus, the time to score all question sets is $O(n_q)$, where n_q is the total number of evaluative questions in the knowledge base.

The time to propagate a leaf node dimension's score up the dimension hierarchy is $O(log\ n_d)$. This is true because such a propagation involves tracing the ancestral path of the leaf node, altering maximum and minimum scores for each node along that path. If we assume that each dimension contains only one evaluative question (a worst-case scenario), then this implies that there are n_q leaf node dimensions (question sets). This sets the upper bound for propagating scores for all leaf node dimensions up the hierarchy at $O(n_q\ log\ n_d)$.

Recommendation Triggering and Presentation

Triggering and presenting recommendations involve three main steps:

1) Triggering the recommendation fragments

2) Suppressing recommendation fragments

3) Sorting recommendation fragments

The basic algorithm for triggering recommendation fragments is shown below:

```
for each recommendation fragment
   if the conditions match for that recommendation
   fragment
     add it to a list of triggerred recommendation
     fragments
   end
end
```

If we let $n_{\underline{rf}}$ be the number of recommendation fragments in the knowledge base and m be the maximum number of conditions per recommendation fragment, then the time to trigger recommendation fragments is $O(mn_{\underline{rf}})$.

The algorithm for suppressing recommendations is as follows:

```
for each recommendation fregment in the triggered list
  for each rec fragment suppressed by the current one
     add the suppressed rec fragment to a list of
               suppressed recommendation fragments
     end
  end
end

for each suppressed recommendation fragment
    delete it from the triggered rec-fragment list
end
```

If we make the worst case assumption that a

recommendation fragment can suppress all others, then the first loop is $O(n_{rf}^2)$. The second loop is $O(n_{rf})$. Thus, recommendation suppression is $O(n_{rf}^2)$.

Sorting the recommendation fragments is done via standard sort techniques, which will be at most polynomial. Thus, if we assume that the maximum number of conditions per recommendation fragment is no more than the total number of recommendations, then the worst case time complexity for triggering, suppressing, and sorting recommendations is $O(n_{rf}^2)$.

Total time complexity for a CEVAL consultation

The above three sections described the time complexity for each portion of a CEVAL consultation. If we add the total time equired for context-based weight adjustment, score propagation, and recommendation presentation, it comes to:

$$O(n_c n_D^2) + O(n_q \log n_D) + O(n_{rf}^2)$$

where:

n_c is the number of context questions in the knowledge base

n_D is the number of dimensions in the knowledge base

n_q is the number of evaluative questions in the knowledge base.

n_{rf} is the number of recommendation fragments in the knowledge base.

Thus, this translates to a polynomial time complexity with respect to the various types of input variables that

CEVAL receives. Keep in mind that the number of recommendation fragments could potentially be exponential with respect to the number of dimensions in the knowledge base. However, in practice this will not occur; rather, recommendation fragments will be created only for "relevant" combinations of dimension-rating values, as deemed necessary by the knowledge engineer.

BIBLIOGRAPHY

Arrow, K.J. (1963) *Social Choice and Individual Values*. Cowles Foundation Monograph 12. Yale University Press. New Haven.

Barr, A., Cohen, P.R., Geigenbaum, E.A. (1989) *The Handbook of Artificial Intelligence*. Vol IV. Addison Wesley. Reading, Mass.

Berliner, H. (1977) Experiences in Evaluation with BKG -- a Program that plays Backgammon. *Proc. Intl Joint Conf on Artificial Intelligence*. pp. 428-433

Berliner, H. (1979) On the Construction of Evaluation Functions for Large Domains. *Proc.Intl Joint Conf on Artificial Intelligence*. pp. 53-55.

Berliner, H. and D. Ackley. (1982) The QBKG System: Generating Explanations from a Non-Discrete Knowledge Representation. *Proc. National Conf. on Artificial Intalligence*. pp. 213-216.

Bhargava, V., Evirgen, C., Mitri, M. and Cavusgil, S.T. (1991) Using Expert Systems in the Classroom: The Case of the Country Consultant. Submitted for publication to *The Journal of Teaching in International Business*.

Boose, J.H. (1989) A Survey of Knowledge Acquisition Techniques and Tools. *Knowledge Acquisition*. Vol. 1 No. 1 pp. 3-37.

Boose, J.H. and Bradshaw, J.M. (1987) AQUINAS: A Knowledge Acquisition Workbench for Building Knowledge-Based Systems. *Proceedings of the 1st European Workshop on Knowledge Acquisition for Knowledge-Based Systems*. Reading University, Sept. pp.A6.1-6.

Brachman, R.J. (1979) On the Epistemological Status of Semantic Networks. In *Readings in Knowledge Representation*. Ed. Brachman and Levesque. Morgan Kaufmann Publishers, Inc. 1985. pp. 191-216.

Breuker, J. and Wielinga, B. (1989) Models of Expertise in Knowledge Acquisition. In *Topics in Expert Systems Design: Methodologies and Tools*. North Holland Publishing Company:Amsterdam.

Brown, D. (1987) Routine Design Problem Solving. *KBS in Engineering and Architecture*. (Ed.) J.Gero. Addison-Wesley.

Brown, D. and B. Chandrasekaran. (1986) Knowledge and Control for a Mechanical Design Expert System. *IEEE Computer Magazine, Special Issue on Expert Systems for Engineering Problems*. July

1986.

Butler, K.A. and Corter, J.E. Use of Psychometric Tools for Knowledge Acquisition: A Case Study. In W.A. Gale *Artificial Intelligence and Statistics*. Academic Press. New York. pp.295-320.

Bylander, T. and Mittal, S. (1986) CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling. *AI Magazine* Vol. 7 No. 3. pp. 66-77.

Bylander, T. and Smith, J. (1986) Mapping Medical Knowledge into Conceptual Structures. *Proc. of the Expert Systems in Govt Symposium*.

Bylander, T. and B. Chandrasekaran. (1987) Generic Tasks in knowledge-based reasoning: The 'right' level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 1987. Vol. 26, No. 2. pp. 231-243

Bylander, T., Goel, A., Johnson, T. (1988) Structured Matching: A Computationally Feasible Technique for Making Decisions. Ohio State University LAIR Technical Report 88-TB-MATCH.

Casson, Mark (1987) Contractual Arrangements for Technology Transfer: New Evidence from Business History. In *The Firm and The Market*. MIT Press. Cambridge, MA.

Cavusgil, S.T. (1981) Internal Determinants of Export Marketing Behavior: An Empirical Investigation. *Journal of Marketing Research*. Vol. XVIII. Feb, 1981. pp. 114-119.

Cavusgil, S.T. (1985) Guidelines for Export Market Research. *Business Horizons*. Nov-Dec 1985. pp. 27-33.

Cavusgil, S.T. (1987) Qualitative Insights into Company Experiences in International Marketing Research. *Journal of Business and Industrial Marketing*. Vol. 2 No. 3. pp. 41-54.

Cavusgil, S.T. (1988) Unraveling the Mystique of Export Pricing. *Business Horizons*. Indiana University. Vol. 31 No. 3. May-June 1988.

Cavusgil, S.T. (1990) Expert Systems in International Marketing. *Proceedings of the 1990 AMA Summer Educators' Conference*.

Cavusgil, S.T. and Nason, R.W. (1990) Assessment of Company Readiness to Export. *Singapore Marketing Review*.

Cavusgil, S.T. and Sikora,E. (1987) Company Strategies for

International Expansion. Advances in Business Studies. Vol. 1 No. 1. 1987. pp. 1-11.

Chandrasekaran, B. (1983) Towards a Taxonomy of Problem Solving Types. *AI Magazine*. Winter/Spring 1983. pp. 9-17

Chandrasekaran, B. (1986) Generic Tasks in Knowledge-Based Reasoning: High Level Building Blocks for Expert System Design. Ohio State University LAIR Technical Report 86-BC-IEEEX.

Chandrasekaran, B; Josephson, J.; Keuneke, A.; Hemans, D. (1986) An Approach to Routine Planning. Ohio State University LAIR Technical Report 86-BC-PLANNING.

Chung, H.M. (1987) A Comparative Simulation of Expert Decisions: An Empirical Study. UCLA Anderson School of Management Information Systems Working Paper #5-88.

Chung, H.M. (1989) Empirical Analysis of Inductive Knowledge Acquisition Methods. *SIGART Newsletter*. April 1989. Number 108. Knowledge Acquisition Special Issue. pp.156-159.

Clancey, W.J. (1985) Heuristic Classification. *Artificial Intelligence* Vol 27 (1985) pp.289-350.

Cohen, P. and R. Kjeldsen. (1987) Information Retrieval by Constrained Spreading Activation in Semantic Networks. COINS Technical Report 87-66. University of Massachusetts at Amherst.

Daser, Sayeste (1985),International Marketing Information Systems: A Neglected Prerequisite for Foreign Marketing Planning, in *Global Perspectives in Marketing*,Erdener Kaynak,ed.,NY. Praeger Publishers. pp.139-153.

Davidson, William H. (1983) Marketing Similarities and Market Selection:Implications for International Marketing Strategy, *Journal of Business Research* Vol.11,(December), pp.439-456.

Davis, R. (1984) Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence* Vol. 24 pp.347-410.

Dawes, R. (1979) The Robust Beauty of Improper Linear Models in Decision Making. *American Psychologist* Vol.34 No.7 pp.571-582.

Dawes, R. (1988) *Rational Choice in an Uncertain World*. Harcourt Bruce Jovanovich Publishers, Inc. Orlando, FLA.

Dawes, R. and Corrigan, B., (1974) Linear Models in Decision Making. *Psychological Bulletin*. Vol.81 No.2. pp. 95-106.

DeCharme, W.M. (1970) A Response Bias Explanation of conservative Human Inference. *Journal of Experimental Psychology*. Vol.85 pp.66-74.

Douglas, Susan and C.Samuel Craig (1988),"Information for International Marketing Decisions",in *Handbook of International Business*,Ingo Walter and Tracy Murray,eds., New York:John Wiley & Sons, Vol.29. pp.3-29.

Duda, R.O., Hart, P.E., and Nilsson, N.J. (1976). Subjective Bayesian Methods for Rule-Based Inference Systems. In *Readings in Uncertain Reasoning*, Ed: Shafer, G. and Pearl, J. Morgan Kaufmann:San Mateo, Calif. 1990. pp. 274-281.

Edwards, W. and Newman, J.R. (1982) *Multiattribute Evaluation*. Sage Publications . Beverly Hills, CA.

Edwards, W., Phillips, L.D., Hays, W.L., and Goodman, B.C. (1968) Probabilistic Information Processing Systems: Design and Evaluation. *IEEE Transactions on System Science and Cybernetics*. Vol.SSC-4. pp.248-265.

Ehrman, Chaim Meyer and Moris Hamburg (1986) Information Search for Foreign Direct Investment Using Two-Stage Country Selection Procedures:A New Procedure. *Journal of International Business Studies*,Summer, pp.83-88.

Engelmore, R. Morgan, T. (Editors). (1988) *Blackboard Systems*. Copyright 1988. Addison-Wesley.

Erman, L.D.; Hayes-Roth, F; Lesser, V.R.; Reddy, R. (1980) The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys* Vol 12. pp.213-253.

Eshelman, L. (1988) MOLE: A Knowledge-Acquisition Tool for Cover-and-Differentiate Systems. in Marcus, S. (ed) (1988) *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Press. Norwell, MA. pp.37-80.

Fischer, G.W. (1975) Experimental Applications of Multi-Attribute Utility Models. *In Utility, Probability, and Human Decision Making*. ed. D.Wendt. Dordrecht. The Netherlands. pp.7-46.

Gale, W.A. (1987) Knowledge-Based Knowledge Acquisition for a Statistical Counselling System. *International Journal of Man-Machine Studies*. Vol.26 pp.55-64.

Gaschnig, J., P. Klahr, H. Pople, E. Shortliffe, A. Terry. (1983). Evaluation of Expert Systems: Issues and Case Studies. In *Building Expert Systems*. Ed. Hayes-Roth, F., Waterman, D.,

Lenat, D. Addison-Wesley Publishing Co. Reading, MA. pp.241-280.

Goldberg, L.R. (1970) Man vs. Model of Man: A Rationale, Plus some Evidence, for a Method of Improving on Clinical Inferences. *Psychological Bulletin*. Vol.73 pp.422-432..

Goodnow, J.D. (1985) Developments in International Mode of Entry Analysis. *International Marketing Review*. Autumn, 1985. pp.17-30.

Goodnow, J.D. and Hansz, J.E. (1972) Environmental Determinants of Overseas Market Entry Strategies. *Journal of International Business Studies*. Spring, 1972. pp.33-50.

Hammond, T.H. (1986) Agenda Control, Organizational Structure, and Bureaucratic Politics. *American Journal of Political Science*. Vol 30. No 2.

Hayes-Roth B. (1984) An Architecture for Blackboard Systems that Control, Explain, and Learn About Their Own Behavior. Stanford University Technical Report No. HPP 84-16.

Hayes-Roth, B. (1985) A Blackboard Architecture for Control. *Artificial Intelligence*. Vol 26. pp.251-321.

Hayes-Roth, B and Hayes-Roth, F. (1979) A Cognitive Model of Planning. *Cognitive Science* Vol.3 pp.275-310.

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

Herman, D., Josephson, J. Hartung, R. (1986) Use of DSPL for the Design of a Mission Planning Assistant. *Proceedings of the IEEE Expert Systems in Government Symposium*. October 1986, pp. 273-278.

Julesz, B. (1975) Experiments in the Visual Perception of Texture. *Scientific American* Vol.14. pp.24-43.

Kelly, G.A. (1955) *The Psychology of Personal Constructs*. Norton. New York.

Klinker, G. (1988) KNACK: Sample-Driven Knowledge Acquisition for Reporting systems. in Marcus, S. (ed) (1988) *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Press. Norwell, MA. pp.125-174.

Knickerbocker, F.T. (1973) Oligopolistic Reaction and Multinationalization Enterprise. Boston: Harvard Graduate School of Business Administration.

Kogut, B. (1984) Normative Observation on the International Value-Added Chain and Strategic Groups. *Journal of International Business Studies*. Fall. pp.151-167.

Kuipers, B., Moskowitz, A.J., and Kassirir, J.P. (1988). Critical Decisions Under Uncertainty: Representation and Structure. In *Readings in Uncertain Reasoning*, Ed: Shafer, G. and Pearl, J. Morgan Kaufmann:San Mateo, Calif. 1990. pp.105-121.

Laird, J., Newell, A., Rosenbloom, P. (1977) "SOAR: An Architecture for General Intelligence. *Artificial Intelligence*. Vol 33. pp.1-64.

Laird, J. and Newell, (1983) A. A Universal Weak Method. Technical Report, Carnegie Mellon University. Dept. of Computer Science.

Langlotz, C.P. (1989). A Decision-Theoretic Approach to Heuristic Planning. PhD Thesis. Stanford University. Report #STAN-CS-89-1295.

Marcus, S. (ed) (1988) *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Press. Norwell, MA.

Marr, David. (1976) Artificial Intelligence -- a personal view. Massachussetts Institute of Technology Paper AIM355. March, 1976.

McDermott, J. (1982) R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*. Vol.19 No.1 pp.39-88.

McDermott, J. (1988) Preliminary Steps Toward a Taxonomy of Problem-Solving Methods. in Marcus, S. (ed) (1988) *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Press. Norwell, MA. pp.225-256.

Michie, D. (1982) High-Road and Low-Road Programs. *AI Magazine*. Vol 3. pp.21-22.

Minsky, M. (1979) The Society Theory of Thinking. In *Artificial Intelligence: An MIT Perspective*. Vol 1. pp.421-452.

Mitri, Michel. (1991) A Task Specific Problem Solving Architecture for Candidate Evaluation. *AI Magazine*. Vol.12 No.3. pp.95-109.

Mitri, M., Yeoh, P.L., Ozsomer, A., Cavusgil, S.T. (1991) Expert Systems in International Marketing. *Proceedings of the 1991 AMA Microcomputers in Education Conference*. August, 1991.

Musen, M.A., Fagan, L.M., Combs, D.M., and Shortliffe, E.H. (1987) Use of a Domain Model to Drive an Interactive Knowledge-Editing Tool. *International Journal of Man-Machine Studies*. Vol.26. pp.105-121.

Newell, Allen. (1969) Heuristic Programming: Ill-Structured Problems. In J. Aronofsky (ed) *Progress in Operations Research*. NY. John Wiley pp.360-414.

Newell, Allen. (1981) The Knowledge Level. *AI Magazine*. Summer 1981. pp.1-20.

Newell, A. and Simon, H. (1957) Empirical Explorations with the Logic Theorist Machine. In *Computers and Thought* (eds) Feigenbaum, E.A. and Feldman, J. New York: McGraw-Hill. 1963.

Nii, H. Penny (1986) Blackboard Application Systems and a Knowledge Engineering Perspective. *AI Magazine* Aug 1986. pp.82-106

Nii, H.P.; Feigenbaum, J.J.; Rockmore, A.J.; (1982) Signal-to-Symbol Transformation: HASP/SIAP Case Study. *AI Magazine* Vol 3 #2. 1982. pp.23-35.

Offut, D. (1988) SIZZLE: A Knowledge-Acquisition Tool Specialized for the Sizing Task. in Marcus, S. (ed) (1988) *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Press. Norwell, MA. pp.175-200.

Ohlin, B. (1983) *Interregional and International Trade*. Harvard University Press. Cambridge, MA.

O'Keefe, R.M. (1989) The Evaluation of Decision-Aiding Systems: Guidelines and Methods. *Information and Management: The International Journal of Information Systems Applications*. Vol.17 No.4. pp.217-226.

Ozsomer, A. (1991) FREIGHT: An Expert System for International Freight Forwarder Evaluation and Selection. CEVED/CEVAL Expert Systems Module. Developed at International Business Centers, Michigan State University.

Page, C.V. (1972) Applications of Signature Table Analysis to Computer-Assisted Health Screening. *Proceedings of the Fifth Hawaii International Conference of Systems Sciences*, Computers and Biomedicine. January 1972. pp.97-99.

Page, C.V. (1977) Heuristics for Signature Table Analysis as a Pattern Recognition Technique. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. SMC-7 No 2. pp.77-86.

Pearl, J. (1977) A Framework for Processing Value Judgements.

IEEE Transactions on Systems, Man, and Cybernetics. Vol.SMC-7. No.5 pp.349-354.

Porter, M. E. (1985) *Competitive Advantage*. New York. Free Press. 1985.

Punch, W. (1989) A Diagnostic System Using a Task Integrated Problem Solver Architecture (TIPS), Including Causal Reasoning. PhD Dissertation. Ohio State University.

Punch, W., Tanner, M., Josephson, J., Smith, J. (1990) PIERCE: A Tool for Experimenting with Abduction. *IEEE Expert*. Vol.5. No.5. pp.34-45.

Punch, W., Tanner, M., Josephson, J., Smith, J. (1991) Using the Tool PIERCE to Represent the Goal Structure of Abductive Reasoning. Ohio State University LAIR Technical Report 91-WP-PEIRCE.

Quillian, M.R. (1967) Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities. In *Readings in Knowledge Representation*. Ed. Brachman and Levesque. Morgan Kaufmann Publishers, Inc. 1985. pp 97-118.

Rangaswamy, A., Burke, R. Wind, J., Eliashberg, J. (1987) Expert Systems for Marketing. Massachussetts Science Institute Working Paper. Report No.87-1107. Cambridge, MA.

Rangaswamy, A., Eliashberg, J., Burke, R. Wind, J. (1989) Developing Marketing Expert Systems: An Application to International Negotiations. *Journal of Marketing*. Vol.53. Oct 1989. pp.24-39.

Root, F. R. (1982) *Foreign Market Entry Strategies*. AMACOM. New York, NY.

Roussopoulos, N. and J. Mylopoulos. (1975) Using Semantic Networks for Database Management. In *Readings in Artificial Intelligence and Databases*. Ed. Mylopolopus and Brodie. Morgan Kaufmann Publishers, Inc. 1988. pp.112-137.

Rugman, A.M. (1979) Internalization: The General Theory of Foreign Direct Investment. Columbia Univ. Graduate School of Business Working Paper No. 218a. April, 1979.

Saari, D.G. (1985) The Optimal Ranking Method is the Borda Count. Discussion Paper #638. Northwestern University.

Saari D.G,. and Newenhizen, J.V. (1985). A Case Against Bullet, Approval, and Plurality voting. Discussion paper #637. Northwestern University.

Samuel, A.L. (1959) Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal*. Vol.3. pp.211-229.

Samuel, A.L. (1967) Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress. *IBM Journal*. Nov 1967. pp.601-617.

Savage, L.J. (1954). *The Foundations of Statistics*. New York:Wiley.

Schank, R.C. and C.J. Rieger. (1974) Inference and Computer Understanding of Natural Language. In *Readings in Knowledge Representation*. Ed. Brachman and Levesque. Morgan Kaufmann Publishers, Inc. 1985. pp.119-140.

Schiffman, L.G.and L.L. Kanuk. (1987) *Consumer Behavior*. 1987. Prentis-Hall, Inc. Englewood Cliffs, NJ.

Sembugamoorthy, V., Chandrasekaran, B. (1986) Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In *Experience, Memory, and Reasoning*. edited by Kolodner and Reisback. Lawrence Erlbaum Associates publishers, 1986.

Sheridan, T.B. and Sicherman, A. (1977) Estimation of a Group's Multiattribute Utility function in Real Time by Anonymous Voting. *IEEE Transactions on Systems, Man, and Cybernetics*. Vol.SMC-7. No.5 pp.392-394.

Shortliffe, E.H. (1976) *Computer-based Medical Consultations: MYCIN*. New York: North-Holland.

Simon, Herbert A. (1969) *The Sciences of the Artificial*. 1969. MIT Press. Cambridge, MA.

Simon, Herbert, A. (1974) "The Structure of Ill-Structured Problems." *Artificial Intelligence*. Vol 4. pp.181-201.

Slagle, J., Wick, M. (1988) A Method for Evaluating Candidate Expert System Applications. *AI Magazine*. Winter 1988. Vol.9. Nbr.4. pp 44-53

Slovic, P, Fischhoff, B., and Lichtenstein, S. (1977). Behavioral Decision Theory. *Annual Review of Psychology*. Vol.28 pp.1-39.

Slovic, P. and Lichtenstein, S. (1971). Comparison of Bayesian and Regression Approaches to the Study of Information Processing in Judgement. *Organizational Behavior and Human Performance*. Vol.6 pp.649-744.

Steels, Luc. (1990) Components of Expertise. *AI Magazine*. Vol

11. No.2. pp.29-49.

Stefik, M; Aikens, J; Balzar, R; Benoit, J; Birnbaam, L; Hayes-Roth, F; Sacerdoti, E. (1983) The Architecture of Expert Systems. In *Building Expert Systems*. 1983. pp.89-126.

Sticklen, J. (1987) MDX2: An Integrated Medical Diagnostic System. PhD Dissertation. Ohio State University.

Sticklen, J. (1989) Problem-Solving Architecture at the Knowledge Level. *Journal of Experiment and Theory in Artificial Intelligence*. Vol.1. pp.233-247.

Sticklen, J.; Chandrasekaran, B.; Bond, W. (1989) Distributed Causal Reasoning. *Knowledge Acquisition*. Vol.1. pp 139-162.

Sticklen, J., Chandrasekaran, B., Josephson, J. (1987) Modularity of Domain Knowledge. *Expert Systems: Research and Applications*, Vol.I. 1987.

Sticklen, J. Chandrasekaran, B., Smith, J., Svirbely, J. (1985) MDX-MYCIN: The MDX Paradigm Applioed to the MYCIN Domain. *Comp. and Maths with Applications* Vol 11 No. 5 pp 527-539.

Subieta, A. (1991) INTJVS: An Expert System for International Joint Venture Partner Evaluation and Selection. CEVED/CEVAL Expert Systems Module. Developed at International Business Centers, Michigan State University.

Tanimoto, S.L. (1987). *The Elements of Artificial Intelligence*. Computer Science Press, Rockville, MD.

Tversky, A. (1972) Elimination by Aspects: A Theory of Choice. *Psychological Review* Vol.79 No.4 pp.281-199.

Tversky, A. and Kahneman, DE. (1986). Rational Choice and the Framing of Decisions. In *Readings in Uncertain Reasoning*, Ed: Shafer, G. and Pearl, J. Morgan Kaufmann:San Mateo, Calif. 1990. pp.91-104.

Von Neumann, J. and Morgenstern, O. (1947). *Theory of Games and Economic Behavior*. Princeton Press, Princeton, NJ.

Von Winterfeldt, D. and Fiscer, G.W. (1975) Multi-Attribute Utility Theory: Models and Assessment Procedures. *In Utility, Probability, and Human Decision Making*. ed. D.Wendt. Dordrecht. The Netherlands. pp.47-86.

Wang, H. (1960) Towards Mechanical Mathematics. *IBM Journal of Research and Development*. Vol.4 pp.2-22.

Whitney, K. (1991) PEREVAL: An Expert System for Expatriate Personnel Evaluation and Selection. CEVED/CEVAL Expert Systems Module. Developed at International Business Centers, Michigan State University.

Wiggens, N. and Kohen, E.S. (1971) Man vs. Models of Man Revisited: The Forecasting of Graduate School Success. *Journal of Personality and Social Psychology*. Vol.66 pp.675-685.

Winston, P. *Artificial Intelligence*. Addison Wesley Publishing Co. Copyright 1984.

Woods, W.A. (1975) What's in a Link: Foundations of Semantic Networks. In *Readings in Knowledge Representation*. Ed. Brachman and Levesque. Morgan Kaufmann Publishers, Inc. 1985. pp.217-242.

Wright, P.L. (1975) "Consumer Choice Strategies: Simplifying vs. Optimizing," *Journal of Marketing Research*. Vol.12 (Feb 1975), pp.60-67.

Yeoh, P.L. (1991) DISTEVAL: An Expert System for Distributor Evaluation and Selection. CEVED/CEVAL Expert Systems Module. Developed at International Business Centers, Michigan State University.