

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Designing Interactive Learning Environments

A dissertation presented in partial fulfilment of the requirements for the
degree of Doctor of Philosophy in Computer Science at Massey University
Palmerston North, New Zealand.

Raymond Henry Kemp

1995

Abstract

The trend towards teaching by facilitating learning rather than by direct instruction is an important one. As part of this movement, there is a growing interest in the concept of interactive learning environments (ILEs), where students learn by experimenting with a computer system that simulates some device, system or situation. Although ILEs can act as effective teaching aids they are time-consuming to create. In this thesis, principles that are useful for guiding the development of these systems are proposed, and design issues are explored.

In order to determine what the principles for development should be, the history of teaching by computer is reviewed, with an emphasis on interactive systems that have a learning rather than instructional bias. The important concepts of modelling, discovery learning and fidelity are examined in some detail.

One of the conclusions of the initial survey is that it is not feasible to think in terms of general design primitives that can be used for the development of all interactive learning environments. Since there is a diverse range of possible environments, two specific types are examined. In each case, a framework for design is proposed.

First, the teaching of procedural skills is considered. These skills include the ability to understand the operation of mechanical devices, to be able to carry out tasks with them, and to correctly assemble and dismantle pieces of equipment. Providing a realistic model which can include informative feedback is seen as important. It is demonstrated that a scheme adapted from AI planning can economically provide an appropriate level of fidelity for modelling device operation. A compatible notation for denoting tasks is also developed.

A methodology for the design of ILEs for teaching procedural skills is proposed, complete with graphical specification for both domains and tasks. It is envisaged that such a scheme would allow domain experts and teachers to take a full part in the design process, even if they are unable to write or understand computer programs.

The second kind of ILE considered involves the simulation of human behaviour. Two schemes for knowledge-based simulation are examined: one based on CYC and one on Schank and Abelson's behavioural model. The former is used to outline a system for simulating problem-oriented policing. The latter is extended to facilitate the

development of knowledge-based simulation teaching systems. This second scheme is then applied to the simulation of domestic disputes.

Since many of the problems of simulating real world events by computer software have yet to be solved, a full computer implementation is not yet a realistic proposition. Instead, the domestic disputes model is tested using a 'Wizard of Oz' approach. Results show that a scheme based on the model proposed is feasible, that subjects can successfully use such a system and that, as a result, they believe their understanding of the issues being presented is improved.

Acknowledgments

First and foremost, I would like to acknowledge the support of my main supervisor Mark Apperley, without whom this research would not have got off the ground. Besides steering me through the project he provided those two most invaluable commodities: time and space.

Next, I would like to express my gratitude to Ross St George, my second supervisor, who has always given me thoughtful feedback on my writing and proffered enthusiastic support throughout. Ross introduced me to the work of Pittsburgh's Learning Research and Development Center and helped me organise study leave in that city to talk to researchers at LRDC and Carnegie-Mellon. This has had a very great impact on my views and on the direction that the project has taken.

Thanks are also due to other people who have provided useful and often detailed comments on drafts of the thesis. These include Noel Craske, Geoff Cumming, Andre Everett, Gord McCalla, Jean McKendree and John Self. Without their input this document would have been completed several months earlier. Hopefully, however, the final product contains far fewer mistakes and is much more cohesive than would otherwise have been the case.

Lastly, I would like to thank various people who have contributed in other ways to the completion of this project and the content of the thesis: Shamus Smith, who has implemented some of the teaching system ideas, tested out the models and helped set up the Wizard of Oz experiment, as well as carrying out other smaller tasks too numerous to mention; John Andreae, who helped me get into shape material for the IJHCS paper, much of which has been used in Chapters 4 and 5; Dave Burns, who suggested and helped with the development of the POPIT police teaching system; Phil Carter, who conducted the domestic violence survey used as a basis for the model in Chapter 7; Stephen Cranefield, who validated the planning schema used in Chapter 4; the Computer Science secretarial staff, particularly Rosemary and Wendy, who were always very helpful; and most of all my family: Rebecca and Stephen for their support, and my wife Elizabeth who kept me focussed and sane.

Publications

The following publications are associated with the research presented in this thesis.

Journal Articles

Kemp, R. H. (1992). Intelligent Computer Assisted Instruction: a knowledge-based perspective. *Australian Computer Journal*, 24(3), 121-129.

Kemp, R. H. and Smith, S. P. (1994). Domain and task representation for tutorial process models. *International Journal of Human-Computer Studies*, 41, 363-383.

Conference Proceedings

Kemp, R. H. (1990). Knowledge-based methods in teaching and training. *Proceedings of Fourth New Zealand Conference on Expert Systems*, Palmerston North: Massey University.

Kemp, R. H. and Burns, D. (1992). POPIT: an intelligent teaching package for police training. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 359-370), Brisbane: University of Queensland.

Kemp, R. H. (1993). Knowledge-based simulation for teaching. *Proceedings of IEEE International Conference on Artificial Neural Networks and Expert Systems* (pp. 192-195), Dunedin, New Zealand.

Kemp, R. H. and Smith, S. P. (1994). Teaching procedural skills by computer simulation. *Proceedings of Second Singapore International Conference on Intelligent Systems* (pp. B73-78), Singapore.

Kemp, R. H. and Smith, S. P. (1994). Using planning techniques to provide feedback in Interactive Learning Environments. *Proceedings of Sixth IEEE International Conference on Tools with Artificial Intelligence* (pp. 700-703), New Orleans.

Smith, S. P. and Kemp, R. H. (1995). Efficient modelling of domains for computer tutoring systems. *Proceedings of 18th Australasian Computer Science Conference*, Adelaide.

Internal Reports

Kemp, R. H. and Smith, S. P. (1994). *Facilitating Feedback in Discovery Learning Systems*. Information and Mathematical Sciences Report No. 94/19. Massey University, Palmerston North.

Kemp, R. H. and Carter, P. (1995). *Knowledge-based Simulation Applied to Domestic Violence Scenarios*. Mathematical and Information Sciences Report Series No. 95/1. Massey University, Palmerston North.

Table of Contents

Chapter 1 Introduction	1
1.1 Background	1
1.2 Learning environments.....	3
1.3 The problem	5
1.4 Scope, goals and limitations.....	5
1.5 Related research	7
1.6 Outline of content.....	8
Chapter 2 Learning by Computer	10
2.1 Introduction	10
2.2 Programmed learning	11
2.3 Early large-scale projects	13
2.4 Intelligent Computer Aided Instruction	15
2.5 Intelligent Tutoring Systems	21
2.5.1 Structure of ITSs	21
2.5.2 Domain module.....	22
2.5.3 Student module.....	24
2.5.4 Tutorial module.....	25
2.5.5 Interface module.....	25
2.5.6 ITS appraisal	28
2.6 Research at LRDC.....	29
2.6.1 Knowledge structures.....	29
2.6.2 Problem solving methods.....	31
2.6.3 The acquisition of expertise	33
2.6.4 Facilitating learning.....	35
2.6.5 LRDC software	38
2.7 CMU research	40
2.7.1 The ACT models	40
2.7.2 CMU software.....	42
2.7.3 A comparison of LRDC and CMU research.....	45
2.8 Summary	46
Chapter 3 Key Concepts	48
3.1 Introduction.....	48
3.2 Knowledge issues.....	48
3.2.1 Expert systems.....	49
3.2.2 Situated cognition.....	56
3.3 Modelling and fidelity	62
3.3.1 Process models	63
3.3.2 Fidelity of models.....	71
3.3.3 Classifying fidelity	77
3.4 Discovery learning.....	82
3.5 Summary.....	87
Chapter 4 Domain and Task Representation for Tutorial Process Models	89
4.1 Introduction	89
4.2 Domain description	90
4.2.1 Transition nets	91
4.2.2 Strips.....	94
4.2.3 An alternative Strips formulation.....	96
4.3 Task specification	98
4.3.1 Domains and tasks.....	99
4.3.2 Domain and task separation	100
4.4 Summary.....	104

Chapter 5 Interactive Learning Environments for Procedural Tutors	106
5.1 Introduction	106
5.2 Implementing domain models	106
5.2.1 The gear assembly domain	106
5.2.2 The VCR domain	108
5.3 Implementing a task model.....	110
5.4 Graphical representation for a procedural task tutor	116
5.4.1 Procedural nets	116
5.4.2 Plan nets	122
5.5 Teaching knowledge.....	125
5.6 Summary.....	128
Chapter 6 Knowledge-based Simulation for Teaching	130
6.1 Introduction	130
6.2 Simulation.....	132
6.2.1 Classifying simulation.....	132
6.2.2 Knowledge-based simulation	134
6.2.3 Knowledge-based simulation in ill-structured domains.....	135
6.3 Scenarios.....	141
6.4 Modelling behaviour.....	143
6.4.1 Schank and Abelson's behaviour model.....	143
6.4.2 A behaviour model for teaching.....	148
6.5 Summary.....	154
Chapter 7 Towards a Knowledge-based Simulation Tutor	155
7.1 Introduction	155
7.2 POPIT: applying knowledge-based simulation	155
7.2.1 The domain.....	156
7.2.2 Outline of system	158
7.2.3 The adviser	160
7.2.4 The student model	161
7.2.5 The interface.....	162
7.3 Domestic disputes.....	163
7.4 Wizard of Oz simulation.....	171
7.5 Summary.....	174
Chapter 8 Conclusions and Further Work	177
8.1 Introduction	177
8.2 Summary of research.....	177
8.2.1 Procedural tutors	178
8.2.2 Knowledge-based simulation in ill-structured domains.....	180
8.3 Contributions of research	181
8.4 Future work	183
References	185

Appendix A Gear Mechanism Description	209
A.1 Introduction	209
A.2 Gear mechanism representation	209
A.3 Summary	209
Appendix B Domain Description using Plan Nets	215
B.1 Denoting VCR operation using plan nets	215
Appendix C Domestic Dispute Simulation	219
C.1 Introduction	219
C.2 Event breakdown	219
C.3 Profiles.....	227
C.3.1 Police officers	227
C.3.2 The batterer.....	230
C.3.3 The battered wife	233
C.3.4 Other actors.....	236
C.4 A sample scenario.....	236
C.5 Summary.....	239
Appendix D Administration of Wizard of Oz Experiment and Results	241
D.1 Introduction.....	241
D.2 Experimental set-up	241
D.3 Sessions, questionnaire and results.....	244
D.4 Conclusion	257

Figures

Figure 1.1	Development of ideas in thesis	9
Figure 2.1	SCHOLAR semantic network.....	15
Figure 2.2	Extract from Scholar dialogue	17
Figure 2.3	Extract from WHY dialogue.....	18
Figure 2.4	SOPHIE dialogue.....	20
Figure 2.5	Organisation of typical ITS.....	21
Figure 2.6	Section of WUSOR's Genetic Graph.....	23
Figure 2.7	Screen from RBT	26
Figure 2.8	Screen from ANGLE 3.0	27
Figure 2.9	Weak problem solving methods.....	32
Figure 3.1	The Alcohol Rule from MYCIN.....	51
Figure 3.2	A tutorial rule.....	54
Figure 3.3	Knowledge types used in MYCIN, NEOMYCIN and GUIDON.....	55
Figure 3.4	Purposes of CO ₂ removal system model.....	62
Figure 3.5	Qualitative models of processes	64
Figure 3.6	Concept map of baroreceptor reflex	66
Figure 3.7	Qualitative description of buzzer	70
Figure 3.8	A linear depiction of fidelity types	78
Figure 3.9	Conceptual simulation	79
Figure 3.10	Qualitative reasoning model	80
Figure 3.11	Components of idealized expert system	81
Figure 3.12	Idealized teaching system model	82
Figure 4.1	Gearing mechanism in Bluebird starter motor.....	91
Figure 4.2	Schematic diagram of starter motor gearing.....	92
Figure 4.3	Part of gearing assembly transition net.....	93
Figure 4.4	Transition in gearing assembly problem.....	93
Figure 4.5	Action table for gearing assembly problem.....	94
Figure 4.6	Analysis of possible feedback from Strips system	95
Figure 4.7	Overlapping subdomains	96
Figure 4.8	Three VCR operations	98
Figure 4.9	Specification for VCR task.....	100
Figure 4.10	Task overlay in the gearing assembly problem.....	101
Figure 4.11	Projection graph for gearing problem.....	102
Figure 4.12	Highlighted task routes in gearing problem projection graph	102
Figure 4.13	Crucial situated control rules for gear assembly task	103
Figure 5.1	Part of gear assembly tutorial session.....	107
Figure 5.2	Analysis of gear assembly dialogue.....	108
Figure 5.3	Essential predicates and parameters for VCR simulation	109
Figure 5.4	Action table for VCR and television operation.....	111
Figure 5.5	Dialogue from VCR simulation program	112
Figure 5.6	Part of projection graph for VCR simulation.....	112
Figure 5.7	Permissible operation sequences for playing a video tape.....	113
Figure 5.8	Situated control rules for playing a video tape	114
Figure 5.9	Part of dialogue for video tape playing task	115

Figure 5.10	Procedural net for setting up camera	116
Figure 5.11	Procedural net for taking photograph	117
Figure 5.12	Action table for high level description of photography task.....	117
Figure 5.13	Procedural nets for parts of the photography task	118
Figure 5.14	Extract of dialogue for program derived from procedural nets	119
Figure 5.15	Procedural net for gear assembly problem.....	120
Figure 5.16	Set of procedural nets for representing gear assembly problem.....	121
Figure 5.17	Plan net enablement and causal configurations	122
Figure 5.18	Plan net for hammering.....	123
Figure 5.19	Plan net for cleaning plugs.....	123
Figure 5.20	Action table for cleaning plugs.....	124
Figure 5.21	Plan net for cleaning plugs with radiator check added	124
Figure 5.22	Automated scheme for developing a procedural task tutor	129
Figure 6.1	Classification of simulation methods.....	132
Figure 6.2	Behaviour networks	139
Figure 6.3	Analysis of knowledge-based simulation packages.....	140
Figure 6.4	Hierarchy of events for knowledge based simulation.....	142
Figure 6.5	Expectancy Rules.....	144
Figure 6.6	Life themes.....	145
Figure 6.7	Interaction of situations, mental states and actions.....	149
Figure 6.8	Possible low-level feedback based on causal semantics.....	149
Figure 6.9	Profiles, goals, plans and actions	152
Figure 7.1	Incident-driven policing	157
Figure 7.2	Problem-oriented policing	157
Figure 7.3	Passage of information through POPIT	160
Figure 7.4	Information used by adviser.....	161
Figure 7.5	Life themes of actors.....	165
Figure 7.6	Actors' themes, goals, plans and actions.....	165
Figure 7.7	Beliefs, goals and actions for actors	166
Figure 7.8	Wife battering event: wife asked out	168
Figure 7.9	Wife battering event: asks husband if she can go	169
Figure 7.10	Wife battering event: evening of social event.....	169
Figure 7.11	Wife battering event: return from evening out	170
Figure 7.12	Wife battering event: the evening after.....	170
Figure 7.13	Wizard of Oz scheme.....	172
Figure 7.14	Screen from Wizard of Oz session.....	173
Figure 7.15	Sample dialogue from a role playing session	175
Figure 7.16	Questionnaire results summary	176
Figure A.1	Gear mechanism.....	210
Figure A.2	Gear mechanism structure.....	211
Figure A.3	Essential predicates for gear mechanism description	211
Figure A.4	Inessential predicates for gear mechanism description.....	211
Figure A.5	Actions for gear mechanism description.....	211

Figure A.6	Static axioms.....	212
Figure A.7	Gear domain description: casing subsystem.....	212
Figure A.8	Gear domain description: link gearing subsystem.....	213
Figure A.9	Gear domain description: drive mechanism subsystem.....	214
Figure B.1	VCR on/off and tape handling.....	215
Figure B.2	Rewinding and fastforwarding.....	216
Figure B.3	Recording.....	216
Figure B.4	Channel selection and television operation.....	217
Figure B.5	Stopping and starting.....	217
Figure C.1	Event breakdown in causal system.....	219
Figure C.2	Domestic violence events outline.....	220
Figure C.3	Stereotypes of Police Officers.....	228
Figure C.4	Personal qualities of different types of police officers.....	229
Figure C.5	Characteristics, aims and actions for uniform carrier.....	229
Figure C.6	Life theme components, etc for the crime fighter.....	230
Figure C.7	Typical beliefs, reasons, goals and actions for police officer.....	231
Figure C.8	Typical life theme profile for batterer.....	232
Figure C.9	Examples of life theme, goal, plan and actions for batterer.....	232
Figure C.10	Some beliefs, reasons, goals and actions for batterer.....	233
Figure C.11	Typical life theme profile for battered wife.....	234
Figure C.12	Beliefs, reasons, goals and actions for battered wife.....	235
Figure C.13	Beliefs for other possible actors in wife battering scheme.....	236
Figure C.14	Responses to initial request to go out.....	237
Figure D.1	Screen from Wizard of Oz session.....	243
Figure D.2	Wizard of Oz configuration.....	244
Figure D.3	Initial role playing system screens (1).....	245
Figure D.4	Initial role playing system screens (2).....	246
Figure D.5	Initial role playing system screens (3).....	247
Figure D.6	Sample role playing session (1).....	248
Figure D.7	Sample role playing session (2).....	249
Figure D.8	Sample role playing session (3).....	250
Figure D.9	Sample role playing session (4).....	251
Figure D.10	Sample role playing session (5).....	252
Figure D.11	Sample role playing session (6).....	253
Figure D.12	Sample role playing session (7).....	254
Figure D.13	Copy of the role playing system questionnaire.....	255
Figure D.14	Summary of results from role playing system questionnaire.....	256

Chapter 1

Introduction

1.1 Background

The use of the computer as a teaching aid has been an attractive possibility right from the early days of computing. Visionaries have pictured students sitting in front of screens and interacting with the machine in the way they might converse with a teacher. Scenarios could be presented, problems set, responses gauged, errors corrected and assessment conducted. This level of individual attention and continuous monitoring is often difficult for human teachers to give, particularly in the increasingly important area of distance education. Foremost, therefore, among the advantages of using a computer for teaching is that a high level of individualised attention can be provided.

The impact of one-on-one teaching can scarcely be overstated. Experiments by Bloom (1984) have shown what has become known as the '2-sigma' effect; that is, individualised teaching can raise a student's performance in a subject by 2 standard deviations. This would have the effect of lifting someone in the bottom 5% of the population up to half way, and the average student could be elevated into the top 5%. These results are corroborated in experiments performed by Reiser, Anderson and Farrell (Reiser *et al.*, 1985). The catch, of course, is that the 2-sigma effect is only currently applicable when the tutor is a person. Computers have not, as yet, been able to emulate the effectiveness of humans within the teaching process.

Before comparing the potential of the computer tutor with that of its human counterpart it is useful to consider what a good teacher can offer. S/he will have a good knowledge of the subject area but this will not be in the form of a set of stark facts. All the information will interlink and be tied up together in some cohesive fashion. Although the teacher will be aware of the underlying reasoning and logic behind what is being taught, s/he will not

necessarily present this to the student, perhaps preferring a simpler but less accurate explanation that the student can readily understand. In addition to knowing what is correct and how to present it, the teacher will also be aware of common pitfalls, how to detect them and how best to remedy them. S/he can adapt remedial procedures to the needs of the students and steer them back on track when they make unwarranted assumptions or develop misconceptions. In short, s/he has a thorough knowledge of the subject matter and of appropriate teaching techniques and is consequently able to adapt to the needs of the student and stimulate learning.

As will be shown in this thesis, much progress has been made towards providing these facilities on computers. Knowledge-based systems (KBS) research has introduced a way of incorporating *domain* (subject) and teaching knowledge into programs (Kemp, 1990; Kemp, 1992). Provision for commonly occurring mistakes made by students can be included, and procedures devised for remediation. Lastly, the level of knowledge and preferred learning style of each individual student can also be taken into account.

It is clear, however, that the features included in most current computer-based tutoring systems fall a long way short of providing a substitute for the human teacher. Furthermore, various critics have suggested that a computer can never entirely replace a person as a tutor. For example, computers, literally, cannot provide the human touch. This is not an inconsiderable problem since, for example, the teacher may use various means to break down the resistance to learning that the student may have, perhaps by using spontaneous humour or topicality. As du Boulay and Sothcott observe 'Good teaching borrows from many of our other human abilities and knowledge in the areas of, say, communication, interpersonal relationships and social skills' (du Boulay and Sothcott, 1987, p345). Weizenbaum (1976) takes a more extreme view and lists teaching among the set of activities that are essentially human ones and maintains that it is a gross offence to moral feelings to entrust them to a computer. Such observations about the limitations of a computer are reasonable but do not invalidate much of the work that has already been carried out in computer tutoring nor should they restrict future research.

Perhaps of more immediate concern to researchers is not producing systems that mimic the activities of human teachers but ones that complement their endeavours. In many well-run laboratories in schools and tertiary institutions, students work on computer projects but have human tutors to turn to for advice and to guide their progress. The computer can perform many tasks that a human would find difficult or impossible. It can simulate the activity of a nuclear power plant, quickly retrieve copies of relevant historical documents or, in the age of multi-media, present appropriate video or aural information

to the student as required. The main focus of the research presented here will be the computer as an aid to, rather than a substitute for, the class teacher.

1.2 Learning environments

Many researchers believe that students can learn particularly effectively in an environment where they feel in control and where they are manipulating some model of reality: see, for example, the work by the Cognition and Technology Group at Vanderbilt (CTGV, 1993), by Lajoie and Lesgold (1989) and by Spensley *et al.* (1990b). This move away from instruction to learning has been noted by many authors and in many contexts. Gagne (1985), for instance, stresses that what is important is creating a suitable environment for learning. Ryba and Anderson, who are looking specifically at the education of children, note that 'Teaching . . . means providing the kinds of environments where children can use the computer as a tool to enhance their intellectual self-management skills' (Ryba and Anderson, 1990, p1). Paul Ramsden, who specialises in tertiary education, observes 'Learning is best conceptualised as a way in which people understand the world around them, rather than a quantitative accretion of facts and procedures' (Ramsden, 1992, p82). Lastly, John Self, one of the foremost experts on the use of artificial intelligence (AI) techniques in teaching, states that 'Current ideas about tutoring system design are moving away from knowledgeable systems remediating ignorant students towards environments which support students in managing their learning activities' (Self, 1992, p703).

Part of the effectiveness of learning environments comes from the interest that the models themselves generate. The popularity of computer adventure games and video arcade games bears this out. People often like to explore some environment with its own, initially unknown, rules and idiosyncrasies and to find out for themselves what happens in different circumstances. They can get real satisfaction from discovering new ways of doing things and attempting to exert some control over what is happening.

The motivation provided by such systems has been documented by Malone (1981). He cites educational games that often develop some skill as a side effect: for example, 'Darts' (Dugdale and Kibbey, 1975) where the user has to burst balloons by estimating their positions on a number line. Here, the interest for the student is in bursting the balloons, and the skill, manipulating fractions, is acquired in passing.

Certainly, interest appears to be an important factor in learning, but other key issues are involvement and realism. It is believed by a number of researchers that knowledge is most easily assimilated and consolidated by involving the student in activities that closely

simulate the ones where the knowledge is to be applied: see, for example, Brown *et al.* (1989), Clancey (1992), and Lesgold *et al.* (1992a). However, in many instances, for simplicity or due to lack of resources, teaching is carried out at an abstract level. Consider the rules of the road that appear in booklets in most countries. Obscure pictures with arrows and dotted lines are often used to show the novice driver what to do in different circumstances such as crossing intersections. How much more impact would the lessons have if the student drove a simulated car and saw the effects of taking appropriate actions and, more significantly, inappropriate ones.

Closely allied to the idea of involvement is that of challenge. Performing trivial actions in some system, however ingenious, does not appear to be sufficient. For example, studies of the use of animations for teaching computer algorithms suggest that unless the students are actively trying to solve problems rather than getting the computer to demonstrate a method, their understanding is not significantly improved (Palmiter and Elkerton, 1991; Stasko and Badre, 1993).

A number of educational programs have been developed where users are placed in a situation in which they have to solve a non-trivial problem or learn to control events and where the skill being developed is similar to that being simulated. Examples include STEAMER (Hollan *et al.*, 1984) where the user has to learn to control a steam propulsion unit and RBT (Woolf *et al.*, 1987) which simulates the functioning of a pulp and paper mill that has to be kept running smoothly.

At the same time, advances in technology focus our attention on learning by doing in a literal sense. Flight control simulators (Rolfe and Staples, 1986) have been available for many years but have become increasingly effective so that now they can, in some cases, replace the actual experience in the air for the purposes of obtaining a pilot's licence. For example, Wright (1992) notes that pilots can now obtain their licenses for a Boeing 747 without having flown one. Multimedia, too, has had a great impact on teaching (Holzl and Robb, 1992), exploiting the stimulation that direct experience of activities gives as opposed to second hand accounts. Lastly, looking to the future, virtual reality systems (Rheingold, 1991; Andrew and Ellis, 1994) have the potential to enhance the learning experience even further.

In the midst of this great technological progress there are dangers. First, there is the possibility of progress being technocentric (Finkelstein, 1992). Electronic gadgets are feasible and available and so are employed even if they are not relevant. They can have an initial novelty value but may have little long term impact or teaching merit. Secondly, sophisticated software must be developed to control the hardware. This has not always

occurred. Reeves (1992), for example, notes that some multi-media systems are driven by software based on the programmed learning concepts of the 1960s and 1970s.

1.3 The problem

Although there has been a great deal of interest in providing learning environments in recent years, the emphasis has been on producing specific pieces of software or exploiting new technologies such as multimedia. In both cases the control is very much in the hands of the software development team. The system produced is often of a very high degree of sophistication but may not be amenable to change. There may be little scope for the educator to modify the software for his/her own purposes.

In early teaching systems an attempt was made to give the educator some control, with the development of so-called *authoring languages* (Barker, 1987). These were simplistic and sometimes tedious to use but at least gave the teacher some say in what was to be presented to the students and in what fashion. Similar tools are not generally available in learning environments. If the teacher is not closely involved in the design process or has no simple notation to describe requirements to the designer then there is a danger that resultant systems could be brittle and serve the user poorly. The lack of communication between designers and users is a well-known phenomenon in the commercial world. Tom Gilb, for example, in his book on software engineering management (Gilb, 1988) has emphasized the importance of close consultation throughout the design process.

The dilemma then, is that, at one extreme, there is simplistic software that can be readily understood by non-computer experts but which is very limited in its effectiveness for producing useful educational packages. At the other end of the spectrum, highly sophisticated software *can* be produced, but it is often difficult for teachers and subject area experts to understand the design techniques being used and therefore to be fully involved at all stages of the development process.

1.4 Scope, goals and limitations

The ultimate goal of the research initiated here is to provide generalized tools, or building blocks, that facilitate the production of learning environments for students. In particular, it is believed that systems produced should be fully interactive, with the computer providing guidance or even taking the initiative if necessary when the student needs help. This kind of *interactive learning environment* (ILE) is in contrast to the *reactive* learning environments (Brown *et al.*, 1975) advocated by some researchers, where the student learns only by trial and error during exploration of the system. Tools for building ILEs

should be available in a package in such a form that non-computing personnel are able to take a major part in the design of the software.

It is anticipated that a single scheme will be insufficient to cover the whole breadth of possible ILEs and that many different kinds of primitives and associated pieces of software will be needed. Within this thesis, an initial framework will be proposed and two schemes will be analysed in detail.

The investigation of knowledge-based methods appears to be a good starting point. These have been successfully used for system development, most notably in expert systems (Jackson, 1990) but also in areas such as natural language processing (Barnett *et al.*, 1990) and Computer-Aided Software Engineering (CASE) tools (Tuthill, 1990). Knowledge and education are inextricably linked in people's minds. It is generally assumed that education somehow has the aim of producing more knowledgable people, although the way or ways in which this might occur are open to debate. The question then arises, can the tools that have been used in knowledge-based software be adapted for use in teaching? In knowledge-based systems, production rules (Hayes-Roth, 1985) and frames (Fikes and Kehler, 1985) are commonly employed. If it transpires that these tools are not suitable then, at least, the overall approach of using appropriate primitives for describing a domain and tasks within the domain can be investigated.

To avoid the danger of proposing techniques and applying them inappropriately, it is important to begin with a general survey to see how computers have been used in education, both successfully and unsuccessfully, and to carefully examine what educationalists currently perceive as the most effective ways of learning. Fruitful themes and patterns that recur in the literature need to be identified and noted.

Next, the role of knowledge needs to be examined. Key questions to be addressed include: to what extent have knowledge-based methods been used so far, to what extent have they been successful and why? A more general question, however, is what kinds of knowledge can be identified and distinguished? Having done that we can see how teaching and related packages fit into the scheme of things. What kinds of knowledge are represented on the computer and how faithfully? To what extent are these considerations important in software development for teaching purposes? A preliminary aim then is to attempt to answer questions of this nature, in order to point the way to practical considerations in the development of learning environments.

The background work provides guidelines for the designing of systems using high-level primitives. Based on these ideas, suitable building blocks have to be determined. It

seems unlikely that a totally new representation would be necessary since so much work on modelling techniques has been carried out in AI generally and in KBS in particular. Suitable modelling tools are probably already available. It is a matter of choosing the most appropriate ones.

Having selected suitable primitives they then have to be tested. It is important to look at diverse kinds of domains to check the applicability of such methods to a range of problems.

In summary, the subject area of the project is educational software design. Of particular interest is the concept of the interactive learning environment for helping students to become familiar with a domain. The main goal is to explore the possibility of providing high-level building blocks for constructing such environments, using concepts from knowledge-based systems and other branches of AI. Ideas need to be tested in different contexts, examining what facilities can be provided for aiding the learning process. The building and testing of complete systems is not within the scope of the project. Implementations would involve solving technical problems that can be addressed in later projects. Before such implementations can be started, frameworks have to be put into place to facilitate the construction of learning environments. The development and testing of such frameworks is the main concern of this project.

1.5 Related research

The idea of simplifying the production of sophisticated teaching software is not new. For example, Towne *et al.* (1990) describe a package called the Intelligent Maintenance Training System that allows a non-programming domain expert to produce an interactive graphical model of a complex device. A scene editor is used for composing diagrams of objects and connections. Once the simulation has been constructed, the learner can experiment with the system, trying out a variety of procedures and observing what happens when faults occur. The system is inspired by STEAMER and other graphical simulations, and has similar advantages and disadvantages. The user can readily observe what is happening during the operation of the system, aided by graphical clues, but the only feedback given is that provided by the simulation activity itself.

Fairweather and his co-workers (Fairweather *et al.*, 1992), have also recognised the dilemma of educational software designers who have to choose between using simplistic but easy to implement authoring systems, and producing elaborate but time consuming learning environments. They seek to bridge the gap by using a standard authoring package in conjunction with their own software for providing additional guidance. The

authoring system presents information to the student whose responses are collated and analysed to allow the provision of feedback at the end of the session. It is an ingenious approach but, as the authors observe, it is not ideal since the student does not have access to immediate, informative guidance during the session itself.

An interesting approach is used in the EXCALIBUR project (Richards *et al.*, 1988; Craske *et al.*, 1991). The EXCALIBUR team has investigated the use of an object-oriented scheme for building a computer tutoring systems shell. The idea is that a library of modules for carrying out various parts of the teaching task should be available and that these modules should be automatically chosen and adapted to provide the best educational approach for any given student. In this way, a flexible teaching system that allows the knowledge organization to be determined at run-time can be provided.

A modular approach is also taken by Beverley Woolf and her associates (Woolf *et al.*, 1988; Woolf, 1992). Both the development of 'tutoring primitives' (lessons, topics, and presentations) and tools for discourse knowledge are discussed. TUPITS (Tutorial discourse Primitives for Intelligent Tutoring Systems) is an object-oriented representation language that provides a framework for defining primitive components of a tutorial discourse interaction. The objects processed by TUPITS include lessons, misconceptions, examples and questions.

The research described in this thesis aims, like all these other projects, to make some headway towards an authoring system for intelligent tutors. As in each of the other projects it is recognized that progress can only be made in specific kinds of domain and using specialised tools. A particular concern of this project is to investigate schemes that permit informative feedback to be given to students as they interact with the system.

1.6 Outline of content

The layout of the thesis is illustrated in Figure 1.1. The boxes show the individual chapters and the arrows indicate the logical flow between them. The shaded areas denote the type of activity involved.

Chapter 2 sets out what has previously been achieved in the area of computer tutoring, examining the contribution of AI in particular. In Chapter 3, the cornerstone principles used as a basis for the research carried out in the current project are examined in some detail. In particular, the concepts of modelling, fidelity and situated learning are discussed. Chapters 1, 2 and 3 constitute the background section in which the main

ideas are described and developed. An attempt has been made not just to provide a literature review but to give a specific knowledge-focussed appraisal and analysis.

In Chapter 4, the idea of high-level design for tutorial systems is applied to learning in procedural domains. Various techniques are examined to assess their suitability as building blocks for such a system. Chapter 5 illustrates the application of these techniques to specific problems including teaching students how to assemble gearing mechanisms, and how to operate a video cassette recorder.

A completely different kind of problem is considered in Chapter 6: teaching in so-called ill-structured domains. Again, an attempt is made to produce high-level design primitives. The use of knowledge-based simulation is assessed in this context, and various approaches to building a system are considered. In Chapter 7 these methods are applied to problem-oriented policing and the representation of domestic disputes.

Lastly, Chapter 8 summarises what has been achieved and proposes future developments and extensions.

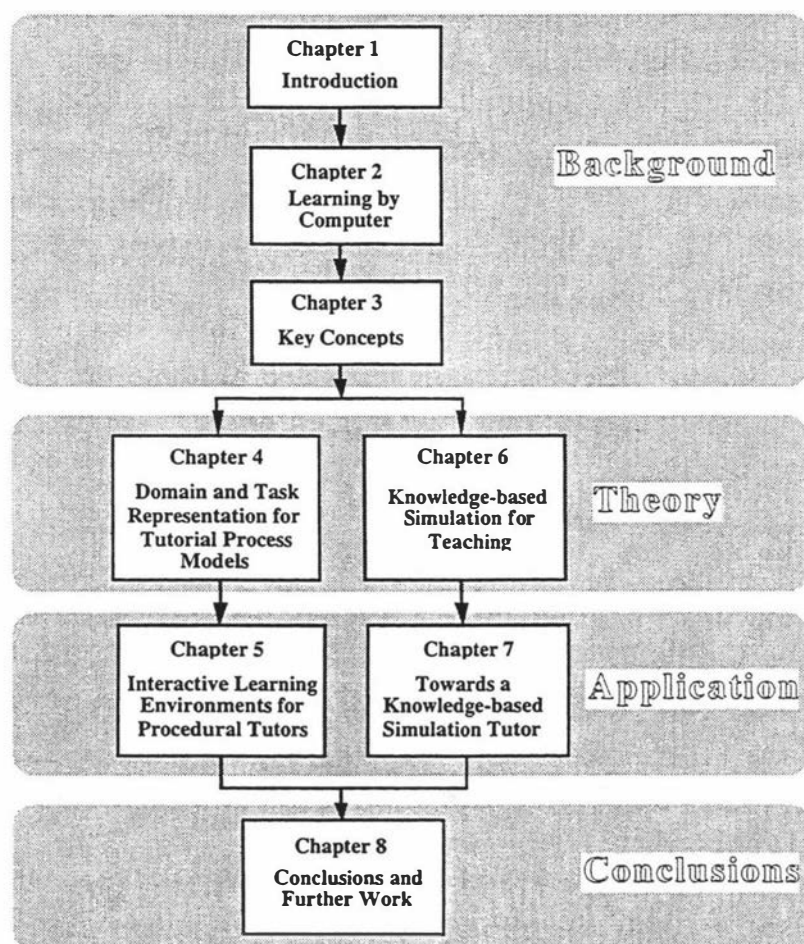


Figure 1.1 Development of ideas in thesis

Chapter 2

Learning by Computer

2.1 Introduction

A great deal of research and development has been carried out in the area of learning by computer. The literature is similarly voluminous. In this chapter an attempt will be made to summarise previous work and to set the scene for the research presented in the body of the thesis. Educationalists, computer scientists, psychologists, cognitive scientists and researchers in artificial intelligence have all had some input into educational software development, sometimes working together, but, at other times, moving in different directions. A survey could be based around any one of these disciplines but, commensurate with the general approach of the thesis, each of these perspectives will be considered separately only when appropriate. What is more important is the development of ideas, from whatever source.

Some of the early work will inevitably appear crude and simplistic compared with what is possible nowadays, particularly in terms of hardware and software technology. It would be easy to focus on its flaws and limitations. Early researchers were very much trailblazers and, as such, made many mistakes. However, although not many of those initial techniques remain in common usage today (and some that do, should have been long ago discarded) their developers often displayed an enormous vision. Their implementation methods may have been necessarily rudimentary but their ideas are still relevant today and are often incorporated into current software, albeit in a more polished form. It is these positive features of early systems that will be emphasized although the drawbacks of each approach will also be pointed out where appropriate.

In keeping with the overall goals of the project, the use of AI methods will be examined carefully and important research on knowledge and learning will be highlighted. Of

particular significance with regard to the latter point is an appraisal at the end of the chapter of work done in Pittsburgh at the Learning Research and Development Centre, and at Carnegie-Mellon University.

2.2 Programmed learning

Much of the early work in computerised teaching systems has been well documented by O'Shea and Self (1983). Programmed learning methods were based on the ideas of behaviourism as advocated by Skinner. He observes that 'if the occurrence of an operant is followed by the presentation of a reinforcing stimulus, the strength is increased' (Skinner, 1938). The educational aim, therefore, is to change students' behaviour so that it is acceptable in the given context. Skinner suggests that this is all we can know, or need to know, about their state of knowledge.

In order to achieve this, programs were developed which ask specific questions. If the student gets the right answer (gives the appropriate response) then they get some positive feedback and move on to the next question. In the pure form of this approach (what O'Shea and Self call *linear programs*) it is assumed that the student will always get the right answer because of appropriate priming. *Branching programs* brought the technology back into the real world by allowing the program designer to cater for incorrect responses from the user. If the student inputs an incorrect answer then appropriate remedial action can be taken by branching to a different part of the program. Remediation may involve going over earlier units again or, perhaps, asking new questions about more elementary material.

Such packages provided the first step on the path to autonomous teaching systems. For the first time students could work by themselves at their own pace and get some remedial attention if their answers were wrong. The novelty of using 'new technology' must have also been appealing.

One of the main advantages as far as educators were concerned is that they could control the content of lessons. Packages on particular applications were available but if the teacher wanted to include some material on a topic of their own then this could be done by using authoring languages (Barker, 1987). The success of such an approach can be gauged by the large number of authoring systems available in the 1980s. Barker describes over 60 courseware development tools and the overwhelming majority of them have a similar form. The basic unit of presentation is the 'frame', a screen of information that may be descriptive or may contain a question. As a result of the student response to

a screen the system may move to the next frame, repeat a sequence of frames or jump to another frame elsewhere in the sequence.

At the heart of such systems, then, are the procedural programming operations of sequencing, repetition and conditional transfer. These concepts are not intrinsically difficult for most people to understand, and, once mastered, the teacher can use them to produce quasi-procedural programs controlling the order of presentation of the material depending upon the performance of the user. Unfortunately, as generations of novice programmers have discovered, although it is necessary to understand sequence, repetition and conditions in order to write procedural programs it is not sufficient. The number of alternative paths through the material that the learner may follow rapidly gets out of control and it becomes, for all practical purposes, impossible to determine what knowledge the student has when they have reached particular points in the program. Structured programming techniques can alleviate the problem but providing these in a form that non-programmers can use effectively is difficult. The inclusion of embellishments such as colour, video and animation do not solve the problem but merely obscure it.

There is still debate, therefore, regarding the contribution of authoring systems to educational teaching. On the one hand, authors such as Kulik, Kulik and Cohen (1980) demonstrate positive gains from the use of drill and practice Computer Assisted Instruction (CAI). Others, such as O'Shea and Self (1983), see no benefit of methods based on what they regard as discredited stimulus-response ideas. As noted here, even if the approach is sound, building bug-free software is not a trivial task, even for experienced programmers.

Whatever opinion is held concerning the usefulness of authoring systems, it is undeniable that involving the teacher directly in the production of tutorial material can be of great benefit, giving them some control over what is taught and how the material is to be presented. At the very least, as programmers in many other disciplines have found, the attempt to write software compels the author to acquire a clear understanding of the process being modelled. This side benefit may not be insignificant. If the teacher is forced to consider what the student is supposed to learn, the order in which concepts should be presented, and what tests can be carried out in order to check that the learning has taken place then, in certain circumstances, the quality of tutoring that can be provided may be improved by the exercise.

2.3 Early large-scale projects

One of the initial drawbacks of authoring software was the fact that, in many cases, the courseware produced was amateurish and ineffective. Two high profile American projects attempted to be more professional about the production of material, and, for a while at least, appeared to get closer to the goal of computer aided learning.

PLATO (Programmed Logic for Automatic Teaching Operations) was conceived at the University of Illinois in 1960. By 1965 it had evolved into a highly interactive timesharing system using touch screens, high quality graphic displays and animation. With the aid of government funding it eventually grew into a system with terminals located at about 140 sites and with 8000 hours of instructional material contributed by over 3000 authors (O'Shea and Self, 1983). Its (retrospectively defined) goals are listed by Alpert (1975). These included 'to demonstrate the technical feasibility of a truly novel computer-based education network system' and 'to prove that the system is manageable, economically viable, and capable of serving a variety of institutions at any educational level' (Alpert, 1975, p181). Other goals are at a similarly administrative level, not considering educational methods or computational techniques.

The technological emphasis was not entirely fatuous. The goal of getting one terminal per student was laudable, rather than having students share a machine. Also, the simplifying of input using touch screens made it much easier for the students to interact in a natural fashion. Lastly, the use of animation and graphics provided a much needed stimulus, particularly to young students, and often overcame their reservations about using the computer, a device that, thirty years ago, was frequently regarded with some trepidation.

Very little control was exercised over the software developed for PLATO. The general guidelines for authors were that they should produce units that exploited the technical devices available such as touch screens and animation. This approach led to the development of material of uneven quality. Some of it was very good (and has been adapted for use on present day micros) whereas other material was poor and ineffective. It was uniformly difficult and time-consuming to write and surveys showed that teachers often found it hard to use.

In the mid 1970s, Mitre Corporation developed TICCIT (Time-shared Interactive, Computer Controlled Information Television). Seeking to learn from the mistakes of PLATO the production of educational material for TICCIT was closely monitored. Each package was produced by a team that comprised an instructional psychologist, a subject

matter expert, an instructional design technician, an evaluation technician and a packaging specialist.

A premise used by TICCIT developers was that the effectiveness of a particular learning strategy is independent of subject-matter. Consequently, an attempt was made to divorce course content from both computer programming and teaching strategy (O'Shea and Self, 1983). Whether this separation can realistically be made is still an open question. Another innovation was the attempt to give the student as much control as possible over the conduct of the lessons. Although this was at a rather crude level of being able to select objectives, request help, ask for examples, etc, it is now generally acknowledged that it is important to give the student a feeling of control over the course of the session.

Educationally, neither TICCIT nor PLATO had any real guiding principles. PLATO material was generally technocentric and aimed to stimulate interest by novel hardware, graphics and animation. TICCIT involved production of material by committee and aimed to incorporate the strengths of experts from various disciplines. Both projects had some success and, for a while, each had a high profile. Their principal achievement, perhaps, was that they raised the awareness of the public, the authorities and teachers to the potential of computer-aided learning.

Assessment of the effectiveness of these projects was inconclusive. According to O'Shea and Self (1983, p96), surveys on PLATO showed there was no significant impact on performance or reduction in drop-out rate although students enjoyed using the material. Students who completed TICCIT programmes generally attained higher post test scores than control groups, but the completion rate itself was low (O'Shea and Self, 1983, p92). In mathematics, for example, the completion rate was 16% compared with 50% for non-TICCIT courses.

It was not the performance obtained, however, but the costs involved that signalled the demise of these projects. A large expenditure of money and effort to computerize any system is normally anticipated, but what really branded PLATO and TICCIT as failures was the realization that the amount of effort required to produce *further* software would not be significantly reduced. That is, for each new package, the developers would more-or-less be starting from scratch.

There are a number of reasons for this state of affairs, two of which are relevant to this thesis. Firstly, many of the early developers had little concept of system design, which at that time was in its infancy. Secondly, teaching is a much more complex activity than was then perceived to be the case. It has been generally accepted more recently that any

system that is going to take over a significant portion of the teaching role must incorporate principles of artificial intelligence.

2.4 Intelligent Computer Aided Instruction

One of the first researchers to advocate the use of AI in teaching systems was Jaime Carbonell. His milestone paper (Carbonell, 1970) precipitated concerted activity in the area which continues until this day. Although he did not use the term, the field that he opened up is often called Intelligent Computer Aided (or Assisted) Instruction (ICAI). Carbonell maintains that any program that is going to teach a student must have some knowledge of the subject area or domain of interest rather than just storing answers to specific questions. As he observes 'In his teaching process, a human teacher is not reciting specific questions, but he is utilizing and processing knowledge he has stored in the form of a semantic information structure' (Carbonell, 1970, p195). Carbonell provides this knowledge by adapting the concept of the *semantic network* that Quillian (1969) had used for natural language comprehension.

The subject that Carbonell chose for his research was South American geography. His program, known as SCHOLAR, stores facts, concepts and procedures in this domain. A simplified version of the semantic network, taken from O'Shea and Self (1983), is reproduced as Figure 2.1.

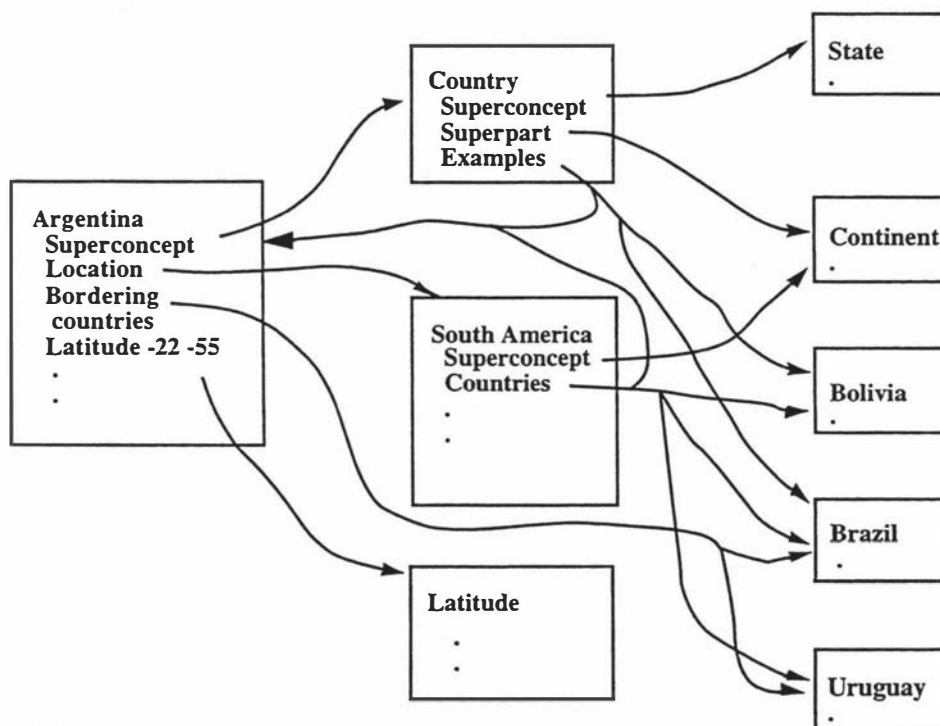


Figure 2.1 SCHOLAR semantic network (O'Shea and Self, 1983, p130)

The information stored in such networks allows the system to answer a range of questions of different kinds. Instead of questions and answers being pre-stored the system can work them out as necessary from its internal knowledge of geography contained in these data structures. In addition to facts and relationships, heuristic procedures are available for determining information that cannot be specifically inferred from the networks. For example, a procedure is available that estimates climate given local conditions such as latitude and altitude.

In addition to being able to answer direct questions such as 'What is the latitude of South America?' SCHOLAR can deal with queries like 'Give me some information about Brazil'. It can then trace through the semantic network listing facts about the country, applying the heuristic that the further away from the Brazil node the information is stored the less relevant it is likely to be.

Although the idea that information for teaching could be stored in a semantic net was a good one, the program has no underlying teaching strategy other than that the student should learn by asking and answering questions. Sessions are undirected and topics are generated at random leading to a lack of cohesion. The mixed initiative interactive style is typified by the extract shown in Figure 2.2.

Jaime Carbonell died in 1973 but his work was continued by others including Allan Collins and Albert Stevens. One of the main lines of research aimed to make the dialogue in SCHOLAR more directed. To this end Collins, Warnock and Passafiume (1975) studied protocols of dialogues between human tutors and students to determine appropriate heuristics for directing the dialogues during sessions. Central to the approach is the incorporation of an agenda that is used to determine what topics and sub-topics should be covered and when.

The model was further enhanced by defining an educational basis for the tutorial approach: the Socratic method. In this approach, originally used by the Ancient Greeks, the tutor seeks to teach students by asking them carefully formulated questions that encourage them to make appropriate deductions. Collins (1977) endeavours to pinpoint the central goals and tenets of the Socratic method. He gives the main goals as attempting to force students to reason for themselves, getting them to derive general principles and encouraging them to apply these principles to new situations. He then shows how these goals can be achieved by using a computer program.

SCHOLAR : THE POPULATION IN CHILE IS APPROX 8500000. PEOPLE.
TRUE OR FALSE?

Student : TRUE

SCHOLAR : YOU ARE CORRECT.

USE ONE OF THE FOLLOWING:

SUCRE
FRENCH
ARGENTINA
SPANISH

TO ANSWER THE QUESTION:
WHAT IS THE LANGUAGE IN CHILE?

Student : SPANICH

SCHOLAR : YOU SHOULD HAVE WRITTEN "SPANISH".
VERY GOOD.

PLEASE INDICATE IF THE FOLLOWING STATEMENT IS CORRECT
OR INCORRECT:

THE AREA OF PARAGUAY IS APPROXIMATELY 47432 SQUARE
MILES.

Student : WHAT IS THE AREA OF BRAZIL?

SCHOLAR : THE AREA OF BRAZIL IS APPROX 3600000. SQUARE MILES.

NOW ANSWER THE QUESTION YOU DIDN'T ANSWER BEFORE.

Student : INCORRECT

SCHOLAR : VERY GOOD.

THE AREA OF PARAGUAY IS APPROX 157000 SQUARE MILES.

APPROX WHAT IS THE POPULATION IN BRAZIL?

Student : 60000000.

SCHOLAR : APPROXIMATELY TRUE, BUT A MORE EXACT VALUE IS 70000000.

Student : Q/A

(student changes mode to question/answer)

Student : WHAT IS THE PRINCIPAL RELIGION IN URUGUAY?

SCHOLAR : THE PRINCIPAL RELIGION IS CATHOLICISM.

Student : THE PRINCIPAL RELIGION IN PERU IS - - - .

SCHOLAR : I'M AFRAID I CAN'T ANSWER THAT.

Student : IS IT TRUE THAT THE AREA OF URUGUAY IS 70000?

SCHOLAR : THAT'S ABOUT RIGHT. THE EXACT ANSWER IS 79000.

Student : THE PRINCIPAL RIVERS IN ARGENTINA ARE - - - .

SCHOLAR : THE PRINCIPAL RIVERS IN ARGENTINA ARE THE RIO DE LA
PLATA, THE PARANA, THE URUGUAY RIVER, THE SALADO, THE
PARAGUAY RIVER, AND THE PILCOMAYO.

Figure 2.2 Extract from Scholar dialogue (Carbonell, 1970, p192-193)

In a related project, Stevens and Collins (1977) investigate the use of the Socratic method in a meteorological tutor called WHY. The main aim of SCHOLAR is to teach concepts and relationships and so the semantic net is a convenient form of representation. For WHY, which seeks to teach the student about the processes that lead to rainfall, another type of structure that could better represent these processes was needed. Stevens and Collins recognised the significance of the parallel research that Schank and Abelson (1977) were carrying out on narrative understanding, particularly the concept of a *script*. Scripts give a method of representing stereotypic sequences of events including the actors, the scenes and the likely actions. Stevens and Collins adapted this idea to store knowledge about rainfall including temporal and causal relationships.

In WHY, scripts are used as a basis of the questions to be asked and to assess the students' responses. The scripts are structured into hierarchies where general scripts give an overall view and the lower level ones supply detailed information about causal chains. Figure 2.3 illustrates the more natural dialogue achieved in this system compared with the early version of SCHOLAR.

WHY:	Do you think the Amazon jungle has heavy rainfall or light rainfall? (Rule 1: Ask for a prediction about a particular case.)
Student:	Heavy rainfall
WHY:	Why does the Amazon have heavy rainfall? (Rule 2: Ask for a prior cause.)
Student:	Because it's near the Atlantic
WHY:	Yes, the Amazon jungle is near the equatorial Atlantic. How does that affect rainfall in the Amazon jungle? (Rule 3: Ask for intermediate cause.)
Student:	The water in the current is warm . . .
WHY:	Do you think that any place with mountains has heavy rainfall? (Rule 6: Form a general rule for an insufficient cause.)
Student:	Yes
WHY:	Southern California has mountains. Why doesn't Southern California have heavy rainfall? (Rule 7: Pick a counter-example for an insufficient cause, and Rule 2: Ask for prior causes.)

Figure 2.3 Extract from WHY dialogue: Wenger (1987, p42)

The SCHOLAR and WHY projects were ambitious but both lacked some important aspects of human teaching, notably the ability to pitch material at a level appropriate to the student's knowledge and to deal effectively with misconceptions. Most importantly, it is very difficult to conduct a sensible conversation on any topic without a great deal of real world knowledge. The technology was not available in the 1970s to deal with these problems but, even so, these programs demonstrated that by storing information in a coherent and integrated form it can be used in a very flexible way.

A third influential project during the 1970s was SOPHIE (Brown *et al.*, 1975; Brown *et al.*, 1982). SOPHIE (SOPHisticated Instructional Environment) was the name given to a series of programs developed between 1973 and 1978 for teaching electronic troubleshooting, initially at the University of California and later at Bolt, Beranek and Newman Inc. (BBN) in Cambridge, Massachusetts.

Rather than just getting students to find faults in given circuits, the project team aimed to improve their conceptual understanding so that they could develop diagnostic procedures of their own, assimilate new information from technical manuals and troubleshoot unfamiliar equipment. A sample dialogue from an early version of the program is shown in Figure 2.4. In this example, the student is presented with a circuit that is not working and checks various components before suggesting possible faults. The system provides results of the tests and helps the student think through the ramifications of his/her hypotheses. As can be seen, the interaction style is very flexible.

Drawing on experience from SCHOLAR and WHY, the developers store knowledge in structures inside the computer. Unlike these earlier programs, however, knowledge is not just stored in a single form. SOPHIE uses three methods of representation: a simulation model of the domain, procedural specialists that contain logical skills and semantic nets for storing time-invariant factual knowledge.

One of the strengths of the later versions of SOPHIE is the modular design employed. An expert troubleshooter capable of locating and diagnosing a wide range of faults in electronic circuits provides the domain expertise. A coach monitors the student's attempts at diagnosis and fault finding, intervening as necessary. Finally, a sophisticated natural language processor is used in the interface. This separation of the different activities of the tutor was something that was being carried out on a regular basis in Britain at the time but, in America, the general emphasis was on domain representation rather than looking at the overall pedagogical strategy. The two approaches eventually converged and led to what are now generally called *intelligent tutoring systems* (ITSs).

SOPHIE : WOULD YOU LIKE TO START WITH A FAULTED INSTRUMENT WHICH NEEDS FIXING?

Student : YES

SOPHIE : THE INSTRUMENT HAS A FAULT AND ITS PRESENT CONTROL SETTINGS ARE:

CC 1.0 (CURRENT CONTROL-FROM 0.0 (MIN CURRENT) TO 1.0)

CR HIGH (CURRENT RANGE SWITCH, HIGH=1 AMP, LOW=0.1 AMP)

LOAD 20 (LOAD RESISTANCE IN OHMS)

VC 1.0 (VOLTAGE CONTROL-FROM 0.0 (MIN VOLTAGE) TO 1.0)

VR HIGH (VOLTAGE RANGE SWITCH, HIGH=30 VOLTS, LOW=10 VOLTS)

Student : WHAT IS THE OUTPUT?

SOPHIE : THE OUTPUT VOLTAGE IS 11.7 VOLTS.

Student : WHAT IS IT IN A WORKING INSTRUMENT?

SOPHIE : IN A WORKING CIRCUIT THE OUTPUT VOLTAGE IS 19.9 VOLTS. THAT VALUE IS INCORRECT.

Student : WHAT IS THE BASE EMITTER VOLTAGE OF THE VOLTAGE LIMITING TRANSISTOR?

SOPHIE : THE BASE EMITTER VOLTAGE OF Q5 IS -0.037 MILLIVOLTS.

Student : WHAT ABOUT FOR THE CURRENT LIMITING TRANSISTOR?

SOPHIE : THE BASE EMITTER VOLTAGE OF Q6 IS 0.349 VOLTS.

Student : WHAT SHOULD IT BE?

SOPHIE : IN A WORKING CIRCUIT THE BASE EMITTER VOLTAGE OF Q6 IS 0.593 VOLTS.

Student : REPLACE Q6

SOPHIE : I AM GOING TO ASK YOU SOME QUESTIONS ABOUT HOW Q6 IS FAULTED. ARE ANY JUNCTIONS SHORTED?

Student : I DON'T KNOW

SOPHIE : ARE ANY TERMINALS OPEN?

Student : YES

SOPHIE : WHICH TERMINALS ARE OPEN?

Student : THE BASE

SOPHIE : Q6 IS NOT FAULTED IN THAT MANNER. Q6 NOT REPLACED.

Student : IS IT POSSIBLE THAT Q6 HAS AN OPEN BASE?

SOPHIE : THAT FAULT DOES NOT EXPLAIN ANY OF THE SYMPTOMS YOU OBSERVED BUT IT IS CONSISTENT WITH THE VALUES FOR THE BASE EMITTER VOLTAGE OF Q5 AND THE VOLTAGE AT N1.

Student : WHAT IS THE OUTPUT OF THE CONSTANT CURRENT SOURCE?

SOPHIE : THE OUTPUT OF THE CONSTANT CURRENT SOURCE IS -0.59 MILLIAMPS.

Student : WHAT SHOULD IT BE?

SOPHIE : IN A WORKING CIRCUIT THE OUTPUT CURRENT OF THE CONSTANT CURRENT SOURCE IS -0.6 MILLIAMPS.

Student : COULD THE PROBLEM BE THAT THE BASE OF Q3 IS OPEN?

SOPHIE : HMMM, LET ME THINK ABOUT THAT FOR A MINUTE.

UNDER THE SETTINGS:

CC=1.0, CR=HIGH, LOAD=20, VC=1.0, VR=HIGH

THE FAULT WHICH YOU PROPOSED WILL PROPAGATE. LIKELY FAULTS WHICH MIGHT OCCUR ARE:

R22 OPEN

WOULD YOU LIKE ME TO CONTINUE TESTING YOUR HYPOTHESIS IN A CIRCUIT IN WHICH R22 IS OPEN ALSO?

Student : NO

Figure 2.4 SOPHIE dialogue (Brown *et al.*, 1975, pp679-683)

2.5 Intelligent Tutoring Systems

In the same way that Carbonell's 1970 paper was a milestone for research into the use of AI in tutoring systems, a paper written by British researchers Roger Hartley and Derek Sleeman (Hartley and Sleeman, 1973) pointed the way towards a general structure for such automated tutors. They proposed that a distinction should be made between domain expertise, student model, teaching methods and the interface. The general acceptance of the concept of an ITS can, perhaps, be traced to a 1979 issue of the *International Journal of Man-Machine Studies* that contains a number of papers on the topic from researchers on both sides of the Atlantic. The papers later appeared in a well-known book (Sleeman and Brown, 1982).

In one of these papers, Sleeman (1982) defines an intelligent tutoring system as a computer-based system intended to provide effective, appropriate and flexible instruction through the application of artificial intelligence techniques and knowledge representations. Under that definition SCHOLAR, WHY and SOPHIE would all qualify. However, the term is often used to distinguish such systems that have separate domain, student, teaching and interface modules and this is the definition employed here.

2.5.1 Structure of ITSs

There are marginal differences between researchers' perceptions of how an ITS should be organised. A typical model is shown in Figure 2.5.

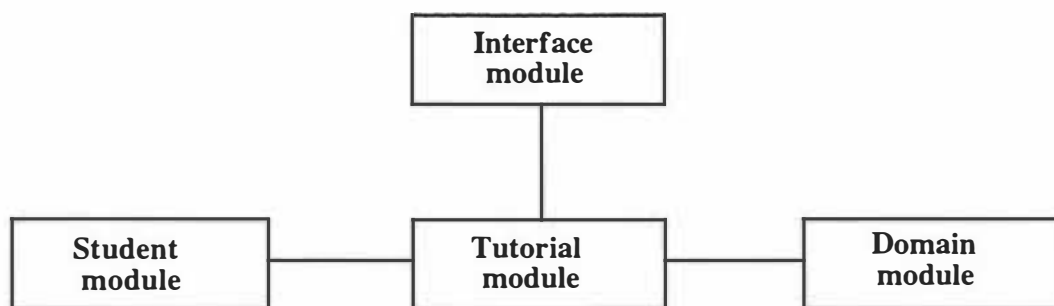


Figure 2.5 Organisation of typical ITS

The interface module converts the user input into a form the system can easily process and *vice versa*. The user may communicate using a restricted form of natural language, WIMPS (Windows, Icons, Mouse, Pull-down menus), multi-media or any other suitable means. The student module contains relevant information relating to the specific student currently using the tutor. User background, level of knowledge, preferred learning style,

misconceptions, capabilities and so on are all examples of the kind of information that may be stored in some form in this module. The tutorial expert embodies pedagogical knowledge and accesses information from the student module in order to determine the best approach to teaching the user. It also consults the domain module which will contain knowledge of the domain being considered.

The above model is idealized and few ITSs will explicitly contain all of these modules. Sometimes an extra module is included that contains the specific course material to be taught. In other systems, particular components are not present at all. For example, Newman (1989) suggests that a student module is unnecessary in some circumstances. In general, then, this four module scheme indicates to developers how they *might* structure their system. Besides aiding the organization of tutors this model focusses the attention of researchers. Addressing all the problems that occur when developing an ITS for a particular domain is hard. As Woolf notes 'The bottom line is that intelligent tutoring systems are AI complete, that is, solving intelligent tutoring problems requires solution of nearly all the problems of artificial intelligence' (Woolf, 1988a, p5). This may be so, but at least by concentrating upon particular modules of the ITS, each with its own problems and challenges, researchers can make some progress towards producing effective tutors.

Many systems that have been developed have concentrated on one or two of the modules of an ITS almost to the exclusion of others. This, as has already been observed, is necessary if any progress is to be made. However, it has had the effect that there is currently a dearth of well-balanced, usable ITSs. Rather than describe one ideal system, brief outlines of research projects that focus on one of the four main ITS modules will be given.

2.5.2 Domain module

It is difficult to gainsay Anderson's observation that the domain (or 'expert') module is 'the backbone of any ITS' (Anderson, 1988, p21). Explicit representation of domain expertise was the crucial characteristic that distinguished intelligent CAI from what had gone before. SCHOLAR's semantic networks, WHY's scripts and SOPHIE's procedural specialists are all examples of structures that may be used for representing expertise within the domain module and have provided a basis for knowledge representation in other projects. Another example of expertise embodied in a structure was provided by Goldstein (1979) in his *genetic graphs*. He developed a program called WUSOR that helps students to play an adventure game called 'Hunt the Wumpus' (Yob, 1975). The human expert at this game has a set of guidelines to help survive and to

capture the Wumpus. Part of a WUSOR genetic graph is illustrated in Figure 2.6. Each of the nodes corresponds to a rule that the expert can use for helping to play. Rules can be related in various ways such as generalisation/specialisation and these are denoted by links in the graph. The different phases correspond to different levels of expertise.

The topic of domain expertise is central to the research presented here and will be considered in greater detail in the next chapter.

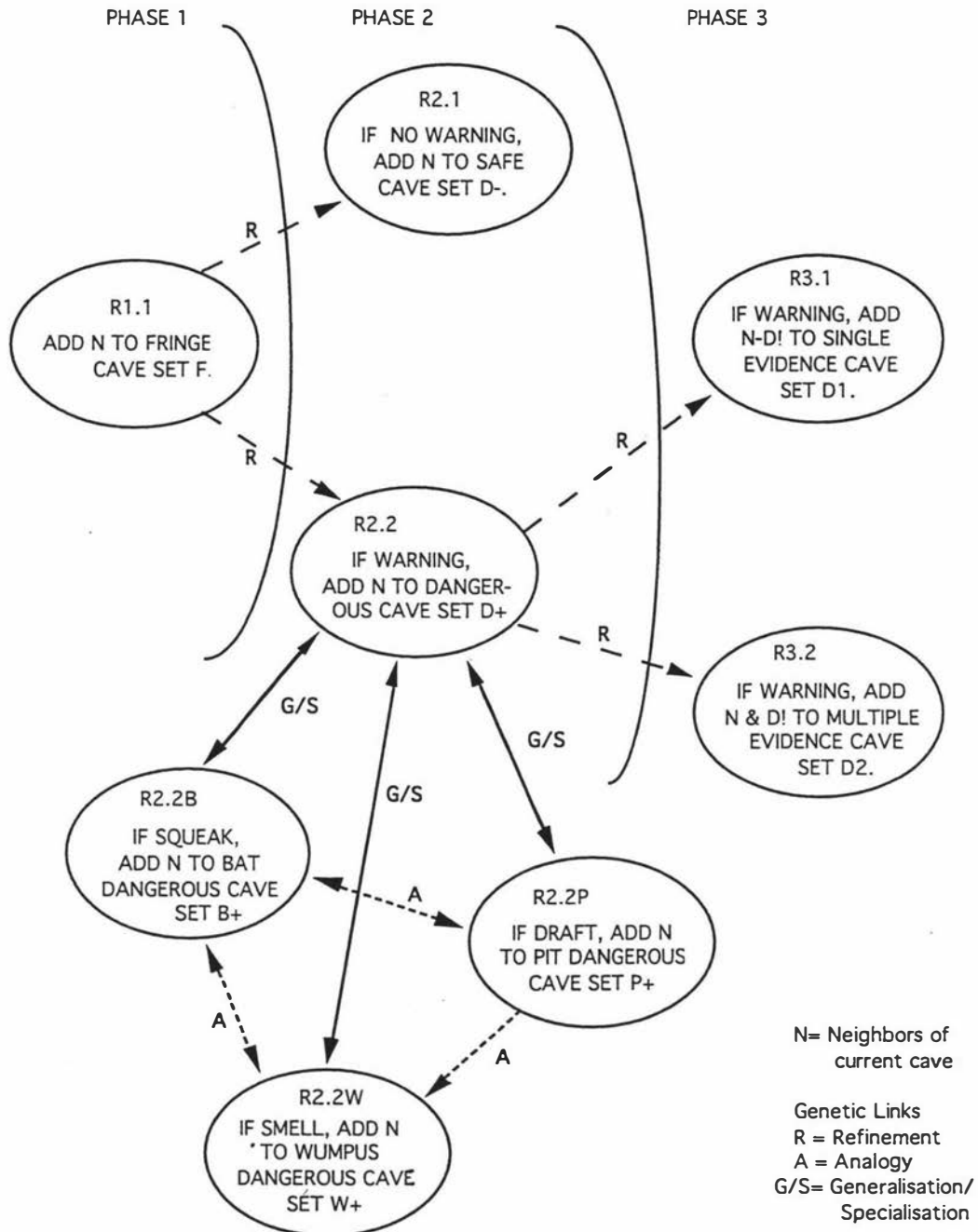


Figure 2.6 Section of WUSOR's Genetic Graph (Goldstein, 1979, p57)

2.5.3 Student module

Much research on student modelling has been carried out by John Self. His early work (Self, 1974) proposes that student models should be represented by procedural programs that could reproduce the student's responses. This approach has been adopted by many researchers including Brown and VanLehn (1980) and Anderson (1984). Laurillard (1990) gives an appraisal of some of this work. An alternative view is that a student's knowledge can be depicted as an *overlay* on the representation of domain knowledge being learned. This view is taken by Goldstein (1979) in his WUSOR program. The computer monitors the user playing the game and attempts to infer which of the skills in the nodes have been mastered. Once a skill is assumed to be acquired the corresponding node is marked. Thus the student model is represented by a sub-graph of the expert model depicted in Figure 2.6. An innovative view is taken by Cumming and Self (1991) who, in their work on collaborative learning systems, permit the user to have some input to the development of the learner model.

Fernandez-Valmayor and Chamizo (1992) take a different approach by simulating the way that people acquire declarative knowledge. They observe that this could also have an impact on how the domain knowledge of experts is organized since the expert will learn in the same way: 'in an ideal ITS, the expert and the student knowledge bases would have different contents, but they must share the same learning procedures' (Fernandez-Valmayor and Chamizo, 1992, p52). To make some progress, the authors are building a dynamic knowledge base to test knowledge representation mechanisms and knowledge acquisition processes needed to build an ITS. Ideally, this system would enable the domain knowledge to be produced (by interaction with experts) and student knowledge to be generated (by interaction with students).

Webb and Cumming (1991) also employ an inductive approach, using what they term *Feature Based Modelling* (FBM). In FBM, attribute-value machine learning techniques are applied in order to produce a cognitive model of the user. It is assumed that such a model can be summarized by relationships between the inputs and outputs of the student's cognitive system. The learning algorithm is used to find a pattern between task features (the inputs) and action features (the outputs).

These are just some of the many ways in which student models may be structured. A more detailed analysis is given by Self (1988). Most of these models are rather informal and Self (1989; 1992) has proposed a more rigorous approach to what he terms 'cognitive diagnosis'.

2.5.4 Tutorial module

Beverly Woolf has been at the forefront of research to provide appropriate tutorial modules in computerised systems. In a paper based on ideas in Woolf's doctoral dissertation, Woolf and McDonald (1984) describe MENO-TUTOR which provides a framework for determining appropriate tutorial activity within an ITS. MENO-TUTOR can be linked in to specific domains and aims to provide a framework for tutor-student discourse. The original approach was to analyse discussions between a tutor and student and to use reverse engineering to construct a program that could reproduce the dialogue. This was applied in the domains of meteorology and Pascal programming, and, from this initial experience, an elaborate *discourse management network* was constructed that distinguished between pedagogic states, strategic states and tactics. Later, Woolf's group at the University of Massachusetts developed a language called TUPITS (Woolf *et al.*, 1988; Woolf, 1992) that can be used for defining primitive components of a tutorial discourse interaction. The tutor then uses this information for reasoning about actions.

Woolf and her colleagues concentrate mainly on the discourse between student and tutor. Other work has looked at providing a range of teaching methods. One such system is DOMINIE (Spensley *et al.*, 1990a) for showing students how to use computer applications effectively. The system embodies eight teaching styles. These include cognitive apprenticeship (where the novice initially watches the expert and then, over time, takes responsibility for more and more parts of the task); successive refinement (the teacher starts by giving an overview and gradually includes more and more detail at a lower level); and discovery learning (where the student investigates the domain and is guided towards discovering important concepts). The strategy for switching between styles takes into account what is currently known about the student's level of understanding and the suitability of the teaching method for the material. For example, at the start of a session the system may not know anything about the student's background and so uses a conventional teaching mode. Once it is determined that the student has reached a given level of understanding, a less interventionist approach is adopted. Another system that uses current educational and cognitive theories is described by Mohan, Greer and Jones (1992). Their *Instructional Delivery Planner* is used for teaching LISP programming and takes into account the learning abilities and cognitive processes of the student.

2.5.5 Interface module

The interface module is, perhaps, the most neglected part of ITSs. Many early systems used a subset of natural language for communicating with the user. A system that

illustrates the potential of *direct manipulation* interfaces (Shneiderman, 1983) is STEAMER (Hollan *et al.*, 1984) with its representation of the controls and other components of the steam propulsion plant of a large ship. STEAMER was used for training operators but hardly qualifies as an ITS since it is essentially a straight simulator. A more elaborate trainer based on simulation that does include explanation and tutoring facilities is RBT (Woolf *et al.*, 1987; Woolf, 1988b) which trains students in the use of a paper mill boiler, and, incidentally, is one of the few ITSs that is in regular use outside the research laboratory. An example of a screen from the program is shown as Figure 2.7.

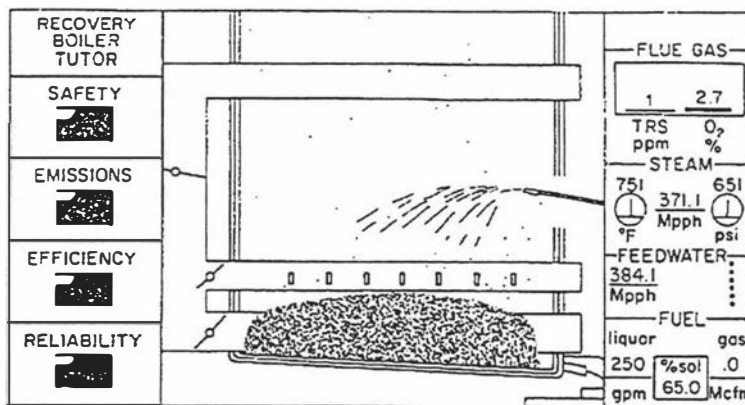


Figure 2.7 Screen from RBT (Woolf, 1988b)

Often the method of interaction not only simplifies the student's task of communication but gives him/her a completely different perspective on the problem. The Geometry Tutor (Anderson *et al.*, 1985) is in this category. The user builds up proofs of geometrical theorems in a diagram. More recently, Koedinger and Anderson (Koedinger, 1991; Koedinger and Anderson, 1993) have produced ANGLE, a sophisticated version of this program, that includes a full WIMPS interface and many other features for aiding the student's understanding of the domain. A screen from ANGLE 3.0 is shown in Figure 2.8. Down the left hand margin are icons denoting different rules that can be applied. A picture in the top left hand corner indicates what has been proved so far. In the main part of the screen the user has built up sequences of statements that can be inferred. The aim is to construct a path between the premises or givens (at the bottom of the screen) and the hypothesis (goal) at the top. Often the development of geometric (and other) proofs by a human teacher is presented in a linear fashion, starting with the premises and working directly towards the hypothesis. In the diagrammatic representation the proof process can be envisaged as attempting to connect the givens and the goal, and the path between the two can be built up from either end (or

starting in the middle). Also the parallel nature of some of the steps can be presented in a clear, visual fashion.

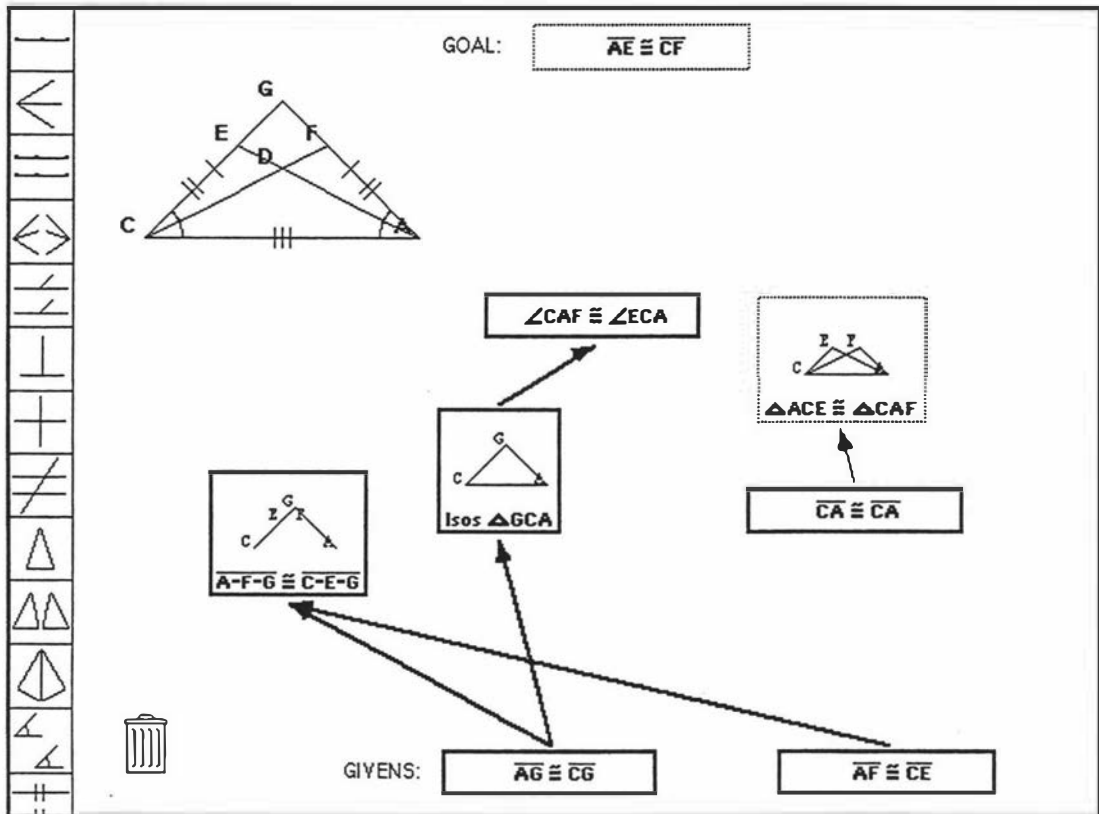


Figure 2.8 Screen from ANGLE 3.0

An elaborate interface is not desirable for every domain. For example, Corbett, Anderson and Fincham (1991) have shown that a menu selection facility incorporated into the Lisp Tutor (Farrell *et al.*, 1984) for code entry appears to impede learning. Students using the menu system rather than typing in programs directly made more mistakes during the computer tutorial and performed substantially worse on post tests.

Generally, however, interfaces that are well-designed and appropriate for the domain under consideration decrease the *cognitive load* (Wastell, 1991) on the students, allowing them to concentrate on the main purpose of the exercise. Graphics, animation, and other features such as video can also increase motivation and interest. Recently, there has been much research into the use of multi-media in education: see, for example, Holzl and Robb (1992), although there is little evidence of the use of AI in the packages produced so far. One of the few attempts to incorporate this technology has been made by Parkes and Self (1988). They investigate the use of video material within ITSs and develop an ingenious notation based on conceptual graphs (Sowa, 1984) for describing the activities

taking place. The aim is to allow the user to interact with a training video in such a way that the normal ITS facilities such as explanation, clarification and student modelling may be included.

2.5.6 ITS appraisal

Although there has been a great deal of research into ITSs, much of it has involved building programs and seeing how they work. Only a few researchers have attempted to review what the aims of their research are and how these might be best achieved. An important theoretical assessment of ITSs was produced by Ohlsson (1987). He considers each of their components in turn and presents principles that the developer can use to focus ideas.

Appraising research on student modelling he suggests that, in some cases, the original aims have been forgotten. There is a temptation to produce a detailed model for its own sake rather than for the real reason which is to help direct the teaching module's activities. Wenger makes a similar observation: 'a perfectly accurate model of the student along all dimensions is not a *sine qua non* for reasonable pedagogical decisions' (Wenger, 1987, p16). Ohlsson puts it more simply when he suggests that the question we need to ask is not 'what goes on in the head of the student?' but 'what does the tutor need to know in order to teach?' (Ohlsson, 1987, p215).

Ohlsson goes on to consider the instructional material. He believes that it is important to distinguish between the subject matter and the formats in which it can be presented, and to be able to generate different presentations as needed. It is a mistake to believe that 'knowing the subject matter' corresponds to a single well-defined cognitive state. The target knowledge can always be represented in different ways and from different perspectives, hence the process of acquiring the subject matter may have many different equally valid end states.

In order to provide adaptive instruction Ohlsson believes that a tutor must have a wide range of instructional actions from which to choose. He distinguishes the different kinds of cognitive skills that may be required by the student and lists different ways for teaching each. For example, to teach a procedure, the tutor might start by defining terms employed, describe the procedure directly, demonstrate its execution and then give the student practice in using it.

Lastly, he considers general strategies that the computer tutor might incorporate. It first needs to generate a teaching plan based on its representation of the student, its knowledge

of the subject matter and its current tutorial goal. This plan must be flexible and the tutor must be able to revise it if it does not fit the student.

Ohlsson is a researcher at the Learning Research and Development Center (LRDC), Pittsburgh University, where, under the leadership of Robert Glaser, a great deal of work has been done on how people learn and how this knowledge can be used in computerized teaching systems. Most of the researchers at LRDC take a cognitive science view. That is, they consider the brain as an information processor and look for models of this complex organ that correlate with observable characteristics. Many cognitive scientists believe that having a detailed model of how the human brain processes, stores and accesses information, particularly during the learning process, will help us find effective ways of teaching people.

Another set of researchers, also situated in Pittsburgh, work from the same premise. This is John Anderson's team at Carnegie-Mellon University (CMU). The LRDC and the CMU groups have different perspectives but both have provided valuable insights into human learning processes. Also, both institutions have applied this knowledge to create educational software that is amongst the most effective produced anywhere in the world. Although there are many links between the organizations, they each have their own research styles and particular topics of interest. For these reasons a section is devoted to describing the work of each these groups.

2.6 Research at LRDC

Many questions concerning human knowledge and expertise remain unanswered but LRDC researchers have provided some general factors that distinguish experts from novices and what happens during the transition between the two. By examining these kinds of phenomena it is believed that appropriate ways of facilitating these changes can be determined.

2.6.1 Knowledge structures

One of the main areas of research during the 1980s at LRDC was the investigation of human knowledge structures and how they differ between novices and experts. In a seminal paper, Chi, Feltovich and Glaser (1981) investigate the problem solving methods of university physics students. They compare the approach of eight advanced PhD students (the experts) with a similar number of first year degree students (novices). In one experiment, students were shown a set of problems and asked to group together similar ones. Novices tended to classify them by surface features such as objects in the

problem, or significant words (such as *friction*), whereas experts classified according to major physics principles governing the *solution* of the problem. Results from other experiments support the theory that novices tend to think in terms of objects whereas experts look at principles and inferences (implying a more procedural approach). By various means, Chi and her colleagues build up separate knowledge structure representations for the novice and expert physics problem solvers. A semantic net type diagram is used to illustrate the differences in structure and objects referred to. They employ the notion of *schemata* or chunks of knowledge representing different concepts. An individual schema may have a number of slots containing information associated with the concept and may also have procedures or rules that are useful or commonly needed in problems referring to it. These procedures have explicit conditions for applicability. The novice may have declarative knowledge but it is not structured appropriately and lacks the abstracted solution methods of the expert.

Chi, Feltovich and Glaser's subjects were university students but similar results have been observed with very different sectors of the population. Glaser and his colleagues (Glaser *et al.*, 1985) conducted experiments with jet engine mechanics to examine how they perceived their subject domain. When asked to produce a mental model of a jet engine by grouping together relevant components, the novices tended to classify them according to their physical position whereas the experts grouped them according to their function or by their mechanism (for example, electrical components).

Experiments with children to determine their knowledge structures have also been carried out. Gobbo and Chi (1986) studied how much children aged between 4 and 5 knew about dinosaurs. Some of the children already knew a significant amount about the subject whereas others knew virtually nothing. From various experiments they inferred that the 'experts' have a more cohesive and structured representation of their knowledge than non-experts. This sophisticated structure enables them to assimilate new information more easily and also to reason and make inferences when presented with unfamiliar examples.

As an interesting aside, some research conducted at the University of Maryland has produced similar results. Weiser and Shertz (1983) asked expert and novice programmers to sort programming problems. The experts sorted them according to solution algorithms whereas the novices sorted them according to areas of application (for instance, whether the program was supposed to create a list of employees' salaries or whether it was supposed to keep a file of current user identifications). In both cases the students obviously partition the problems into conceptual categories but novices only take surface features into account whereas experts consider principles.

Experts, then, appear to have their knowledge structured in a different fashion from novices. It is better organised, and is oriented towards solution techniques rather than superficial aspects of the problem. Another important question is whether the two classes use different problem solving methods.

2.6.2 Problem solving methods

Before addressing the question of what kinds of problem solving techniques different people use, there is a need to classify methods. Glaser and his colleagues (Glaser *et al.*, 1985) distinguish weak methods (general problem solving techniques) and strong methods (based on the domain of interest). Both are important and are reflected in techniques used in AI programs. Newell and Simon (1963) were two early researchers to use weak methods applicable in a variety of domains. The well-known computer program called GPS embodies many of their ideas. In contrast, most knowledge-based systems use strong problem solving methods, very much geared to the domain of application.

Research into more than a dozen weak problem solving methods has been conducted by LRDC researchers (Magone, 1977; Glaser *et al.*, 1985; Resnick, 1985a; Gobbo and Chi, 1986; Rabinowitz and Glaser, 1986). These methods, with explanations where appropriate, are listed in Figure 2.9.

Results of experiments suggest that experts tend to use some of these problem solving methods rather than others. For example, Rabinowitz and Glaser (1986) believe that experts prefer to use a data directed strategy whereas novices work backwards using a kind of means-end analysis. Self-regulation is believed by Glaser and his colleagues (Glaser *et al.*, 1985) to be an important weak method used by experts. They quote Piaget who distinguishes three stages of self-regulation: *autonomous regulation* in which learners continually regulate their performance, fine tuning and modulating actions, but with very little conscious awareness and control of what they are doing; *active regulation*, which is similar to trial and error, where the learner constructs and tests theories in action; and *conscious regulation*, which involves mental formulation of hypotheses capable of being tested via imaginary counter-examples or confirmatory evidence. The LRDC team suggest that this sequence is a common one in the development of expertise.

Strong methods appear to be central to problem solving performance. These methods are highly dependent on the structure of the knowledge about the domain. To be able to solve problems one must have specialist knowledge of the problem domain and it must be appropriately structured. This is more important than processing ability. The

experiments of Glaser and his co-workers (Glaser *et al.*, 1985) with avionics technicians reveal that expert technicians use much more effective strong methods than novices but that they do not necessarily employ better weak methods. Further indirect evidence for experts' reliance on strong methods comes from Resnick (1985a). She cites work that has shown that expertise cannot usually be transferred from one domain to another. In other words, it appears that knowledge of the domain rather than any general methods or native intelligence seems to be the principal factor that distinguishes the expert from others.

Weak problem solving method	Explanation
inference	
induction	producing general principles or structures that are consistent with current information
analogy	
metaphor	
planning	
self-monitoring	
self-regulation	modifying approach and procedures used if they do not seem to be working
meta-cognition	being aware of how much one knows
means/end analysis	taking decisions that appear to make progress towards the solution
generate and test	producing trial solutions and checking whether they satisfy solution constraints
data-directed problem solving	working from data towards solution
goal-directed problem solving	working back from solution to data
question-asking	
top-down problem solving	
bottom-up problem solving	

Figure 2.9 Weak problem solving methods

By definition these strong methods are somewhat problem-specific but do have general characteristics. Drawing an analogy with expert systems, one might assume that heuristics play a big part. According to Glaser (1986) this is not so. In fact, he finds that heuristics are used heavily by novices but only in unusual situations by experts.

Glaser and Chi believe that matching of schemata is a key feature of strong problem solving methods used by experts. This allows problems to be solved quickly and 'noise' to be reduced. It also allows the subject to prune the problem space quickly (Glaser and

Chi, 1988). Thus, for example, Chi, Feltovich and Glaser (1981) show that experts at solving Physics problems use the initial data to suggest a problem category and then use what else they know about the problem category to generate other required information. This is like choosing a schema and then filling the slots. The fact that these experts have trouble in solving problems that are unusual or nonsensical provides evidence for this since, presumably, they do not have the corresponding schemata and have to solve the problems another way. Chi and her colleagues show that novices may also use this schema matching approach but that the schemata used are less effective.

Having examined the distinction between novices and experts in both the way they structure knowledge and their problem solving methods, it is important to understand the transition between the two. As Resnick (1985b) observes, unless we have a theory of acquisition of expertise we cannot develop a theory of instruction.

2.6.3 The acquisition of expertise

The transformation process whereby the subject develops skills in a domain is one that has been of interest for many years. Piaget (Boden, 1989) used the idea of 'equilibration' to indicate how humans are able to smoothly change their perception of a domain to improve performance. The LRDC group has also looked at particular facets of this. Important ideas in this transformation seem to be schemata changes, the development of automaticity, chunking and self-monitoring.

Schemata changes are considered by Glaser. At different stages of learning there exist different integrations of knowledge. Smaller units get integrated into larger ones. Also the expert 'compiles' the knowledge into a suitable form for usage in problem solving, perhaps using situation-action rules. Glaser observes that the 'progression from declarative to tuned procedural and goal-oriented information is a significant dimension of developing competence' (Glaser, 1989, p273). In an earlier work (Glaser *et al.*, 1985) his team also cites the importance of appropriate mental models in the development of expertise. The more appropriate the model is, the less the subject's memory and inferencing capabilities will be taxed. They suggest that this may be what gaining of expertise is all about, ie the development of increasingly accurate models of the domain.

Chi, Feltovich and Glaser (1981) note that intermediate physics students seem to represent the problem domain in a form that straddles the explicit object representation of the novice and the procedural orientation of the expert. This suggests that learning involves a possibly gradual change of the schemata used within a domain from object oriented to solution procedure oriented.

Resnick (1985b) has also studied the transformation of schemata. Some initial frame of reference (represented by initial schemata) is vital to the absorption of new knowledge. She cites examples from English comprehension and from Physics. Although new knowledge can be easily accommodated into an existing schema as long as there is no conflict, there will often come a time when there is inconsistency between the two. This will occur when the person has an inaccurate or inappropriate model of the domain (misconceptions).

In an earlier paper, Resnick (1984) suggests that instantiation of already established schemata is the central method by which people interpret new information. Specific events become interpretable when they provide the information needed to 'fill slots' in schemata. Schema activation and instantiation are also aspects of learning. One learns about situations by interpreting them in terms of already held schemata. At the same time, the process of successive reinstantiation of schemata with new specifics enriches the schemata themselves and extends their range of applicability. Close consideration of the way in which schema activation and instantiation works in various models of comprehension can, then, provide an important set of hypotheses about the processes of learning.

There is some evidence to suggest that the development of new schemata correspond to 'plateaux of learning' that have been observed in various domains. Glaser and his colleagues believe that there are times when the student does not improve for a while. These could correspond to points where the student is changing his/her model. Evidence comes, for example, from their study of the effectiveness of training methods for mechanics and technicians in the American Air Force (Glaser *et al.*, 1985). They believe that the subjects' abstract models of the domain change with experience.

The automating of useful and commonly required processes ('automaticity') accompanies the development of ability in the domain. Resnick (1985b) suggests that automating part of the processing of information ensures that the limited short term capacity of the brain is not over-burdened. Schneider (LRDC, 1986) also looks at the shift towards automaticity in experts. He observes that as automaticity is developed the subject is able to increase speed and cope with high work load and higher problem complexity. Glaser (1986) notes that the development of automaticity allows the expert to spend more conscious time on novel features of the problem and on general strategy and planning. This is obviously so in such activities as driving and tennis but, according to Glaser, is also crucial in intellectual activities where strategies become important in solving difficult problems.

2.6.4 Facilitating learning

Before assessing the effectiveness of teaching methods one needs to define what is meant by the term 'instruction' and to have a theory of instruction. Resnick (1985a; 1985b) maintains that instruction is any activity that facilitates the process of learning in another person. That is, it is an intervention procedure in a learner's ongoing processes of knowledge acquisition. She postulates that a theory of instruction should have three components: a theory of intervention (how the instructor operates), a theory of acquisition as described in the previous section, and a theory of expertise (some criteria for measuring the student's competence level in the domain). Resnick believes that theories of expertise are well advanced but that acquisition is not so well understood. Even less is known about effective methods of intervention.

She looks at various aspects of the problem of appropriate intervention, observing that we can no longer think of teaching as a process of directly putting knowledge or skill into people. Instead 'effective instruction must aim to place learners in situations where the constructions that they naturally make as they think about the events and information that impinge on them are maximally likely to be correct and efficient' (Resnick, 1985b, pp28-29). Also, the instructor should not necessarily be looking for ways of presenting information that directly match the thought or performance patterns of experts. Rather s/he should be trying to find instructional representations that allow learners to construct those expert representations gradually for themselves.

Glaser and his colleagues (Glaser *et al.*, 1985) come to a similar conclusion when testing avionics technicians. The technicians had previously been taught by demonstration of the algorithms, but, since they had not devised these themselves, they learned nothing of the diagnostic and procedural skills that are needed. The students needed to apply problem solving in the specialty in order to develop stronger problem solving methods from the known weak ones. The LRDC team recommend that the students be supplied with more declarative knowledge of the specialty, rather than having to determine it, and be given more opportunity to practise problem solving. They propose a simulation method whereby the technicians may develop these skills, improving their competence by tackling and solving novel problems. They observe that experts in a routine domain often do not handle novel problems well.

One of the important characteristics of effective problem solving and of learning in the process seems to be self-monitoring. Glaser and his colleagues (Glaser *et al.*, 1985) discuss meta-cognitive strategies or *comprehension monitoring* (ie establishing learning goals, assessing the degree to which these goals are being met, and modifying strategies

and methods when necessary to better achieve these goals). Glaser (1989) suggests that the ability to self-monitor (for example, to assess how far through the solution process one has progressed and how much work each part of the problem will take) is a measure of expertise.

Resnick (1985b) observes the same kind of phenomenon. Some students automatically monitor their own progress and adjust their activities accordingly. For example, she notes that students will often re-read a piece of text that they could not make sense of first time through. Further improvement can be achieved by encouraging the subjects to suggest questions about a piece of text that could be asked of others. Also they might be asked to predict what happens next.

Weinstein and Rogers (1984) at BBN find that comprehension monitoring can be quickly and effectively taught by encouraging students to engage in periodic self-questioning when studying textbook materials. One approach involves the tutor and student taking turns in paraphrasing, summarizing and asking appropriate questions on alternate paragraphs of a piece of text: using a so-called *Reciprocal Teaching Method*.

It is suggested by one group (Chipman *et al.*, 1985) that it is difficult to teach general problem solving skills to someone who does not have knowledge of a specific domain. This confirms the findings of the LRDC team (Glaser *et al.*, 1985) that specific problem solving skills seem to be the main discriminant between the more and less skilled workers. Seeming to contradict this is the notion that specific problem solving comes from applying general problem solving with the special knowledge of the domain. Glaser and his colleagues recognise this apparent contradiction and suggest that the two should be developed in parallel with a specific domain but, once this is done, the techniques developed can be applied in other domains: 'An intelligent practice environment should provide considerable external support in the form of general prompts for planning, monitoring, and modelling, for general problem solving skills which may not be developed well enough to be exercised autonomously . . . The goal of intelligent training systems should be to provide practice in applying general problem solving skills to specific problems that are of the same kinds as the trainee will face on the job. This practice then will allow specialized, expert forms of the higher-order skills to be proceduralized and strengthened' (Glaser *et al.*, 1985, p292). Although this means that the subject will have only specific knowledge of a single domain, it is believed that s/he will have learned enough to be able to adapt quickly to related ones.

The importance of an appropriate model of the domain is also mentioned by Chi (LRDC, 1986). She suggests that, instead of trying to teach children better ways of accessing

their knowledge, we should be ensuring that they build up better internal knowledge structures. If the student has an inappropriate model then misconceptions can result (Resnick, 1985b). The instructor should try to ensure that the subject does not develop 'buggy' models. In cases where the subject has developed obvious misconceptions the instructor must decide whether to confront the subject directly with the new model or let the student discover the inconsistencies him/herself.

Champagne, Klopfer and Gunstone (1982) find one effective way of reducing misconceptions is to encourage students to discuss their beliefs among themselves and to adopt an initially more qualitative view of what they are learning than is normally taken. In later papers (Champagne *et al.*, 1985b; Champagne *et al.*, 1985a) this approach is formalised as an 'ideational confrontational strategy'.

Minimising the chances of misconceptions can be achieved by proceeding with caution. Learning is facilitated by presenting subjects with problems that they should be able to solve with their current knowledge but which extend them and encourage them to modify their knowledge structures. This is particularly effective if the subject has a large declarative knowledge of the domain and discovery techniques are used for exploration (Glaser, 1987).

Misconceptions can often arise when inappropriate heuristics have been taught. Often heuristics are used to aid the learning process of novices. This is understandable since these rules of thumb provide shortcuts and can be applied without requiring a detailed knowledge of the domain. Glaser and Chi (1988), however, observe that inappropriate heuristics can, in fact, hamper novices' development of expertise since the rules often have no logical basis, the student may become too dependent upon them, and they may not work with a more elaborate model of the domain.

Short term memory capacity seems to be another limiting factor on learning (Resnick, 1985b), consequently the tutor should try to reduce the demands on memory. This can be done by encouraging automation of simpler tasks and by helping the subject to develop appropriate groupings into chunks. Resnick believes that the development of understanding is affected by the kinds of practice afforded by instruction and by the ways in which procedural practice is interspersed with invitations to reflect and construct explanations. Shute, Glaser and Raghavan (1989) also maintain that understanding is not necessarily a pre-cursor to performance. Often it follows on later from the ability to solve problems. Procedural knowledge within the domain is more crucial than understanding. In particular, subjects learn well in a discovery environment where they can formulate hypotheses and then test them.

The use of a discovery environment that students can explore is examined by Shute and her colleagues (Shute *et al.*, 1989). By proposing and testing hypotheses the student learns about the environment. If the learning is done by getting the subject to solve problems then these problems should be carefully chosen to extend the subject's knowledge gradually so that s/he can build on existing knowledge.

As noted in Section 2.6.2, experts seem to use different (and fewer) weak problem solving methods from those of novices. Can expertise be improved, then, by getting novices to adopt the weak problem solving methods used by experts? The answer seems to be no. The distinction between weak problem solving methods used by experts and novices would appear to be an effect rather than a cause. For example, self-regulation helps in solving problems but only if one has enough knowledge of problem solving methods within the domain to know how well they are working. Rabinowitz and Glaser (1986) observe that although experts tend to use a working forward strategy, this would not necessarily help novices since they do not have the domain knowledge to exploit such a technique. Gobbo and Chi (1986) find that inferencing is an effective technique used by experts for problem solving but that the amount of inferencing that can be done is very much dependent on their knowledge of the domain. Thus, it is *because* the expert has a well-structured knowledge base, s/he can effectively apply certain weak problem solving methods. It may not be appropriate to attempt to teach novices similar methods.

Although strong methods appear to be ultimately more important than weak ones in solving problems in particular specialties, there are important inter-dependencies between them. For example, Glaser's team (Glaser *et al.*, 1985) suggest that strong methods are developed from the operation of weak methods on declarative knowledge. It is also clear that without specific domain knowledge many weak methods would be ineffective.

2.6.5 LRDC software

A significant amount of software has been developed at LRDC either to test ideas on knowledge acquisition and learning or, under contract, to supply particular needs. A package that comes into the first category is Smittown (Shute and Glaser, 1990; Shute and Glaser, 1991). This program provides a 'laboratory' for teaching students the elements of economics. In a simulated environment various products can be bought and sold. Prices vary according to such factors as availability and demand. By experimentation the student has to determine what the relationships are between these factors. Although the subject is economics, the real purpose of the investigation is to look at how effectively people can learn by using an experimental approach and what facilities can be provided to aid this process.

In this guided discovery environment, students are encouraged to generate and test hypotheses, observe, record and organise test results, modify hypotheses in accordance with the results, and induce regularities and laws. Thus the focus is on developing effective weak problem solving methods in an unknown domain. Some of the experiments with students (Shute and Glaser, 1990) suggest that appropriate use of weak methods is a better predictor of success than general intelligence.

As has already been noted, self-regulation is an important weak problem solving method. A facility for encouraging this process has been added to Smithtown (Raghavan *et al.*, 1991). DARN (Discovery And Reflection Notation) allows the student to view activities in the laboratory from three different perspectives: the Student View, the Plan View and the Expert View. The Student View traces the actions that the student has taken, presenting this as a search path. The Plan View relates what the student has done to the plans they have proposed, indicating which actions are and are not consistent with that plan. The Expert View illustrates what the expert would have discovered, given the same data.

Other laboratory-style environments that have been developed at LRDC include Voltaville (Schauble *et al.*, 1991) where the subject is electrical circuits, and REFRACT (Reimann *et al.*, 1988) which is a discovery environment for geometrical optics. Schauble and his colleagues look at the impact of effective models of the domain on the learning ability of students. Reimann's group examine the effect of inductive learning.

All the previously mentioned programs have been written to investigate ideas about learning. The Sherlock project (Lesgold *et al.*, 1988) has put many of these ideas into practice. The American Air Force use a test station to locate problems in electronic navigation equipment for F-15 aircraft. If the test station is functioning properly then the faults can be isolated quickly. Unfortunately, the test station, a very complex piece of equipment, quite frequently breaks down. Technicians have to be able to track down faults in this test station, a skill that can take up to 5 years to develop. To fast track this learning process the Air Force has tried various means, principally giving the apprentices a detailed course on how the test station works. As has been observed by Glaser and his co-researchers (Glaser *et al.*, 1985), this approach is not very effective. The best way of teaching problem solving is to set up faults in the test station and get the student to locate them. Because of costs and the possibilities of damaging the test station this method is impracticable.

To get over this problem Sherlock was developed. Sherlock provides a coached practice environment that simulates the activity of the test station including the effect of many

different faults. It is easy to set up the system to demonstrate the effect of any of these faults so the tutor can focus on a particular problem, perhaps one that the user has never seen before or one that s/he has previously had trouble diagnosing. As noted by Lajoie and Lesgold (1989) the help that Sherlock provides can be seen as a form of *cognitive apprenticeship* (Brown *et al.*, 1989) where the system initially gives the student a significant amount of help in problem solving but gradually withdraws that assistance as the student improves. To achieve this *fading*, Sherlock stores a model of the student's current level of competence.

Not only was Sherlock developed by workers with a deep understanding of all aspects of learning and teaching but the original version was also a very sophisticated piece of software. It was highly successful in its aim of providing the Air Force with a system that could accelerate the teaching of novices. Lajoie and Lesgold (1989) observe that experiments have shown that after twenty to twenty-five hours using Sherlock, less experienced technicians could achieve a comparable trouble shooting ability to colleagues with four or more years on-the-job experience.

Building on the experience of the initial version of the package, researchers have developed Sherlock II (Lesgold *et al.*, 1990; Katz *et al.*, 1992a; Katz *et al.*, 1992b; Lesgold *et al.*, 1992a). The new version incorporates a deeper model of the device and expert knowledge, a sophisticated interface with video graphics, uses fuzzy logic for student modelling, and plans appropriate exercises for the student to attempt.

2.7 CMU research

In the same way that much of the impetus for LRDC research came from one person, Robert Glaser, the research at CMU was initiated by and was built around the work of a single researcher: John Anderson. Anderson has a background in psychology and cognitive psychology. His original reason for moving into computing was to provide a test bed for his models of cognition. Not only has this research proved highly fruitful, but tutoring systems developed using his ideas have been particularly successful.

2.7.1 The ACT models

From an early stage, Anderson appreciated the power of using the computer to model aspects of cognition. His first notable contribution was a model of memory called HAM (Anderson and Bower, 1973) where word concepts were stored in nodes of a graph and associations stored in links. Although the behaviour of HAM was found to mirror quite closely the behaviour of humans in memory recall experiments, Anderson felt that it was

very limited since it could only store facts and could not model processes. This is when he first developed one of his central tenets that there is a clear distinction in the human cognitive system between declarative knowledge (facts that we know) and procedural knowledge (skills that we know how to perform). He had used propositional nets for storing facts and relationships in HAM but these were inadequate for representing processes. Anderson eventually found what he believed was the ideal building block for procedures when he read about Alan Newell's production-system models. As he later notes 'I became more and more convinced that they [production systems] contained key insights into the nature of human procedural knowledge' (Anderson, 1983, pviii). Putting together the propositional nets with a production rule interpreter he developed an early version of the ACT (Adaptive Control of Thought) system (Anderson, 1976). In this work he shows how production sets may be used to model various tasks.

When Anderson moved to Carnegie-Mellon in 1978 he began the process of refining his model. Later versions of ACT each addressed specific problems. For example, ACTF showed how production rules could be created, thus modelling skill acquisition. ACT* (Anderson, 1983) was the final system in the sequence and addresses problems related to pattern matching, goal representation and, in addition, contains a more refined model of production learning.

ACT* has a declarative memory (facts and relationships), a production memory (procedures) and a working memory. All knowledge is initially declarative and is interpreted using skill-independent productions. This roughly corresponds to the use of weak problem solving methods. Gradually this declarative knowledge is replaced by rules representing specific skills. These rules may also be modified over time by merging. These two processes (called *proceduralization* and *composition*, respectively) together form what Anderson calls *knowledge compilation*. Anderson likens this to the automatization of skills and knowledge assimilation observed among human learners. Like the LRDC researchers he concludes that this automatization is necessary to reduce the load on memory and to speed up operations.

According to Anderson's model, expert problem solving corresponds to appropriate interpretation of the production rules. The model can be data-driven, when patterns on the left hand side of rules are matched and corresponding actions on the right hand side of rules are invoked. Alternatively, actions can be matched to see what circumstances would be appropriate for the use of a specific action, simulating goal-driven problem solving. Like the LRDC researchers he concludes that experts predominantly use a data-driven approach and consequently he develops elaborate pattern matching methods for the left-hand side of his rules. A key factor is which rule to choose when several could

be activated. The ACT* pattern matcher uses five principles for this *conflict resolution* including degree of match (how closely the current situation matches the rule), production strength (how useful the production has generally been found to be), and specificity (choosing the production that has the most specific match with the data). These are mainly justified on logical grounds. Practical justification of his model was attempted by constructing software.

2.7.2 CMU software

During the early 1980s, Anderson and his co-researchers developed two significant tutoring systems: the Lisp Tutor (Reiser *et al.*, 1985) and the Geometry Tutor (Anderson *et al.*, 1985). One of the main aims was to justify and refine the production system model of learning but, as noted by Anderson, Boyle, Farrell and Reiser (1987), they also believed that a model of cognition was essential for constructing any computer tutor. Such a model would have to provide an accurate assessment of the student's current cognitive state, a precise definition of the desired cognitive state and a learning theory to explain how the change from one to the other takes place.

The Lisp Tutor is a program that provides assistance to students as they work on elementary Lisp programming exercises. It has been used for introductory Lisp teaching at CMU since 1984 and has been successfully marketed (one of the few ITSs to have achieved this status). Evaluation studies (Anderson, 1990) have shown that students learn more quickly and effectively using the tutor than in self-study.

The expertise in the Lisp Tutor, not surprisingly, is represented by production rules. Each of these rules corresponds to some concept in the language. An example, taken from Corbett, Anderson and Patterson (1990, pp87-88) and transliterated into English, is shown below:

```

IF      the goal is to form a list
        by inserting newitem
        at the beginning of
        an existing list oldlist
THEN   code a function call to CONS
        and set subgoals to code
        newitem and oldlist.
```

The production rule organisation is slightly different from ACT* in that a goal stack is maintained and that a single active goal is popped off the stack during each cycle. This restricts the student to operate in a top-down, left to right fashion. This refined version

of ACT* is called GRAPES (Goal Restricted Production System Architecture). An advantage of this organization in the tutor is that explanations can be related to current goals.

The 325 rules in the tutor are designed with novice learners in mind and do not necessarily correspond to the way that an expert might store the information. Instead, they reflect what protocol analysis has shown to be simple concepts that novices could relate to and remember. As a further variant on ACT*, the original Lisp Tutor stored 475 *mal-rules* that denote common misconceptions that students may have.

The ideal rules and mal-rules are used for attempting to monitor the student's problem solving procedure. The student is guided along the idealized path represented by the correct production rules (so-called *model-tracing*). The mal-rules allow the system to check and monitor certain kinds of errors and, as long as the student makes errors that the tutor can recognize, s/he will be allowed to continue. Such deviations from a correct line are strictly controlled, however, since the ACT* model does not cover learning by making mistakes. Consequently, the student is usually quickly brought back into line after an error is detected. Anderson feels strongly that it is unnecessary to attempt to diagnose student misconceptions in detail. He believes 'there is little role for delivering cognitive diagnosis of errors in feedback and in many cases the tutor's feedback should be minimal even to the point of being non-existent' (Anderson, 1989, p1).

Another feature of ACT* is the use of weak problem solving methods in a problem domain and how it can be used (via knowledge compilation) to produce appropriate domain rules. How this might occur in teaching Lisp is described by Anderson (1987). He shows how both means-end analysis and analogy may be used by students to produce the kind of rule that is embodied in the Lisp Tutor.

The evidence for knowledge compilation amongst humans was gathered by logging the problem solving times of students using the Lisp Tutor (Anderson, 1987). Analysis shows that students apply rules slowly first time around (presumably using weak methods on declarative data) but rapidly improve thereafter (knowledge compilation). After this initial significant improvement there are smaller gains corresponding to strengthening of production rules.

Improvements in the Lisp Tutor continue to parallel modifications to the ACT* model. For example, a more flexible control mechanism, PUPS (PenUltimate Production System) (Anderson *et al.*, 1990), has a more flexible control system than GRAPES so

that a top-down, left-to-right development system is not imposed on the student. PUPS also contains schema-like structures containing special slots for problem solving.

The lower profile but equally ingenious Geometry Tutor (Anderson *et al.*, 1985) has been another CMU vehicle for investigating cognitive models. The domain, elementary problem solving in geometry, is very different from that of the Lisp Tutor but, again, the program was found to be successful when used in a teaching situation, in this case tutoring children in Pittsburgh high schools.

As usual, idealized student knowledge is encapsulated in production rules. In keeping with the nature of the domain, both forward and backward inferencing rules are used since it is equally valid to work from the hypothesis or from the premises. Examples of both forward and backward inferencing rules (Anderson *et al.*, 1990, p10), are shown below:

IF the goal is to prove triangle XYZ is congruent to triangle UYW
 and X, Y, W are collinear
 and U, Y, Z are collinear

THEN conclude angle XYZ = angle UYW because of vertical angles

IF the goal is to prove triangle XYZ is congruent to triangle UVW
 and XY=UV
 and YZ=VW

THEN set a subgoal to prove angle XYZ = angle UVW so SAS can be used.

An interesting feature of later versions of this program, as noted in Section 2.5.5, is the clever interface. Not only does it make the package easy to use but also reduces the student's cognitive load by providing a visual analogue of the proof procedure, a technique that is sometimes called *reification* (Wenger, 1987). It is this aspect of tutoring in geometry that has been the focus of recent research in the area (Koedinger, 1991; Koedinger and Anderson, 1993).

The visual representation of the proof graph and the automatic linking of logically valid statements to the subgraphs already built up provide positive guidance to the student. However, as yet, the tutor does not contain mal-rules and does not provide feedback on logical errors. Hints can be given when the student has gone wrong but these are not dependent upon the *kind* of mistake the user has made.

Other, more recent, projects at CMU include the development of a tutor for high school algebra (Lewis *et al.*, 1990) and the extension of the Lisp Tutor system to include facilities for teaching Pascal and Prolog (Anderson *et al.*, 1992). In this last paper, some principles for a general architecture for tutoring systems are proposed, paving the way for the development of an intelligent authoring system.

2.7.3 A comparison of LRDC and CMU research

It is useful to compare the approaches and results of LRDC and CMU researchers who are essentially looking at the same kinds of phenomena. LRDC has spent much time gathering data on learning methods in a variety of settings from primary school to Air Force training programmes. From these results, aspects of cognition have been modelled. In this respect their work may be regarded as mainly data-driven. In contrast, John Anderson is very much model oriented. He tackles various problems of cognition (memory, language acquisition, learning) by setting up and modifying models to fit his observations. This can be seen as mainly a goal-driven approach. The two strategies can be likened to the different methods used to determine the structure of the DNA molecule (Judson, 1979; Watson, 1981). Maurice Wilkins and Rosalind Franklin at King's College, London took a traditional approach of gathering data and attempting to construct a model to fit their observations. In contrast, the Cambridge team of Crick and Watson built models which they then compared with observations, modifying or replacing their models as necessary.

At LRDC, a great deal of research has been done into how people learn, but most of this has been in a traditional setting. It could be argued that the learning that takes place in a computer environment is different and that new models of this process are needed. In contrast, the CMU group have spent a great deal of time observing how students learn on the computer. They have monitored the progress of their students in this environment and made adjustments to their model as necessary.

At the core of all of Anderson's models is the representation of knowledge in production rules. Anderson chose this mechanism because he believes it gives insight into the nature of human procedural knowledge. Whether this is so is still open to debate. Perhaps the main advantage of this mechanism has been that it provides a homogeneous basis for the development of ideas at CMU. At LRDC, the schema model of expert knowledge has been dominant and influenced the development of software. Significantly, the later PUPS architecture from CMU introduces schema-like structures.

The process of automatization has been observed at both institutions. Anderson has modelled this phenomenon as the transformation of declarative knowledge into a set of production rules. At LRDC, this is seen as the transition between different schema forms. The phenomenon of learning plateaux, according to Anderson, corresponds to the development of problem-specific production rules, whereas many researchers at LRDC believe that it is due to changes in schemata that are occurring. Knowledge assimilation has been likened to the transformation of these structures from superficial descriptions of the domain to complex schemata denoting more abstract concepts. These schemata would be more problem oriented and contain procedures for solving commonly occurring problems within the domain. According to Anderson (1983) this blurs the distinction between procedural and declarative knowledge. Interestingly, although the two groups have different mechanisms, both emphasize the importance of learning by problem solving in the domain.

Both groups have used their knowledge to develop successful software. Anderson's group has taken a rigorous approach, adhering very closely to the ACT* model and only making modifications that can be justified both in terms of their perception of human learning and in terms of a production rule architecture. At LRDC, although principles of learning have been followed in developing the software, these have been at a more general level and no specific model of cognition has been adopted.

When considering each group's observations on the nature of learning it is tempting to recall the story of the visually challenged people who all touch a different part of an elephant and come to different conclusions about what it is like. Certainly, we are still groping in the dark to determine exactly how people learn but research at both CMU and LRDC has provided a great deal of insight into the process, and it seems likely that their research, and that of other cognitive scientists, will eventually produce a coherent and comprehensive theory of learning: arguably a necessary pre-requisite to the development of an effective computer tutor.

2.8 Summary

In this chapter, a selection of projects carried out in computer tutoring over the last thirty years has been described, with the merits of each being highlighted. Programmed learning systems, for all their faults, introduced the idea of using the computer for teaching. Authoring languages had the laudable goal of giving the teacher some say in what was to be taught to the students and in what order. If nothing else, they forced the teacher to consider exactly what the student should be learning and how this could be tested. The initial ICAI projects such as SCHOLAR and SOPHIE demonstrated how the

use of appropriate information structures could improve the flexibility of teaching programs. Intelligent tutoring systems consolidated that approach and introduced a useful template for system structure. Lastly, the work by cognitive scientists makes researchers reflect on their goals and methods and reminds us that the ultimate aim is to produce tutoring systems that are actually going to help people learn.

There is much that can be learned from the research described in this chapter. In terms of the design emphasis of this thesis some important points are listed below:

Educators and subject matter experts should be involved throughout the system development process.

AI techniques can be used to improve the transparency and flexibility of systems, using high-level concepts such as semantic nets and scripts.

System developers need to think in modular terms to make problems tractable. The ITS structure of domain, student, tutorial and interface modules provides a useful framework for system development.

Cognitive science is a fruitful source of ideas about learning. We need to be aware of such concepts as declarative and procedural knowledge, problem solving methods, automaticity, and appropriate modelling of the domain.

Of particular importance is how we might facilitate learning: placing learners in situations that help this process. They need to be solving problems in an environment that is as realistic as possible.

Opinions vary on the value of feedback, but a flexible system should allow educators and domain experts to be able to include as much or as little as desired.

These observations together with the principles expounded in the next chapter provide guidelines for the research to be presented in the second part of the thesis.

The aim of this review has not been to give a comprehensive history of computer tutoring or even to summarise all the important concepts. For example, significant work on collaborative learning (Cumming and Self, 1989; Cumming and Self, 1991; Dillenbourg and Self, 1992a; Dillenbourg and Self, 1992b), handling misconceptions (Burton, 1982), and planning the teaching of material (Peachey and McCalla, 1986; Brecht *et al.*, 1989; McCalla and Greer, 1990) has not been described. Instead, the emphasis has been on aspects of teaching systems that are relevant to the research to be described in this thesis. The discussion of some systems such as Logo and GUIDON has been deferred because they embody important ideas that have had a direct bearing on the research. In the next chapter, some of these key concepts will be presented.

Chapter 3

Key Concepts

3.1 Introduction

The main aim of this research is to develop high-level primitives to aid the design of computer teaching software. In order to do this, the nature of knowledge has to be examined, particularly how knowledge can be represented on the computer. The study of knowledge-based systems is an obvious starting point here.

Next, we need to consider how humans acquire knowledge. How do we learn? There are many points of view on this question, as has already been noted in the previous chapter. Further aspects of this topic will be considered in the light of the implications of knowledge representation.

A link between the different views on learning is provided by the concept of modelling. We need to consider both models that humans form and those that can be represented in the computer. Computer teaching models can be appraised according to their fidelity, an important concept that is examined in detail.

Lastly, the ideas of knowledge, models and learning are brought together by considering the central concept of discovery learning, its development, implications and variations.

3.2 Knowledge issues

Discussing the nature of knowledge has been a preoccupation of philosophers and other thinkers for thousands of years. Although these debates have been of a mainly academic nature in the past, the need for a way of describing knowledge, rather than

just being able to recognise when it is being applied, is seen as crucial in computerising instruction (Kemp, 1992).

Knowledge issues that are relevant to this thesis include:

Aspects of knowledge have been represented in expert systems. Are these representations useful for modelling *human* knowledge?

Can knowledge be described in a way that is useful for teaching?

Assuming knowledge has a particular form, does this suggest specific teaching methods?

Is the kind of knowledge embodied in expert systems useful to humans?

Can knowledge be communicated from computer to human in the same way that it can apparently be transferred from human to computer?

3.2.1 Expert systems

Expert systems technology has been a highly publicized success story. The idea of attempting to get better performance out of a computer program by incorporating human expertise was first tried out in DENDRAL (Buchanan *et al.*, 1969) and subsequently became very popular.

There is no doubt that, in a number of domains, expert systems methods have been highly effective. The term *knowledge* is used freely and widely in the context of expert systems and since education is all about imparting knowledge to students we may wish to examine whether expert system technology can give any pointers in this regard. Some observations on ways in which expert system technology could help are given below:

Examining the data organization of expert systems might give some insight into how experts' knowledge is stored.

Knowledge acquisition (Hart, 1992) is a well-researched area. We could regard the aim of extracting knowledge from an expert and putting on the computer as being similar in some ways to communicating it to the user.

An expert system can be used to solve problems. Could it also be used to teach people? In the same way students could learn from watching a human expert at work, could they also learn from watching an expert system solve a problem?

Perhaps a student can learn about a subject by attempting to construct an expert system to solve problems in that area.

Some of the earliest work on using an expert system for teaching was carried out by William Clancey using MYCIN as a basis. The MYCIN project (Shortliffe, 1976) was a highly successful enterprise, producing an expert system that was able to diagnose a selection of infectious diseases and to advise on antibiotic therapy. Clancey joined the project in 1975 and, as his PhD topic, attempted to construct a tutorial system for medical students. The program he developed, called GUIDON, was operational by 1979 but research on the project continued for another 6 years.

The original aim of GUIDON was to produce a set of teaching rules that could be applied to MYCIN's knowledge base. Expert systems are generally believed to store their representations of knowledge in an explicit form. They can display chains of inference that have been used to reach their conclusions and also indicate what proposition the program is currently attempting to establish. In view of these features, and because MYCIN was such an effective problem solver, Clancey initially believed that he would be able to easily adapt the program for teaching purposes. This proved not to be the case. As he notes 'In building MYCIN, we did not make explicit *how an expert organizes his knowledge, how he remembers it, and strategies he uses for approaching problems*' (Clancey, 1987b, p200). These are crucial factors in teaching and so Clancey had to consider how MYCIN's knowledge base could be re-organized and augmented to provide this information.

Like most expert systems MYCIN had been developed to solve problems efficiently by computer. A set of rules describes the knowledge of the domain that a human expert would have. These rules are processed using a rather simple but inflexible interpreter. An initial set of questions is used to determine possible diseases (the *differential*). Characteristics of all the diseases in the knowledge base are known and so the user can be questioned about each of those in the differential until the one that most closely corresponds to observed symptoms is found.

Since the interpreter is fairly simple, the power of MYCIN comes from its production rules, and this is where Clancey's main difficulty lay. The productions are usually geared towards problem solving rather than transparency. An individual rule may have

been initially developed to deal with a particular concept but over a period of time it will probably have been altered substantially to improve the overall efficiency and effectiveness of the program. This point can be illustrated by examining the 'alcohol rule' (Clancey, 1987b, p203) that relates alcoholism to particular organisms (see Figure 3.1).

RULE535	

If:	<ol style="list-style-type: none"> 1) The infection which requires therapy is meningitis, 2) Only circumstantial evidence is available for this case, 3) The type of the infection is bacterial, 4) The age of the patient is greater than 17 years,-- 5) The patient is an alcoholic,
Then:	There is evidence that the organisms which might be causing the infection are diplococcus-pneumoniae (.3) and e.coli (.2).

Figure 3.1 The Alcohol Rule from MYCIN (Clancey, 1987b, p203)

The first condition is what Clancey calls a screening clause. That is, it is used to determine whether the rule is applicable. In this case, it is relevant if the system is checking for meningitis. Clauses 2 and 3 serve a similar function. The ordering is important since it reflects the taxonomic hierarchy of infections. Each condition need only be checked if the preceding one has been satisfied. Clause 4 relates to real world (commonsense) knowledge and acts as a guard for the next part of the question. The user might lose faith in a system that asks whether a young teenager is an alcoholic. Condition 5 is based on the heuristic that alcoholics are susceptible to diplococcus-pneumonia and e.coli organisms.

Thus, in a single rule, several different kinds of knowledge have been incorporated in a non-obvious fashion. Clancey (1983) attempts to provide an 'epistemological framework' for rule based systems, distinguishing the purpose of each of the different kinds of knowledge that may be used in production rules.

Structural knowledge consists of relationships within the domain that are useful for problem solving. The first three clauses in the alcohol rule incorporate this kind of knowledge. They implicitly describe part of the taxonomy of diseases: in this case, meningitis as an infection, of which bacterial meningitis is an example. In this instance, knowledge relates to the problem space but knowledge about the solution space is also frequently incorporated into rules. For example, in the alcohol rule the

fact that *diplococcus-pneumoniae* is given a rating of .3 and *e.coli* is rated at .2 suggests that the former is more likely than the latter.

Strategic knowledge relates to the ordering of goals and hypotheses during the problem solving process. Clauses in a rule are always executed in sequence, from first to last, but by carefully arranging the conditions in a rule particular strategies can be implicitly implemented. For example, the ordering of the disease tests in the alcohol rule (checking for meningitis first and then bacterial meningitis) imposes a top-down strategy. Similarly, the elements in the consequent of a rule are executed from left to right so, by putting the possibilities in descending order of likelihood, an implicit strategy that investigates common causes first can be realised.

Other kinds of knowledge are incorporated into MYCIN's rules. Clancey distinguishes these as identification, world fact, domain fact and cause.

An example of an identification rule is 'if the organism is gram-negative, anaerobic rod, its genus may be *bacterioides* (.6)'. Effectively, it is a description of a property of a class that aids identification.

An instance of a world fact has already been cited, in condition 4 of the alcohol rule. An example of a complete rule that embodies this kind of knowledge is 'if the patient is male then the patient is not pregnant'. Others may not be so obvious, such as one rule that links military recruits with environments in which disease may spread quickly (since they live in crowded conditions).

Domain fact rules link hypotheses on the basis of domain definitions: 'if a drug was administered orally and it is poorly absorbed in the GI tract then the drug was not administered adequately' (Clancey, 1987a, p179). This is not a cause-effect relationship, but describes a particular circumstance: an instance of 'not administering a drug adequately'.

A crucial kind of knowledge for problem solving is embodied in causal rules since these link symptoms, diseases and treatments. Clancey follows Szolovits and Pauker (1978) in distinguishing three kinds of causal link:

an empirical association (a correlation where the process is not understood and where it is not clear which is the cause and which is the effect);

a complication (where the direction of causality is known but the reasons are not understood);

a mechanistic association (where the process is well-understood and can be modelled).

MYCIN includes rules that embody each of these kinds of links. Many are in the second category but even where the association is well-understood it is rarely explicitly stated since this would be inefficient. Clancey notes that in many cases the experts seem not to have deliberately 'compiled' the knowledge but, over a period of time, gradually optimized steps.

Clancey believed that it would be necessary to separate out these different kinds of knowledge in order to produce a teaching system. It seemed logical to assume that students would learn a process better if they could be given, for example, the rationale behind questions or told why a particular hypothesis was being checked.

This *decompilation* of knowledge in MYCIN produced an entirely different program called NEOMYCIN (Clancey and Letsinger, 1981). The new program solves problems at least as well as the old one (Clancey, 1988a, p375) but, since the different kinds of knowledge are separated out and labelled, it is easier for a human to follow the logic of the problem solving process. An extra kind of knowledge of a causal nature that Clancey calls 'support knowledge' is also included. This provides justifications for inference rules in the program. The justification links (which may fill in intermediate steps or may indicate the logic of a rule) aid understanding but may have been superfluous in the original system where efficiency was the main consideration. An example is support knowledge for the rule 'if the patient is less than 8 years old, don't prescribe tetracycline'. There are several intermediate steps that are missing in this rule: a patient under 8 years old has growing bones, there is chelation of the drug in growing bones, this leads to discoloration of permanent teeth which is an undesirable body change.

Although the explicit identification of knowledge types and the inclusion of support knowledge made it easier to produce a tutor, another form of knowledge was also needed: teaching knowledge. Clancey organised GUIDON to teach by what he calls a *case-method dialogue*. That is, a specific patient history is available and the student is given the results of the initial standard tests. Based on these s/he has to ask further questions and determine an appropriate treatment. The computer system monitors the student's problem solving process and can provide guidance as necessary.

In his book on the GUIDON project, Clancey (1987a) lists almost 200 tutorial rules (*t-rules*) which control the tutorial process. These rules do not refer to the specific knowledge in MYCIN or NEOMYCIN but only to the types of rule in the knowledge base. An example of a t-rule is given in Figure 3.2

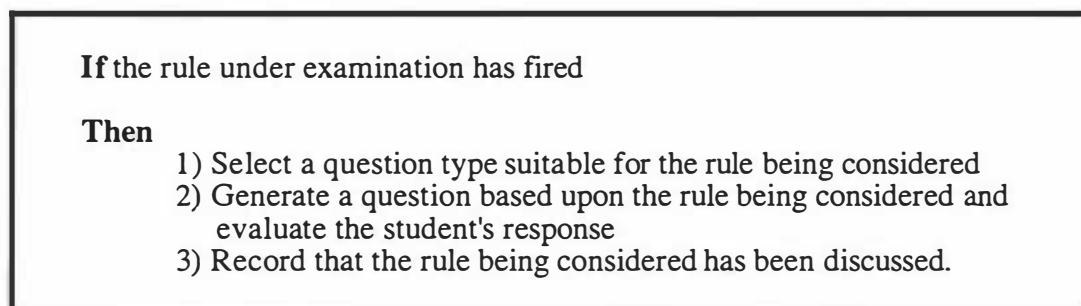


Figure 3.2 A tutorial rule (Clancey, 1987a, p359)

Thus, throughout Clancey's work on GUIDON and NEOMYCIN there has been a general assumption that knowledge can be separated out into different types and that these knowledge sources can be slotted together to provide knowledge-based programs for different domains and different purposes. So, for example, the t-rules from GUIDON could be attached to a knowledge base for another domain. Clancey (1987a) describes how the rules from GUIDON were applied in two other domains: aiding the interpretation of results from a structural analysis program (SACON) and pulmonary function interpretation (PUFF).

An analysis of the different forms of knowledge that Clancey distinguishes is presented in Figure 3.3. He, of course, was working mainly in one domain and with one kind of problem (diagnosis) but the categories described provide a fairly comprehensive summary of the kinds of knowledge that are generally used in expert systems.

A simple overlay approach is used for student modelling in GUIDON: rules that are assumed to have been understood and used by the student in the correct context are marked. In some sense this makes the assumption that the student's knowledge can be represented in a similar rule form to the way knowledge is represented in an expert system, and that the aim of teaching is to transfer this knowledge. Clancey refers to the notion of 'transfer of expertise' (Clancey, 1987a, p8). He likens the communication between teacher and student to the process that occurs when the knowledge engineer puts his/her expertise into a program, or when the expert system conveys information to a user.

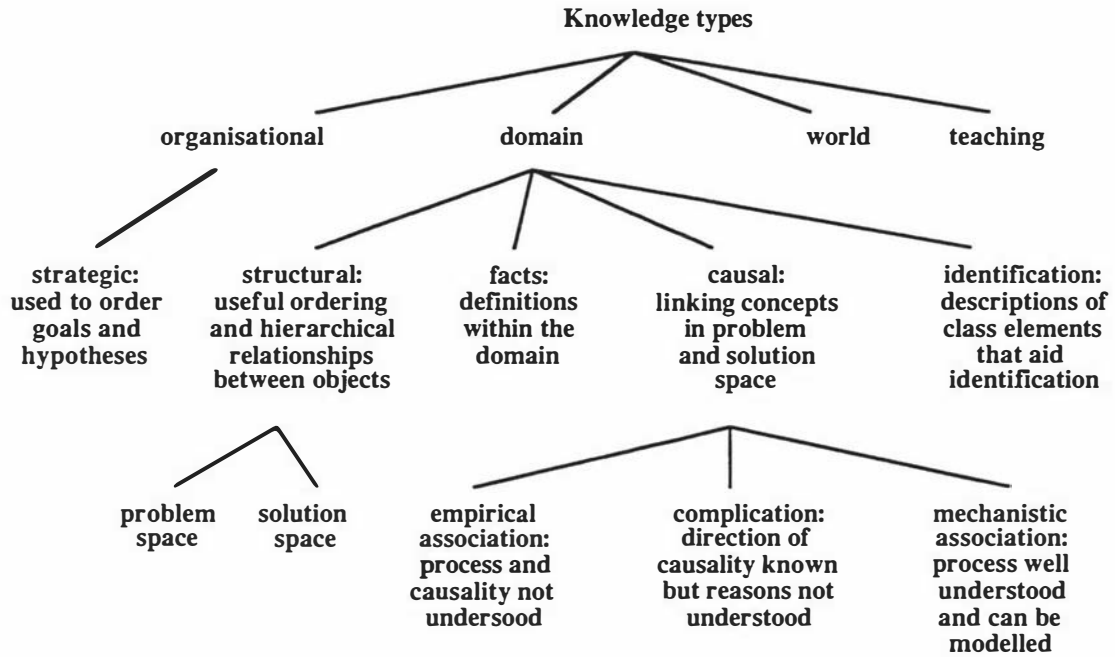


Figure 3.3 Knowledge types used in MYCIN, NEOMYCIN and GUIDON

This simplistic view of teaching is not too different from what is assumed to happen in programs like the Lisp Tutor. One significant difference is that the Lisp Tutor stores rules representing the knowledge the student is to assimilate rather than that of the expert.

Clancey's views on learning and teaching were expressed most directly in a paper that describes some of the features of GUIDON2 (Clancey, 1988b). Here he likens the procedure that the knowledge engineer employs in order to capture the expertise of the specialist in a model, to the process that a student might employ to learn effectively.

The knowledge engineer will use certain techniques for learning about the domain depending upon its nature. So, for example, when a causal network is to be constructed s/he might go through such steps as identifying the fundamental terms and relations in the domain, finding out about categories of substances and processes, and seeking causes of abnormalities. A learner could be encouraged to use the same kind of procedures to find out about the domain. This is the basis of the GUIDON2 approach where the student becomes familiar with a particular problem solving strategy, *heuristic classification* (Clancey, 1985), and then uses it in conjunction with the domain knowledge provided by NEOMYCIN to solve diagnostic problems. Once the problem has been solved, the student observes NEOMYCIN tackle the same problem, and, since s/he is familiar with some of the workings of the program, can better understand what is going on.

Lastly, in a proposed part of the system called GUIDON-EXPLAIN, the student and computer compare solution approaches. This last phase, Clancey likens to the process of knowledge engineering. In the same way that the knowledge engineer attempts to ascertain the expert's knowledge and procedures, the student attempts to understand the program's methods and domain knowledge.

Knowledge communication is a major theme in Wenger's book on intelligent tutoring systems (Wenger, 1987). The human expert maps his/her knowledge into some physical medium. It is then conveyed to the learner who re-constructs it. Writing a text book is a simple example. The specialist explicates his/her knowledge on paper. Students read the book and assimilate what is there to augment their own knowledge. The main advantages of using the computer for this are seen to be its capability for easily demonstrating procedures, and for engaging the student in an interaction.

Related to the notion of knowledge representation is that of *knowledge state*. Wenger, here, regards learning as being like progressing through a series of such states. Knowledge communication is defined as 'the instigation of modifications in a student's knowledge state via an alternation of diagnostic and didactic steps' (Wenger, 1987, p417). That is, the tutor alternately determines what the state of knowledge of the student is, and based on this, initiates appropriate teaching steps.

The preoccupation with knowledge representation and transfer that has been a feature of expert systems, knowledge engineering and ITS is not one that has received universal support. Among the dissenting voices has been that of Terry Winograd who developed an influential language understanding program in the 1970s called SHRDLU (Winograd, 1972). He later criticized the beliefs behind his own program and behind similar AI work. Together with Fernando Flores he produced a book (Winograd and Flores, 1985) that questions the value of a 'syntactic' approach to understanding human reasoning. This view was taken up by some influential researchers in the ITS area and an alternative model of learning was proposed: situated cognition.

3.2.2 Situated cognition

Modularity and abstraction have become key features of program and system design. Inevitably, when computer scientists attempt to model the brain, such ideas tend to be thought of as relevant. The design of the computer itself, with its memory, processing unit and interface could be seen as a useful analogue to the functioning of humans.

Any model, by definition, is incorrect in some way. If it was not it would not be a model, it would be the real thing. To be useful, however, any model must have one or more features that reflect reality. Some researchers question the usefulness of models that compartmentalise concepts such as memory and knowledge. There seems to be little physical evidence for knowledge existing in the brain as a separate entity. Some neurological research (Rosenfield, 1988) even calls into doubt the existence of *memory* as a separate phenomenon.

Scientists remain puzzled about how the physiological activity in the brain relates to what we know of people's thought processes (via reflection and protocol analysis for instance). One way of deducing the nature of thought processes would be to examine how people appear best to learn.

Humans have developed methods of teaching that emphasize abstraction and so it might be inferred that the brain stores information in an abstract form. Abstraction involves finding patterns in activities, extracting them and representing them in some formal way. For example, operations on numbers are used to teach children about buying and selling, weighing, measuring volumes, etc. Frequently, the actual tasks of purchasing, weighing and so on are taught later, as applications of the processing of numbers. At a later age, the same approach is used in topics such as electronics where the theory of electrical currents is taught before the student is exposed to electronic devices. Although these methods have been used for years and, in some cases, centuries, there are some experts who question the validity of this approach and the corresponding inferences about the nature of thought. Among those who have adopted a different view are two prominent ITS researchers, John Seely Brown and Alan Collins.

In 1989, Brown, Collins and Duguid provided support for the view that knowledge cannot be separated out from the context in which it is used. They believe that components of knowledge index the world and are 'inextricably a product of the activity and situation in which they are produced' (Brown *et al.*, 1989, p33). This view on the nature of knowledge is termed *situated cognition*.

Given this premise, the authors argue that students will best learn by carrying out tasks rather than by first being given a theoretical model. A comparison is made with the learning of words divorced from their context. They claim that students who have acquired vocabulary from dictionary definitions invariably use the words wrongly since the description in a dictionary can never be adequate. Only by observing others using a word, by reading it in the proper context or by trying to use it oneself can one come to

a proper understanding of its meaning over a period of time. Similarly, they suggest, knowledge of how to do a task cannot be obtained by starting with a theoretical description which is then applied. Thus some form of *situated learning* is required in order to best assimilate knowledge.

An analogy is drawn with the learning of crafts where the apprentice initially watches the expert at work and, after a while, takes over some of the operations (*stealing moves*). Eventually, the apprentice is allowed to carry out the complete task under the gradually reduced supervision of the teacher. Brown believes that a similar approach can be used with cognitive tasks as well as for manual ones. In fact, it is suggested that there is no clear distinction between physical and mental tasks since virtually all tasks require both faculties to some degree. This method of learning from the expert is termed *cognitive apprenticeship*.

In this initial paper, the ideas are not developed to any great degree and the authors admit that they are deliberately speculative. For example, details of exactly how cognitive apprenticeship could work in the classroom or university environment are not given. Even so, the paper generated a great deal of debate. Much of the cognitive science view of knowledge seemed to be challenged. Certainly, Wenger's idea of knowledge being communicable in some direct way and almost palpable is regarded as mistaken.

The reaction of advocates of knowledge communication was awaited with interest. Much of the work by Clancey on GUIDON, for example, had assumed that the knowledge of the expert could be represented in rules and that the aim of teaching was to communicate these procedures to the student. In an address to the Annual Meeting of the Educational Research Association at Boston in April 1990 (later published in a set of IFIP collected papers) Clancey (1991) gives his response. As Winograd had done, he had changed his view and decided that his earlier work had been based on false premises. He and others had previously believed the human memory contains representations that could be processed and that this was the essence of knowledge. However, it was now clear that this was not the case.

He admits that representations such as semantic nets, grammars, schemata and so on are useful and even necessary in some circumstances but that they should not be closely identified with what occurs in the human brain. These notations can be used to model how people interact with the environment and provide a succinct description of objects and processes, but what goes on in the heads of humans is something completely different.

Previously, Clancey had assumed that the human memory was very much like a knowledge base containing some equivalent to *if-then* rules. He now believes that the brain does not contain such static information but constructs necessary internal representations as needed by processing the external representations (perceptions). We can consciously try to analyse what is going on and devise appropriate representations but these bear no relation to *how* we are thinking.

Like some earlier researchers, Rosenfield (1988) for example, Clancey regards memory and knowledge not as having any physical properties but as abilities. Memory is seen as a 'capacity to directly reenact and compose previous ways of behaving' (Clancey, 1991, p61).

In an address to ITS '92, Clancey (1992) further develops his ideas. Knowledge is perceived as a capacity to interact which comes into being during the interaction itself. This ability improves as more and more cases are considered of a similar type, but the evolution never stops since circumstances are always changing as are our perceptions of reality. Any attempt to model this phenomenon by rules or structures is doomed to failure since we deal with each new situation as it arises depending upon the circumstances. These circumstances may not have been met before or even anticipated but they must be considered. How people cope with new situations is generally explained by using such terms as *judgment* and *common sense*. Clancey claims that no systematic structures or algorithms can denote them - although we can describe them in this fashion well enough *after* the event.

One could attempt to generalise the rules and structures by having meta-rules for interpretation but this only transfers the problem to another level since a further set of rules would be needed to interpret the meta-rules and so on *ad infinitum*. Such an approach would limit our actions in a way that does not reflect reality. As a final refutation of the schema model, Clancey states that this is not how the brain works since there is nowhere to store representational structures.

Clancey describes the implications of his beliefs for teaching and the development of ITSs. He suggests that the schemata view can prejudice methods of education if we believe that what we are attempting to do is to build up structures and matching processes in the student's mind. By attempting to impose set procedures and structures upon students we inhibit their learning and their capacity for innovation. Learning should take place in a community and involve interaction with other learners and with experts. Learning by doing becomes not just a good way but the *only* way of developing understanding. Not only this, but others should be involved too. For

example, medical students should learn about how hospitals work, what nurses do, patient care and so on, and not just about diagnosis of diseases.

In order to create more useful and usable systems Clancey now works with a wide range of groups particularly social scientists. He is taking what he calls a *socio-technical* approach to system development where everyone who is relevant to the final system works together in a democratic fashion. This includes experts, teachers, users, designers, programmers, administrators and anyone else who may have a contribution to make.

Although some of the most prominent researchers in teaching by computer have advocated the situated cognition approach, no overall methodology for teaching has been developed. It is unclear to what extent learning by doing is seen as a necessary component of learning. For example, the Cognition and Technology Group at Vanderbilt University (CTGV, 1993) cite the programmes that they have developed as examples of situated learning. These include the Jasper Woodbury Series of videodisks for teaching mathematical principles. In each of these videos, Jasper is in some predicament or is trying to help someone and, in order to achieve his aim, he needs to apply some mathematical methods. So, in one story he has to work out whether a micro-light aircraft can reach an injured eagle. In order to help Jasper, the class has to find out the relationships between time, distance and speed. In another story various geometrical principles are useful. The authors argue that this is in line with situated cognition since realistic contexts are provided in which the users are encouraged to construct their own knowledge. They stress the importance of not using the videos to consolidate theoretical material but to use the stories to motivate the students to find out about the theory.

Critics have argued that these, and similar programmes, are not strictly in line with situated learning. For example, Tripp (1993) believes that the Jasper programmes do not use the principles of situated cognition since users are not learning in the 'real world' from experts. He points out that the principles to be learned have already been abstracted out and have been embedded in an artificial scenario. What the students are exposed to is not the real world but a simulation. Certainly, if one accepts Tripp's view then situated learning becomes very difficult to provide since nothing but the real world provides the kind of experience that the student needs.

Tripp suggests that, in view of this, we should continue using traditional teaching methods. Not only are they the only currently feasible ones but they still have advantages. Principles can be developed in a classroom situation where noise that may

distract students from learning can be eliminated. Since the nature of what is being taught is clearly delineated it can be tested by dependable means. Once the student has this knowledge consolidated it can then be tried out in practical situations and augmented as necessary.

The situated learning supporters would argue that it is precisely the separate manipulation of symbols in the classroom and their meaning in the real world that they wish to avoid. Brown (1990) observes that symbols as developed in the classroom take on a meaning of their own, totally unrelated to what they are modelling in the real world. In the same paper, he cites work which shows that learning in an abstract situation may impair the ability to apply such knowledge in practical situations whereas students who learn during carrying out tasks seem to be able to apply the knowledge they have acquired to different problems.

The views of supporters of situated cognition can be summarized as follows. Knowledge cannot be disembodied from the user and the context. It can be represented, perhaps in a diagram, by logic, or on the computer but in the translation process vital ingredients are lost which cannot be replaced. The models can be used in a restricted way for problem solving but do not equate in any way to the knowledge that a human would bring to bear on the same process. The computer representation is fixed, disembodied and cannot adapt to circumstances. User's knowledge is constructed as needed and is totally dynamic.

It might seem that the rift between the supporters of knowledge communication and situated cognition would mean that a system developer has to take one view or the other in order to produce a coherent piece of software. This does not have to be the case, however. There is common ground and the link between the views is the idea of modelling. If one considers models of the different aspects of the teacher/student interaction, such as a model of the domain and of the problem solving process, then the main point of debate is not what these artefacts should look like but how they are constructed. An advocate of knowledge communication would attempt to get the student to understand a computer model in a fairly direct way. A supporter of situated cognition would attempt to get the student to build up their own model. Brown (1990), for example, sees the education process as facilitating the building of adequate mental models by the user. So, it would appear that modelling is a central concept in education.

3.3 Modelling and fidelity

A model may be defined as the 'cost-effective use of something in place of something else for some cognitive purpose' (Rothenberg, 1989, p75). The cost is not necessarily monetary: it could be convenience, feasibility, safety or any one of a number of factors that make it more desirable to deal with a substitute rather than the original (the *referent*). Generally, the model will have some relationship to the referent but often the relationship may be implicit or vaguely specified. At least one feature of the referent will carry over, otherwise there would be no point in producing a model. Conversely, it will not be an exact duplicate since, in this case, there would be no savings. Often the model will be simpler than the original, but, as will be demonstrated later, that is not necessarily the case.

The purpose of models gives one common method of classification. At a broad level a model may be descriptive (for describing or explaining the world) or prescriptive (used for solving problems). In each case there is a notion of *understanding* phenomena, and this provides the cognitive link. The purpose is crucial and must be included in the specification, otherwise it is impossible to determine what features of the original should be represented. A good example of a clear explication of purpose is given by Kunz *et al.* (1989) when they describe the building of a model of a carbon dioxide removal system to be used by NASA engineers (see Figure 3.4). Here, all the reasons for building the model are laid out in a precise fashion, enabling the designers to work out exactly what features of the original need to be represented and how to approach building the model.

- To demonstrate the functions of the modeled system when operating normally
- To identify all likely single faults in the modeled system and characterize modeled system functions when operating with any such fault
- To develop diagnostic data collection, fault detection, isolation, compensation, correction and reporting procedures for the CO₂ removal system
- To investigate changing system operation mode in event of a detected fault to protect it and its environment from further damage or failure
- To test diagnostic procedures, including procedures to make the system safe while preserving ability to collect diagnostic data
- To introduce faults into the model and fix faults in the model
- To explore heuristics for design and diagnosis

Figure 3.4 Purposes of CO₂ removal system model (Kunz *et al.*, 1989, p93)

Listing the purposes of a model will help determine what *aspects* of the referent are to be modelled and what *form* the model is to take. The form specifies its manifestation which may be as a physical object, a computer program, a description in some formal notation or it may exist in the mind of a person (a *mental* model). Although the form is one of the most obvious properties of a model it is not the most crucial. The aspects or properties of the original that are to be copied will usually indicate or even dictate what form the model is to take.

The aspects of the original to be modelled will be determined, of course, by what is of interest in the original. An important feature of an object that is often modelled is its change over time. In this case the object is being regarded as a *process*. As Lenat and Guha (1990, p205) observe, every object changes over time and so can be considered as a process. Whether we treat it as such depends upon whether those changes are of particular interest and significance.

3.3.1 Process models

Processes are often represented by physical analogues or by mathematical equations. An example of the former is the use of a model of an aircraft in a wind tunnel. Measurements can be made on this and the likely effects on a full-sized plane of similar composition and design can be assessed. An example of a mathematical model is the use of probability functions to represent what happens in a queue over a period of time with particular patterns of service and arrivals.

With the advent of computers a new medium for representing processes became available. A computer program, when it is executed, progresses through a sequence of states. If these states have some correspondence with the states of a process then a useful model may be developed. This is the basis of *simulation* which can be regarded as a dynamic model of a process that 'unfolds over time' (Rothenberg, 1989, p82). Computers are not essential for simulation, of course. The changes in a model could be recorded on paper. Often, however, there are complex or large numbers of calculations to be carried out which makes anything but a computer model infeasible.

In all the examples described so far there has been a requirement to measure *quantitative* aspects of a model whether it is the level of vibrations in a model aircraft or the number of elements in a simulated queue. If the model is to be used for teaching purposes then numerical calculations may be less significant than *qualitative* considerations.

Qualitative models are of great interest in artificial intelligence since, as observed in the previous chapter, both experts and novices seem to rely on descriptions to guide their problem solving. Clancey notes that such models describe systems in terms of 'causal, compositional or subtypical relationships among objects and events' (Clancey, 1989, p10). From this viewpoint, knowledge bases are predominantly qualitative whether they represent expertise at a *deep* or *shallow* level. More information on the contrast between these two approaches to knowledge representation can be found in papers by Chandrasekaran and Mittal (1983), Hart (1982) and Michie (1982). Basically, shallow knowledge bases represent observable rules and heuristics that experts employ. In contrast, deep methods attempt to represent a fuller knowledge of the subject area and the reasoning that the expert uses.

For our purposes, Clancey's categorization of process models is a useful one and is reproduced as Figure 3.5. Classification models are often at a shallow level and seek to provide some pattern with which the original can be compared. The heuristic classification provided by the rules in such systems as MYCIN (Shortliffe, 1976) and MUD (Kahn and McDermott, 1984) are typical examples. Frames and scripts are tools that are often used to aid classification.

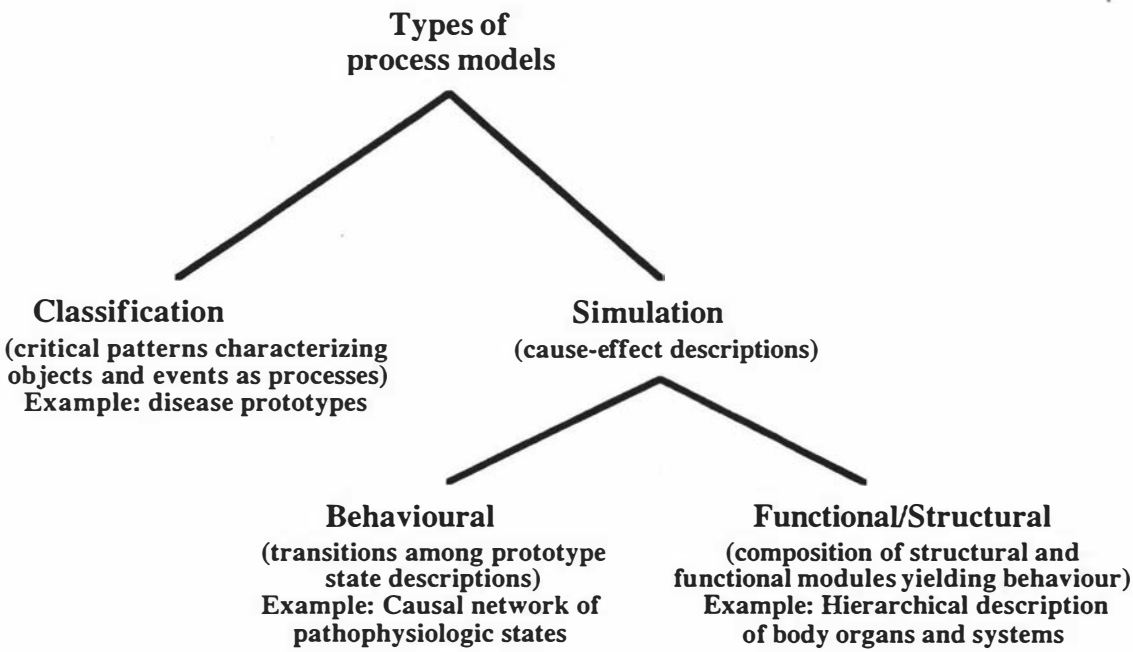


Figure 3.5 Qualitative models of processes (Clancey, 1989, p12)

The distinction between behavioural and functional/structural simulation is important and has been a source of confusion in the area of expert systems. For example, Bobrow (1984) in a special issue on qualitative reasoning considers only functional/structural models. It is useful to elaborate further on these two different kinds of simulation. Broadly, the distinction is between modelling *observable* characteristics of a system taken as a whole, and modelling the individual components, describing what they do and how they interact.

If there is a detailed model of how the observables in a system are caused (for example, NEOMYCIN gives a description of how diseases exhibit particular symptoms) then, by applying a reasoning procedure to this model for a specific case, a causal, *situation-specific* model (Clancey, 1988c) of the process can be produced. The behaviour of the system in particular circumstances is being simulated. This simulation, the kind that occurs in many deep expert systems, can be effective but is very limited. There is no attempt to model the behaviour of systems in any general way (for example, the physiology of a healthy patient) or to simulate the functioning of individual components (such as the heart or lungs in a medical model). Instead there is an emphasis on manifestations of particular kinds of phenomena and their causes. Often the model is expressed as system states linked in a causal associational network.

An example of a system that uses this idea is CIRCSIM-TUTOR (Khuwaja *et al.*, 1992). This is an ITS for teaching students the principles of the baroreceptor reflex, the part of the cardiovascular system that controls blood pressure. A qualitative causal model in the form of a *concept map* (Figure 3.6) is used as the basis for teaching the students the causal relationships between the different features of the system. For students who have problems with this high level view there are two more detailed maps which include more concepts and which also show the ways in which the different components affect one another (for example, neural or physical/chemical interactions).

In functional/structural simulation, there is an attempt to explain the behaviour of the system in terms of its composition and the interaction of components or processes in a very general way. There are several techniques for achieving this and the general approach is called *qualitative reasoning*. The term, as Clancey observes, is a misleading one but has been adopted by the developers of functional/structural models and so will be used here. The technique has been used for investigating phenomena, for developing expert systems, for system design and various other purposes (Weld and de Kleer, 1990) but many of the basic ideas were developed during research into computer teaching systems, particularly the work of Stevens and his colleagues on WHY, and the BBN team involved in the SOPHIE project.

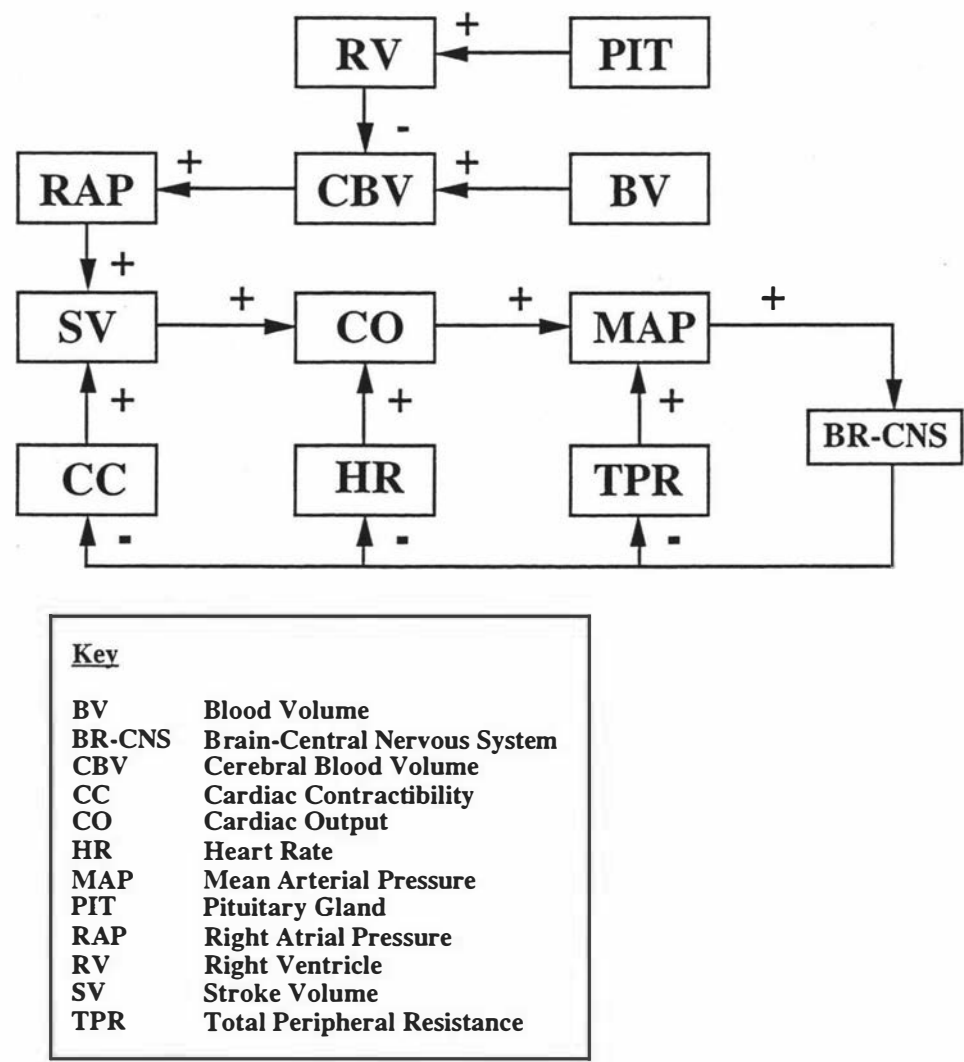


Figure 3.6 Concept map of baroreceptor reflex (Khuwaja *et al.*, 1992, p218)

The WHY system for teaching the basics of meteorology has been mentioned in Chapter 2. It has a script-like structure for representing causal and temporal relations and takes what Clancey would regard as a classificatory approach to processes. As Stevens, Collins and Goldin (1982) observe, this produces a useful but limited model. By executing the script the system can demonstrate what is happening and can match the student's explanation. If the student goes wrong at some point or misses a step then the system can detect this and explain what should be done. However, it cannot deal with certain kinds of misconceptions and has difficulty explaining the inter-relationships of the components of systems other than in a general fashion. For these reasons, Stevens and his colleagues propose another (functional) viewpoint for filling in these gaps in the system's explanatory and diagnostic capabilities.

In this alternative model, each of the elements is considered as a separate entity and the role that each plays in the process is described. In the rainfall domain the process of

evaporation can be viewed from this perspective. The ocean can be regarded as an actor with the role of providing moisture. Each of the actors has attributes that may affect the process (for example, temperature and altitude of air-mass). Functional relationships between factors explain the effects of interactions. For instance, the temperature of an air-mass is inversely related to altitude and so the air cools as it rises.

The domain of electronic circuits is very different from that of meteorology, yet the workers on the SOPHIE project (Brown *et al.*, 1982) came to a similar conclusion to the WHY researchers; that is, in order to provide insight into the operation of a model, the functioning of the constituents needs to be considered. The original SOPHIE was built around a quantitative simulation of electronic circuits. A knowledge base of information about the simulator was built on top of this that allowed the user to interact with the program and learn about fault detection in electronics. With this interpreter between the user and simulator the program could monitor the student's attempts at troubleshooting to see if s/he was using a logical approach, answer hypothetical (what-if) questions about the circuit, and suggest hypotheses for the student to try.

As with WHY, the weak point of SOPHIE I was explanation. As Wenger notes in his assessment of the package: 'it makes no use of the kind of causal reasoning performed by human troubleshooters . . . causality is pedagogically important because it is the main ingredient of the kinds of explanation human students can understand' (Wenger, 1987, p62). SOPHIE II (Brown *et al.*, 1982) attempts to remedy this defect by providing a troubleshooting expert that can find faults in the circuit. Thus, for example, the user can introduce a fault and see how the program detects it. The procedure that the program follows, however, is quite brittle (essentially a decision tree). Although the system can provide causal explanations of its steps, these are pre-stored and of a general nature and do not relate to the specific circuit. Also, the program cannot monitor a student's troubleshooting procedure if this deviates at all from its own actions and ordering.

In summary, SOPHIE II papers over the cracks in SOPHIE I. At the core of the problems with both systems is the use of a numerical program for simulating the behaviour of the circuit. In circuit analysis, as in many other domains, problem solving involves not just learning particular procedures but having a deep understanding of the system under consideration. The numerical simulator is essentially a black box and the interactions between components and the overall structure of the circuit has been coded in a mathematical form to which the user cannot relate.

SOPHIE III (Brown *et al.*, 1982) provides an alternative approach. The simulator is replaced by an electronics expert that is knowledge-based and has a qualitative component. The expert has knowledge about the principles of electronics and uses this in conjunction with an automated troubleshooting expert to generate a situation-specific model of a given circuit which is represented as a *behaviour tree*. In this tree the module behaviours, structure relations and interactions between modules are stored. The information in this tree can be interpreted to provide a kind of simulation of the behaviour of the circuit.

The SOPHIE III approach still involved interposing a layer of processing between the description of the behaviour of the circuit and the user. Two of the researchers on this project, de Kleer and Brown, wanted to produce a more direct causal model of circuits and of other mechanistic devices, based on people's perceptions of how they operate. In a 1981 paper (de Kleer and Brown, 1981) they expound their views on how this might be achieved. They believe that the key to providing a model that people can relate to is to make it as close as possible to the mental model of the device that they have or that an expert might employ. It is claimed that this model is essentially qualitative and based on cause-effect principles.

When seeking to produce mental models, de Kleer and Brown aim for an ideal rather than the reality. As Norman (1983) observes, most people's perceptions of how devices work are incomplete, unstable, confused, illogical and limited. de Kleer and Brown are looking for models that are descriptive, have simple to understand cause-effect sequences of operations, and that correspond to the structure of the original in some direct fashion. The models must also be logical, general, robust and consistent.

In order to preserve the structure of the original device each of the components of the referent must be modelled. The behaviour of each component must also be described, and, since each component may be a module in itself, this is recursive. A key concept is *no-function-in-structure*. That is, when describing how a component works this description must not depend upon the behaviour of the system as a whole but only on how it is directly affected by, and affects other components of the system. As well as allowing localized behaviour to be explicitly described, the no-function-in-structure principle makes the system robust. If the circuit is altered or faults are introduced at a local level then only the portion of the model corresponding to the modified part of the original needs to be altered. Essentially, this is an *object-oriented* approach.

Related to the principle of no-function-in-structure is that of 'weak causality' which states that every event must have a direct cause. Again, the implication is that the

reasoning used for determining the next state change should be localized and should not use any indirect information. A development strategy using these principles can, according to de Kleer and Brown, provide a qualitative model that corresponds closely to that an expert might employ and which could explain many aspects of the behaviour of a system.

To illustrate their ideas, they develop a sequence of models of an electro-mechanical buzzer which can be used to answer useful questions about its behaviour. It is important not only to be able to describe how it operates normally but to be able to determine what would happen if various changes were made such as reversing the leads of the battery or shorting the switch contact. Figure 3.7, taken from Wenger (1987), illustrates one model. Figure 3.7a shows the original circuit in typical schematic form, indicating the individual components and the electrical connections between them. Figure 3.7b shows state diagrams for each of the components and for the three components linked together. Lastly, in Figure 3.7c a causal model shows how behaviour is propagated through the system as changes are made. de Kleer and Brown believe that these kinds of models, by illustrating how individual components interact and by making any implicit assumptions explicit, provide a useful target for students learning about devices.

There are two other main strands to qualitative reasoning that should be mentioned although they have had less influence on the research conducted in this thesis than has the work of de Kleer and Brown. Qualitative process theory (Forbus, 1984) takes a different approach to conceptualizing mental models. Forbus focusses on processes, how they are enabled, relationships that are satisfied during their occurrence, and how they influence individual entities involved. In line with his work on STEAMER, Forbus attempts to describe systems at a conceptual level. Also he looks at systems as perceived by people in general rather than experts. This work is related to Hayes (1985) research on naive physics.

Kuipers (1986) takes a third view by looking at qualitative measurements within a system. The term qualitative measurement may seem like an oxymoron but refers to attempts to describe values of variables in a general way. For example, we might distinguish between an electrical current value that was negative, zero or positive without determining its exact value. Again, causal methods are used to show how the system changes, although, in this case, the causal reasoning is a form of constraint propagation. Values are propagated through the system and constraints determine which are feasible. So the emphasis is on a mathematical model of the system rather than on the interaction of components or processes.

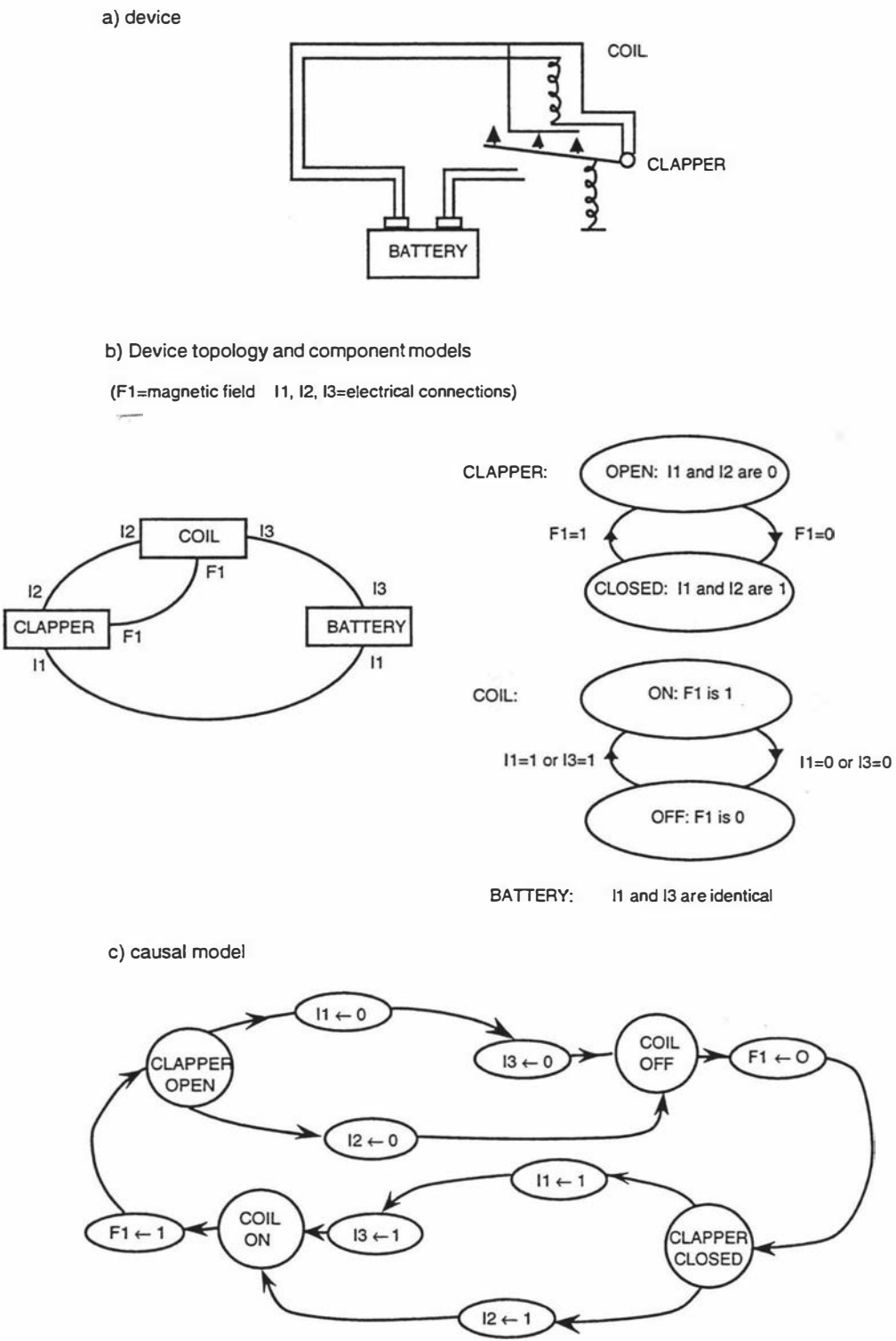


Figure 3.7 Qualitative description of buzzer (Wenger, 1987, p72)
based on de Kleer and Brown (1983)

3.3.2 Fidelity of models

It has already been noted that one of the key characteristics of a model concerns the aspects of the referent that are copied. If a model is going to be used for teaching purposes we have to consider what features of the original should be included in the model and how they are to be represented. A second, associated, consideration is the model we want students to have once they have used a computer program. These ideas can be encompassed within the notion of *fidelity*. This relates to the degree of correspondence between the model inside the computer and the model that we wish the user to acquire (normally a mental model).

Historically, when producing software for the computer, programmers have sought to exploit the machine's potential to perform large numbers of computations in a short time and to sift through massive amounts of data. Humans do not have the same capabilities and so cannot be expected to follow the steps that a computer takes in order to get results.

Computer chess programs provide an interesting example. These have been particularly successful (Marsland and Schaeffer, 1990; Coles, 1994) and the best ones, it can be reasonably claimed, play at the level of grand masters. This has been highlighted by the defeat of the Professional Chess Association world champion, Garry Kasparov, by the Pentium Genius computer program, albeit in a speed chess competition (Barden, 1994). However, the strategies used in computer programs do not generally correspond to the ones that humans appear to employ. The most effective algorithms sift through millions of possible moves, searching for ones that satisfy some numerical assessment of 'good' board positions. Although this produces reliable results, justifying the moves that are selected in terms that a novice or even an expert could understand would be extremely difficult, if not impossible.

Burton and Brown discuss the form that domain expertise may take within a computer program. A *black box* program has data structures and algorithms that do not mimic those used by human beings. In a *glass box* or articulate model, they note that 'each problem-solving decision it makes can, in principle, be explained in terms that match (at some level of abstraction) those of a human problem-solver' (Burton and Brown, 1982, p81). As will be demonstrated, there is a spectrum of possibilities between black and glass box models and also other subtle but important related distinctions.

Burton and Brown (1982) describe a game called WEST, that users can play against the computer, and that is designed to improve their arithmetic skills. A set of numbers

can be manipulated using the arithmetic operators of addition, subtraction, multiplication and division, with the result being used to determine moves on a playing board. The human opponent often has the choice of several ways of manipulating the numbers, but getting the largest result will not necessarily be the optimal choice since one can take short cuts, 'bump' the opponent, or get a bonus on certain squares. Advice is available from a coach that has a complex algorithm for determining good strategies. This is essentially a black box algorithm since it is not one that a human is likely to employ. However, Burton and Brown seek to bring in a glass box component by having a module that examines any inferred sub-optimal moves by the human, relates that to a general strategy that the user seems to be missing, and then has some canned text describing the benefits of that strategy.

To illustrate this approach it is convenient to consider chess rather than the WEST game since the former is more familiar to most people. Consider a situation where the computer is monitoring a user's moves during a game of chess and giving advice. At some point, suppose the user contemplates a move that places his/her queen in such a position that it can be taken next move. If the computer is asked for its advice on what move to make, it might use a complex algorithm to determine that the best choice would be to castle. When asked why the user's proposed move is not advisable the computer might delve into its store of canned explanations and find a piece of text that explains why it is not usually a good idea to sacrifice a queen. Similarly, to justify its own suggestion, it could describe the general benefits of castling.

Two major problems with this so-called *issue-based* tutoring (Burton and Brown, 1982, p83) are apparent. First, it pre-supposes that the computer is employing a better algorithm than the user and that the latter has not spotted a superior move (perhaps by making a spectacular sacrifice). In these cases, the computer might actually inhibit learning. Secondly, the advice given does not relate to the specific situation that has occurred. The computer has attempted to recognise the pattern or issue that has caused the mismatch and has produced an explanation related to that issue, not to the context of the specific problem. If the user suspects that his/her move is better or cannot see any justification for the computer's move then the system will be very limited in the help it can give. Also, it could not attempt to explain how the user got into the predicament or what the long term, or even short term goals should be after castling.

There are many other examples of attempts to use a black box for computation and to interpose some kind of facilitator between it and the user to provide 'transparent' explanations. For example, the early versions of SOPHIE had a black box, numerical

program for computing results with pre-stored, general explanations. Also, STEAMER's operation was completely hidden from the user.

A numerical model is at one extreme of the black box / glass box spectrum. In one sense, it is a model of a model, since a mathematical description of the referent's behaviour will have been produced and then a numerical model constructed that satisfies the equations. This is far removed from how a human might learn the basic principles of a domain. An expert system provides a more intelligible representation of its referent: a specialist's knowledge. Even so, many shallow expert systems have hidden knowledge and compressed causal sequences when heuristics are employed. Also, they may have complex algorithms for computing likelihoods that would be difficult to explain to a human. Most expert systems, of course, give some form of explanations such as describing how a result has been computed or justifying why a question needs to be asked. Such explanations are of limited value in teaching for several reasons. Firstly, as already noted, the heuristics often compress steps or have no meaning for a non-expert. Secondly, such audit trails inevitably leave out a large number of rejected possibilities that the system may have examined. Thirdly, the overall strategy that the system uses may not correspond to how a human would solve a problem.

Another aspect of models that needs to be considered is what is to be conveyed to the user: what mental model will s/he have once s/he is familiar with the system? The designers of STEAMER (Hollan *et al.*, 1984) deliberately set out to convey a model to the user that the expert would have. An expert might have a detailed knowledge of a mechanism but some of it would be irrelevant for reasoning purposes whereas other parts of the system might be re-conceptualised into a form that makes it easier to consider in the problem solving process. Much of the work at LRDC considered in Chapter 2 examined how the expert's perception of a system differed from the novice's. Hollan sought to fast-track the learning process by presenting the user with a view of the system that corresponds to that of the expert. To achieve this, various methods were used including the presentation of graphical material, the ability to examine transient phenomena, and ways of manipulating the system that were not directly available to an operator. Hollan used the term *conceptual fidelity* to refer to the ideal of faithfulness to the expert's abstract conceptualization of the system rather than to its physical properties.

It could be argued that all of these models are predominantly black box since the method used by the computer to work out results is fundamentally different from that which humans would be expected to use. In the expert system field, as already noted,

Clancey's NEOMYCIN system was amongst the first where an attempt was made to use reasoning processes that a human might understand rather than the faster, more efficient, but ultimately opaque methods that had previously been employed in knowledge-based systems. A model of the deep-level reasoning expertise of medical specialists was incorporated into NEOMYCIN.

Although NEOMYCIN incorporates much expert knowledge in a transparent form its method of problem solving matches neither that of the expert nor the novice. As observed in Chapter 2, the expert tends to use data-driven methods whereas the novice tends to use a goal-driven approach. Also the t-rules that incorporate tutorial knowledge (Clancey, 1987a) help to bridge the gap between the steps that the student is taking and the method used for problem solving by the system. The inference engine in NEOMYCIN is fairly simplistic and generally uses a goal-driven strategy but can present it to the user in a form that is more similar to the 'expert' method. Unfortunately, not enough is known about how an expert really uses his/her knowledge to solve problems to be able to convey this to a novice. What the user can be shown is an explanation of how results were obtained and what they might do next. They may be able to readily understand this but it often corresponds neither to the way that the expert might think nor the sequence of steps that the user might take. This degree of faithfulness to human problem solving is termed *epistemic fidelity* by Wenger (1987, p313).

A system such as GUIDON that has a high degree of epistemic fidelity can be used as an effective teacher. All the appropriate medical terms can be used in the right contexts; knowledge of links between symptoms and diseases and reasons for these can be learned and consolidated; plausible sequences for determining diseases from symptoms and for deciding what other information is required can be observed. The only missing component from the computer system is the capacity to solve problems in the same way as a human and to follow and comprehend the problem solving logic of the user. To achieve this, the computer must solve the problems in the same way that an expert might do and must be able to consider the alternative methods that a user might employ and check on them.

The extent to which a system's methods are similar to those used by a human is briefly mentioned by Wenger (1987, p15) when he discusses *psychological plausibility*. The concept is further described and developed by Anderson in his discussion of cognitive modelling: 'The goal of the cognitive modeling approach is to effectively develop a simulation of human problem solving in a domain in which the knowledge is decomposed into meaningful, humanlike components and deployed in a humanlike

manner' (Anderson, 1988, p34). In other words, not only is the human knowledge represented explicitly in the computer but also the ways in which a human might use this knowledge are also modelled. The term *cognitive fidelity* is used by Anderson to refer to the degree to which the computer models human problem solving methods.

Beeson describes a system called MATHPERT for teaching mathematics that attempts to achieve a high degree of cognitive fidelity. He states that 'MATHPERT produces not just "answers", but full step-by-step "solutions", with each step intelligible to the user and justified on-screen, and does so not by secret high-powered algorithms, but by the same methods students are supposed to use' (Beeson, 1989, p9). At the heart of the system are so-called *operators* for transforming expressions. The operators have been chosen to correspond to identifiable chunks of knowledge that can be taught and learned. For example, one operator called 'collect powers' takes an expression of the form $x^n x^m$ and transforms it to x^{n+m} . A student model (basically, an overlay) checks which of the operators the student appears to know (from previous usage) and if the student is already very familiar with the operator then the operation will be carried out automatically. Although it is claimed that the system solves problems in the same way that a user might, there is no detail on exactly how this is achieved. This is a key consideration since, as noted earlier, having the knowledge, and applying it in an appropriate fashion are two different matters.

A good example of a scheme where a cognitive modelling approach has been used is QUEST (Qualitative Understanding of Electrical System Troubleshooting) developed by White and Frederiksen (1990). They have implemented a teaching system for electric circuitry that is based on de Kleer and Brown's work on qualitative reasoning. The method often used in schools to teach electric circuits is a classic example of throwing the students in at the deep end, presenting them with complicated concepts such as Ohm's Law before they fully understand the basic principles of current flow. Consequently, as the authors note, even college Physics students have trouble with *zero-order* concepts (that is, relating to straight on/off phenomena such as whether a current will flow in a specific circuit).

QUEST aims to teach basic zero-order concepts and then, when the student gains mastery of these, to move on to more complex notions. This reflects the view of Glaser's research team (Glaser *et al.*, 1985) that gaining expertise is like developing more and more realistic models of the domain and its working. The system exhibits a high degree of fidelity. Knowledge about the circuit is represented in a glass box form. There is a direct mapping of the device topology and of the physical components of the model. In addition, the voltages and currents in each circuit are computed by the

system, by employing the same techniques that the student is expected to use. These methods are intuitively easy to understand, depending upon a notion of causality where disturbances are propagated through the network. de Kleer and Brown's 'no function in structure' principle is adhered to but only at the cost of limiting the kinds of circuit that can be considered to fairly simple ones.

Even at the zero-order level there is a sequence of increasingly sophisticated ideas that need to be conveyed, starting with basic notions of voltage and conductivity, and progressing to troubleshooting 'shorts to ground' in a series circuit. Corresponding to these ideas is an increasingly sophisticated sequence of models. The problem solving methods are based on causal techniques developed by a human circuit expert.

Since the internal representation of circuits in QUEST is based on principles and descriptions of device components that are circuit independent, it can handle any circuit given a topological description. Thus the student can set up his/her own circuits with which to experiment.

The advantages of White and Frederiksen's approach include the ease with which effective explanations can be incorporated. The algorithms are written in such a way that the reasoning processes used when simulating the behaviour of a given circuit, or when troubleshooting, can be easily explained to the user. Also, the student can build circuits and get the system to analyse them and report back using the same terms of reference. This last property allows the system to be used in a discovery mode. Any circuit can be built up, its properties observed and, significantly, its characteristics explained by the system.

Results from the QUEST project seem promising. A group of students who were having trouble with the basic concepts of electricity were, after five days using QUEST for one hour per day, able to make accurate predictions about circuit behaviour and could solve elementary trouble-shooting problems. The next challenge is to modify the system to incorporate first order ideas using a modified qualitative approach that takes into account relative values.

Potential drawbacks to the QUEST approach include its use of a cause-effect model and its domain-specific approach. This relates back to the idea of one phenomenon causing another. However, it is not clear what meaning this has when considering instantaneous changes such as the relationship between current and voltage: does a change in the current lead to a particular voltage, or is it the voltage that produces a

current? The problem could be addressed by taking the slightly different view of time-dependent phenomena adopted by Kuipers (1986) in his qualitative simulation model.

One of the few other systems where a cognitive modelling approach has been used is FysioDisc (Winkels and Breuker, 1992) for training physiotherapy students. Winkels and Breuker note that it is important during knowledge acquisition not to model observed expert behaviour descriptively but to attempt to discover the process models that cause this behaviour.

3.3.3 Classifying fidelity

Whether one believes in knowledge communication or in situated cognition, it is still important to strive for glass box technology. As Brown notes, 'The goal of design of any tool or device . . . should be to produce "glass boxes" which, first and foremost, connect users to the world' (Brown, 1990, p279). The advocates of knowledge communication would perceive that the model within the system is being conveyed directly to the user. Supporters of situated cognition such as Brown would use terms like 'facilitating the building of adequate mental models by the user'. Members of both camps would agree on the form of model, the disagreement concerns the cognitive mechanism by which the student comes to understand it.

As may be seen from the summary in the previous section, the term *glass box* is open to many interpretations. How do notions such as conceptual fidelity, epistemic fidelity, cognitive fidelity and qualitative simulation relate to this?

It is tempting to regard the transition from black box to glass box models in a linear fashion using the fidelity notion (see Figure 3.8), but this is a simplistic view. As will be demonstrated in this section, what is being modelled is as crucial as how it is modelled. A more detailed analysis is, therefore, necessary in order to better distinguish the different concepts and to determine with what aspects of the system they are associated. A useful approach is to consider the features of the domain and expertise that are being modelled.

First, consider the domain. This may be an electronic circuit, the working of human physiology or a game. We can often produce a model that simulates the behaviour of the referent: a mathematical model of an electric circuit, for example. This kind of model, particularly if implemented on a computer, is rarely open to inspection.

A human expert will have a model of the conceptual abstractions and causal relationships within the domain. The conceptual abstractions can be illustrated on the computer by using appropriate interface techniques such as graphics and animation. This is what is attempted in STEAMER. Various expert concepts are reified to aid the student's understanding of them. The causal relationships (as understood by the expert) cannot be depicted since the black box model will not have been developed with these as components.

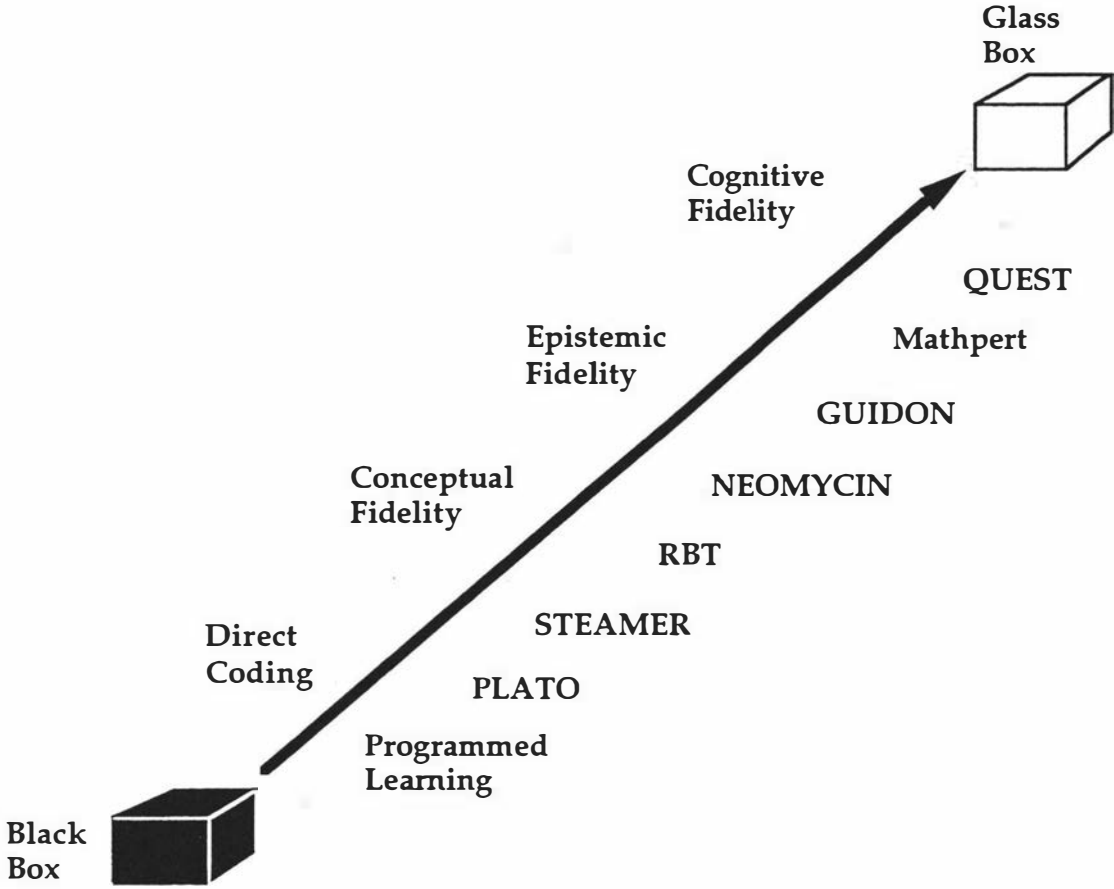


Figure 3.8 A linear depiction of fidelity types

Figure 3.9 illustrates the creation of a conceptual simulation program. On the left, the domain is depicted, in the middle is the expert mental model, and on the right the final program. Consider, first the domain box on the left hand side of the figure. This represents domain knowledge, comprising the knowledge of how the system operates and the problem solving knowledge that might be useful within the domain. A conceptual simulation system such as STEAMER does not attempt to model any problem solving and so this component is greyed out. Simulation of the behaviour of the system has two aspects. There is a black box model of the operation of the device, probably represented in terms of mathematical formulae, and numerical methods. This is denoted by the arrow connecting the domain description to the final program.

The main feature that distinguishes STEAMER and similar systems from simple numerical simulations (and what makes it an effective teacher) is the inclusion of expert knowledge of concepts reified in the graphics. An example is the use of icons showing the rate of change of variables. As Hollan and his colleagues note 'putting a layer of interface computation between a user and a quantitative model provides a graphical qualitative view of the underlying model . . . allowing a user to more directly apprehend the relationships that are typically described by experts' (Hollan *et al.*, 1984, p19). This aspect of the system is denoted in Figure 3.9 by the arrow linking the functioning of the system to the conceptual abstractions box in the expert mental model. These abstractions are used to augment the black box model of the system for teaching purposes. The precision with which the conceptual abstractions are combined with aspects of the black box model gives the degree of conceptual fidelity. The 'process of cognition', denoted by the dashed line, refers to the creation, by humans, of a high-level model. This model will represent components of the expert's understanding.

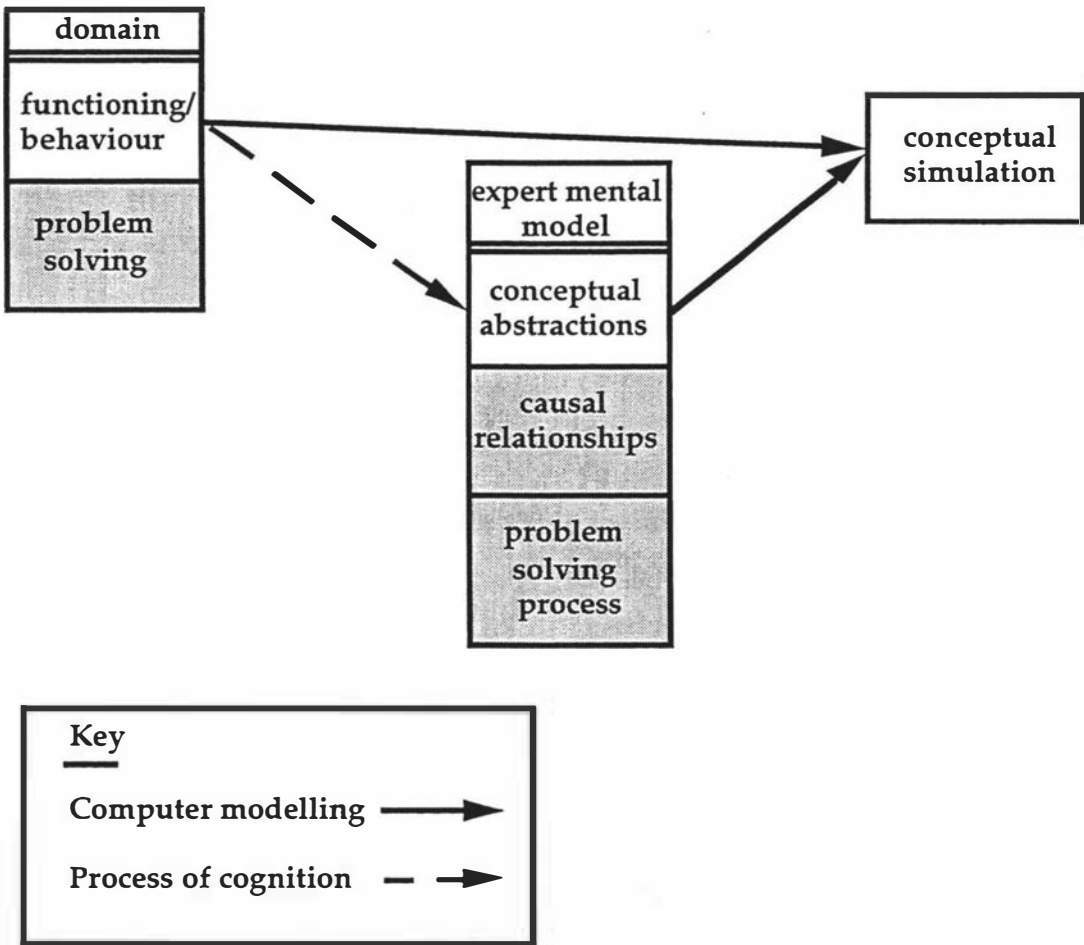


Figure 3.9 Conceptual simulation

Various simulation packages for teaching developed during the late 1970s and the 1980s fall into this category, including STEAMER, RBT and SOPHIE. A more sophisticated design for this kind of model has more recently been used by van Joolingen and de Jong (1992) as the basis for an intelligent simulation learning environment.

If, instead of a black box model of the domain, the conceptual and causal relationships of the expert are represented on the computer then this gives a qualitative reasoning model (Figure 3.10). Here, there is a deliberate attempt to model the behaviour of the system in terms of the expert's view of the concepts within the domain and the causal relationships that apply. A typical example would be de Kleer and Brown's model of a buzzer shown in Figure 3.7. Again, there is no problem solving component present. The aim of such models is to give a causal representation of how the system operates, not how problems might be dealt with.

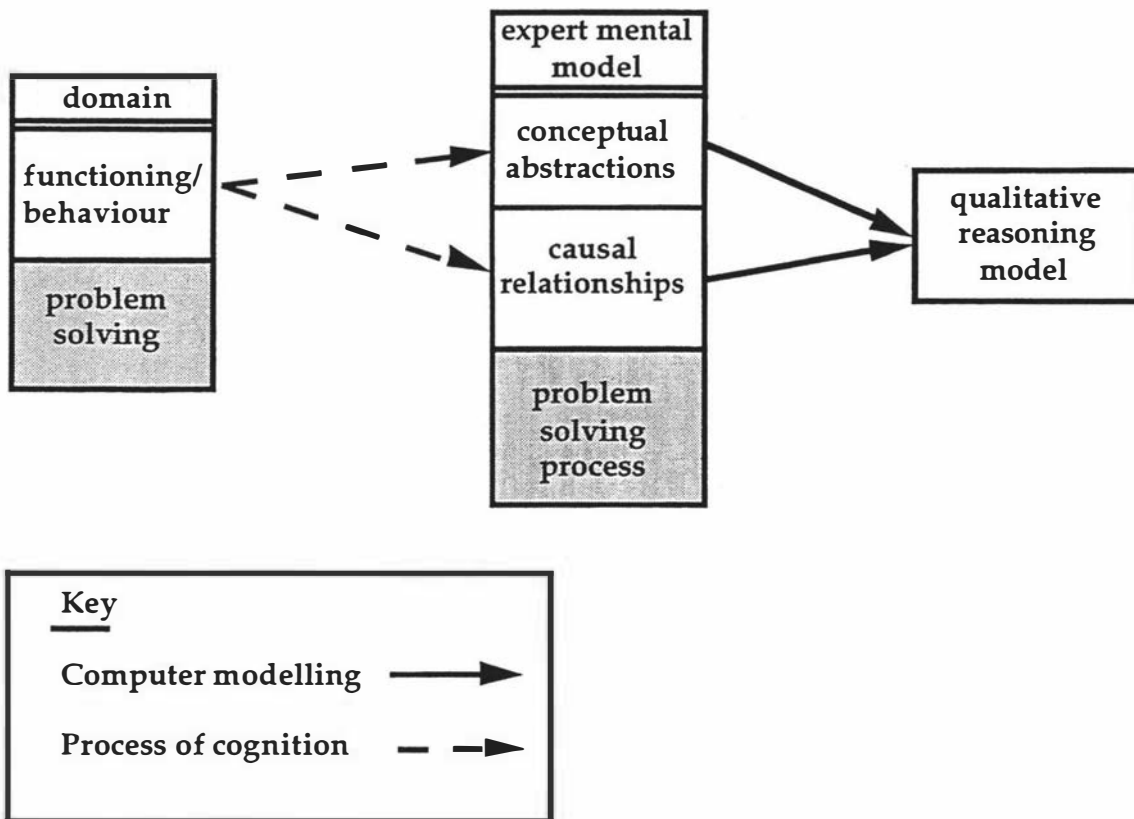


Figure 3.10 Qualitative reasoning model

Expert systems take into account not only the expert's knowledge of the domain but also his/her problem solving methods (such as the strategic and structural knowledge described in Section 3.2.1). The contributions to an idealized expert system are shown

in Figure 3.11. The problem solving knowledge may be coded directly as in many shallow expert systems or may be based on a model of the problem solving used by the expert as in NEOMYCIN.

Epistemic fidelity gauges the accuracy with which the expert's conceptual, causal and problem solving knowledge is represented explicitly in the final system. The more accurate the model is, the closer it will be to this figure. A faithful system will be able to use the expert's knowledge to solve problems in the domain and the expert will be able to understand the steps that the system is taking to get its solution (although these may not reflect the method that s/he would use in practice).

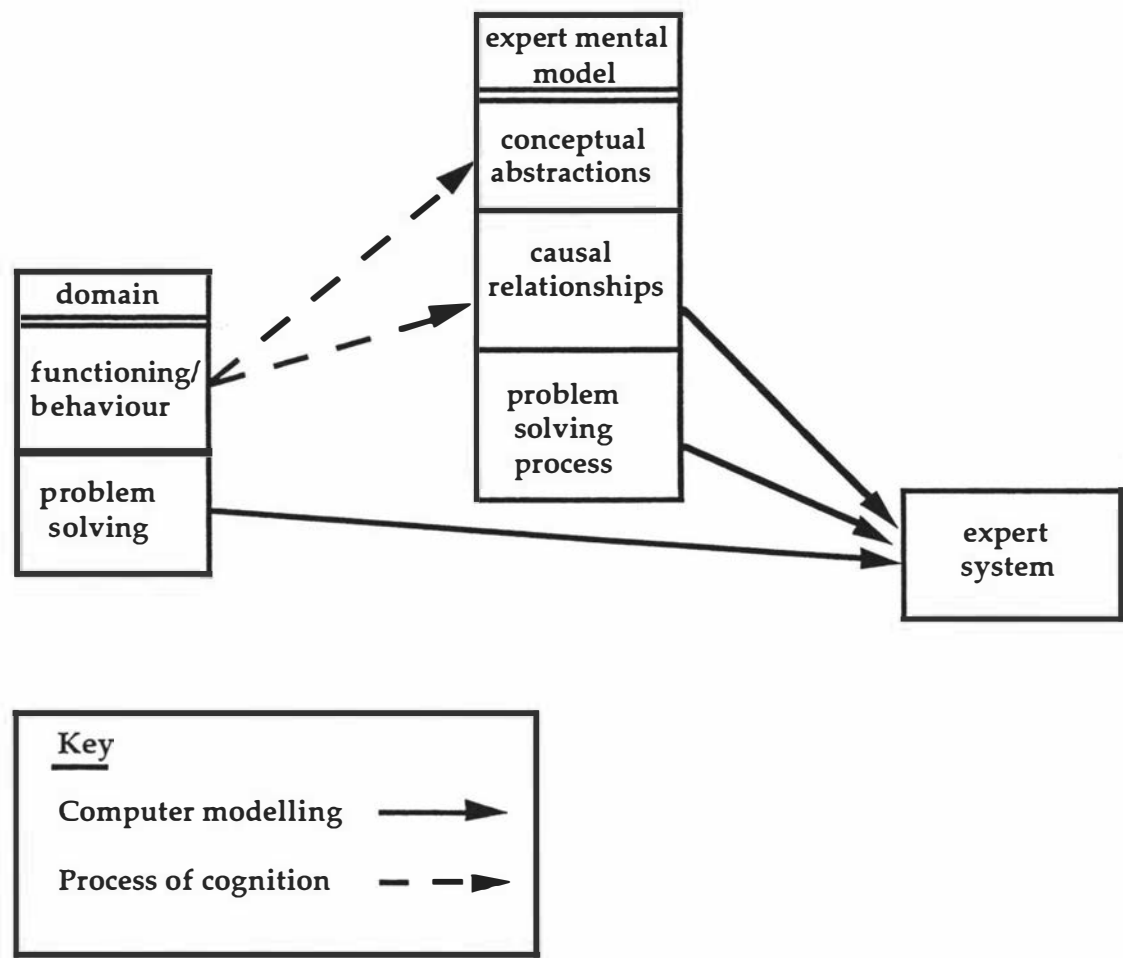


Figure 3.11 Components of idealized expert system

The main reason why many expert systems are poor for teaching purposes, even if they have a high degree of epistemic fidelity, is that the method used to solve the problem takes into account the capabilities of the computer and does not attempt to mimic the behaviour of the human problem solver. The inference engine is often very simple and attempts to cover as many alternatives as possible. If, instead, an attempt is made to simulate the human expert problem solving activity then this moves towards cognitive

fidelity (see Figure 3.12). In an idealized teaching system, cognitive models of conceptual abstractions, causal relationships and the expert's problem solving processes would all be produced, and explicitly represented in the final computer program.

To summarize, then: to better appreciate the different subtleties of black box and glass box systems we need to look at fidelity of the domain representation, of the expertise and of the problem solving process. Also, we may wish the user to acquire a model that is faithful to some conceptual representation of the domain as in STEAMER.

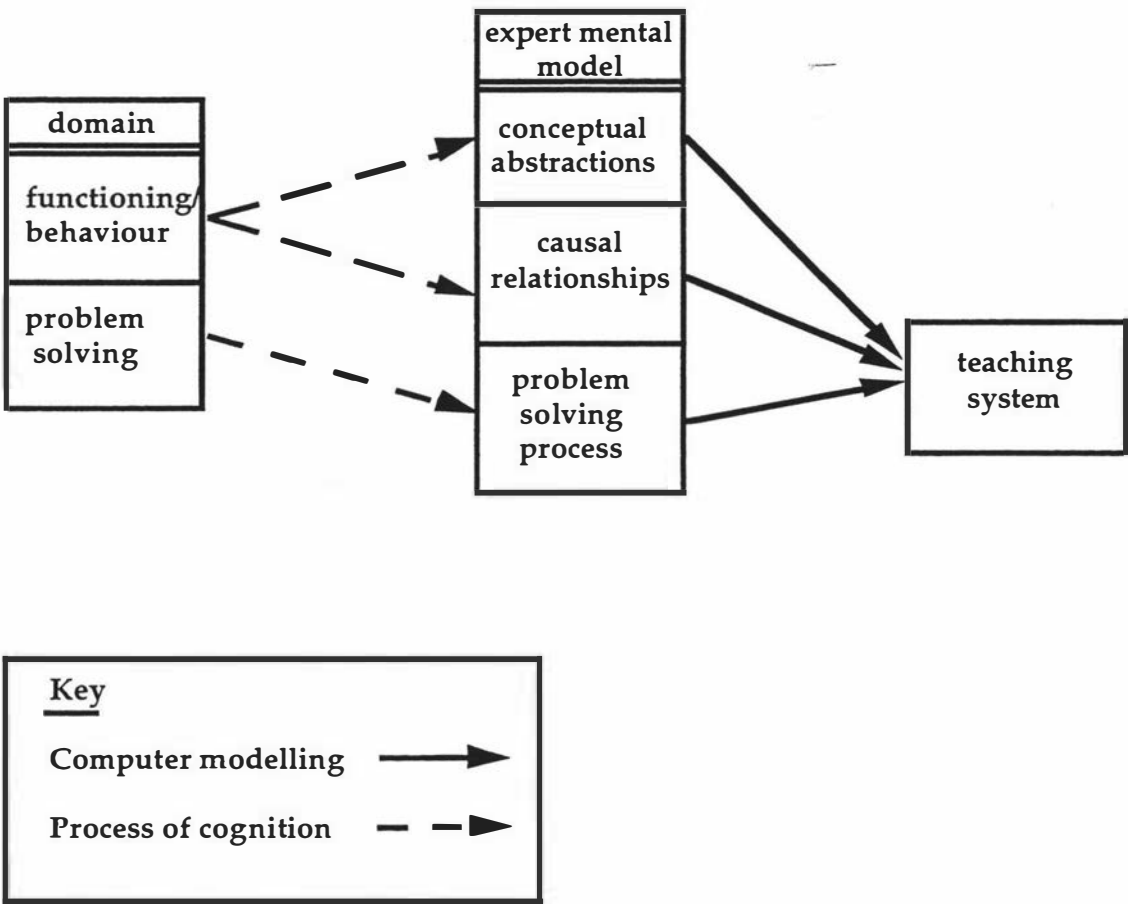


Figure 3.12 Idealized teaching system model

Even the above detail only scratches the surface. We may wish to consider different levels or kinds of expertise, or to include different perspectives on the knowledge. All of these factors add to the complexity of the model. However, for our purposes the scheme described in this section has the detail required for the models used in the subsequent chapters of the thesis.

3.4 Discovery learning

It is clear from Section 3.2 that there are many and diverse forms of knowledge that can be employed in problem solving. It would also seem that 'knowledge communication' is, at the least, a questionable process whereby students can learn. If we wish to get students to understand the kinds of models developed in Section 3.3 then are there better ways of achieving this, using the computer? Discovery learning is an obvious contender. As noted by O'Shea and Self: 'computers enable students to develop, investigate and experiment with their own ideas and theories in a relatively enjoyable and informative fashion' (O'Shea and Self, 1983, p108). If we seek to distil knowledge of a process or situation and then teach it to the user via lists of facts and examples then clearly this can work, but involving the student more directly in the process seems likely to be more effective. It is difficult to verify this hypothesis but statistical tests on student performance (Mark, 1991) seem to suggest that, at least for teaching procedural skills, learning by discovery produces better results than learning by repeated presentation.

Before discussing further the advantages and disadvantages of discovery learning it is instructive to consider how the idea of this form of learning developed and what it entails. The approach seems to have its roots in the socratic method. Although the method, as originally devised, was highly structured and teacher-driven the onus was on the students to derive their own principles and to apply these in novel situations. As Lewis, Bishay and McArthur (1993) observe, the modern version seems to have emerged from the 'Progressive Education Movement' spearheaded by Dewey in the early part of this century. Dewey believed that students should take an active part in their learning, and this idea was developed further by Bruner (1961) in the 1960s. A key conference held at Stanford University in 1966 brought together many of the important researchers in the area. The proceedings were later published as a book (Shulman and Keislar, 1966). Among the presenters was Robert Glaser who suggests that discovery learning involves 'teaching an association, a concept, or rule which involves "discovery" of the association, concept, or rule' (Glaser, 1966, p15). As an unknown delegate at the conference pointedly observes: 'Hence, the major question in the issue of learning by discovery is the extent to which you get better pedagogy by not telling the student what the teacher already knows' (Shulman and Keislar, 1966, p27). At the same conference, Gagne (1966) associates the method closely with problem solving. Clearly, one of the major motivations that the student should have is a curiosity to see how a system behaves in different circumstances and to learn how to deal with problems from observing this behaviour.

Perhaps the best known system that incorporates discovery learning is Logo (Papert, 1980). Workers from BBN and from MIT, most notably Seymour Papert, attempted in the late sixties and early seventies to develop an education-oriented, conversational programming language for children. They produced one based on Lisp, with its recursive capabilities and symbolic approach but much simpler in form. Although it was originally thought of primarily as a first programming language, Papert envisaged Logo as a 'philosophy of education' and as a means of empowering children to learn. Students can learn programming from the turtle language within the system, but, as noted by Ryba and Anderson (1990) and by Chang (1990), it has also been shown to have a positive effect on problem solving skills, reasoning skills, learning of geometry and planning ability.

One of the central tenets of Logo is that students should build their own knowledge by working in an environment that encourages exploration. No guidance is given within Logo itself (although, of course, human tutors could provide help as necessary). The only feedback that students receive is the response the system gives to the actions that they carry out. From these, they are expected to infer whether they are heading in the right direction. Smithtown (Shute and Glaser, 1990; Shute and Glaser, 1991), the LRDC program for teaching students simple economics, has a similar approach. Users have free range to test out various theories in the expectation that they will, thereby, come to some understanding of the domain. As noted by Elsom-Cook (1990), the level at which users operate is self-regulating since they will usually only try out ideas that are at their current level of competence.

In order to get students to examine the methods they have used, and what they have learned, it is common to include some facility for self-appraisal. For example, in a later version of Smithtown (Raghavan *et al.*, 1991), a reflective tutor called DARN is added to get students to analyse their actions at the end of a session, but there is no intervention by the system during the main exploration phase. This attempt to get students to actively reflect on what they have done is in line with constructivist theory of learning, in which the student constructs his/her own knowledge. According to Forman and Pufall (1988) constructivism embodies three properties: epistemic conflict, self-reflection and self-regulation. Initially, there is some discrepancy between what we expect to happen from our knowledge of the world and what actually occurs. We reflect on this, determining how and why our mental model differs from reality. Finally, we change our model to fit in with the facts. As noted by Collins and Brown, the computer provides an ideal environment for supporting reflection since it is easy to keep track of what the student and the computer have done, and these *process trails* can be used as a basis for study. In this way 'the computational medium . . . can provide a

powerful, motivating . . . tool for focusing the students' attention *directly* on their own thought processes' (Collins and Brown, 1988, p2).

It is not universally accepted that a pure exploratory environment is the ideal. Certainly, in computer environments, many systems have been developed that give the student some extra feedback, in the form of guidance or explanation of errors. WUSOR (Goldstein, 1979), for example, uses its genetic graphs to determine what kind of advice might be appropriate to students depending upon their current level of knowledge. In WEST (Burton and Brown, 1982), the system keeps an eye on the choices that users make and gives advice if they seem systematically to be making poor choices or overlooking good ones. Other systems, such as QUEST (White and Frederiksen, 1990) and Sherlock (Lesgold *et al.*, 1992b), supply varying amounts of advice depending upon circumstances. In experiments to test the impact of different levels of feedback, Mark (1991) finds that a scheme that allows the student to freely explore the environment, but which gives feedback on poor choices, seems to be effective for teaching purposes. In an empirical study, Sebrechts and Marsh (1989) also find that users appear to benefit more in an environment where they are set tasks and given limited feedback, rather than where they are just allowed to explore without constraint or guidance.

Elsom-Cook (1990) provides persuasive arguments for including feedback for students. He believes that, ideally, students should learn from observation and inquiry without being told, but, if the student is to absorb new ideas, these have to be presented in some way. He regards exposure to the environment as insufficient by itself for teaching purposes. *Cognitive hooks* (analogies with which the student is already familiar) facilitate the transition towards new concepts but more assistance is sometimes needed. Often the student needs help in drawing the comparison and in other cases there is no obvious analogy to make (for example, in learning Lisp recursion). For this reason, and because the student may flounder without being able to make the required conceptual leaps, a tutor can be advantageous.

However, Elsom-Cook suggests that the tutor should not be a 'coach' as in WEST, for example. He advocates a completely symmetrical organisation where the student and tutor can interact with each other and can observe and manipulate the environment in a similar fashion. The tutor should always be monitoring progress, ready to intervene when requested or when needed, but the intervention should decrease as the student progresses.

To an extent, the appropriate amount of advice may depend upon many factors: the nature of the domain, the nature of the task, the familiarity of the student with either, the difficulty of the task, etc. However, researchers generally agree that extremes are not desirable. A system that gives no guidance may cause the user to flounder making errors and gaining no benefit from the experience. At the other extreme, a rigid system that gives the student no leeway in the exploratory process may be equally ineffective.

In this thesis, a computer simulation is advocated in which the user can explore the environment but in which some guidance can be given to enable new ideas to be presented and to avoid frustration and misconceptions. A computer model is desirable for many reasons including:

The computer operates by advancing through many states. If these states can be mapped onto a model of operation of a procedure or device etc, then a degree of realism can be provided.

The computer model can be stopped and replayed at any stage to allow important points to be understood and consolidated.

The user can immediately see the effect of his/her choices.

There is no danger to the user or the equipment being simulated. The dangers in the real world may inhibit either the tutor or the student.

The operation of the computer model can be monitored and an assessment of the student's performance made.

Another advantage of a computer simulation is that the user can try out possibilities that would be completely infeasible in practice. This may seem pointless but sometimes valuable educational benefits can ensue. An interesting example of demonstrating the impossible is provided by DiBi (Spada *et al.*, 1989), a package for teaching students about elastic collisions between spheres. The spheres are represented pictorially on the screen and the student can observe how collisions occur and the effects. The system has a student module that attempts to assess the user's current state of knowledge and also any misconceptions. If the student clearly has a mistaken idea about the nature of impacts of spheres then this misconception can be incorporated into the model that drives the collision animation. As a consequence, the student can observe the unusual behaviour of the spheres that results from the mistaken assumption. Clearly, this is a direct and effective way of generating the 'epistemic conflict' of constructivist theory. Research along similar lines was carried out by Smith (1986), who developed the Alternate Reality Kit, a program in which the student can

manipulate the laws of physics (such as changing the law of gravity) and see the effects in comparison to their predictions. A more sophisticated version of this system with full graphics and direct manipulation was developed at the Open University in Britain and has been used for teaching physics students (Spensley *et al.*, 1990b).

Besides providing a plausible model and guidance, it seems natural in teaching some skills to provide tasks for the student to carry out. If the student is learning troubleshooting in electronic circuits then getting the student to solve problems in the domain would appear to be advantageous. Glaser and his team (Glaser *et al.*, 1985) advocate this approach and other work has tended to support this view, for example, articles in the set of papers edited by Boud and Feletti (1991).

3.5 Summary

From Clancey's work it is clear that knowledge of many forms can be represented on the computer and that, for teaching purposes, it is important to distinguish each kind. There is still some debate about how learners acquire knowledge, whether it is by knowledge communication or by building up their own knowledge structures but, in either case, modelling is an important factor for representing both student knowledge and the knowledge inside the computer.

A key feature of computer teaching models is their fidelity: how they relate to the material being taught, and the problem solving methods being learned. We have distinguished conceptual, epistemic and cognitive fidelity, each with its own characteristics. One of the main distinctions between the forms of fidelity is the way each different type of knowledge is treated. Cognitive fidelity, with its faithfulness not only to the system being modelled but to the problem solving methods being taught, is the ultimate aim of an idealized teaching system, although this ideal may not be completely realizable with the current state of knowledge about models and teaching.

Within the framework of knowledge and models, a suitable teaching method is needed. The use of discovery learning, particularly in a problem solving environment, is advocated. To avoid floundering by the student, some feedback would appear to be desirable but still allowing users sufficient leeway to discover and rectify their own mistakes.

The concepts examined in this and the previous chapter provide the guidelines for the models developed in the remainder of the thesis. These guidelines are listed overleaf.

Knowledge of many different forms may need to be represented within a teaching package. As far as possible they should be separated out with appropriate linkage.

Two specific forms of knowledge that should be distinguished are knowledge of the domain of interest and methods of problem solving within the domain.

System development should be considered as a modelling process with the aim of simulating as closely as possible the operation of the system and the activities of the problem solver.

We should aim to describe important concepts by high-level primitives. Not only the features of the domain, but also of the problem-solving process within the domain should be modelled in this way.

In order to help the learner understand activities within the domain, a model showing cause-effect relationships is often effective. Where cause-effect is unclear or too complex, then associational links should be provided that fit as seamlessly as possible into the main model.

Discovery learning provides an effective environment for teaching students a wide range of subjects. It is compatible with the theory of knowledge communication since one can envisage the knowledge embodied in the program being communicated to the student. It is also consistent with situated cognition where students build their own knowledge structures by observation and interaction.

Since educationalists have differing views of how much guidance should be given in a discovery learning environment, any framework should allow the developers scope for providing as little or as much guidance as required.

The term *interactive learning environment* has been used in various contexts over the years. White and Frederiksen (1987), for example, use it to classify their system for teaching elementary electric circuit analysis. Here, the term will be used to focus on systems that satisfy the guidelines given above. In the following chapters, the development of interactive learning environments in two different areas will be investigated, and primitives will be proposed for aiding their design. As noted in section 2.5.1, it does not seem feasible within the scope of a single project to consider in detail the whole of an ITS. Here, the emphasis will be on the domain module, arguably the linchpin of the system, although, where appropriate, there will be discussion of other modules.

Chapter 4

Domain and Task Representation for Tutorial Process Models

4.1 Introduction

In this and subsequent chapters, the ideas and approach developed in Chapter 3 and listed in its summary are used to determine a basis for system design. It is clear from all that has been said that there are many different kinds of domains and numerous alternative teaching methods that might be employed. There is not going to be a single panacea that solves all the problems of automating teaching. The best that can be achieved is that for specific kinds of domains, particular teaching methods can be advocated and that, for these combinations, principles of system development can be identified.

The kinds of systems that will be examined in this chapter and the next are ones where the student is learning about procedures and developing problem solving skills. Fairly tightly defined domains and procedures are considered. In Chapters 6 and 7, problem solving in ill-structured domains will be studied. In both cases, it will be assumed that the goal is to produce interactive learning environments where appropriate guidance can be included as desired.

The aim will be to determine common features in the domains, tasks and tutorial guidance that can be used to specify building blocks (high-level primitives) for systems. If these can be identified, then the process of constructing such systems can be simplified, allowing domain and teaching experts to help in their development.

Systems have been built for teaching in a variety of well-defined procedural domains including:

- steam engine control (Hollan *et al.*, 1984);
- control of a pulp and paper mill boiler (Woolf *et al.*, 1987);
- troubleshooting of an electronic test station (Lesgold *et al.*, 1990);
- video cassette recorder operation (Mark, 1991);
- maintenance of a rifle (Miller and Lucado, 1992).

Although a program to model a domain can be designed and coded directly, it is helpful to start with higher level building blocks for describing the system. The techniques employed should facilitate development without unduly constraining the designer. The aim is to provide a tool that can be used to build a glass box simulator. In short, we are trying to achieve Anderson's cognitive fidelity: faithfulness to both the operation of the system and to the problem solving techniques of an expert within the domain.

We wish to allow the user to step through the procedure and so a discrete-event model seems appropriate. For some domains, this may not be feasible. Modelling the operation of an electronic test station, for example, may require a numerical model, although in SHERLOCK II (Lesgold *et al.*, 1990) researchers have gone some way to providing a full qualitative model, at least for sections of the system.

Next, appropriate feedback may be desirable. As noted by McKendree 'no consistent theory of feedback has emerged' (McKendree, 1990, p381). Two features of feedback that she considers important are presenting information about *why* a particular step is incorrect and also representing information at different levels of detail. In one experiment she notes that the subjects 'learned the application of individual operators well enough but failed to acquire a working knowledge of the higher goals which the operator served' (McKendree, 1990, p407).

In this chapter, we consider alternative ways of attempting to achieve a level of cognitive fidelity and of facilitating the provision of appropriate feedback. Various bases for a model are examined. The important concept of domain and task separation is then considered and a convenient way of incorporating this idea into a model is developed.

4.2 Domain description

As noted in Chapter 3, the way that the domain is represented will have important repercussions for the fidelity of the system and the ease with which feedback can be provided. Three high-level methods for representing domains for procedural tasks will be considered: transition nets and two variants of the *Strips* scheme.

4.2.1 Transition nets

Transition nets have often been used in an educational setting. For example, Mark (1991) uses them in her description of video cassette recorder (VCR) operation. VCRs will be considered later but some basic ideas are developed for a simpler domain.

A common requirement is to teach apprentices how to assemble and dismantle equipment. For instance, a mechanic may become familiar with how cars work by examining various subsystems. The example to be considered here is the gearing subassembly of the starter motor used in some models of Bluebird cars and is taken from the Nissan Bluebird Owners Workshop Manual (Churchill, 1990). A photograph of the assembly is shown in Figure 4.1. Three types of component are of interest: the drive gear (being held), the reduction gears (three identical cogs linked to the drive gear) and a sleeve (enclosing the reduction gears). The components are all enclosed within a mounting. A schematic diagram of the system is shown in Figure 4.2.

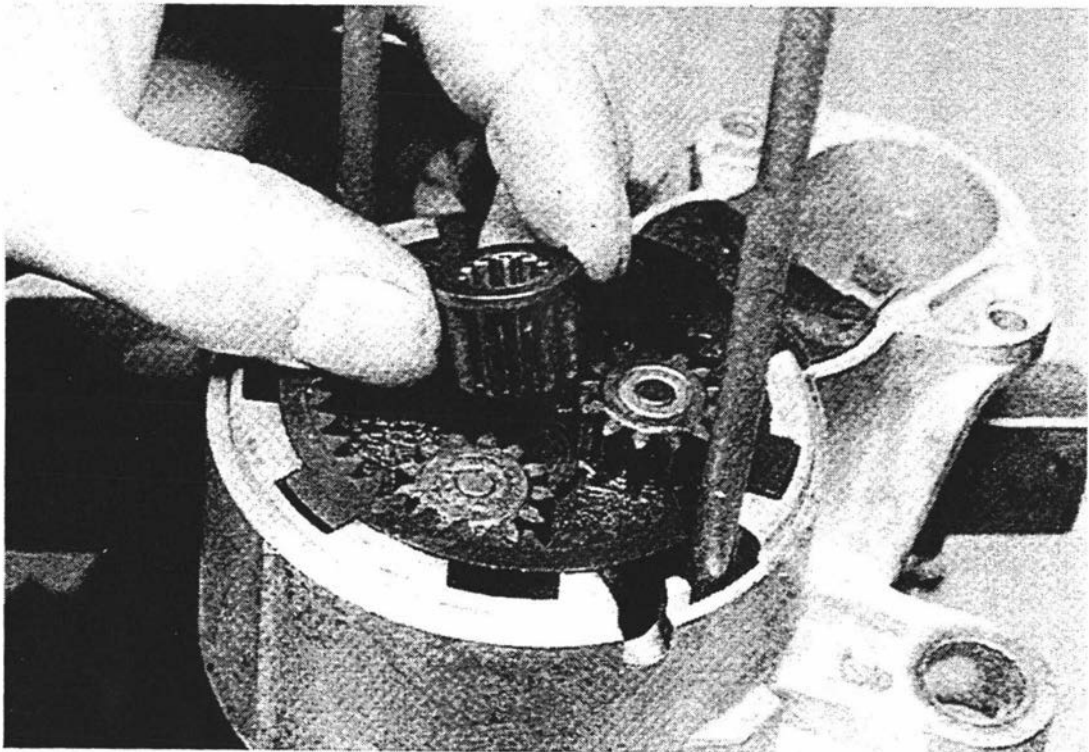


Figure 4.1 Gearing mechanism in Bluebird starter motor (Churchill, 1990, p330)

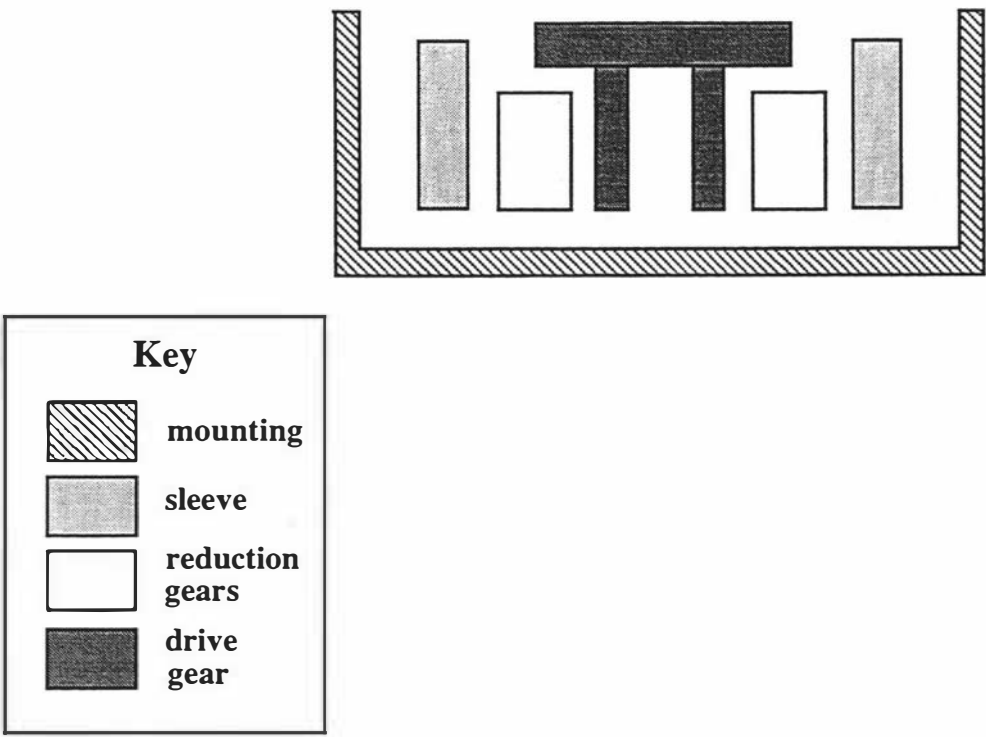


Figure 4.2 Schematic diagram of starter motor gearing

To simplify the problem, we shall assume there is just one reduction gear (or, equivalently, that they are all added or removed as a unit). Given that the mounting is fixed, the actions possible in the domain will be to insert or remove each of the three parts: the sleeve, the reduction gear and the drive gear.

As implied in Figure 4.2, assembly and dismantling can only be achieved in certain ways. While the drive gear and sleeve are in place the reduction gear cannot be removed. It can only be extracted after either the sleeve or the drive gear has been taken out. Similarly, when assembling the unit, the reduction gear cannot be put in place last.

Even for such a simple domain the full transition net representation denoting all possible actions becomes quite complex, containing 786 transitions (Kemp and Smith, 1994b). To reduce the size of the net, two assumptions are made: firstly that the parts are already to hand and secondly that each component cannot be removed once placed in position.

The resulting transition net description of possible actions during assembly is given in Figure 4.3. The letters S, R and D in the nodes indicate whether the sleeve, reduction gear and drive gear are in place, respectively. When the same letters are used on

branches they denote the *actions* of putting the respective parts in position. In the diagram a distinction is made between R_M (putting R in place from the middle - when the drive gear is absent) and R_E (putting R in place from the edge - when the sleeve is absent). Although the effect is the same, it is often useful to distinguish such slightly different actions for pedagogical purposes. Note that the situation S_D , where the reduction gear cannot be put into place, is allowed, although it may not be desirable. This issue will be considered later.

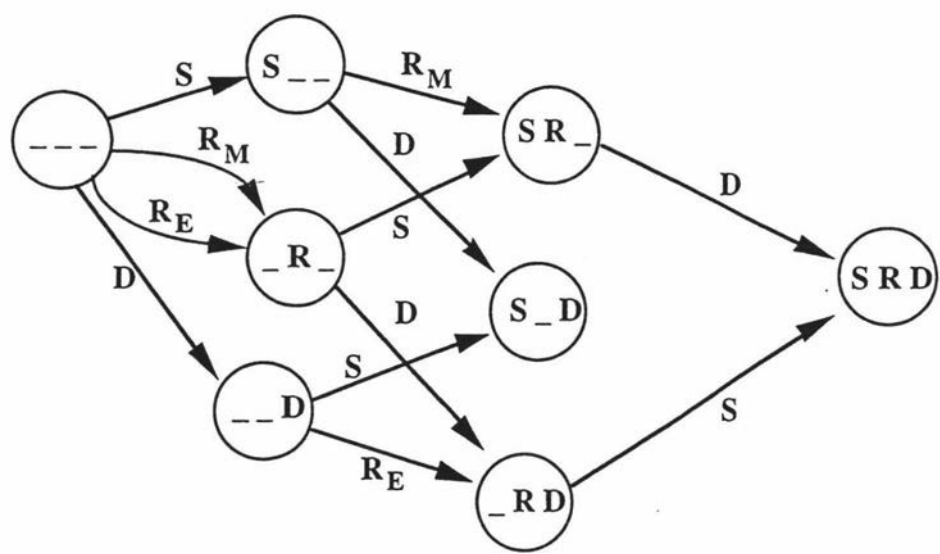


Figure 4.3 Part of gearing assembly transition net

Complexity is not the only problem when using transition nets, opacity is another. There is no indication of the effect of a particular action or why it is feasible. Consider, for example, the operation shown in Figure 4.4. Although a possible transition is given, there is no justification of *why* we are allowed to 'put R in place from the edge', or any direct indication of its effect. The names used for state symbols in this particular example give some clue to what is happening but this is not an essential part of the state transition notation. One would have to annotate the representation to clarify its meaning. Annotations for a few situations are feasible but for the large number of transitions occurring in real-world applications this would soon become laborious and difficult to check. What is needed is an economical representation that more closely reflects the semantics of the operations to be carried out.

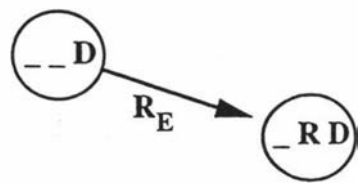


Figure 4.4 Transition in gearing assembly problem

4.2.2 Strips

Another approach to states and their transitions was developed in the archetypal AI planner known as Strips (Fikes and Nilsson, 1971). The basic idea of a Strips-style planner has occasionally been used before in teaching. For example, Peachey and McCalla have assessed its potential in constructing and carrying out tutorial plans (Peachey and McCalla, 1986). Here, its potential for designing procedural tutors is assessed. A fuller description is given in Kemp and Smith (1994a).

The notation to be described is a simplified version of Strips based on the TWEAK planning system (Chapman, 1987). Corresponding to states in the transition net are *situations*. A situation comprises a set of states (conditions) each describing some current aspect of the system. Each state can be denoted by a predicate $f(c_1...c_n)$ where $c_1...c_n$ are constants. Each *event* (or operator) affects the current situation. This effect is denoted by specifying postconditions that can be associated with the event. Each event also has an associated list of preconditions that have to be satisfied before it can take place. An *action table* can be used to summarise the preconditions and effects of each operation. A TWEAK representation for the gearing assembly problem is given in Figure 4.5. As earlier, a distinction is made between inserting the reduction gear from the edge *Eplace* and from the middle *Mplace*.

preconditions	event	postconditions
~available(S)	get(S)	available(S)
~available(R)	get(R)	available(R)
~available(D)	get(D)	available(D)
available(S) ~inplace(S)	place(S)	inplace(S)
available(R) ~inplace(R) ~inplace(S)	Eplace(R)	inplace(R)
available(R) ~inplace(R) ~inplace(D)	Mplace(R)	inplace(R)
available(D) ~inplace(D)	place(D)	inplace(D)

Figure 4.5 Action table for gearing assembly problem

In essence, this formulation depicts transitions between states in the same way that a transition net does. However, there are two reasons for preferring the above model. First, it is more succinct than a transition net representation. In the unconstrained gearing assembly and dismantling problem only 27 table entries are needed as opposed to the 786 states in the net. Also the Strips notation gives more meaningful denotations. We can consider the preconditions as *enabling* the event, and the postconditions to be *caused* by the event.

This representation of the domain and its interactions can then give a complete description of how it will behave in different circumstances. Of significance, from a teaching point of view, is that the user can be given feedback on why actions cannot be performed (preconditions not satisfied) and what impact certain actions have (postconditions). A summary of types of feedback that are facilitated is given in Figure 4.6.

User action	System action	System feedback
ask about current situation	determine conditions satisfied in current situation	list conditions satisfied currently
carry out legal operation	check preconditions satisfied	show result of carrying out action
attempt to carry out non-existent command	verify that command is not in action table	indicate illegal operation
attempt operation currently not possible	verify that one or more precondition not satisfied	explain that action currently not possible
ask why operation currently not possible	determine preconditions currently not satisfied	specify condition to be satisfied before operation can be carried out
ask how operation can be made possible	determine how precondition can be satisfied	suggest action making progress towards carrying out specified operation
ask for all actions currently possible	find all actions with preconditions satisfied	list all currently legal actions
ask how 'impossible' operation can be made possible	verify that preconditions cannot be satisfied	explain that operation cannot be carried out

Figure 4.6 Analysis of possible feedback from Strips system

A further advantage of the Strips approach is that it can easily be used for multiple domains. Problems that cut across domains can have a representation that incorporates the conditions for each one. Figure 4.7 illustrates how we might have several subdomain specifications within a main one. Combining subdomains is relatively simple in Strips since each of the actions is independent of the others. As will be indicated later, subdomain break-up has important ramifications concerning efficiency and feasibility.

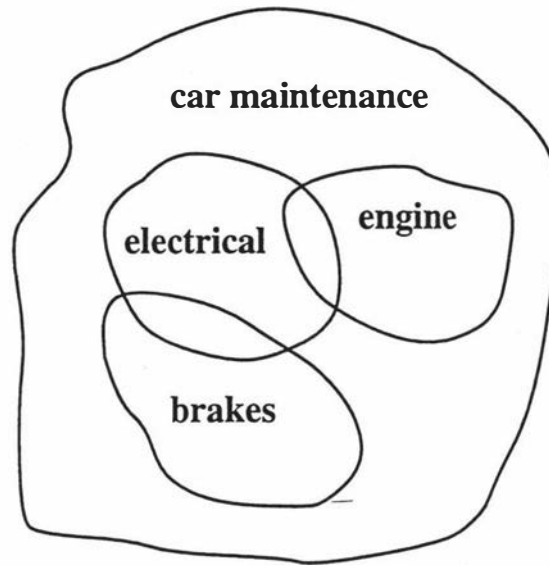


Figure 4.7 Overlapping subdomains

4.2.3 An alternative Strips formulation

Although the TWEAK formulation is adequate for simple problems it becomes unwieldy in more complex domains. Large numbers of preconditions and postconditions required for each operation can make the action table unmanageable. What is needed is some scheme that is as powerful as Strips but without the complex expressions required in Fikes' initial model. Tenenberg (1991) provides a suitable formulation where the preconditions and postconditions are simple atoms as in TWEAK but where additional *static axioms* are included denoting relationships that are assumed to be always satisfied. The use of static axioms adds greatly to the power of the domain description since there are often a large number of conditions that remain the same throughout the execution of a process. Also, it enables the description of what *does* change to be formulated much more succinctly.

Consider an example from the domain of operating a VCR. 'VCR off' can be regarded as an independent variable (what Tenenberg calls an *essential predicate*). Consequently, 'VCR on' is a dependent variable, or *inessential* predicate. The two can be related via the static axiom:

$$\sim \text{VCR off} \rightarrow \text{VCR on}$$

As another example, consider the representation of the 'stop tape' command. In many VCRs, a single button can be used to stop recording, stop playing, stop rewinding or

stop fast forwarding. In a standard Strips description this would involve the use of a disjunction in the precondition for 'stop tape' of:

tape playing \vee tape recording \vee tape rewinding \vee tape fastforwarding

Using Tenenberg's formulation we can define the inessential predicate 'tape in motion' as the sole precondition of 'stop tape' and set up the static axioms:

tape playing	\rightarrow tape in motion
tape recording	\rightarrow tape in motion
tape rewinding	\rightarrow tape in motion
tape fastforwarding	\rightarrow tape in motion

Besides making the action table more succinct it gives a way of including additional information in the model. This information can be exploited for generating feedback: allowing hierarchies of operations to be set up, for example. 'Tape in motion' becomes the more general description of a group of four possible conditions. This could be incorporated into an ISA hierarchy as described by Tenenberg (1991). An alternative approach to providing feedback at different levels has been investigated by Greer and McCalla and their colleagues at the ARIES Laboratory, Saskatchewan. They explore the concept of *granularity* (Brecht *et al.*, 1989; Greer *et al.*, 1989; McCalla and Greer, 1990).

Some of the operations for a VCR are given in Figure 4.8. As with the gearing assembly problem, meaningful responses can be constructed using the table as a basis: checking whether operations can be currently performed, explaining why not, if they cannot, and indicating what steps the user should take to enable them to carry out an action. Using the static axiom table, however, further kinds of information can be included. For example, if the tape is already playing there could be a sequence such as:

Enter command : **eject tape**

This is currently **not** possible

Enter command : **why?**

Because the tape is in motion

Enter command : **give details**

The tape is **playing**

The ability to elaborate on **conditions** is provided by having an implicit hierarchy within the static axiom table.

preconditions	operations	postconditions
tape not in VCR	insert tape in VCR	tape in VCR ~VCR off
tape in VCR tape not in motion tape not at end VCR on	play tape	tape playing
tape in VCR tape not in motion	eject tape	~tape in VCR

Figure 4.8 Three VCR operations

One further important point regarding this scheme should be noted. Care has to be taken during the construction of action tables and static axioms. As Lifschitz (1987) observes, it is easy to modify the basic Strips formulation and unwittingly produce a system that generates incorrect results. To ensure this does not occur, a clear distinction has to be made between dependent and independent variables and the way they are used.

This has two main repercussions for the Tenenberg formulation. Firstly, the values of essential predicates may not be inferred from the static axioms. So, for example, in the VCR problem, 'tape playing' is regarded as an essential predicate. Consequently, its value cannot be deducible from the static axiom table because this would imply that it was dependent on one or more other predicates.

Secondly, inessential predicates cannot occur in postconditions: their values can only be determined by examining the static axioms. For example, if 'tape in motion' was a postcondition of some operation then there would be a possible conflict if its value could also be deduced from the static axiom table.

4.3 Task specification

What has been considered so far is the representation of a domain. We can create a model that shows all possible construction and dismantling operations for a piece of equipment or that has all the appropriate functionality of a device. For example, we can show users what will happen if they flick particular switches or press specific buttons on a VCR. It provides a totally self-exploratory environment, where the only feedback given relates to operations that are possible within the domain. If the user attempts to record a program when there is no tape in then the system may tell them

that this is impossible. However, if they alternately insert and eject a video tape several times then the system will not interfere. The latter are plausible operations but may not be sensible ones. To distinguish what is required from what is plausible, we need to specify tasks within the domain.

4.3.1 Domains and tasks

Many ITSs use ideas from knowledge-based systems (KBSs), both for domain and procedure representation. Central issues in this area have historically included the contrast between deep and surface knowledge (Hart, 1982; Michie, 1982; Chandrasekaran and Mittal, 1983) and the use of frames and production rules (Fikes and Kehler, 1985; Hayes-Roth, 1985). For many, the two are intertwined with deep knowledge being associated with frames, and surface knowledge with production rules. More recently some authors have altered the emphasis and talk instead about domains, tasks and problem solving methods: see for example Woods and Hollnagel (1987) and Steels (1991). Steels, in particular, believes that these factors are more crucial to the effective development of KBSs. This reflects the approach taken in KADS (Breuker *et al.*, 1987) and KADS II (Porter, 1992) where there is a clean separation between domain and task layers. Although he does not suggest that this separation is always possible or desirable, Steels does believe that it should be considered throughout the knowledge acquisition phase. A similar approach can be used in the development of ITSs. Instead of considering the task within the domain directly, it is advantageous in some circumstances to develop a suitable domain representation and then build the task description on top of this.

An example of an integrated approach to domain and task specification is shown in Figure 4.9. Here, Mark (1991) represents, in state transition form, the steps in setting a VCR to record a program. By combining the constraints of the domain and the guidance to be given to the problem solver for a specific task a certain conciseness is achieved. Her work is more concerned with studying the behaviour of the users of systems rather than the development of the systems themselves and so this formulation may be appropriate in such circumstances.

Our contention is that increased insight can be given to users by producing a more elaborate model, separating out the two considerations of domain and task. Suppose that a tutorial program tells a student that a selected action is 'not allowed'. There is no way of telling whether the system means that the operation is literally impossible (such as rewinding a video cassette tape when the VCR is empty) or that the action does not

contribute to the completion of the task in hand. Often such a distinction can be made on a case by case basis but it is such *ad hoc* methods that we are attempting to avoid.

Moreover, system design is facilitated by separating the concerns of modelling the device and the tasks to be carried out when using it. Task specification can be overlaid onto the original model, each task being designed and implemented separately as necessary.

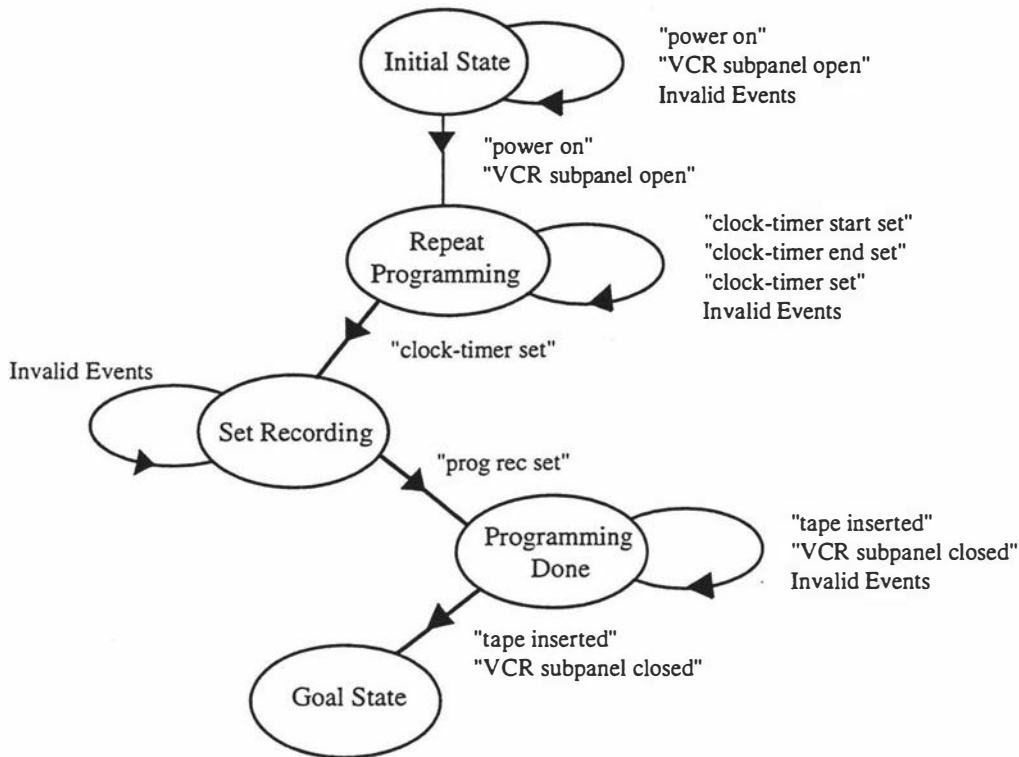


Figure 4.9 Specification for VCR task: based on Mark (1991, p33)

4.3.2 Domain and task separation

One way of separating out domain and task considerations is to set up a transition net representing legal operations within the domain and then overlaying permissible or impermissible operations for each task. For example, given the gear unit introduced in Section 4.2.1 we could overlay the domain description by indicating choices that cannot lead to its correct assembly (see Figure 4.10). The two operations that lead to the state 'S_D' are flagged as impermissible as denoted by the crossed lines on the corresponding transitions.

The problems with using transition nets in domain modelling are still present when they are applied to task representation. For example, the incorrect action can easily be detected but telling the student why it is not allowed will have to be provided as a

separate annotation. It would not be helpful to tell the student that a particular move is not allowed because it 'leads to an impermissible state'.

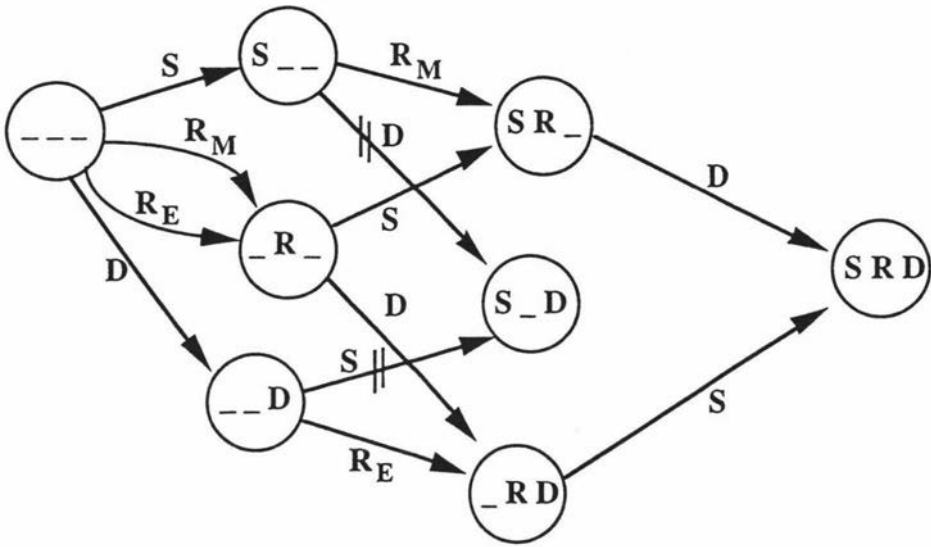


Figure 4.10 Task overlay in the gearing assembly problem

Given that a Strips-style representation is more user-oriented and economical than the transition net approach, the question arises, how can permitted task moves be computed and represented in such a system? Drummond (1989) considers this problem in the planning context but his results are equally relevant for teaching.

His approach is to consider possible sequences of actions that may be carried out and to pinpoint ones that are critical to the completion of a given task. As an intermediate step in this process he gives an algorithm for producing a *projection graph* of possible 'behaviours' of the system (Drummond, 1989, p107). Basically, from a given start situation, all the possible next situations may be produced in turn by checking preconditions of operators. The main complication is checking for causal independence. If two or more operators are 'free from interference' (that is, the postconditions in one do not change the preconditions for the others) then their order of application is irrelevant, and this can be indicated in the corresponding graph. The projection graph for the gearing problem is given in Figure 4.11.

It may be noted that the projection graph is only marginally different from a transition net but the distinction is crucial. In the former, some transitions encompass more than one step. So, for example, to move from ' _ _ _ ' to 'S _ D' two actions have to be taken: placing S in position and placing D in position. These actions are independent, however, and so can be carried out in either order. By grouping together such sets of

independent actions we can distinguish situations where the order of execution may be crucial and those where it is not.

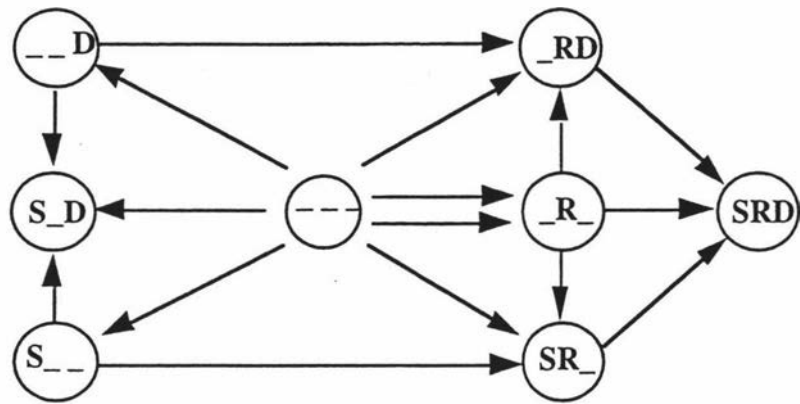


Figure 4.11 Projection graph for gearing problem

Once a projection graph has been produced we can overlay the routes that a user may select to achieve a specific goal. Consider the task of getting all three parts in position and allowing any sequence of actions to be taken that achieves this goal. The possible alternatives for this are highlighted as thick solid lines in Figure 4.12. Given this information, three critical points can be identified from which the user may make an incorrect selection (or sequence of selections): ' _ _ _', 'S _ _' and ' _ _ D'. Drummond (1989, p1 10) gives an algorithm for generating the conditions satisfied at each of these points and the possible actions that may be taken from here. These give 'situated control rules' (SCRs).

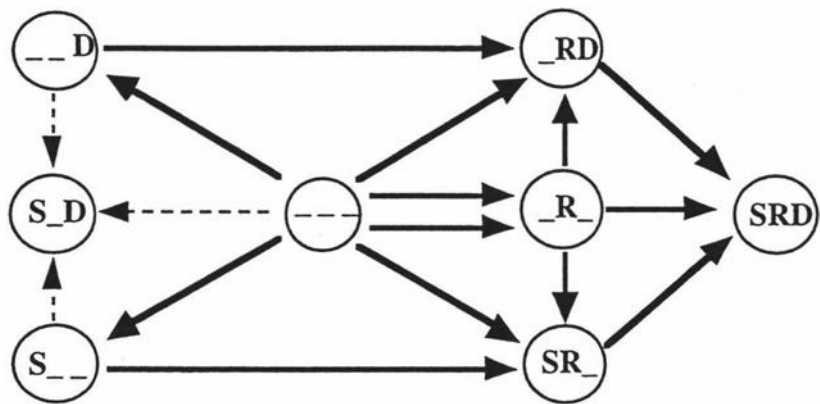


Figure 4.12 Highlighted task routes in gearing problem projection graph

The SCR algorithm is fairly straightforward. For a given situation and set of acceptable paths through a graph, the conditions in this situation that are relevant to the traversal of these paths are generated. This is done by tracing along the paths, noting the preconditions of the operators encountered. The resulting set of conditions form

the antecedent for a rule. The consequent is generated by accumulating all the sets of operators that can be applied from the given situation to get to an adjacent one. For example, in figure 4.12, the acceptable next situations from '___' are 'S __', '__ D', '_ R D', '_ R _' and 'S R _'. The sets of operators leading to these situations form the consequent for the SCR rule from '___'.

The SCRs for the critical points in the gearing problem are shown in Figure 4.13. Column 1 entries denote the current situation, column two shows the list of conditions that *all* have to be satisfied before the rule can be invoked (the rule antecedent), and column three lists *alternative* paths that can be followed if the corresponding SCR conditions are satisfied (rule consequents). The conditions in the first rule correspond to the situation '___' from which point there are several alternative actions. Note that, in addition to the single steps such as placing S in position, there are multiple steps such as inserting D and placing R in position from the edge. These multiple steps denote sets of independent actions that can be taken. The second rule corresponds to the situation 'S __' and, from this point, the only action that ensures the goal is still achievable is to place R in position from the middle. The third rule is similar in form to the second one.

situation	conditions	acceptable moves
---	available(S) available(R) available(D) ~inplace(S) ~inplace(R) ~inplace(D)	place(S) Eplace(R) Mplace(R) place(D) { place(S), Mplace(R) } { place(D), Eplace(R) }
S __	inplace(S) available(R) available(D) ~inplace(R) ~inplace(D)	Mplace(R)
--D	available(S) available(R) inplace(D) ~inplace(S) ~inplace(R)	Eplace(R)

Figure 4.13 Crucial situated control rules for gear assembly task

Consider how these rules may be used in a teaching environment. In a dialogue, the user is at some point: for example, with all three parts available but none in position. Once the user reaches this situation all the conditions in SCR rule 1 are satisfied and so the rule becomes applicable. Various forms of guidance are now feasible for the user. For example, a list of all the actions that will lead to the required goal can be given or we can check whether a choice made by the user can possibly lead towards the goal. Alternatively, the conditions under which a particular operation is possible may be specified.

This representation of guidance rules fits in with the Strips-style domain description and is again economical in storage. An intermediate transition graph-like structure is needed, but only temporarily, to generate the SCRs. Also, various economies of storage can be employed during this generation. Once the SCRs have been produced for a given task then only the original domain table and the task SCRs themselves need to be retained.

Using an analogy with finding one's way through a maze, the SCRs may be viewed as signposts at various points to ensure that bear pits and monsters are avoided. Generally, methods that employ some sequencing control are constrained to pull the user back to the place where they went wrong. They may be told right away, or only after they have wandered around for a while, but they will usually be required to join the path where they left it. In this alternative approach, they may be led back to the optimal path at some other point since local context is used for guidance. This gives extra flexibility and may improve the convenience.

SCRs can be used in more complex domains in a similar fashion. In the next chapter, the production of SCRs for tasks in the VCR domain will be considered.

4.4 Summary

Various possible frameworks for building procedural tutorial systems have been examined. The goal has been to determine a scheme that is powerful, consistent, flexible, easy to apply and could be used as a basis for providing helpful feedback to students.

Transition nets have the expressibility required but at the expense of clarity and economy. They do not lend themselves to clear expression of cause-effect steps, do not have any natural hierarchical structure and can become quite cumbersome to handle.

TWEAK is a consistent version of the Strips model and is able to represent cause-effect relationships quite succinctly. It is limited in expressibility, however, and can become quite cumbersome for complex domains.

Tenenberg's formulation of Strips is both economical and powerful. It has been carefully formulated to be consistent, and has the bonus of static axiom tables for representing relationships that do not change. These tables can be used to include hierarchical information.

Although domain constraints can be readily described using Tenenberg's method, there is no obvious way of including task information. A method for providing this has been described. It involves adapting a method of planner guidance devised by Drummond. His situated control rule formulation can be used for denoting task information, overlaying it on the domain description.

What has been proposed in this chapter is a scheme for building teaching systems that employs techniques adapted from AI planning. The cause-effect scheme and hierarchical features allow appropriate feedback to be provided as necessary. Distinguishing domain and task issues permits these aspects to be considered separately. The situated control rule formulation facilitates the provision of appropriate task information. Lastly, partitioning into sub-domains is simple and not only allows further enhancements to the feedback but permits economies of storage and significant improvements in speed to be made (Smith and Kemp, 1995).

Although the modified strips representation is convenient for modelling certain kinds of domains and tasks it has its limitations. It is based on a state-space approach and so in domains where this is inappropriate (for instance, mathematical modelling of computer circuits) it would be difficult to use in isolation. Another potential problem is the size of the projection graph for large domains. This latter problem is one that is still being addressed (Smith, 1994f).

Chapter 5

Interactive Learning Environments for Procedural Tutors

5.1 Introduction

What has been described so far has been a succinct knowledge representation method for procedural tutors that facilitates the provision of feedback. In order to provide a system built along these lines, various other problems have to be solved. Addressing these problems provides the focus for this chapter. In Section 5.2, models of procedural simulators are presented, including results from programs. Task specifications are added in Section 5.3. In Section 5.4, a way in which a graphics interface could be employed to aid the building of a procedural task teaching system is considered. Although this project does not address the issue of what kind of feedback is most effective, some of the different forms that guidance might take are considered in Section 5.5. Finally, in Section 5.6, a scheme for system development is proposed.

5.2 Implementing domain models

5.2.1 The gear assembly domain

Implementing a simulator for the gear assembly domain, described in Section 4.2, can be achieved quite simply. An interpreter for TWEAK-style specifications is needed, with a feature that includes feedback for different situations that occur such as those listed in Figure 4.6. Any given action table can then be fed in as data with specific initial conditions. Such a program has been written in Prolog (Smith, 1994b). The action table shown in Figure 4.5 was used as data, and part of the output from a test session (slightly modified and including diagrams for clarity) is presented in Figure 5.1.

Note that computer prompts and responses are shown in plain font whereas user input is in bold. The numbers on the left are for reference and are not part of the dialogue.




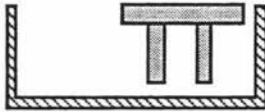

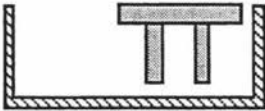

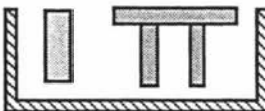
		
1	Enter command : insert drive gear	
		
2	Enter command : insert grommet This is not a legal operation	
3	Enter command : insert sleeve This is currently not possible	
4	Enter command : why Because sleeve is not available	
5	Enter command : how get sleeve	
6	Enter command : get sleeve	
		
7	Enter command : help currently available actions are:	insert reduction gear from the edge insert sleeve
8	Enter command : insert sleeve	
		
9	Enter command : insert reduction gear from the edge This is currently not possible	
10	Enter command : why Because sleeve is in place	
11	Enter command : how This cannot be achieved	

Figure 5.1 Part of gear assembly tutorial session

Figure 5.2 gives a detailed summary of the different kinds of user action including erroneous ones, the cause of each error and the feedback given to the student. Most of these are easy to provide by examining the entries in Figure 4.5. Note that the user gets stuck at command 9 in the dialogue because there is no way that the reduction gear can be inserted once the sleeve and drive gear are in place. This is not a flaw in the program. The program describes the domain constraints and possibilities, not any particular task within the domain.

Command Number	User action	Source of problem	System response
1	carry out legal operation		show result of carrying out action
2	attempt to carry out non-existent command	operation is not in command list	indicate illegal operation
3	attempt operation currently not possible	precondition(s) not satisfied	indicate operation currently not possible
4	ask why operation currently not possible	precondition(s) not satisfied	give precondition not currently satisfied
5	ask how operation can be made possible	precondition(s) not satisfied	give operation making progress to precondition satisfaction
7	ask for all actions possible		list all actions with preconditions satisfied
11	ask how operation can be made possible	operation cannot be carried out from current position	explain that operation cannot be carried out

Figure 5.2 Analysis of gear assembly dialogue

5.2.2 The VCR domain

The gear assembly problem has been used to show how the Strips approach can be adapted to provide a simple tutor. It is not a domain, however, in which any useful teaching could take place. The question arises, can such methods be applied in practical domains? Mark (1991) tests out people's ability to learn tasks by simulating the operation of a VCR. She demonstrates that people generally have difficulty in carrying out any but the simplest tasks for this device. Her statistical analysis supports a commonly held view that VCRs are difficult to master.

Since many people are familiar with the VCR, if not its detailed operation, it seems an appropriate domain to test out models of representation. An initial attempt was

therefore made to simulate VCR operation using the TWEAK scheme. Unfortunately, the resultant system was unwieldy and opaque.

As noted in the previous chapter, by employing Tenenberg's approach of including static axioms, a much more succinct domain description can be achieved. This method was tried out in the VCR domain. The specific devices simulated were the Panasonic NV-F70 HQ video cassette recorder and Philips 20GR1250 television. Most, but not all, of their relevant features have been included. The major omission is the timing facilities component. This consists of a more-or-less independent set of operations, and so, as previously noted, could be easily added to the current model.

The actions covered are:

insert tape	press eject
press VCR button	press play
press stop	press pause
press ff	press rewind
press record	select VCR channel N (N=1..99)
press tv power button	select tv channel N (N=0..39)

The essential predicates and their parameters are summarised in Figure 5.3.

predicate	parameter	parameter values
VCR	S	S=on, off
tape	S	S= playing, fast forwarding, rewinding, review playback, cue playback, recording, playing double speed, stationary, out of VCR, paused play, paused record
VCR tuned	S	S=channel N (N=1..99)
tv tuned	S	S=channel N (N=0..39)
tv	S	S=on, off
tape tab	S	S=present, absent

Figure 5.3 Essential predicates and parameters for VCR simulation

In addition, it is convenient to include two inessential predicates: 'tape in VCR' and 'tape activated'. The static axioms are shown below:

tv(on)	→ ~tv(off)	
tv(off)	→ ~tv(on)	
VCR(on)	→ ~VCR(off)	
VCR(off)	→ ~VCR(on)	
tape (S)	→ tape activated	(S≠stationary, S≠out of VCR)
tape (S)	→ ~tape (T)	(S≠T)
tape tab(present)	→ ~tape tab(absent)	
tape tab(absent)	→ ~tape tab(present)	
~tape(out of VCR)	→ tape in VCR	

The action table for the complete simulation is shown in Figure 5.4. Note that for channel selection, a variable N denoting the channel number is used. This produces what Tenenberg (1991, p229) calls an *operator schema*. Such schemata have to be grounded by instantiating variables before they can be applied.

A program has been written (Smith, 1994c) that can interpret any Tenenberg domain specification and simulation activities within the domain, giving the kind of feedback described in Figure 4.6. Output from a program for VCR simulation is given in Figure 5.5.

5.3 Implementing a task model

As has been emphasized already, there are good pedagogical reasons for separating out the domain and task considerations. Using the model described in Chapter 4 this separation can be readily accomplished. For a given domain, a projection graph can be produced and acceptable sequences of actions for completing a given task can be overlaid on the graph.

A program has been written (Smith, 1994d) to generate the projection graph from a Tenenberg domain specification. Part of the output from applying this program to the VCR domain has been re-constructed as Figure 5.6. As noted previously, the graph is not dissimilar to a transition diagram, except that any operations where the ordering does not matter (such as 'press eject' and 'press VCR button') are grouped together. This may facilitate feedback, since an interpreter could allow either operation to be carried out first.

preconditions	operation	postconditions
tape tab (present) tape (out of VCR)	insert tape	VCR (on) tape (stationary)
tape tab (absent) tape (out of VCR)	insert tape	VCR (on) tape (playing)
tape in VCR ~tape (recording)	press eject	tape (out of VCR)
VCR (off)	press VCR button	VCR (on)
VCR (on) tape in VCR	press VCR button	VCR (off) tape (stationary)
VCR (on) tape (out of VCR)	press VCR button	VCR (off)
VCR (on) ~tape (paused record) ~tape (playing) ~tape (paused play) tape in VCR ~tape (recording)	press play	tape (playing)
tape (playing)	press play	tape (playing double speed)
tape activated	press stop	tape (stationary)
tape (playing)	press pause	tape (paused play)
tape (paused play)	press pause	tape (playing)
tape (recording)	press pause	tape (paused record)
tape (paused record)	press pause	tape (recording)
tape in VCR VCR (on) ~tape (playing) ~tape (paused play)	press ff	tape (fast forwarding)
tape (playing)	press ff	tape (cue playback)
tape (playing)	press rewind	tape (review playback)
tape in VCR VCR (on) ~tape (playing) ~tape (paused play)	press rewind	tape (rewinding)
VCR (on) tape (stationary) tape tab (present)	press record	tape (recording)
VCR (on) tape(stationary) tape tab(absent)	press record	tape(out of VCR)
VCR (on)	select VCR channel N	VCR tuned (channel N)
tv (off)	press tv power button	tv (on)
tv (on)	press tv power button	tv (off) tv tuned (channel 1)
tv (on)	select tv channel N	tv tuned (channel N)

Figure 5.4 Action table for VCR and television operation

Given a projection graph, tasks can be specified by highlighting acceptable paths from a given state to a given goal. Of course, the choices that are considered acceptable will depend upon the pedagogic approach judged appropriate by the teacher. This issue is considered further in Section 5.5. For the purposes of illustration it is assumed the teacher has decided that the student can choose any action that makes progress towards achieving the goal without performing redundant operations. The resultant goal graph of acceptable paths for the task of playing a video tape is given in Figure 5.7. Note that the command 'select video channel' denotes the operation of selecting the designated tv channel for viewing the video tape.

A teacher might be even more restrictive and insist that certain actions are carried out in a specific order, in which case, the goal graph would be smaller. Conversely, if the need for efficiency was not paramount, redundant operations such as 'press VCR button' could be allowed (not strictly necessary since inserting the tape automatically turns the VCR on). Obviously, the graph becomes larger and more complex as more allowable task sequences are included.

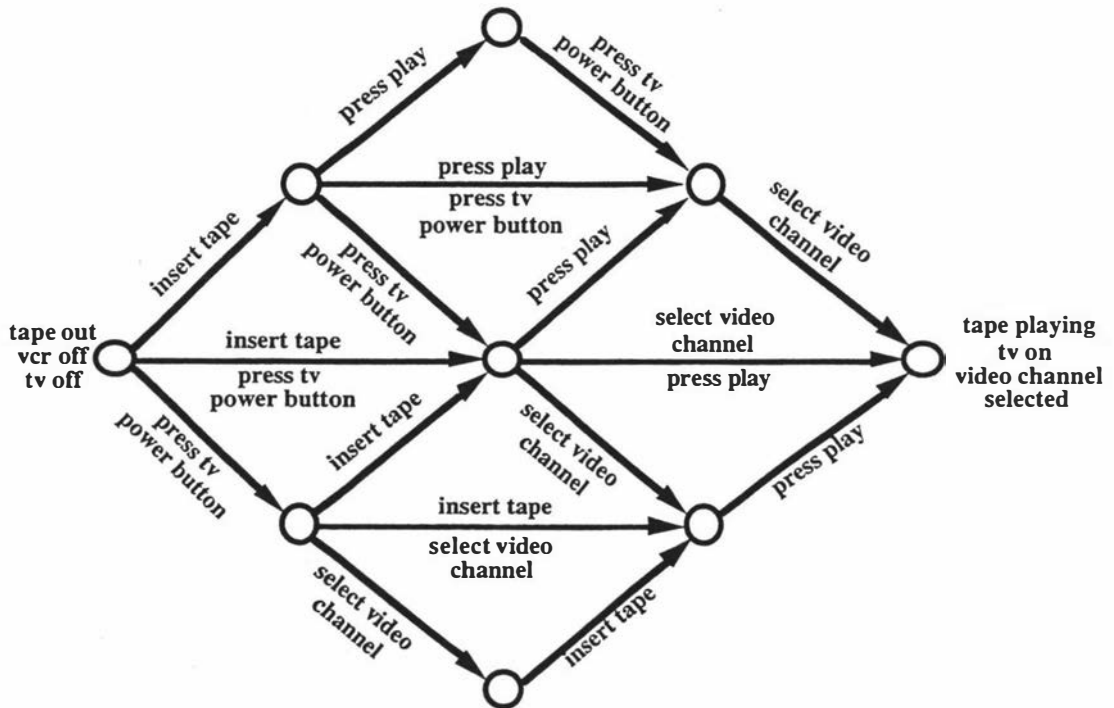


Figure 5.7 Permissible operation sequences for playing a video tape

The next stage is to produce the situated control rules from such a task specification. As noted in the previous chapter, situated control rules indicate for each of the crucial choice points, what the acceptable actions or sets of actions are. Once these rules have

been produced, the projection graph and goal graph are no longer needed. An interpreter can follow through the action table and use the SCRs to ensure that the student does not take an unacceptable path. Shamus Smith (1994a) has written a program to produce the situated control rules from any given task specification. Some of the SCRs for the video tape playing task are given in Figure 5.8.

SCR No.	conditions	acceptable moves
1	tape out tv off	insert tape { insert tape, press tv power button } press tv power button
2	tape in VCR VCR on tv off	press play { press play, press tv power button } press tv power button
3	VCR on tape in VCR	press play

Figure 5.8 Situated control rules for playing a video tape

A program to teach a user how to carry out given tasks in a particular domain has been written (Smith, 1994e). Output from this program is shown in Figure 5.9. As in the earlier example, some of the computer responses relate to domain constraints (answers to commands 1-4, for example). Other responses are task oriented and can be checked via the situated control rule table. These include the responses to commands 5, 9, 10 and 11.

Appendix A contains an action table description of assembling and dismantling a piece of equipment. Like the assembly described in Section 4.2.1 it is a gear mechanism, but much more intricate. Consequently, to make the model manageable, static axioms have to be included. It is interesting to note that, even for complex domains, task specification has not been a problem. The number of user actions to be considered has been quite manageable. This is in line with the experience of Glaser *et al.* (1985) who used cognitive task analysis to investigate the troubleshooting tasks carried out by F-100 jet engine mechanics and F-15 avionics technicians. They found that the number of plausible choices during the execution of complex tasks, once impossible ones were eliminated, was surprisingly small. Further work has been carried out to investigate and improve the efficiency of the representation (Smith and Kemp, 1995).

5.4 Graphical representation for a procedural task tutor

Although action tables provide a useful representation for procedural domains and tasks they are less than ideal for development purposes. It would be useful to have some notation that was more transparent and could be readily used by non-computer specialists.

It may appear that a pictorial scheme for representing algorithms such as structure diagrams (Doran and Tate, 1972a; Doran and Tate, 1972b) or flowcharts might provide such a basis. These notations are too constricting, however, since, like most other methods of algorithm description, they are deterministic. For many of the tasks considered here the ordering of steps is not fixed, and there may even be several different valid ways of carrying them out. A more appropriate scheme is needed.

5.4.1 Procedural nets

One way of denoting alternative possible orderings of activities within a task is to adapt *procedural nets* (Sacerdoti, 1975) originally developed as an aid to solving AI planning problems. How the notation can be employed is best described by an example. Suppose someone is to be guided through the process of taking a photograph. At the top level, two initial activities may be envisaged: 'load the camera' and 'set up tripod'. It will be assumed that the order in which these tasks are carried out is unimportant. This may be conveyed in a procedural net as shown in Figure 5.10. The arrows denote the direction of time, and the co-terminal branches indicate that the ordering between the subtasks shown within the boxes is not important.

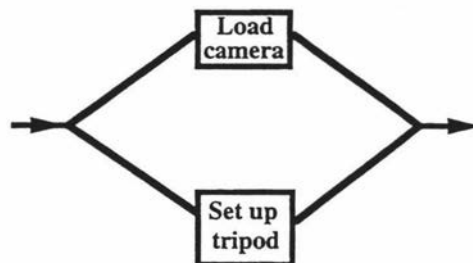


Figure 5.10 Procedural net for setting up camera

After these initial tasks have both been completed, the photographer can then attach the camera to the tripod. Next, the image is framed, camera adjusted, and lastly the shutter is clicked to take the photograph. These four activities have to be carried out sequentially and must follow the initial set up and so we can add them to the diagram as shown in Figure 5.11.

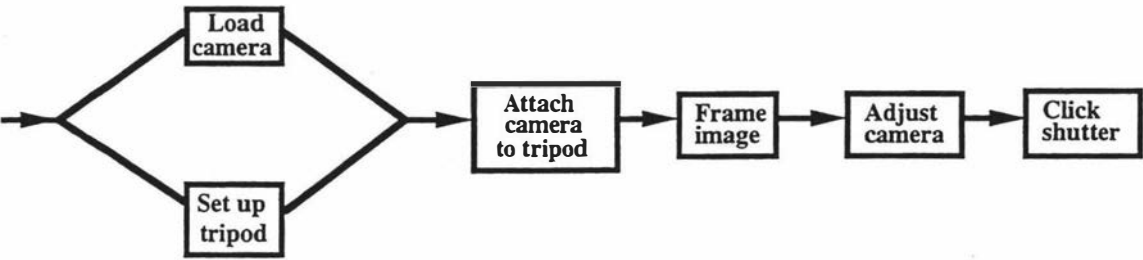


Figure 5.11 Procedural net for taking photograph

An action table corresponding to this specification can easily be constructed by tracing through the diagram and determining appropriate pre and postconditions for each node. Figure 5.12 shows the table for this high level description of the photography task.

preconditions	operation	postconditions
	load camera	camera loaded
	set up tripod	tripod set up
camera loaded tripod set up	attach camera to tripod	camera loaded tripod set up camera attached to tripod
camera loaded tripod set up camera attached to tripod	frame image	camera loaded tripod set up camera attached to tripod image framed
camera loaded tripod set up camera attached to tripod image framed	adjust camera	camera loaded tripod set up camera attached to tripod image framed camera adjusted
camera loaded tripod set up camera attached to tripod image framed camera adjusted	click shutter	camera loaded tripod set up camera attached to tripod image framed camera adjusted shutter clicked

Figure 5.12 Action table for high level description of photography task

Any of the nodes in the net could be made expandable to allow more detail to be given about the actions involved. Other operations could be designated as primitive,

permitting no further elaboration. So, for example, it may be determined that 'Set up tripod', 'Attach camera to tripod' and 'Click shutter' are in the simplest form that these operations are to take. The others could be represented in more detailed form in other procedural nets as shown in Figure 5.13. Again, action table entries could be included for each of these activities.

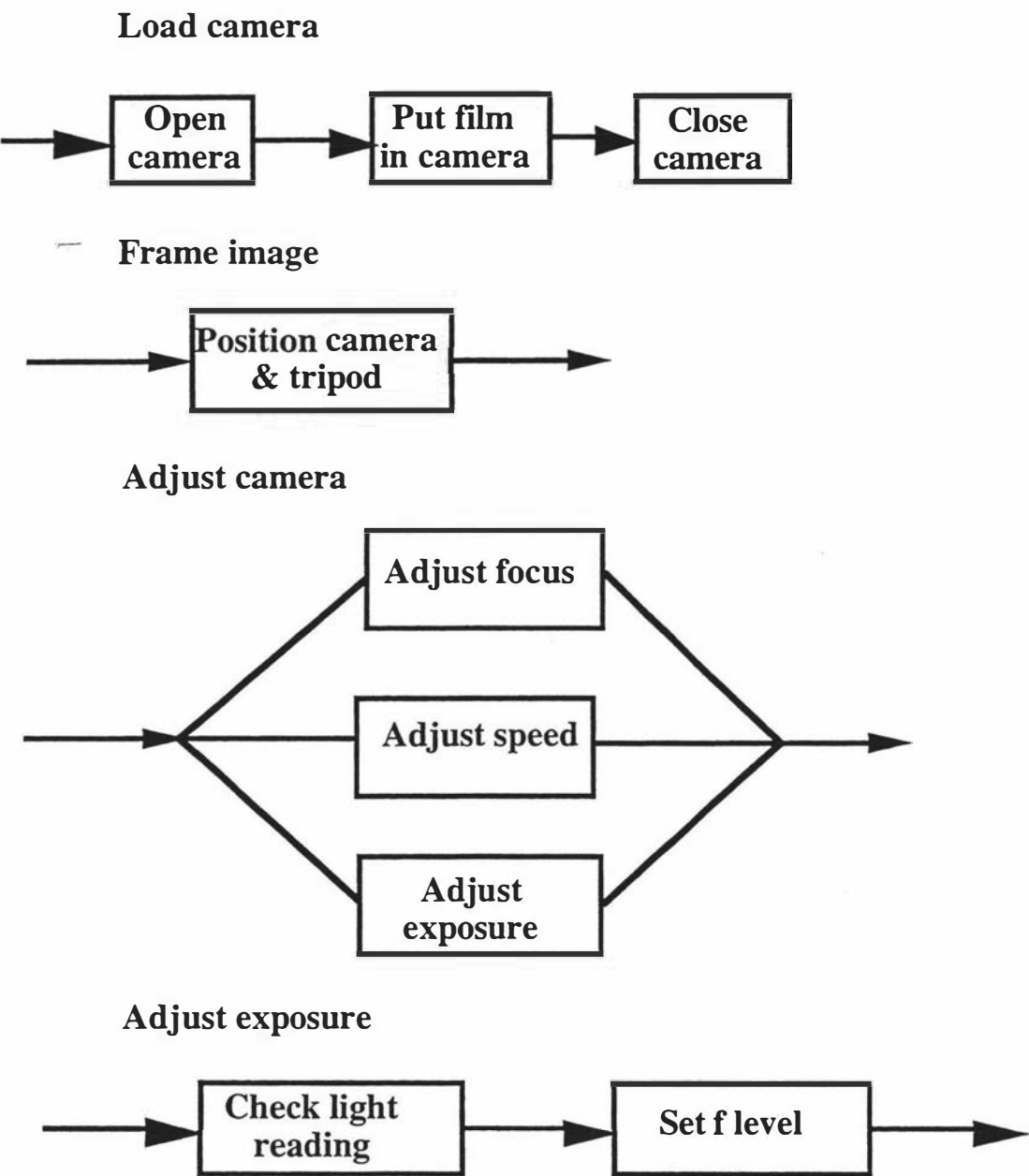


Figure 5.13 Procedural nets for parts of the photography task

Given this hierarchical representation of the operations involved in the taking of a photograph, it becomes possible to envisage a program that could use the embedded information in a tutorial for helping a student who is seeking guidance on how to take a photograph. An extract from a possible dialogue is shown in Figure 5.14.

Enter command :	load camera
	how?
Enter command :	open camera
	OK
Enter command :	put film in camera
	OK
Enter command :	help
	you have not completed loading the camera
Enter command :	help
	you should close camera
Enter command :	close camera
	OK. The camera is now loaded
Enter command :	frame image
	this cannot be done at this stage
Enter command :	help
	attach camera to tripod

Figure 5.14 Extract of dialogue for program derived from procedural nets

For simple task specification, procedural nets provide an effective means of denoting activities. Their pictorial basis means that a suitable graphics interface could be provided for task description. A program could then be easily implemented from this, either manually or even, perhaps, automatically. As can be seen from Figure 5.14, both hierarchical and sequential information can be used to provide effective feedback to the student. For example, the system can indicate what to do next, show that a subtask is incomplete, or indicate the completion of a higher level task.

The hierarchical framework can also be used to streamline the teaching process. If the student already knows how to carry out some high level task then it would be tedious and pointless to require him/her to fill in details. For instance, the student may be allowed to enter the high level command 'load camera' if it has been determined that s/he already knows the detailed operations involved.

Although procedural nets are easy to understand and could provide the basis for a user-friendly interface, they are limited in various ways. Firstly, if the hierarchical form is adopted then inconsistencies can arise. The obvious way of converting such structured nets to an action table is to use multi-level preconditions, postconditions and operators.

A similar scheme was used by Sacerdoti (1974) in ABSTRIPS but, as shown by Tenenberg (1991, p250), inconsistencies can result from having a mixture of conditions and actions at different levels where some of the high level ones may subsume others at a lower level.

Secondly, even if care is taken to check for and avoid inconsistencies, the procedural net is limited in its expressive power. For example, alternative ways of reaching the same state cannot be specified. This causes no difficulties for activities where there is only one set of actions that can be carried out to achieve a goal, but even for relatively simple problems, such as that of gear assembly introduced in Section 4.2, there may be alternative ways of performing a task. If there were no constraints on the order in which the gearing components were assembled then the procedural net in Figure 5.15 would suffice.

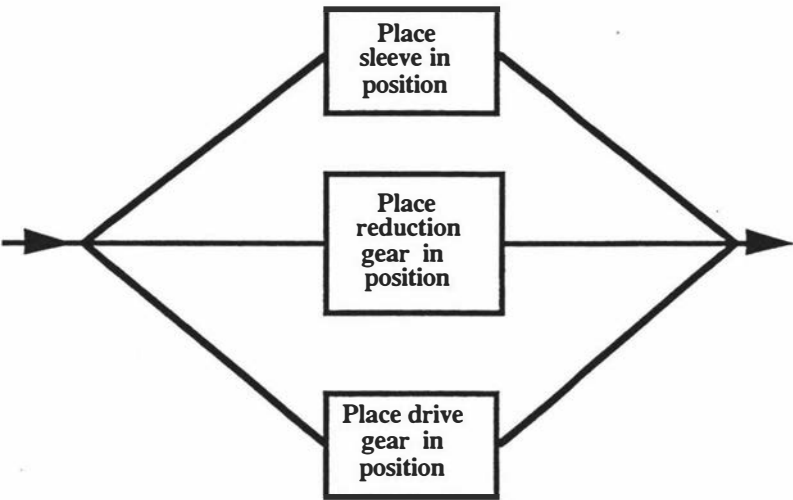


Figure 5.15 Procedural net for gear assembly problem

However, a further stipulation is that the reduction gear cannot be slotted in between the sleeve and drive gear, but has to be inserted from the edge or middle. Consequently, there are four alternative orders in which the task can be completed: (R, S, D), (R, D, S), (D, R, S) and (S, R, D). Because the operations are inter-dependent, there is no way the preconditions of each of the subactions 'place R', 'place S', 'place D' can be specified to ensure the task is completed correctly.

The procedural nets representation could be augmented to deal with this kind of situation by allowing sets of alternative expansions of any action node. For example, the node 'assemble unit' would expand to three alternative sub-nets as shown in Figure 5.16.

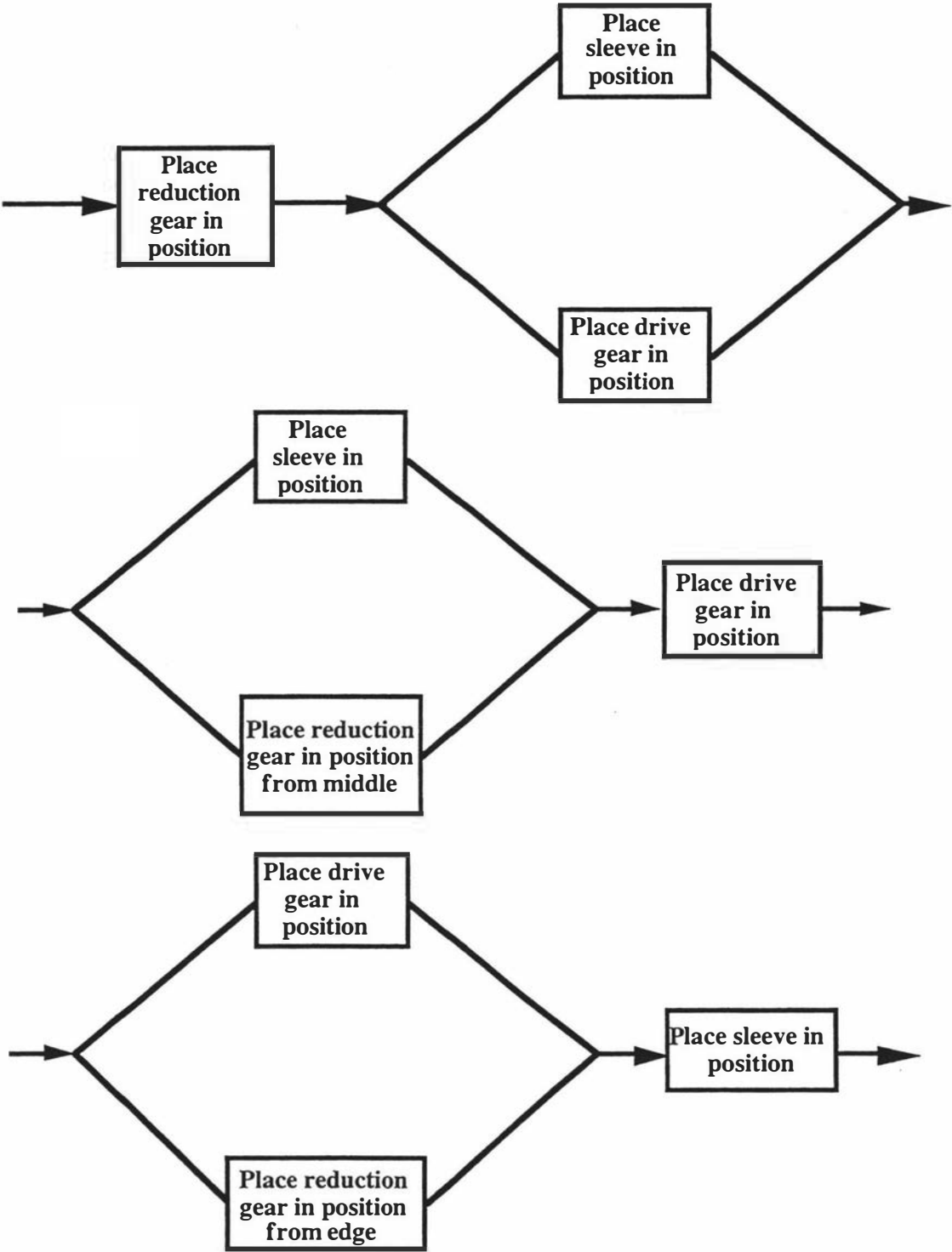


Figure 5.16 Set of procedural nets for representing gear assembly problem

This notation would make task representations difficult to construct, to follow and to interpret by computer. Even if it could be employed in this form it still lacks power. As demonstrated by Drummond (1985), because of their nature, procedural nets cannot be used for tasks involving repetition.

5.4.2 Plan nets

An alternative diagrammatic representation proposed by Drummond (1985) for planning is more suitable for our purposes. His *plan net* represents conditionals and consequent actions explicitly. It is not traversed in the same sense that a procedural net is, but the whole configuration is more like a petri-net (Reisig, 1985) in that an action node can 'fire' when all of its inputs have been activated. We can distinguish operations that have occurred by what has fired. A condition becomes satisfied if any of its predecessor actions has fired.

For example, in Figure 5.17 (a) we have an *enablement* configuration. Once both conditions C1 and C2 have been satisfied, the action A is enabled and so the node can fire. An example of a *causal* configuration where actions feed into conditions is shown in Figure 5.17 (b). Here, the actions A1 and A2 both feed in to condition C, but only one of the actions needs to fire to cause condition C to be satisfied.

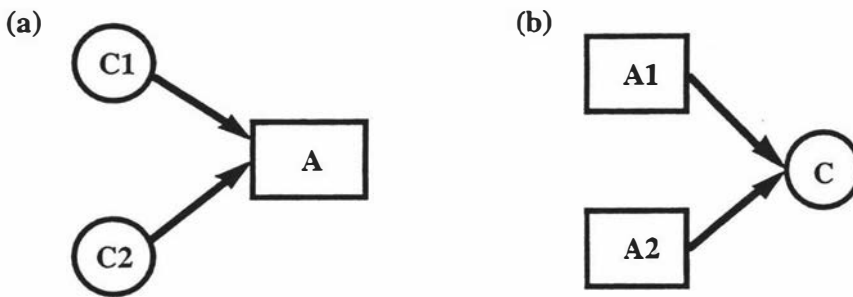


Figure 5.17 Plan net enablement and causal configurations

In the procedural net there is some sense of flow through the operations, each being carried out in turn until the target end node is reached. Advancing time is explicitly represented via the traversal of the arcs. This means that no repetition of operations can be directly represented. In plan nets, arcs do not specifically represent advancing time. A node can be re-activated if conditions are satisfied. For example, for hammering we might have the net shown in Figure 5.18.

A net depicting the task of cleaning the plugs in a car is illustrated in Figure 5.19, and a corresponding action table is shown as Figure 5.20. Note that not only does this notation clearly show what has to be done at each stage and the general flow of activities but that there is a direct correspondence between the diagrammatic representation and the action table that enables a conversion between the two to be considered.

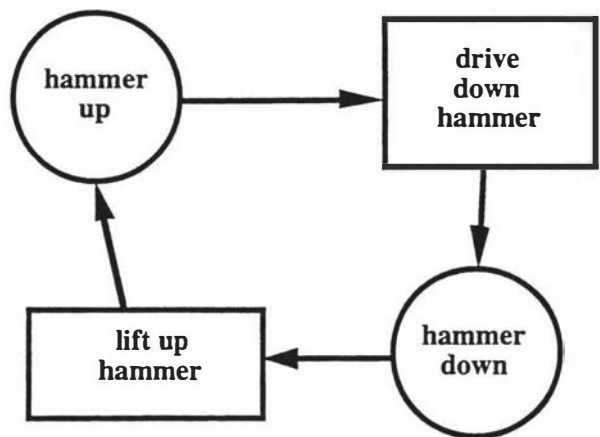


Figure 5.18 Plan net for hammering

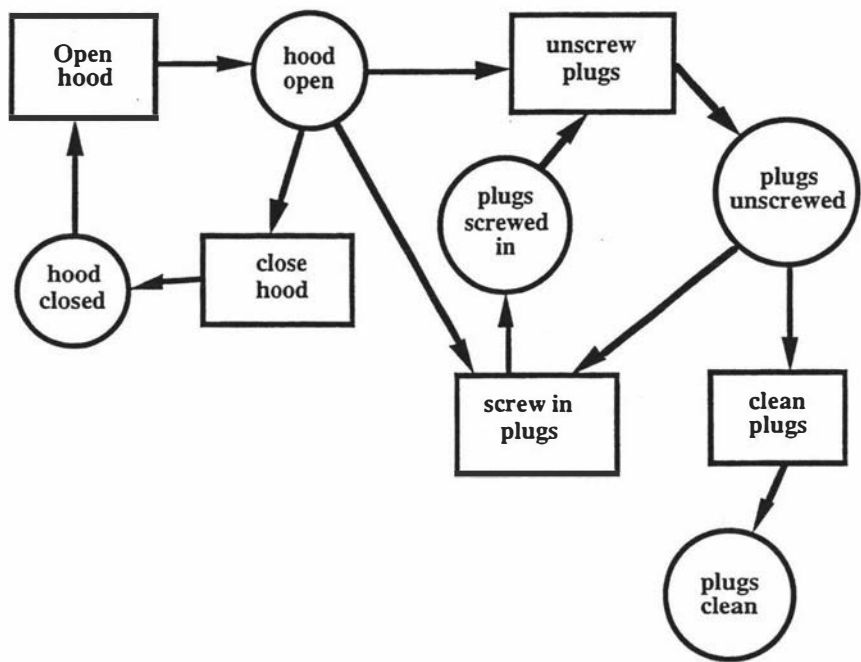


Figure 5.19 Plan net for cleaning plugs

Like the action table, the plan net notation allows modifications to be made quite simply, particularly where the changes are relatively independent of the original task specification. The result of adding the subtask 'check the radiator' to the car maintenance task is shown in Figure 5.21.

Since there is a direct correspondence between plan nets and action tables it is possible to convert automatically from graphical descriptions, such as those shown in Figures 5.19 and 5.21, to suitable action tables. The corresponding action tables can then be used as a basis for providing a dialogue system that can guide students in learning

about a domain. As a further example of the use of this graphical approach, plan nets for VCR operation are given in appendix B.

preconditions	operator	postconditions
hood closed	open hood	hood open
hood open	close hood	hood closed
hood open plugs screwed in	unscrew plugs	plugs unscrewed
plugs unscrewed	clean plugs	plugs clean
plugs unscrewed hood open	screw in plugs	plugs screwed in

Figure 5.20 Action table for cleaning plugs

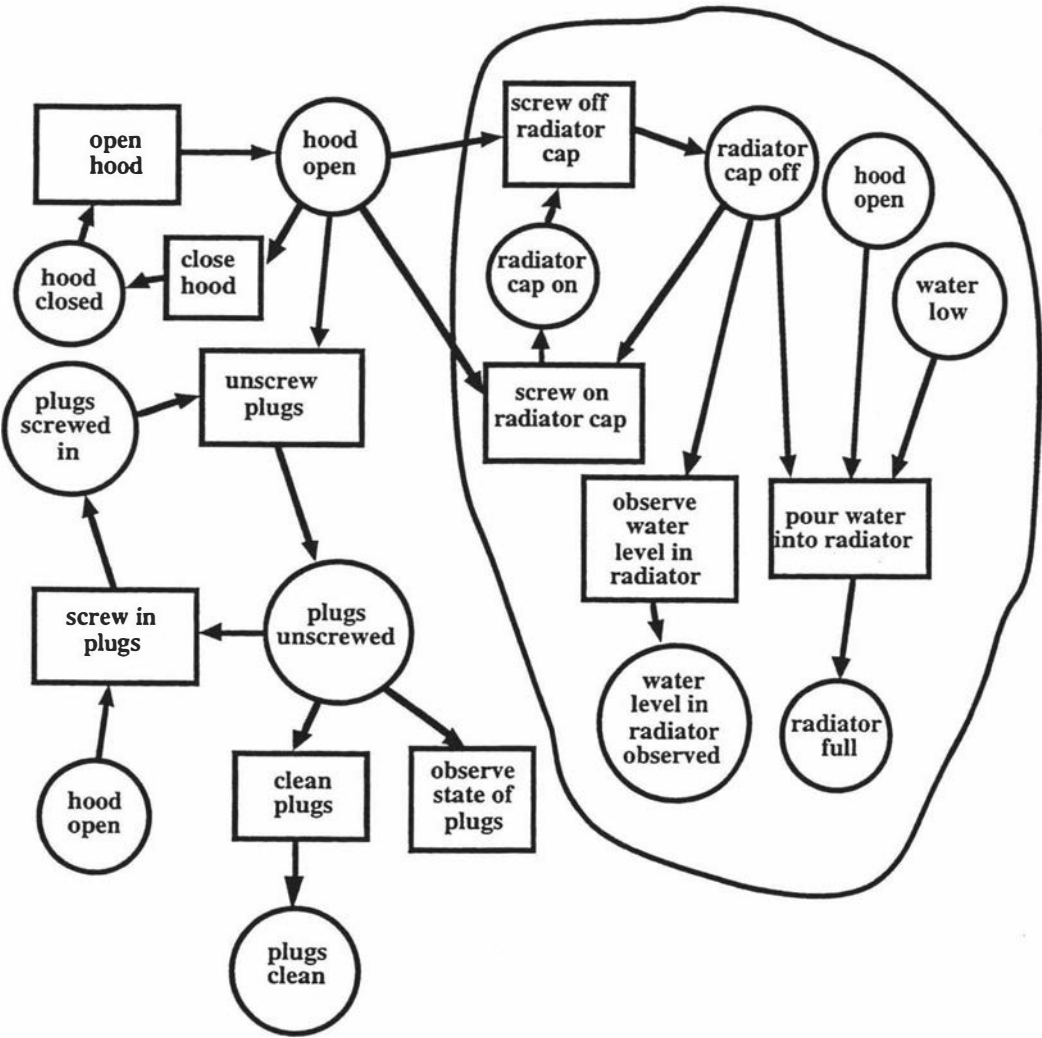


Figure 5.21 Plan net for cleaning plugs with radiator check added

5.5 Teaching knowledge

The emphasis in this thesis is on providing high-level primitives to allow educators to implement whatever teaching strategy is deemed appropriate, within the framework of a discovery learning system. However, it is appropriate to speculate on some of the strategies that might be used in this kind of environment.

Some educators are vague when it comes to specifying what teaching knowledge involves and how it should be used. A researcher who not only has a practical knowledge of teaching but has also given pointers towards a coherent theory of the subject is Stellan Ohlsson (1987). Gagne (1985), McKendree (1990) and Mark (1991) have also provided guidelines for feedback that are relevant to this research.

The user's performance will improve if s/he learns as a result of using the system. The kinds of learning that may occur are:

- changes in the user's domain knowledge (correcting or increasing his/her knowledge);
- an improvement in the user's problem-solving ability;
- an improvement in the interpretation skills of the user.

The role of the teacher is to facilitate learning. In order to do this s/he needs to be able to determine what aspects of the user's knowledge or performance are deficient. This may be ascertained from interviews, from question-answering or from observation of the user's actions during one or more sessions.

Timing is an important aspect of intervention. A computer tutor may be activated at any point. For example:

- when requested;
- when the user appears to be making a poor move;
- at some point later than when a poor move has been detected (eg when the user appears to be lost);
- at an appraisal point in the session.

One of the key issues is whether a detected error should be corrected straight away or whether the student should be allowed to continue. Allowing the student to continue may give him/her chance to detect errors or at least to see the consequences of the error. Some research suggests that the student benefits most when an error is pointed out straight away: see, for example, Corbett and Anderson (1989).

What the teacher does may depend upon the situation (and also the intended effect). The tutor may be required to:

- give minimal feedback (for example, just point out that an error has occurred);
- describe the nature of any error;
- supply information directly (for example, facts that the user does not know or has forgotten);
- supply information indirectly (for example, by presenting it in a different form such as graphically);
- give directions;
- make suggestions;
- provide practice for the student (for example, further case studies);
- give an alternative interpretation (perhaps an 'authoritative' view or one from another agent);
- provide a hint (for example, a subgoal that the user should aim for);
- get the user to consider his/her attitudes or actions.

An important factor that will impact both when the tutor is activated and what action is taken is the kind of error that the user is assumed to have made. Errors occur when the user interacts with the domain. The causes of the error may be complex inter-relationships between the user's lack of knowledge of the domain, lack of general expertise, or due to specific pitfalls within the domain itself.

Some useful work has been done by Jean McKendree (1990) on error analysis as applied to the domain of geometry tutors but the ideas generalize to many other domains where deduction is being performed. She distinguishes four types of error:

- 1 right premise/wrong rule;
- 2 wrong premise/right rule;
- 3 wrong premise/wrong rule;
- 4 wrong conclusion.

Ways in which these could correspond to errors that a user might make in a procedural domain can be surmised.

A person who knows what the current situation is but who, nevertheless, makes an unacceptable choice is committing an error of type 1. For example, in the VCR domain, they may realise that the television is off but may assume they can still select the video channel. This could be checked by asking the user if they are aware of the preconditions for the operation being attempted. If they are then it is likely that they are committing an error of this type.

If the student attempts to select the video channel thinking that the television is on then they are committing an error of type 2. This may not be a conceptual problem but just an oversight on the user's part. Here, it might be determined that the user realises what the preconditions for an operation are but does not know, or has overlooked the fact, that the conditions are not currently satisfied.

An error in the video domain might be classified as type 3 if, for example, the user presses the rewind button in order to play the tape when a tape is not even in the machine. Here the student would appear to be having problems both with how the device works and how the task is to be completed. If the user does not know either the preconditions or the effects of an operation that is currently being attempted then we might infer that this kind of error had been committed.

Lastly, the user might choose an acceptable rule that is applicable at the particular point in the session but may be under a misapprehension concerning its effect. Here, it could be determined that the user knew what the current situation was, what the preconditions for the operation were, but did not realise what the result of the operation would be.

Whether the system should be using any or all of these kinds of criteria for judging the student's knowledge of the domain or ability to carry out specific tasks is not considered here. It is merely observed that such mistakes can be readily detected using the representation developed.

How the system should react *after* an error has been dealt with is a further issue. One may constrain the system so that the user must take a particular path or one may allow him/her to go on. Carroll and McKendree (1987) suggest that a 'penalty-free' exploration may be effective. That is, the user may go on to see the impact of a poor decision but then may backtrack to the decision point to try alternative courses of action. In certain circumstances particular courses of action may be appropriate. For example, for the right premise/wrong rule situation we may provide a sub-goal for the user to work towards which is easier to attain. Because of the way the simulation is organised this kind of information is relatively easy to provide.

5.6 Summary

The methods for domain and task representation developed in Chapter 4 have been applied to various problems. The basic simulation approach that involves interpreting an action table and providing feedback on the allowable and non-allowable operations has been programmed. Output from simulation programs in both the gear assembly and the VCR domains has been given, demonstrating the different kinds of feedback on device operation that can be easily provided.

The method has then been adapted to include task information. Drummond's concept of situated control rules has been applied to the problem of providing appropriate feedback for a user learning how to carry out a task. It has been shown how the teacher and domain experts can decide upon the restrictions of choice to be placed on the user during a learning session and how these can then be used to generate SCRs. A program that uses these SCRs to provide task feedback has been demonstrated.

The basic action table representation is not easy to produce or check. Alternative graphical schemes have been investigated, and the use of plan nets has been proposed.

Together with the use of projection graphs, the plan net approach provides a 'developer friendly' environment. Most of the planning of device operation and of user feedback could be completed using computer graphics. An additional advantage of projection graphs is that they can be used for testing a simulator during development. A transparent, visual representation of what the user can and is doing can be shown to the designers, facilitating the detection of flaws in the logic.

Some possible feedback schemes have been suggested with an indication of how they could be implemented within this framework. As emphasized earlier, it is not the purpose of this project to investigate, in detail, how effective different feedback

strategies are. Instead, it has been demonstrated that the knowledge level representation proposed allows for a wide range of possible tutorial methods.

A complete system has not yet been built: it is the main objective of another project (Smith, 1994f). The eventual aim is for the domain and tutorial experts to cooperate to design a suitable procedural task tutor using an appropriate package. Many of the components of such a package have already been tried out. Figure 5.22 shows how they could be combined to provide a procedural task tutor design system. Note that the process could be terminated once the domain simulator has been produced if a task-oriented system is not required.

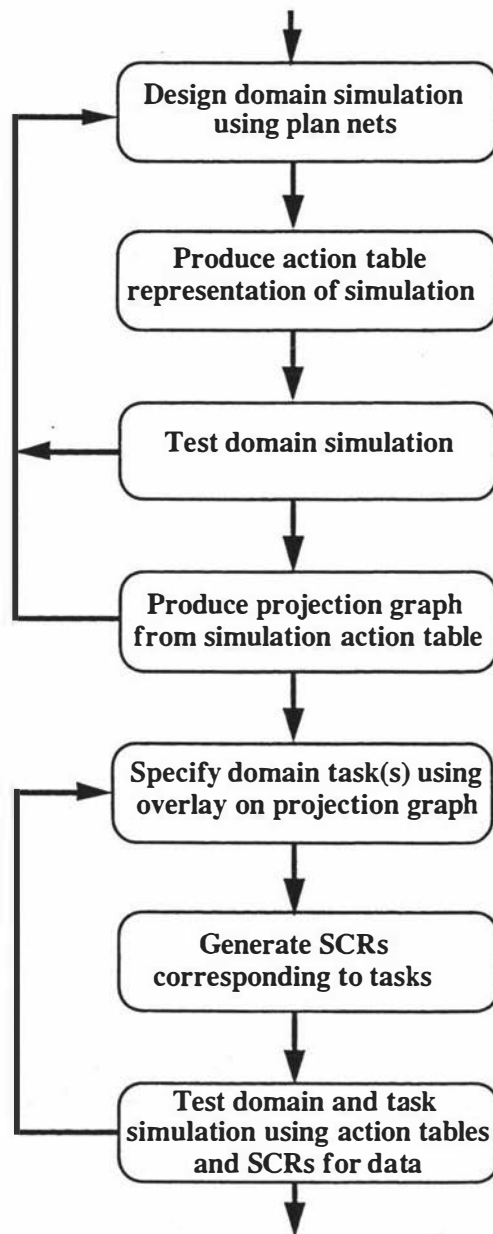


Figure 5.22 Automated scheme for developing a procedural task tutor

Chapter 6

Knowledge-based Simulation for Teaching

6.1 Introduction

It has been demonstrated that a modelling approach based on planning can provide a degree of cognitive fidelity and useful feedback in procedural domains. The question arises whether a similar approach can be applied for other kinds of problems. Possible application areas include game playing (Syms, 1992) and so-called ill-structured problems (Kemp, 1993). In this chapter, the latter are considered: in particular, the possibility of providing an interactive learning environment based on simulation of human behaviour.

Many simulation systems are opaque and so anything substantially more than observing and (possibly) manipulating the system becomes difficult. The main form of feedback in this kind of system is direct, from seeing what happens as various adjustments are made. van Joolingen and de Jong (1992) attempt to develop an educational environment to improve the feedback obtained from a simulation program. Their approach is to assume that the original program may be opaque but that, by appropriate knowledge acquisition from experts who know about the simulation, an environment can be built up to sit on top of it and provide the user with additional information and feedback. This is counter to the belief of Hartog (1989) that learning should be seen as acquiring mental models by students and that these should be the same models that drive the simulation. In this thesis a middle view is taken: that some simulation models will necessarily be opaque because of the nature of the domain. For such problems, a transparent teaching system cannot be developed, but transparency is highly desirable wherever possible. Some improvement

can be obtained using the van Joolingen and de Jong approach but here the focus is on domains where a cognitively faithful model seems feasible.

In some ways the models investigated here are similar to adventure type games that have been readily available for several years. In adventure games one is introduced into an environment (often a fantasy world) and expected to achieve some aim. Sometimes the aim is positive, like retrieving a crock of gold, at other times it involves just staying alive for as long as possible. Often, part of the fun of such games is discovering the internal logic of the world which may be very different from how things happen in the real world, but where (at least in good games) there is some consistency and plausibility in what occurs. Inevitably, one finds out something about the objects, creatures, and relationships in the make-believe world. Sometimes the games are geared towards teaching as in 'Dyno-quest' by Mindplay® although there is rarely any direct teaching component. The student learns almost incidentally during the running of the game.

There has been little study of the impact of adventure games in an educational setting and the work that has been done has concentrated on the effect on problem solving abilities of exposure to these games, rather than learning about particular subjects (see for example Sherwood (1990)). What we are looking at are simulation environments that facilitate learning in specific subject areas.

The overall goal, then, is to provide a system involving the depiction of human behaviour which will:

- present authentic-looking situations;
- allow the user to interact with the system, make choices and observe consequences;
- give the student feedback during the running of the simulation: providing guidance, suggestions, supplying reasoning, etc.

It is assumed that the simulation will be transparent. This makes it harder to develop (and impossible for some domains) but makes the problem of providing a seamless supportive environment more tractable.

In this chapter a number of ideas are brought together: simulation, knowledge-based simulation and representation in an ill-structured domain. Each of these topics has to be explored before the model that incorporates all the ideas into a discovery learning environment can be developed. First, the central technique of simulation is considered.

6.2 Simulation

6.2.1 Classifying simulation

Kreutzer (1986) distinguishes two types of symbolic model: analytical and descriptive. The latter is equated with simulation and involves 'model animation'. In a later work (Kreutzer, 1990), he uses the term 'simulation' to refer to any experimental exploration of symbolic models. Basically, in simulation we have some framework of information about objects, relationships and patterns of behaviour. This is then interpreted by executing the model according to these specifications. Some random elements may be introduced to simulate chance or to represent features of the model that are difficult to predict exactly. Different variations on the theme are illustrated in Figure 6.1, based on the analysis in Kreutzer (1986). Note that it is only a rough partition since there are some hybrid or unusual systems that might incorporate features of different classes.

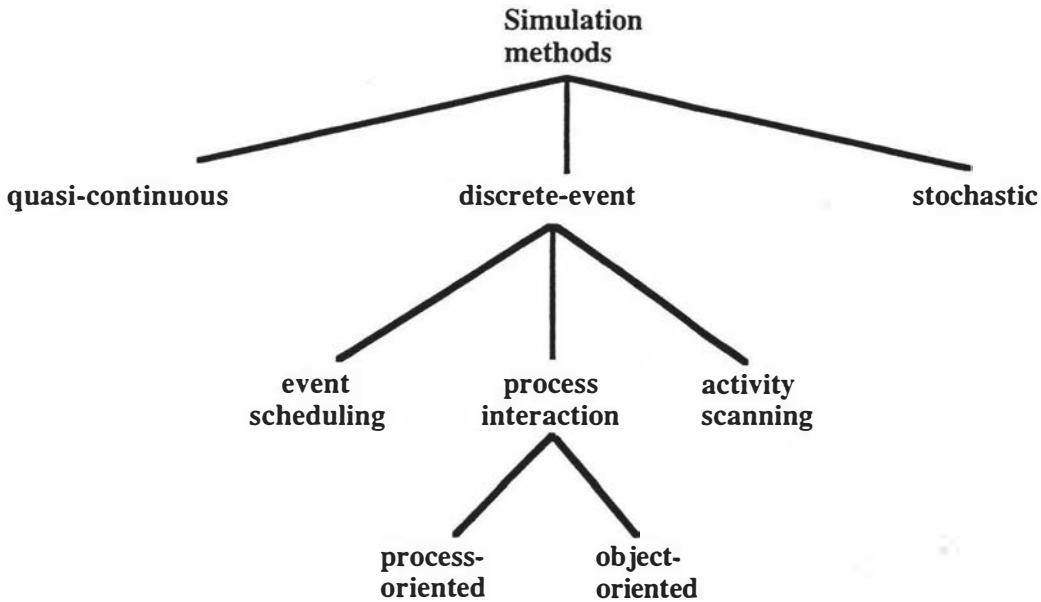


Figure 6.1 Classification of simulation methods

Two kinds of simulation that are only of marginal relevance to this thesis are quasi-continuous and stochastic. In the first of these, relationships between objects are reflected in rates of change which are typically represented in differential or difference equations (Karplus, 1977). These equations are stepped through in model time. In stochastic simulation, all relationships are defined in statistical terms, for example, by employing Monte Carlo techniques (Hammersley and Handscombe, 1964; Kreutzer, 1986). Both kinds of simulation are centred on quantitative methods.

In discrete-event simulation, as the name suggests, the central concept is that of the event. An event is associated with one or more concurrent changes in the system. These 'state transitions' are assumed to occur instantaneously. The assumption is also made that no significant changes occur between these state transitions. Changes are determined by reference to relationships between entities (such as cause-effect) and also by using statistical techniques. Statistical methods are often used where detailed analysis would be difficult or is irrelevant.

Time is measured by a 'clock' that is incremented as necessary. The clock mechanism is normally asynchronous in discrete-event simulation. That is, the clock jumps from the time of one event to the time of the next one rather than by equal time slices, although, generally, the time intervals are a multiple of some minimum time-slice. The model is evaluated at each event time to determine what changes have taken place and to note ramifications that may affect future states of the system.

Two other important features of discrete-event simulation are the monitor and the agenda. The monitor or 'run-time control system' sequences state transitions and generally uses an agenda (event list/noticeboard/scheduling list) to keep track of pending events.

Within this framework there are important sub-classes depending upon the view that is taken of the entities within the system. This view may be:

event-based - the emphasis is upon the events themselves (basically, the effects that the interacting entities have);

process-based - concentrating upon the processes that precipitate the events;

activity-based - this focuses upon activities within the system (the different kinds of interaction that occur between the entities during the simulation).

The event-based and process-based paradigms are usually implemented in an imperative style. In imperative models the emphasis is on commands carried out, messages sent etc, and the programming of the interaction of the components may be quite complex. Activity-based systems are often programmed using a declarative approach. Here, the focus is upon descriptions of entities within the system and the conditions under which they are activated. The control structure for such a system is often, therefore, much simpler but a great deal of work needs to be put into the design of the components and their behaviour.

Finally, we may distinguish two different methods of implementing process-based simulation. We may implement processes and their interactions in a standard algorithmic fashion, or alternatively, we may give more autonomy to the processes by using an object-oriented style (Booch, 1991), allowing processes to have private methods that can send messages to one another.

All the above classical simulation systems have a very elementary representation of the entities within the system and their interactions. Certain aspects are approximated by employing statistical distributions. Others that do involve logical sequencing of events and cause/effect type descriptions are very simplistic. For example, a typical 'logical' sequence of operations is 'customer arrives, customer joins queue, customer gets served'. A cause-effect pair could be 'customer joining empty queue causes server to start service'. If we want to give feedback to students as a simulation is in progress, or want the student to be able to take part in the simulation then more detail is needed. There needs to be a reduction in the reliance on statistical or numerical approximation and the capability to expand or reduce the detail given as necessary. Also it would be desirable to include a larger selection of entities, a wider variety of different actions and more detailed descriptions of possible interactions. These facilities can be provided in knowledge-based simulation.

6.2.2 Knowledge-based simulation

Knowledge-based simulation has been applied to a wide range of activities and is often associated with decision making in the sphere of operations research: see, for example, Fishwick and Luker (1991). The term was coined by Klahr and Faught (1980) where they used the approach for simulating military air battles. The aim was to provide increased transparency in simulation programs by explicitly representing experts' knowledge of the domain.

Motivation for the work came from the observation that large scale simulators previously developed were generally unintelligible, difficult to modify and lacking in justification of results. Expert knowledge was generally used to produce these simulations but was hard-wired into the code in such a fashion that the software became totally opaque. Often this was done in the name of efficiency or because suitably descriptive programming languages were not available.

Klahr and Faught take the view that a decision based simulator is a knowledge-based system. The behaviours, interactions of objects, decision making rules and so on 'are all pieces of knowledge that can be made explicit, understandable, modifiable, and can be

used to explain simulation results' (Klahr and Faught, 1980, p181). They use a language called Director (Kahn, 1979) which is object-oriented and the general system, as written, fits into the discrete event, process interaction, object-oriented category of Figure 6.1. Although an object-oriented approach is taken for describing static elements of the system the dynamic aspects are modelled as causally dependent event chains.

Subsequently, knowledge-based simulation has been used in a variety of settings. Although there is no universal agreement on the exact meaning of the term, a knowledge-based simulation system would be expected to have the following features:

a simulation component: a behavioural model of sequences and (possibly) timings of events in the real world;

explicit representation of knowledge of one or more human experts in the domain;

causal modelling of sequencing of events.

The knowledge-based simulation approach has been found to be particularly useful when modelling systems involving human behaviour. Although the utility of such models can be readily appreciated it is not clear how best to develop them. As a first step, the nature of these models will be considered, and then some systems that incorporate these ideas will be surveyed.

6.2.3 Knowledge-based simulation in ill-structured domains

Computers, historically, have been used almost exclusively for solving well-defined problems. Virtually all problems in the real world, however, start off by being vaguely specified. It is generally accepted that computers need to work on a precisely defined problem specification and, in fact, the whole discipline of systems analysis and design is based on this belief, and has the aim of producing a clearly defined description of a problem and of proposing a method of solution.

Even in the area of artificial intelligence this quest for precision is paramount. Minsky sets the ground rules by stating that 'for each problem we are given some systematic way to decide when a proposed solution is acceptable' (Minsky, 1963, p408). He goes on to specify the problem solving process in terms of clearly defined states and transitions between states.

Some authors have observed that, for many important problems, such a clear-cut description is difficult, if not impossible. Reitman (1964) introduces the notion of *ill-*

defined problems: ones that, in Minsky's terminology, are incompletely specified either in terms of initial state, transitions or final state. Reitman distinguishes six categories for such problems that include tasks like writing a fugue, making a stereo cheaper to produce, and using operations research in an organization. Curiously, he omits the ubiquitous problem of teaching.

In education, the aim of 'teaching a student' is vague but difficult to make much more precise. Educationalists, over the years, have struggled to develop effective strategies for teaching and for measuring achievement. To an extent, satisfactory methods of teaching and assessment depend upon the nature of the domain. Among the more problematic areas for teaching are what Spiro, Feltovich, Jacobson and Coulson (1991) call *ill-structured domains*. These are ones where the content is particularly complex and where the behaviour of the system becomes difficult to predict. Often these domains have human agents or groups as components within the model, and involve attitudes, opinion, beliefs, and judgement rather than facts and set procedures. Some packages that have been developed for ill-structured domains will now be considered.

The Strategic Automatic Discovery System (STRADS) (Oresky and Lenat, 1991) uses knowledge-based simulation to help intelligence analysts understand and anticipate events in 'geo-political' situations. For example, it has modelled possible behaviours of political terrorists and extreme religious groups in particular areas of the world.

The knowledge expert is an intelligence analyst who will have deep knowledge of a region with respect to politics, geography, religion, economics, etc. The aim is for the system to aid the analyst in considering different scenarios and outcomes. S/he can vary many aspects including the general situation, the protagonists, the initial conditions and assumptions about behaviour. Random factors can be included to cope with unexpected outcomes.

The knowledge base of STRADS contains five different types of objects: actors, events, scripts, rules and slots. Each hierarchy allows for inheritance of default values, thereby improving efficiency and transparency. Actors, events and scripts are particularly relevant for our purposes and will be described in more detail.

The *actors* represent real world entities that respond to different situations. An actor may be an individual who is influencing events, such as a president or prime minister. Groups such as terrorists or the Senate also count as actors since they can react and make decisions as a body. Even whole countries or blocks of like-minded nations can be

considered as actors if they are responding in unison. Likely behaviour of actors can be anticipated by storing knowledge about their beliefs, attitudes and goals.

An *event* is associated with an action that occurs in the world at a particular time and place and corresponds to the normal understanding of the term in simulation. A STRADS event would describe who was involved, what action was performed, how it was performed, when, where and why it took place, and who knows or believes it took place. Events occur when actors respond to previous events that they know or believe took place. They can also be caused by naturally occurring conditions.

Actors often have set behaviour patterns in stock situations. For example, a country may take a stance of being initially outraged when an unfriendly country conducts nuclear tests, demanding that the tests stop, and, if they do not, claiming that their own country will have to conduct tests to ensure they do not lose ground in nuclear capability. Such patterns are referred to as *scripts* and, loosely speaking, correspond to the term as coined by Schank and Abelson (1977), although, in this setting, they are more closely associated with goals rather than stereotypic situations.

Using the scheme outlined in Section 6.2.1, the control structure of STRADS would be classified as discrete-event, event-based. There is an asynchronous clock and future possible events are kept on an agenda (termed a 'global event list'). A variation on the standard organisation is that although each event has an (estimated) time for occurrence this is not immutable. For example, other events may have made the stored event redundant (if a dictator has been assassinated then an event that has him/her making some decision is no longer relevant). Also an event may have pre-conditions associated with it that need to have been satisfied before it can take place. If these conditions have not been met then the event may stay on the agenda with a new estimated start time generated. As with classical discrete-event simulation, the processing of an event may cause new ones to be anticipated and these may be added to the agenda.

Since STRADS is conceived as an aid to analysts there is not a great need for a teaching component within the system. So, for example, it is expected that the analyst will be able to appraise events as developed in the simulation and to be able to understand them in the context of his/her knowledge. No explicit guidance is given to the user. Any learning that takes place is solely through observing the events that occur and by seeing the impact of one's own input.

The CACTUS system (Hartley *et al.*, 1992) is a simulation in a very different kind of domain. The trainee is to gain experience in command and control of police units at

major incidents where public order may be at risk, such as large demonstrations and marches. The basic scenario is that a large demonstration and march is to take place in a city centre. The proposed route of the march is known and various police units are available for deployment at points on the route. The trainee attempts to facilitate the smooth running of the march.

The actors within the simulation are the crowd groups (leader groups and demonstrator groups), police units (both deployed and reserve) and the trainee him/herself who takes the role of the base commander of these units. The actors have autonomous components such as the behaviour of the crowd and the reaction of the police in certain circumstances. These are stored as localized methods and have a probabilistic basis. The trainee can over-ride the pre-programmed behaviour of the police units by giving appropriate orders (either direct orders or advice on strategy).

The simulation has a world state that gives information about what each of the actors within the system is doing at any given time. The system is synchronous with time being advanced in equal units of twenty seconds. At the end of each interval the actors review the current situation and see if a change in activity or behaviour is to be generated. Within the classification of Section 6.2.1 this can be regarded as a process-based simulation employing an object-oriented approach.

The behaviour of the groups is represented in probabilistic transition nets linking related behaviours. So, for example, there are direct transitions between such actions as 'marching angrily' and 'throwing missiles'. This is the core of the knowledge-based activity and was produced in consultation with the police. A diagram depicting the nets is shown as Figure 6.2.

The authors note that the advantages of having a 'knowledge-based world model' include the variety of incidents that can be represented and the customizing that can be done to concentrate on specific kinds of problems. Thus the scenarios can be tailored to the objectives of the trainer and the background of the trainee.

CACTUS does not have any coaching component, and it would be difficult to envisage how one might be incorporated since the knowledge used in the system has been processed and converted to a form that is not readily explainable. One of the authors has noted (Williams, personal communication) that if the system were to be re-written then the knowledge would be incorporated in a more explicit form so that feed-back could be more readily given to users.

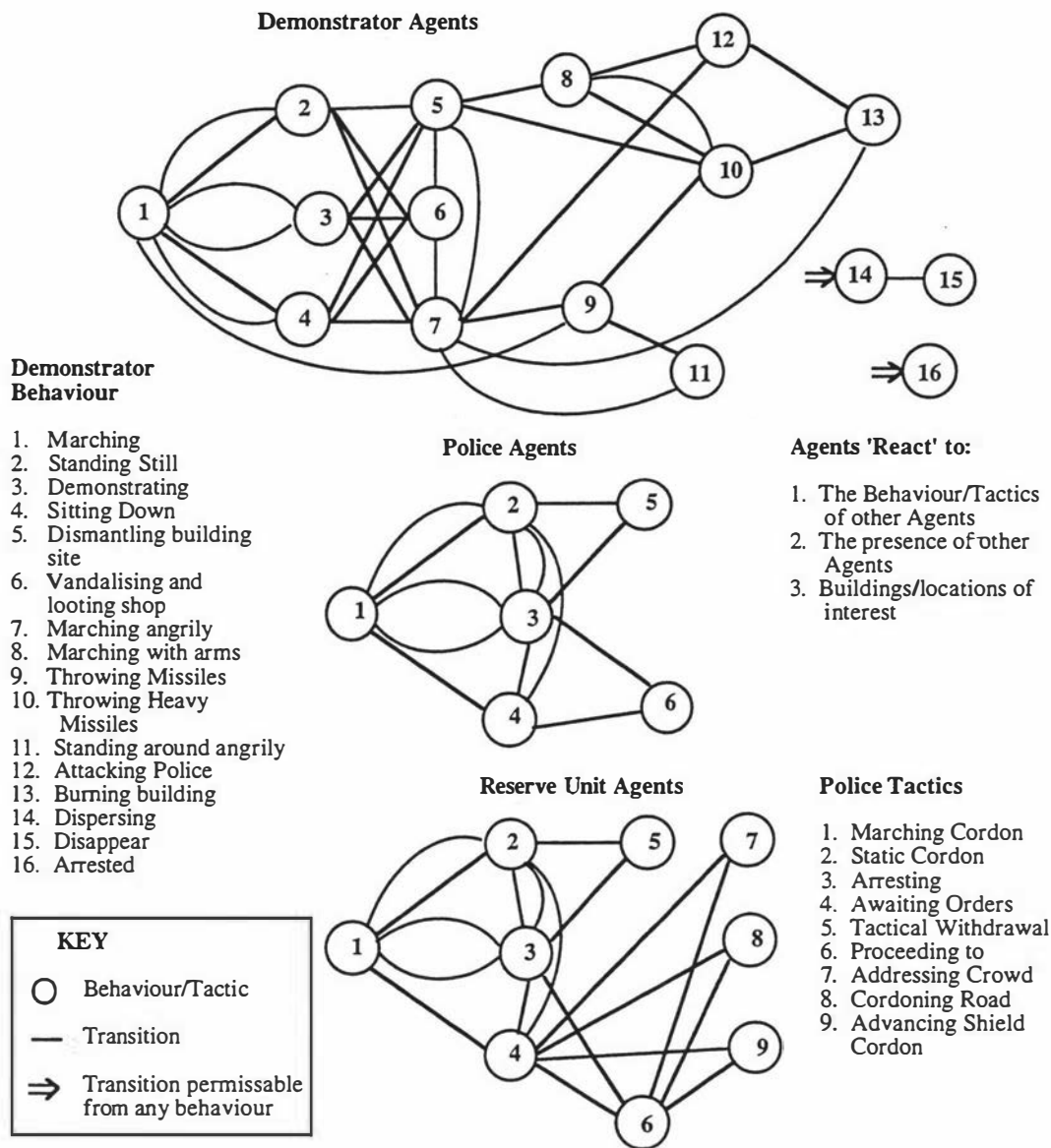


Figure 6.2 Behaviour networks (Hartley *et al.*, 1992, p131)

Within the simulation there is a large degree of flexibility. The size of the march, its intended route, the size of subgroups within the main group, the number of police units available and their deployment can all be determined beforehand. The authors specifically note that this pre-event planning task can be made part of the training exercise. Another important aspect of the teaching function of the system is that all decisions made and activities within the system are logged and can be used in a de-briefing session.

A knowledge-based simulation package that makes extensive use of multi-media is the fire-fighting simulator called ICCARUS (Powell, 1992). It is used to train senior fire brigade officers in command and control of major fire incidents. The trainee has to learn how to assess the likely progress of the fire, what precautions to put in place, potential hazards, how to deploy the fire pumps that are available and whether to request further

assistance. S/he reacts to situations and gives commands in an audio-visual presentation. Screens showing videos of fires and sound cues are included.

The simulation follows an unplanned path determined by the 'semi-autonomous actors' (up to 200) with a preset description including the scale of the fire, the number of 'seats' of fire, hazards in rooms, etc. As in the real world, neither the fire nor the fire fighters behave in totally predictable ways and there are plenty of hidden traps such as cans of kerosene left in rooms. ICCARUS embodies knowledge of experts on the behaviour of fires and personnel in this kind of situation, and the problems that might occur with particular kinds of buildings. The behaviour of the actors is represented by PLANNER-type specifications (Hewitt, 1969). The organization of the simulation is not described in any great detail but it appears to be a discrete-event simulation with a process-based approach.

Like CACTUS, the emphasis is upon learning by being involved in realistic situations. No feedback is given during the simulation. The user learns by seeing the impact of decisions s/he has made. As with CACTUS it would be difficult to give intelligible feedback since there is a high probabilistic component in the simulation.

It is instructive to classify the knowledge-based simulation systems that have been considered in this section, taking into account the important issues in an educational setting. Results of an analysis are shown in Figure 6.3.

package	causal transparency	user interaction	explanation facilities
STRADS	high	high	none
CACTUS	medium	high	none
ICCARUS	low	high	none

Figure 6.3 Analysis of knowledge-based simulation packages

To an extent, a proper assessment of explanation facilities cannot be made since CACTUS and ICCARUS simulate real time activities where speed of reaction and thinking is important. It would therefore be inappropriate to evaluate choices during the session itself. However, explanation of where people went wrong or what alternative choices could have been made could be included in a post-session appraisal. In all of these systems it is difficult to include explanation facilities because of the way the knowledge has been compiled into the system.

STRADS, CACTUS and ICCARUS all attempt to simulate some activity that involves human beings. Modelling human behaviour is hard but there is an increasing demand for systems that can do so. Examples of recent packages include Regimental Surgeon (Henderson, 1992), REALPOLITIK II (Livergood, 1993), 'The Case of Dax Cowart' (Reeves, 1992) and Try's Management Training System (Try, 1992). In Regimental Surgeon, the user is in command of a mobile army surgical hospital (a tent) and has to make decisions on what to do with casualties as they arrive and what priority to give each one. REALPOLITIK II looks at how international companies and corporations might behave based on business goals and local political circumstances. 'The Case of Dax Cowart' is a multi-media package that explores philosophical issues related to life-and-death situations. Lastly, Try's system simulates how a junior bank manager might deal with different inter-personal situations relating to the staff.

As can be seen the application areas are many and diverse. The packages all have in common that they attempt, at some level, to deal with human behaviour and try to involve the user as one of the participants in some developing situation. None of them incorporates a deep model of the actors and their behaviour and so any feedback that is given is added as an overlay.

In order to make progress towards a deep model of knowledge-based simulation that could be used for teaching, two central issues will now be addressed: the representation of situations (scenarios) and modelling the behaviour of people in these situations.

6.3 Scenarios

The scenario incorporates all the information about a domain and the particular events that are occurring therein. For example, if a training package for police officers was required then the scenarios might include domestic disputes, late-night public disturbances and more serious crimes of violence. In each case a full analysis of individual incidents and how they fit into a higher level pattern needs to be taken into account.

The approach taken to scenario design here is based on scripts and frames. One of the most comprehensive development schemes involving these objects is that used in CYC (Lenat and Guha, 1990). In this formulation the structure of an object defines the set of constraints that hold between its parts. Since there may be orthogonal ways of viewing an object (eg physical and functional), there is not just a single structure associated with an object but a set.

This view of how objects should be represented extends to temporal phenomena. Each dynamic process is represented by a special frame called a script. In addition to normal frame slots it has three other elements:

pre-conditions on the actors in order for the script to occur (may be necessary or sufficient);

the 'parts' of a script (these are of two distinct types: the actors, and the events in which they participate);

the changes that occur if/when the script is carried out (changes in constraints, final constraints, etc).

Based on these ideas a structure for events within the system has been formulated and the outline is given in Figure 6.4.

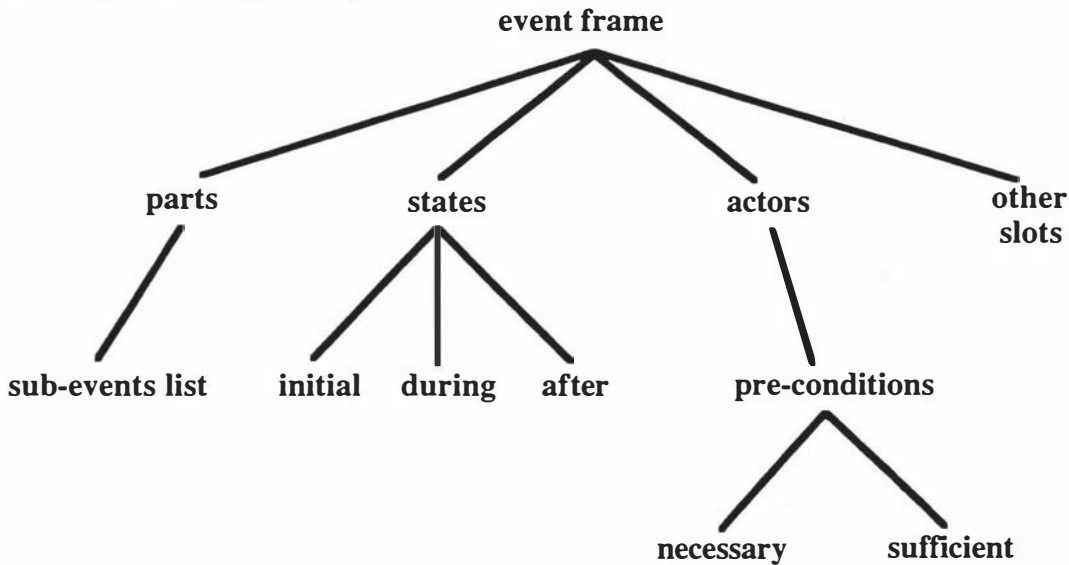


Figure 6.4 Hierarchy of events for knowledge based simulation

Although Lenat is not specifically working in the area of tutorial systems, various elements of his design facilitate the development of material for teaching. Sub-event break-up is important because of the issue of granularity which was explored by Hobbs (1985), and taken up by others such as Greer, Mark and McCalla (1989). Not only the amount of detail may be important but also the view that the actors take of the situation. Each event may be seen from different perspectives. This leads to the idea of multiple viewpoints, another issue that has received attention recently (Moyse, 1991).

Note that the pre-conditions on the actors, besides driving the system, can provide advice to the user. For example, in a domestic disputes scenario, the conditions that might apply

before a domestic crisis occurs include marital problems (long term) and the husband arriving home drunk (short term). Assumptions that a police officer might make also influence his/her behaviour. An officer's beliefs that, in some sense, the wife has caused the violence or is free to leave the marital home give a rather one-sided view of the situation that can be questioned by appropriate intervention during the interaction.

Although this scheme addresses some of the problems involved in modelling human activities, giving a useful framework for the development of sequences of events, it does not look at human behaviour or social interaction in any detail. To drive the simulation, it is highly desirable to include information about what people are likely to do in particular circumstances. How this might be achieved is considered in the next section.

6.4 Modelling behaviour

In order to provide the level of agent interaction required, some detailed consideration of human behaviour would appear to be needed. Only if this is incorporated, can justification, explanation and alternative actions be communicated to the user. CYC as described in the 1990 text (Lenat and Guha, 1990) does not give the kind of detail that is required for a system giving user feedback. Alternatives considered were models by Dyer (1983), Davis (1990), Sanders (1989) and Schank and Abelson (1977). The last of these was chosen as a basis, since it gives the detail and generality required for this particular application.

6.4.1 Schank and Abelson's behaviour model

Perhaps the most influential early research on computer modelling of behaviour came from Roger Schank and his colleagues. Their initial motivation for representing human behaviour was to automate the analysis of natural language, particularly stories and scenarios. In an early work, Schank (1975) uses conceptual dependency theory (CD) to analyse sentences and break down the meaning of words into primitive concepts such as 'physical transfer of possession' (encompassing such activities as giving, taking, stealing and buying), and mental activities such as 'constructing new information from old information' (including deciding, concluding, imagining and considering). Although Schank was satisfied that at some level he could encompass the meaning of all words with these concepts there seemed to be a missing dimension. A frequently cited example is the concept of a kiss, which can be represented in CD by primitives showing the moving of two sets of lips so that they touch. However, the significance or repercussions of such an act, which may be crucial in the development of a story, are not considered at all in this notation.

One of the aims of Schank's subsequent research was to put the meaning of words and activities into human contexts. In conjunction with Robert Abelson (Schank and Abelson, 1977) he considers human beliefs, behaviour, goals and motivations. Since it has greatly influenced the approach used for the model in this chapter the relevant ideas as they relate to the current research will be presented here.

Human behaviour can be analysed at various levels and from different perspectives. At the highest level 'universal' human traits can be represented: that is, patterns of behaviour that can be assumed to be shared by the vast majority of humankind. For example, the assumption is made here that people will normally attempt to alleviate their own pain (physical or mental). Consideration of these general traits gives what Schank and Abelson call 'expectancy rules', a term coined by Hemphill (1973).

Expectancy rules indicate how people are likely to react generally. A partial list of such rules is given in Figure 6.5.

1. If a person's state becomes negative (either physically or emotionally) then that person will take action to attempt to improve the situation.
2. If a person believes that his/her state (physical, emotional or mental) may deteriorate if nothing is done and taking some action may avert this then the person will carry out the action.
3. If a person performs an action which causes a negative change in another person's state then the second person may perform another to cause a negative change in the first person's state.
4. If one person performs an action which causes a positive change in another's state on some level then that person may perform another action to cause a positive change in the first person's state.

Figure 6.5 Expectancy Rules (Schank and Abelson, 1977, p122)

The first rule deals with alleviation of a condition that is perceived to have deteriorated (for example, taking an aspirin to combat a headache). The second rule considers circumstances where the actor believes that things will get worse if nothing is done (for example, climbing a tree when being chased by a dog). Here, the aim is to ensure that the situation does not deteriorate rather than actually improving it. Rules 3 and 4 denote the classic revenge and gratitude scenarios, respectively.

Although everyone might have the same high level goals, we may have different perceptions of our current situation and may have different ways of achieving our aims.

Here is where individual aspirations and circumstances come in: what Schank and Abelson call *themes*. Themes provide the main driving force for people's goals and plans. They consider three variations: life, interpersonal and role themes.

Each individual will have some guidelines that influence his/her behaviour. These may be fostered by parents, assimilated as advice from others or developed as a result of experience. We inherit or develop attitudes that are used to determine how to behave in certain circumstances: honesty, aggressiveness, greed, courtesy. Life themes reflect each person's general attitude to life, aspirations, belief in how to achieve success and so on. These attitudes can often be encapsulated in maxims such as 'honesty is the best policy', 'do as you would be done by' and 'get rich quick'. A person will often have many of these guidelines that interact and that may interfere with one another. For example, a good way to get rich quick might be to carry out a robbery but an honest person may baulk at this possibility.

Schank and Abelson's classification of life themes is shown in Figure 6.6.

personal quality: honesty, loyalty
ambition: success, getting a certain job
life style: luxury living, living frugally, travelling, being adventurous
political attitude: anarchist, right wing, radical
approval: fulfil parent's expectations, be liked by women
physical sensations: staying high, constant sex

Figure 6.6 Life themes (Schank and Abelson, 1977, p148)

Although life themes influence people's behaviour when interacting with others, they will be tempered by their attitude and relationship to them. If we have a high regard for a person or that person has some hold over us then we may behave in a different fashion from when we dislike and can dominate them. Commonly occurring relationships include spouse, parent/child, mentor/acolyte, lover, enemy, boss/employee. Some of these may be mixed together. For instance, a son may be the enemy of his father, a boss may be a mentor of an employee. These are *interpersonal themes* and indicate attitudes to others and reactions to them.

Interpersonal themes help explain people's behaviour to others. They can be used to draw up expectancy rules for goals. For example, if A is in love with B then:

if C has caused harm to B then A may attempt to cause harm to C;

if B is in a particular mental state then A may be inclined to be in the same state;

if B has a goal then A may promote this goal;

if B is not in love with A then A may attempt to change the situation.

It may be surmised how these interpersonal themes can impact on the general guidelines. For example, the last two expectancy rules in Figure 6.5 indicate how a person might behave to another as the result of some action (by taking revenge or showing gratitude). These reactions may be heightened, reduced or even negated by one's relationship to the person performing the original act.

At a more mundane level, a person's general behaviour may be guided by what job they have, what position they hold or what role they have taken upon themselves (doctor, garbage collector, nurse, president, burglar). This provides what Schank and Abelson call *role themes*. A person with a job has certain actions that s/he will take, certain things s/he will do. A person's role gives them a reason and justification for doing things.

A person's role is just one of many factors that governs their actions and may be overridden by other considerations. A bank-teller may be persuaded to fiddle the takings from motives of greed. A waiter may, in anger, pour soup into a customer's lap. Lastly, a person may have several conflicting roles. The archetype of someone in this predicament is Pooh-Bah (in Gilbert and Sullivan's *The Mikado*) who holds a number of high offices of state:

'... as First Lord of the Treasury, I could propose a special vote that would cover all expenses, if it were not that, as Leader of the Opposition, it would be my duty to resist it, tooth and nail. Or, as Paymaster-General, I could so cook the accounts that, as Lord High Auditor, I should never discover the fraud. But then, as Archbishop of Titipu, it would be my duty to denounce my dishonesty and give myself into my own custody as First Commissioner of Police' (Gilbert, 1992, p11).

A person may aspire to become rich and how s/he does this may be affected by other themes. For example, if the person believes in honesty then some of the means of achieving this end, such as robbing banks or blackmailing, may be out of the question. An actor may be very environmentally aware and this may significantly affect their actions or goals. They may become a vegetarian or quit their job as an aerial topdresser. Thus there is much interaction between different themes and kinds of themes that has to be taken into account.

Once goals have been determined, means (*plans*) for achieving these have to be devised. This will involve a sequence of actions that will hopefully lead to the desired goal. Novel

plans have to be generated somehow: using means-end analysis (Newell and Simon, 1963; Ernst and Newell, 1969), for example. Other plans occur quite frequently and so do not need to be worked out from scratch. Schank and Abelson call these 'named plans'. A more elaborate form of this concept, called a MOP (Memory Organization Packet), is described in a later paper by Schank (1982). MOPs are similar to scripts except that various intentions and expectations are built into the structure, enabling detailed feedback to be provided.

Although expectancy rules, themes and goals provide the background to determining the general inclinations of people as they interact, a blow-by-blow account of individual interactions is still required. Cause-effect chains can be used to generate plausible sequences of interactions. The kind of cause-effect phenomena that Schank and Abelson distinguish in human behaviour is not very different in form from that of device functioning. They list the kinds of transitions that might occur (Schank and Abelson, 1977, pp25-27):

- actions can result in state changes;
- states can enable actions;
- states can disable actions;
- states (or acts) can initiate mental states;
- mental states can be reasons for actions.

The last two often occur in sequence: a situation leads to a person taking an action. It is the basis of the planning idea: one considers the current situation and decides what to do. The above *causal semantics* (Schank and Abelson, 1977) is at the bottom level in the progression of any session, guiding or monitoring the progress of the developing scenario. It can also be used as the basis of feedback in a systematic fashion as will be demonstrated later.

Although there will normally be some logic (rationale, excuse) for actions, these may be ritualized, based on convention, etc. This is where *scripts* come in. There have been various interpretations of the term since, but Schank and Abelson originally envisaged them as stylized sequences of expected events. They view them as giant causal chains where humans automatically fill in the missing steps. Steps can sometimes be justified in logical terms but, more frequently, just reflect 'the way things are done'.

When trying to understand someone's actions, Schank and Abelson suggest it is better to try scripts first since these are already worked out, and may contain conventional social practices etc that cannot be readily understood without a reasonable amount of real world knowledge.

6.4.2 A behaviour model for teaching

In the simulation of a physical system the general flow, at the most simplistic level, can be considered to be: situation \rightarrow action \rightarrow situation \rightarrow action At each stage, a new situation enables an action to be executed and each action causes a change in the situation. There are some similarities between the causal chains that are modelled in physical systems and the way that humans act and respond. How far this analogy can be taken remains an open question, both for philosophers and psychologists. A philosophical perspective is given by Davis (1979) who believes that causal events have an inevitability and are logically deducible, whereas human actions have some element of choice. He concedes, however, that although human behaviour is not totally predictable, it can be surmised and can usually be explained once it has occurred.

When considering human behaviour, the psychologist John McClure (1991) distinguishes three aspects: intentions, reasons and causes. *Intentions*, he states, 'articulate a plan of action' and may be conceived as the class of response one might give to a question such as 'what are you planning to do?'. *Reasons* give a motive or rationale for an action and may be typified by answers to the question 'why are you doing this?'. Lastly, *causes* 'comprise a mechanical explanation of behaviour' and correspond to replies to the query 'what makes you do this?'. This extends the earlier ideas of, for example, Davis (1979) who distinguishes intention and reasoning. He characterises intention as how the agent thinks of the action, relating it to the notion of *purpose* (although the latter term is sometimes associated with the idea of reasons). Davis uses the word purpose exclusively to mean 'what the agent aimed for'. Reasons are more related to beliefs and desires. More precisely, beliefs are held that a particular desire can be fulfilled by a particular action.

Some of these ideas are incorporated into the model of behaviour considered here. In scenarios where actions are being executed by humans, it is useful to distinguish the mental states of these actors. It is assumed that the way that someone feels will influence how they behave. This augmented model of cause and effect is shown in Figure 6.7.

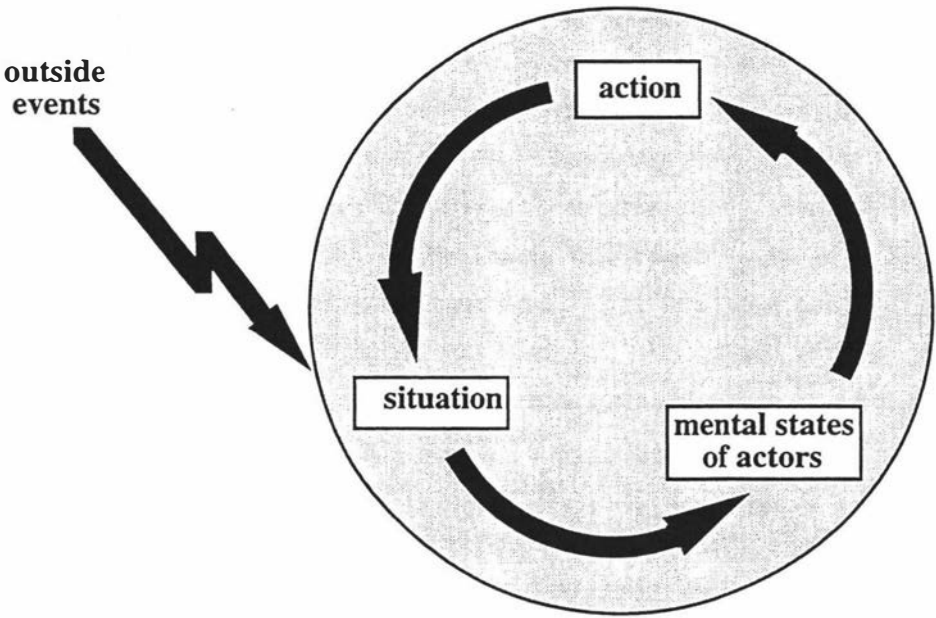


Figure 6.7 Interaction of situations, mental states and actions

The shaded area shows the scope of simulation of interpersonal relationships. Outside events occur and have to be considered but their justification cannot necessarily be included in any detailed fashion. To be able to include a full qualitative simulation would require an intricate model of the whole world on the scale of the CYC project (Lenat and Guha, 1990). However, the reaction of people in the system to situations influenced by outside circumstances can, arguably, be simulated. The initial impact is upon their mental state, and this then will result in some action that changes the current situation. The kinds of action that may be taken include verbal as well as physical.

Even within this simple framework some feedback can be given. The different kinds of guidance are summarised in Figure 6.8.

operation	possible feedback
actions resulting in state change	alternative results of action indicated desirable action recommended action given to achieve specific state
states initiating mental states	effect of current situation on mental state of actor current mental state of actor situation that might induce given mental state
mental states initiate actions	what person is likely to do

Figure 6.8 Possible low-level feedback based on causal semantics

It might appear that the model given in Figure 6.7 takes a behaviourist approach where a situation causes someone to react in a certain way that then changes the situation. This is not, in fact, the case. Rather than thinking of the diagram as denoting sequencing of operations, it is perhaps best to think of it as a petri-net (Reisig, 1985) where branches show direct influences but where a specific sequencing is not implied. For example, although a situation may change the mental state of a person, this does not necessarily lead immediately to an action. A person's mental state may incorporate short (or even long) term goals that dictate their actions over a period of time. The changing situation may give them feedback on whether their aims are being achieved.

Another assumption implicit in this model is that a person's actions depend not upon the situation but upon their *perception* of the situation. This will be interpreted using their current beliefs and predilections to determine their reaction. These distinctions may be crucial to help explain seemingly irrational behaviour.

More information is needed in order to animate this model. In particular, the mental states of actors within the system have to be described in enough detail to be able to determine how their behaviour is affected. A detailed psychological model of behaviour of human beings is, of course, currently out of the question. It is argued here, however, that known patterns of behaviour can be simulated and corresponding plausible development of situations can be presented. The aim is not to predict how everyone will behave in every circumstance, but to be able to string together credible sequences of events based on knowledge of human behaviour. In addition to being credible, it is hoped that the knowledge that has been gleaned by psychologists, social scientists and so on can be conveyed at some level and in sufficient detail to enable users to gain a better understanding of certain types of scenarios and, perhaps, to enable them to consider alternative courses of action.

To be able to develop a working model, various simplifying assumptions have to be made. In this process, as in any modelling exercise, the aim is to ensure that the baby does not get thrown out with the bath water. The main model used as a basis is the one developed by Schank and Abelson, described in Section 6.4.1.

First it is assumed that a profile can be generated for each of the actors within a system. This will include information about relevant beliefs, feelings, and other aspects of character. From this information potential goals can be hypothesized. To achieve these goals specific plans are made. Lastly, the plans involve carrying out sequences of actions. Things do not always run so smoothly, of course. For example, other people's goals may clash with our own or the plans that we have may need to be changed due to

unforeseen circumstances. Although every eventuality cannot be covered this gives, at least, a framework for human behaviour. Now, consider each of these ideas in more detail.

Much of what is required in a person's profile is incorporated into Schank and Abelson's life themes. People's behaviour is influenced by their personal qualities, their ambitions, their life styles, what approval they seek and by their need for physical sensations. To this may be added their belief system. According to Bootzin and his co-authors a person's beliefs consist of 'knowledge structures about objects and events' (Bootzin *et al.*, 1986, p607). Along with these they include *motivations* which they define as the reasons for beliefs. Clearly, beliefs will be personal and may be totally at odds with those of the general population. Motivations, therefore, provide personal justification at some level.

Bootzin and his colleagues note that one cannot necessarily predict behaviour from beliefs since, for example, people may be perverse or there may be some unknown interplay between what they believe and what they want to achieve. However, for the most part, a good idea of how they are liable to behave can be predicted from their beliefs.

Assuming that a person's profile can be used to determine what they would like to do and be, goals can be generated for people, both long term and short term. These may change due to circumstances (for example, if an actor believes a goal is no longer realistic) or if the attitude of the actor changes. Goals may be unattainable, fanciful, limited or destructive depending upon the character of the actor.

Besides the individual goals of an actor, Schank and Abelson's expectancy rules give some guidelines for what they might try to achieve: for example, to show gratitude or to exact revenge. Expectancy rules provide the glue for fixing together the whole system. Goals may be split up into sub-goals, or *instrumental* goals for achieving the main one. For example, seeking revenge may involve the instrumental goal of bankrupting the unfortunate target of one's wrath.

Also, each actor may have different views on the best way of attaining their goals. The variation in methods of achieving goals may be due to different beliefs about the current situation, different capabilities of the actors, different moral codes, etc. Given a specific goal for a particular person in given circumstances it is possible to formulate plausible means by which the person may seek to achieve the goal. This leads to the idea of a *plan*.

A plan may be good, poor, watertight, flawed or just plain stupid. However, it is a means by which the formulator intends to aim for a particular end. It may deal with all kinds of eventualities with provision for modifications as circumstances change. On the other hand, it may fall down at the first hurdle and be abandoned. In any case it provides the guidelines for the actions that a person carries out. This corresponds to McClure's notion of an intention.

At the bottom level of this hierarchy is the action. An action is motivated by some aim. It may almost be a reflex response based on immediate circumstances or it may be part of a thoughtful strategy for achieving a particular goal. In Figure 6.7 the action of an actor was depicted as being triggered by their mental state. This mental state is presumed to incorporate long term as well as short term goals and plans.

At a high level, a scheme that culminates in action is needed. Such a scheme is illustrated in Figure 6.9. Profiles are assumed to consist of themes and beliefs. The types of goals, taken from Schank and Abelson's book (Schank and Abelson, 1977) are self-explanatory except for instrumental goals (intermediate goals) and delta goals (changing the current state). As with Figure 6.7 this diagram shows the general impact of one stage upon another and does not imply an absolute sequencing in particular circumstances. Obviously, the current situation may change what a person does, but each level is more resistant to change than the one to its right. Plans can be changed according to what happens, goals are less likely to be influenced by events and a person's profile is unlikely to be affected except in unusual circumstances ('seeing the light', disillusionment, etc).

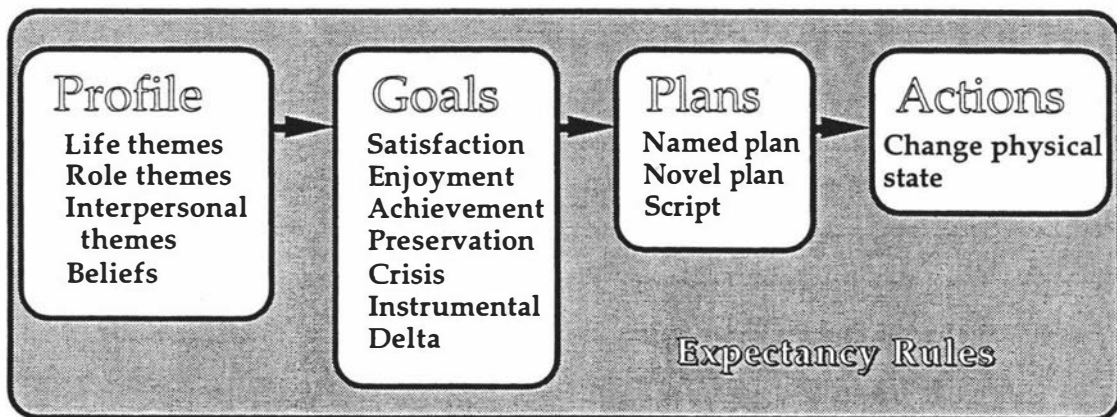


Figure 6.9 Profiles, goals, plans and actions

The models presented in Figures 6.7 and 6.9 contribute to the development of a feedback system. Figure 6.7 indicates the various actions that are enabled, based on the current mental state of the actors. So there may be several plausible alternatives that a person

may choose from (that is, enabled actions within the current state of things). Thus any one of a number of actions may be taken. If the user assumes the role of an actor then a choice of the enabled actions may be presented. Any of these can be justified via the preconditions for a particular action. The effects can be shown via postconditions.

The higher level description in Figure 6.9 can be used to show longer term and more considered reasons for behaviour. If it is required to show *why* a person may have carried out a particular action rather than any other feasible ones then this can be justified by the plan that this person has. The plan can be shown to have emerged from a general goal, and this goal can be justified in terms of the actor's beliefs and attitudes. As well as justifying what actors have done, this model enables the computer to suggest actions that might be carried out which are consistent with the user's goal (or any other goal suggested by the system), or an attempt can be made to infer information about users from their behaviour. Their plans might be deduced from their actions, their goals from their stated plans, and their attitudes from their stated goals. In this way, a meaningful dialogue can be built up and tailored to the needs of the user.

Not much has been said about specific circumstances but these obviously play some part in how someone behaves. To some extent a prediction of what a person will do can be based on the current situation (or more precisely upon their perception of the current situation) but this may be influenced by factors other than just their attitudes and beliefs. Stock situations can be represented by scripts and these can have a great deal of influence on how people react. In addition, the role that someone sees for themselves (Schank and Abelson's role theme) can have a significant impact. In a realistic model these have to be included. Care is needed when employing scripts, however, since, to an extent, they gloss over the details of why actions are carried out. A successful Script Applier Mechanism (SAM) was developed by Cullingford (1981), but, as noted by Dyer (1983), SAM cannot answer questions such as *why* someone did something.

Comparing the above model with that developed in Chapter 4 it is clear that it is going to be more difficult to implement. Any given situation is going to be more complex to analyse because of the requirement to consider individual attitudes, beliefs, reasons for behaviour and so on. It is also clear, however, that there are similarities that can be exploited. The causal semantics involving behaviour can be likened to the cause-effect model developed previously. Each situation that is considered will contain the extra component of mental states of the actors but, again, plausible reasons can be found for the choices made. This extra information about people's plans, goals and expectations would hopefully provide feedback to guide the user. The plausible actions that the user might take (in line with their profile and goals) could be considered to be the domain

constraints. Specific recommended courses of action could be provided by an overlay on this, and is analogous to the task specification developed in Chapter 4.

6.5 Summary

A scheme for modelling human behaviour and interaction within a framework of knowledge-based simulation has been developed. In such an ill-structured domain as simulation of human behaviour much simplification is required to enable any kind of computer model to be constructed. The aim has been to simplify the system without losing the main driving features of human interaction: attitudes, beliefs, goals and plans. A prototype system based on Lenat's scheme for event description has been presented. A more detailed model has also been described that is based on Schank and Abelson's research into human knowledge structures. It is proposed that such a model can be adapted for teaching purposes.

Using the attitudes, beliefs and goals of the actors to drive a system based on the above model has several advantages:

- a high level of transparency is maintained so the user can see what is happening;
- since the plot line is not hardwired in, there is flexibility in what can occur;
- the domain expert can be involved in the package development to a greater degree;
- the methods of Chapter 4 for highlighting routes in the system can be employed, enabling the teacher to focus on specific situations;
- feedback can be handled in a similar fashion to that described in Chapter 4.

Chapter 7

Towards a Knowledge-based Simulation Tutor

7.1 Introduction

In order to gain a fuller understanding of some of the issues involved in using knowledge-based simulation for modelling human behaviour, two projects have been undertaken, both of which are described in this chapter. First, the framework of a prototype system for helping police officers is covered. Secondly, a scheme that includes detail of behaviour simulation in domestic dispute scenarios is considered.

7.2 POPIT: applying knowledge-based simulation

POPIT (Kemp and Burns, 1992), which stands for Problem-Oriented Policing Intelligent Tutor, is a project with the aim of using the computer for developing the problem solving skills of both police recruits and more experienced officers. The intention is that the user should work through a computer simulation, based on experience of typical situations that an officer would have to face in the line of duty. The system would not only allow the user to see the impact of decisions via the simulation but would also include a coaching facility to provide additional feedback.

Two of the main problems that need to be considered are, first of all, to provide a convincing representation of scenarios and, secondly, to utilize this model within a teaching framework. It is suggested that these two problems are closely linked and that, by employing an appropriate model of the domain, the difficulties of appraising the decisions of the student and justifying alternative courses of action can be effectively addressed.

7.2.1 The domain

Most police work involves reacting to calls for service generated externally to the police organisation. Upon the receipt of a call, a decision is made to dispatch a police officer to the scene or to handle the call in some other manner. These calls can range from instances of serious crime that demand an emergency response, such as an armed robbery or an assault, to those of a less urgent nature such as a burglary, loss of property or auto theft where the situation can be handled over the telephone or via a delayed response. Regardless of the type of incident, the police process the call in the most efficient manner possible. That is, the officer responds to the event with the aim of handling the incident quickly and expediently: hurry to the scene, investigate the event, adopt a solution and return to service to await the next call. In most instances, only the most immediate and superficial symptoms of what might be a much deeper problem are addressed.

This approach to the problems confronting the police has been labelled 'incident-driven policing' (Eck and Spelman, 1987; Goldstein, 1990). Within this approach, each incident is treated as if it were isolated and disjoint from other similar ones. The exception to this is when the pattern of crimes is analysed for the purposes of identifying or apprehending an offender. While this approach addresses each incident efficiently, the outcomes are largely ineffective. That is, the symptoms are addressed but not the factors contributing to the problem. This means that the incident is likely to be handled over and over again. Figure 7.1 shows the general characteristics of incident-driven policing.

A new approach is clearly required. Problem-oriented policing is defined as a strategy directed at solving persistent problems that reside within the police sphere of operation. Thus, a problem must involve a group of incidents that share a common connection, be of concern to the public and be within the scope of police functions. Eck and Spelman characterize a problem as the main cause of 'a group of incidents occurring in a community, that are similar in one or more ways, and that are of concern to the police and the public' (Eck and Spelman, 1987, p42). The problem-oriented approach encourages the police to examine the relationships between incidents (patterns of behaviour, location), persons involved (victim, witnesses and perpetrators) and to analyse these factors in greater depth in the search for a more effective approach. The new approach is summarized in Figure 7.2.

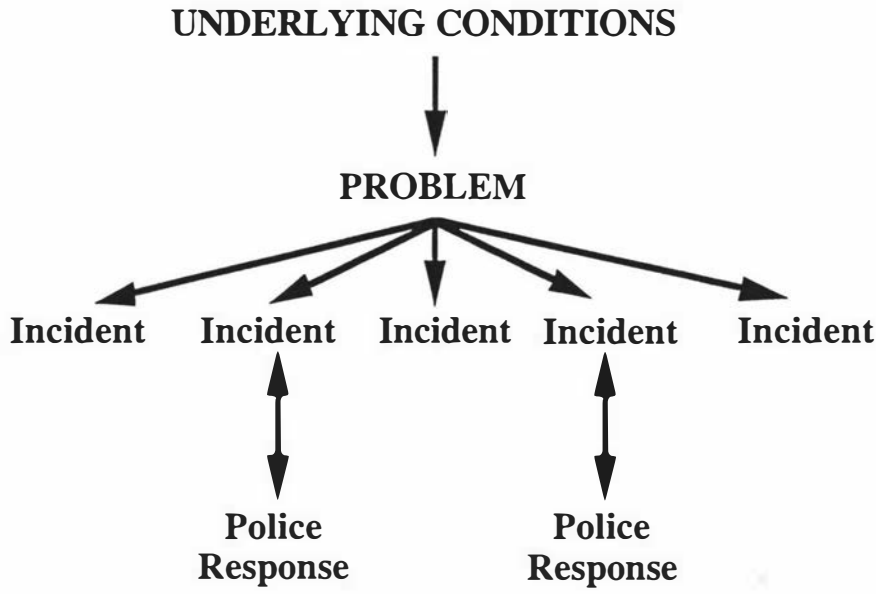


Figure 7.1 Incident-driven policing (Eck and Spelman, 1987, p4)

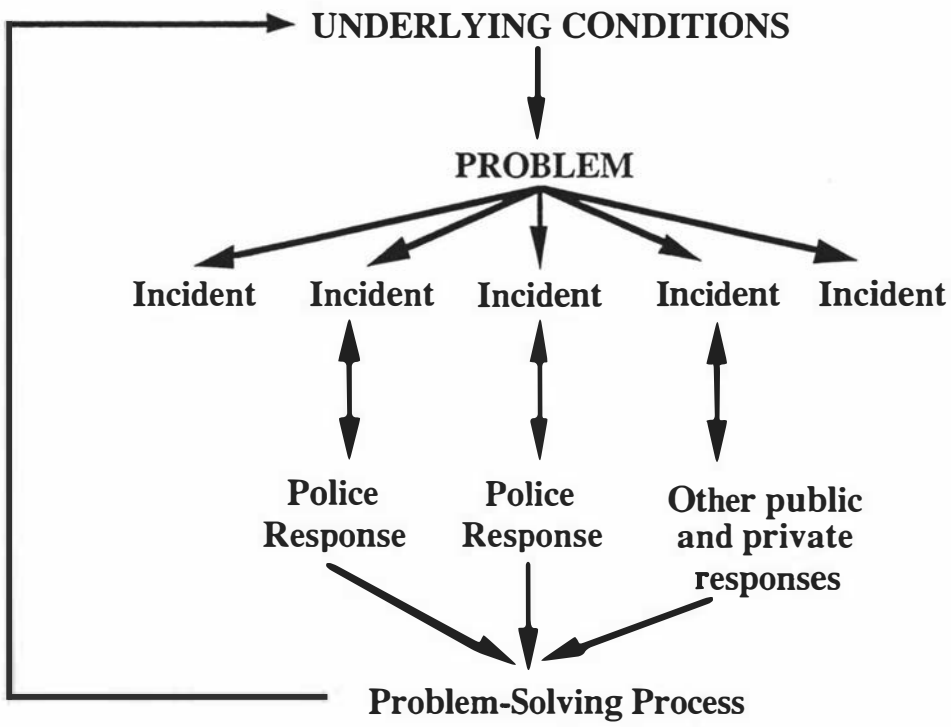


Figure 7.2 Problem-oriented policing (Eck and Spelman, 1987, p4)

Some problems can be eliminated, but it would be unrealistic to expect this as a final outcome in more than a very few instances. A few of them can be dealt with more satisfactorily by organizations other than the police, but simply transferring a problem to another agency is unacceptable unless they can deal with it more effectively. The

most realistic expectation of improved effectiveness will occur in the two areas of reducing the number and seriousness of the resultant incidents.

In New Zealand, problem-oriented policing is currently taught during training at the Royal New Zealand Police College, and there are also courses run for more experienced officers. In addition, there is a one-day practical exercise on applying the methods that is extremely labour intensive and also very intrusive in the community. For these reasons and others, a more effective method of consolidating the ideas presented during lectures is required.

7.2.2 Outline of system

POPIT aims to improve the teaching of problem-oriented policing concepts by simulating common situations on the computer, allowing the user to interact and obtain feedback on performance. The objective is to give police officers some appreciation of how they might re-consider their approach to problems and problem solving. They are encouraged to think more seriously about what information within a community is overlooked or under-used, what steps they could take to become more informed and what alternative actions could be taken to achieve various outcomes. Typical kinds of situation that provide useful training materials include domestic disputes, controlling rowdy teenagers in a shopping centre, investigating a stabbing at a local nightspot, dealing with a prostitute who has been beaten up or evicting street kids who are using an abandoned building as a drug dealing centre.

Obviously, producing software that can give a realistic impression by simulating such situations is no easy matter. However, the popularity of adventure-type games demonstrates the interest in this kind of package and also indicates that such an approach is feasible. Malone (1981) was one of the first to point out the motivational effect that computer games can have on student learning. More recently, sophisticated systems such as GEO (Duchastel, 1989) have incorporated games into an intelligent tutoring system structure. If the student is to get meaningful feedback the operations that the user can perform are limited and have to be anticipated at some level by the computer but, even so, users can get a sense of being involved in a realistic situation and of making decisions that have plausible consequences.

Simulation, too, has been shown to have a stimulating effect on students: see Woolf *et al.* (1987) and Spensley *et al.* (1990b), for example. The question arises, to what extent can one simulate reality without straining the student's credulity? Obviously, one needs to anticipate to some degree what the user may do but still give him/her the feeling of

having unconstrained decisions to make. This is a difficult problem that will not be totally solved in the near future, but the success of adventure games and, in a more academic environment, Bonar and Cunningham's innovative Bridge system (Bonar and Cunningham, 1988), demonstrates that carefully selected scenarios can be analysed in sufficient detail to provide a semblance of reality.

In simulating the activity of a police department and the way that it deals with problems, one is in a complex problem solving environment with its own underlying logic, rules and guidelines. At one level a problem is solved by either making an arrest, administering a warning or by claiming there was no problem in the first place. A decision has to be made at this level (which, of course, may be affected by other actions the officer takes) and which has further ramifications. At a higher level, the officer is encouraged to look for patterns of offences: either people involved, places where they occur or similarities of types of offence. Once these patterns are established, more effective methods of resolving problems can be considered.

There is a temptation to set up a simulation with which the user can experiment and use this as it stands as the teaching system. Such an approach, at least in isolation, is very limited because:

the user may not realize what alternative and more appropriate courses of action were available;

the user may not realize that a particular outcome is unsatisfactory;

even if it is obvious that something has gone wrong the user may not believe that s/he could have taken any action to improve the situation;

it will often not be clear at what point things started to go wrong.

For the above reasons, it was decided to incorporate a coach/adviser into POPIT that interacts with the user as appropriate. Figure 7.3 illustrates the general passage of information through the system during a consultation.

The user interacts with the system via the interface. S/he has access to scenario events and can also influence events in the scenario. The adviser supplies advice to the user based on information that has been derived from the scenario and from the student model. Note that the adviser picks up the interaction between the scenario and the user via the scenario events record. Also, the student model may be updated from sessional information but this does not take place during the session itself. The scheme described

in Section 6.3 is used for scenario representation. Other aspects of this model (the role of the adviser, the teaching knowledge required, student modelling and the interface) relate to the learning environment rather than the domain model and will be considered in the following sections.

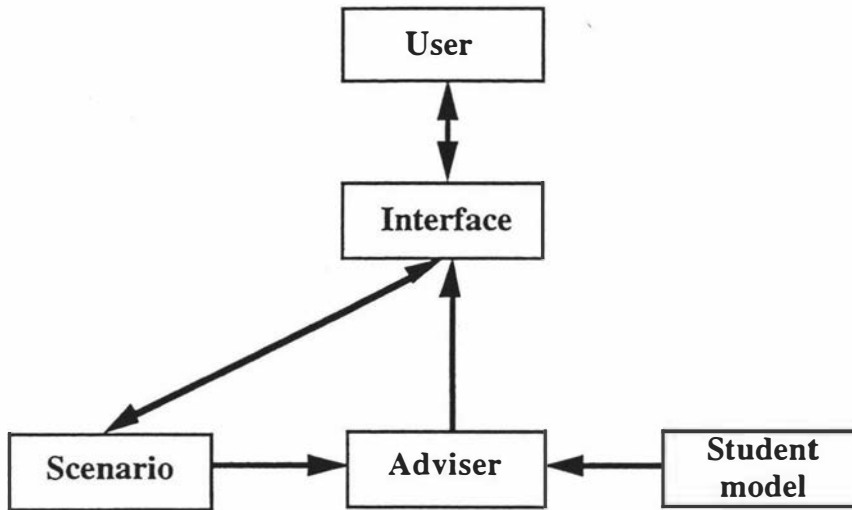


Figure 7.3 Passage of information through POPIT

7.2.3 The adviser

The adviser/coach is at the heart of the system. It must monitor the progress of the student, appraising the situation, deciding whether to intervene and, if so, in what way. The starting point here was the seminal work by Burton and Brown (1982). A general view of the information that the adviser in POPIT uses in order to determine its strategy is given in Figure 7.4.

This model suggests that we have an adviser that takes into account:

the domain model: a general outline of a type of situation.

the current scenario: the current instantiation of the situation. This includes everything that has happened in the system, including the advice that the user has been given so far.

agent models: the main model is the student who is interacting with the system, but models of other agents are also relevant, both for comparison with the user, and also to present him/her with alternative points of view: see, for example, Todd (1988).

teaching knowledge: this includes information relating to common types of errors, misconceptions and how to deal with them, the facts that need to be conveyed and the problem solving approach that is to be instilled.

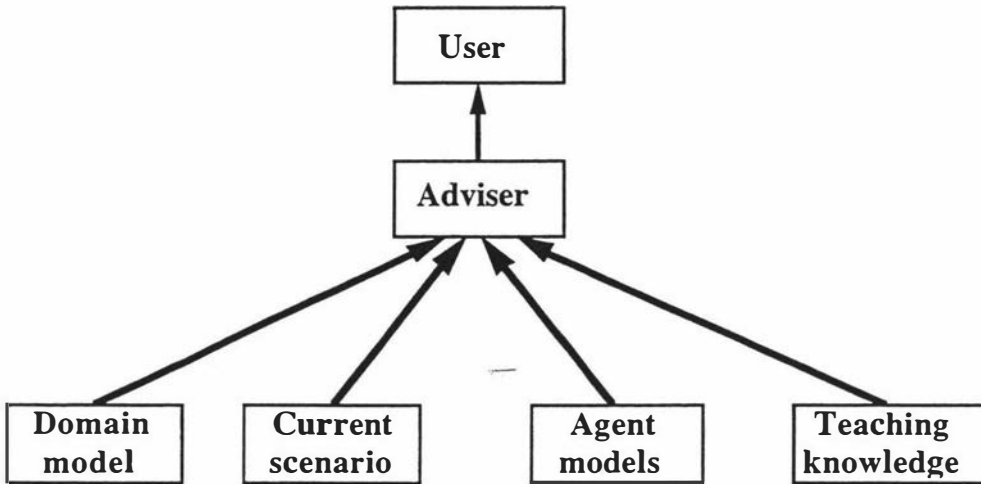


Figure 7.4 Information used by adviser

From the above sources, the adviser is able to access all the information needed to formulate advice. For example, it is possible to anticipate the user's view and likely actions in any given situation. Partly because of the nature of the domain, the coach is not regarded as an infallible agent. Also, in common with Cumming and Self (1989), it is believed that the student will learn more effectively if s/he has to weigh up the significance and even the veracity of the advice given and use this in conjunction with his/her own judgment.

7.2.4 The student model

Kass (1989) takes a qualitative approach to student modelling and this is a view that is in keeping with the overall scheme proposed here.

What kind of information about the user is needed? Anything that aids the understanding of his/her actions, and the generation of an appropriate response is relevant. As noted in Chapter 2, this may be significantly less than a full model of the subject's personality and knowledge.

A student model is needed to be able to predict or interpret the student's behaviour. This may affect the information that is given to him/her (for example, what scenarios are supplied or how they are presented). Secondly, if the user's performance is to be improved the system needs to know strengths and weaknesses.

With regard to this project, parallel work on the relationship between human factors and the interface (van der Veer *et al.*, 1985; Benyon and Murray, 1988) has been found to be of particular relevance. The factors that may be taken into account include: personality (likely to influence how s/he deals with a situation), intellectual capabilities (limiting his/her understanding of a situation), impulsivity/reflexivity (the degree to which the user considers actions before carrying out tasks) and knowledge of what may be done.

Each of the above factors is relevant to the way in which the subject behaves in particular situations and how the coach should approach the task of giving feedback. It is important to consider how resistant to change each of them are, so that it may be decided which ones, realistically, can be modified. Obviously, personality and intellectual capabilities are difficult to adjust but other factors, such as knowledge and problem solving ability, are open to modification.

Various sources of information about the student may be tapped to formulate the student model. One of the trickiest problems is getting a personality profile of the user. This could be obtained by questioning the student him/herself although self-assessment is notoriously unreliable. Bias is also present when taking information from colleagues of the user (from a supervisor for instance). A much more productive source of information would appear to be a psychological test. Again, this is reliant upon the judgment of the student although a good test will have questions that inter-relate to check for consistency and which discourage the user from presenting a false picture. Further information can be gleaned during the sessions themselves: the reactions to situations, and the reasoning behind decisions. Lastly, stereotypes can often provide default information where explicit facts are not known: see, for example, Rich's work (Rich, 1979; Rich, 1989).

7.2.5 The interface

A suitable interface is crucial to the success of any system and should be developed in parallel with the other modules: see, for example, Kemp and Boorman (1987). This is particularly true in an application where so much depends upon an effective interaction between the user and the system. It is proposed here that a suitable presentation of scenarios and options can be simulated by using a similar approach to that of pictorial adventure games, and a prototype has been developed using Hypercard (Carter, 1993). Many of the design decisions taken so far, concerning the coach and the student model, for example, have an implicit inter-dependence with the form of the interface. The final shape that this will take will be the goal of another project.

One way in which different interfaces can be tested out in parallel with the development of the rest of the system is by using the so-called *Wizard of Oz* approach. As will be shown in Section 7.4, this scheme not only allows interface design to be carried out early enough to impact the rest of the system, but also allows the feasibility of the scheme to be checked before a full implementation is initiated.

7.3 Domestic disputes

Although the scheme outlined in the previous section gives a framework in which a knowledge-based simulation system could be developed there is no feature that allows agent interaction to be simulated at a detailed level. An enhanced scheme was presented in Section 6.4.2 for modelling human behaviour within a teaching environment. That model is now applied in the domain of domestic violence.

The subject of domestic violence is one that has over many years received a great deal of attention from sociologists, psychologists, criminologists and a large number of other groups. Often, domestic violence is only detected or dealt with in any concerted fashion when someone is badly injured or perhaps even killed. Numerous case studies have been compiled and considerable analysis has been carried out. Although there are many variations on the basic theme some basic patterns of behaviour can be discerned and there is some general agreement about contributory causes, effects and methods of averting some of the worst excesses.

There is also general agreement that domestic violence is rife and that, ultimately only a change in public attitudes will give an overall improvement. As in other ill-structured domains there are not necessarily hard and fast rules about how to deal with every situation and what is fact and what is fiction. Even so, if people understand some of the circumstances then there is more chance of making progress. For example, it is widely believed in the police force that battered wives deserve what they get, and are free to leave the marital home (D. Burns, personal communication). This may significantly influence the way the police act when called to a domestic dispute. Other aspects of police behaviour and also the attitudes of others likely to be involved in such disputes (neighbours, social workers, friends, judiciary and so on) need examining.

One of the fundamental tenets of this thesis is that people learn most effectively by doing. They can read books, listen to lectures, watch videos, etc but these are essentially passive activities that may not have the desired impact. If users can be involved in an activity that closely relates to one they are attempting to understand, it is believed that their knowledge and understanding can be greatly increased. By

confronting users of a system with a possible conflict to be resolved they are encouraged to think and to reassess their beliefs and corresponding behaviour.

Who will the users be? In fact, there are a number of groups who could benefit from interacting with such a system including sociology students, social workers, police recruits, batterers, battered wives, and any other member of the public who wants to learn about the issues and situations that might arise.

Obviously, a great deal of research is needed to determine what the major issues and views are in this particular subject area. A technical report (Kemp and Carter, 1995) contains the results of a study of a large number of books and papers on domestic violence, information from interviews of over 100 people with knowledge of the problem (academics, victims, social workers, police officers, the judiciary, etc), and material from a conference on domestic violence. From this report, a detailed analysis of scenarios, actors and potential feedback has been produced. An outline of the important points is given in this section. A detailed model is presented in Appendix C.

Before a start can be made on analysing goals, plans and actions it is necessary to produce profiles of people involved in these kinds of scenarios. Schank and Abelson's life themes (Schank and Abelson, 1977) give an overall picture of people's nature and motivations. As mentioned earlier, they distinguish exactly six components of a life theme: personal qualities, ambition, lifestyle, political attitude, approval and physical sensations. Following their model, typical life themes for three potential actors in family dispute scenarios have been produced (see Figure 7.5).

Each life theme element will, in general, generate a complete package of goals. Each goal leads to plans for achieving the goal and each plan has a sequence of actions that will be activated in specific circumstances. Some of the goals, plans and actions corresponding to specific life theme elements are shown in Figure 7.6.

General beliefs of the actors may incorporate a great deal of information and will significantly influence their behaviour. In order to keep this information overload under control, a decision can be made to consider only those beliefs that are relevant to the scenarios to be represented. Relevant views of the batterer may be that 'a woman is a man's property' and that 'a man handles all the money in a household'. These beliefs may have been developed because of being socialized in a patriarchal system. As before they can be used to influence goals, plans and actions. Some examples of goals generated from beliefs are given in Figure 7.7.

	batterer	battered wife	police officer
personal quality	moody, insecure, aggressive, suspicious, passive, self-pitying, possessive nature	voluble, timid, withdrawn, friendly, submissive, low self-esteem, self-effacing, excitable, self-critical, scared, complaining, critical	authoritarian, inflexible, placatory, male-oriented, self-reliant
ambition	to feel safe and secure, to feel in control	to raise family, to keep family together, to have peaceful life, to feel supported	to get promotion, to be successful police officer
lifestyle	self-gratification, pleasure from indulgence, patriarchal, traditional view of sex roles	isolation	fitness enthusiast Spartan
political attitude			right wing
approval	to be respected (or feared) by friends and wife	to be loved by husband and children, to be respected by outsiders	of peers, of superior officers
physical sensations	getting drunk		

Figure 7.5 Life themes of actors

actor	life theme component	general goal	plans	actions
batterer	possessive nature	ensure no one else gets close to spouse	sever links between spouse and friends	stop spouse going out, stop friends coming to house, threaten friends of spouse
battered wife	ambition to have peaceful life	to keep husband happy	not go against husband's will	follow husband's instructions
police officer	placatory	avoid arresting people unless necessary	calm down domestic situation without provoking husband	sympathise with husband when criticizes wife, persuade wife not to make complaint

Figure 7.6 Actors' themes, goals, plans and actions

actor	belief	reason for belief	goal	actions
batterer	inculcating fear is best way of ensuring obedience	patriarchal upbringing	make wife fearful	threaten wife, hit wife
battered wife	punishment is due to own inadequacies	attitudes of others	to correct deficiencies in character	to get meals on time, to avoid annoying husband
police officer	arresting batterer has no effect	own experience, lack of support from superiors and courts	avoid arresting batterer	try to get protagonists to make peace
family court judge	a man should be in charge in a marriage	religious convictions	women who challenge domination of husband may be punished by husband	be lenient with husband who punishes wife in 'justifiable' circumstances

Figure 7.7 Beliefs, goals and actions for actors

There are other factors that influence a person's behaviour in addition to their beliefs and life themes. For example, in any given situation, one's attitude to other people present may determine how one behaves. Interpersonal themes supply this kind of information.

So far, it has been assumed that people's behaviour will be governed by their characters, attitudes and feelings towards others. In certain circumstances, personal attitudes and beliefs may be suppressed, or at least curbed. Much of the behaviour of officials such as police officers, social workers and the judiciary is determined by their job and, at least in theory, personal feelings do not come into play. To represent this feature Schank and Abelson's role theme may be used. For example, a police officer's goals may be specified as keeping the peace, arresting anyone who is suspected of committing a crime, etc. In order to achieve these aims the police officer may make various plans, and may go through various procedures. The procedures may be represented by scripts such as the *arresting a suspect* script. In fact, scripts are crucial to the understanding of many situations since people often behave in predictable ways in set circumstances.

The notion of scripts can be employed in a simulation of domestic disputes. Not only are there specific scenarios but also general ones in which specific scripts are contained. This is in the style of events/sub-events discussed in Section 6.3. A general

scenario showing some of the common stages in a violent relationship has been proposed by Giles-Sims (1983). These are:

1. Establishment of relationship;
2. First violent incident;
3. Stabilization of violence;
4. Crisis;
5. Leaving the system;
6. Resolution.

Each of these stages is considered in more detail in Appendix C. The information provided by Giles-Sims has been modified to fit in with our events approach. Thinking in terms of a computer simulation it would be counter-productive to follow through the complete relationship of a couple with all the constituent events, since this would be long, complex and confusing. However, some appreciation of what is happening in each of the phases of the relationship is important. The approach taken here is one often adopted by operatic composers: there is a complete story to tell but most of what happens occurs between acts. The librettist just concentrates on a few key scenes that delineate important relationships and pinpoint significant changes in them. Similarly, presenting a meaningful incident from each stage of a domestic relationship would be sufficient to show general trends.

Consider an example of such an incident (taken from stage 2 of the relationship). It is a hypothetical situation where the couple are married but things are just starting to go wrong. The wife is asked out by a friend and has to decide what to do. Among the alternatives are agreeing to go, asking the husband and refusing to go. Following on from this there will be further choices. So, for example, if the wife asks the husband and he refuses then how will she proceed?

We can envisage a number of scenes (sub-events) illustrating how the situation develops. Figures 7.8 to 7.12 give a summary of actions and reasons behind them (the incidents are analysed in detail in Appendix C). It may be seen, as with the procedural domain, that there is a sequence of events with causes and effects, and the user can be kept informed of why something occurred and why certain events may be infeasible within the domain, or inadvisable (relative to a specific goal).

Note that there are a number of issues highlighted by these scenes. These include the desire for control by the male, his recourse to violence, the conciliatory nature of the female, and the importance of maintaining contact with the outside world since isolation obstructs any feedback on the acceptability of the situation.

The user of a computer system based on these ideas could observe the interaction of the players or take one of the parts and be involved in the scenes. In either case, the generated actions could be explained to the user in terms of the driving factors: attitudes, beliefs and goals. As with the procedural example, we can explain why a particular action occurred, give a list of feasible next actions by each of the participants or explain why a particular action is infeasible.

It should be noted we have just considered reasonable alternatives based on the natures, goals and beliefs of the people involved. If it is required to demonstrate particular points then a projection graph of actions leading to a specific end could be overlaid on feasible events and the user could be shown what happens in particular circumstances and why. The 'acceptable' routes through the system do not have to be ones that are seen as the most desirable in practice: they may just illustrate particular points. For example, paths could be limited to those which lead to a violent episode, in order to illustrate the way one might occur. Alternatively, to demonstrate how violence can be avoided, other routes and actions could be highlighted. A third possibility is to examine the effects upon the outcome of particular approaches by the participants. The impact of a conciliatory attitude by the wife could be illustrated, for example.

Action	Reason/Source
phone call from old girl friends asking wife out for evening	external event
wife agrees in principle	she is sociable and enjoys company of friends
but says she will ask husband first	believes husband has final say on wife's movements

Figure 7.8 Wife battering event: wife asked out

Action	Reason/Source
husband says she cannot go out	jealous, possessive, suspicious; weak nature pre-disposes him to attempt to maintain control
husband demands to look in wife's handbag	suspicious, gesture to demean wife and assert authority
she refuses	pride, self-respect
he grabs handbag anyway and looks through it	he uses physical superiority
argument ensues	both parties distraught
both retreat in bad mood	both have tendency to sulk
he apologises and says he gets jealous because he loves her	he is prepared to concede points when he feels there is no pressure to do so
she accepts apology	she has need to be loved
he still maintains that she should stay in since he needs company	attempts to get own way by evoking sympathy
she accepts this argument	sympathetic nature

Figure 7.9 Wife battering event: asks husband if she can go

Action	Reason/Source
he goes out on night in question anyway since it is friend's birthday	external event
she goes out, thinking she will be back before him and he will not find out	desire to socialize, hoping to avoid problem

Figure 7.10 Wife battering event: evening of social event

Action	Reason/Source
she returns home to find husband already back	external event
he is angry	believes his authority has been undermined
she is defensive and apologizes	believes she is in wrong for disobeying husband
apology angers husband more	further consolidates his belief that he has been badly treated
he smashes her favourite vase	he seeks to punish by physical violence, initially directed towards objects
she abuses him verbally	physically weak and so retaliates verbally
he gets more angry	she has undermined his fragile self-image
he slaps her	frustration, need to break tension, feeling of having no alternative since cannot back down, escalates level of violence, belief that he has the right to be violent.
wife withdraws in distress	physically and mentally hurt, has no other recourse
husband becomes melancholic	gratifying his self-pitying nature

Figure 7.11 Wife battering event: return from evening out

Action	Reason/Source
wife still distressed	situation unchanged
husband brings flowers and apologises	retains control by conceding on his terms
she accepts apology	need to be loved and desire to resolve situation in least stressful way

Figure 7.12 Wife battering event: the evening after

7.4 Wizard of Oz simulation

Although the problem of incorporating knowledge of human behaviour for particular scenarios has been addressed, there is much more to be done before a realistic role playing program based on such methods can be written. In particular, much real world knowledge that we take for granted would need to be painstakingly input into the computer. For example, the implications of the calling of the police, the authority that they exert, their power, and the repercussions of defying them, would need to be included. Legal issues, issues of ownership of property, family relationships, are all relevant. Even observations that humans would take for granted (such as the impossibility of hitting someone else if they are in another room with the door closed) need to be considered if the situations and events are to appear authentic.

Each of these points can be considered separately but there are literally millions of such trivial observations that would be needed in a flexible scheme. Doug Lenat and his colleagues (Lenat and Guha, 1990) have spent ten years building up a knowledge base of such facts about the world.

A further problem is that natural language dialogue might be considered desirable in such a system. However, many of the problems of natural language interaction still have to be solved.

It is argued here that such obstacles should not deter research into the use of role playing in computer teaching systems. It would be a mistake to wait for all the other problems to be solved, since there is no reason why research cannot be carried out in parallel. The question then arises, how can one test the kind of scheme proposed here?

The problem has been addressed by using a 'Wizard of Oz' method (Chin, 1984; McKevitt, 1990; Maulsby *et al.*, 1993). This method involves users interacting with what they believe is a program to carry out a specific task but, in actual fact, no such program exists. Instead there is a person (the Wizard) at another terminal who is providing the responses to the users' input. The term 'Wizard of Oz' derives from the book by Baum (1969) in which the main character, Dorothy, is confronted by a frightening monster. However, this is a hoax since a little old man (the Wizard) is working the levers (see Figure 7.13).

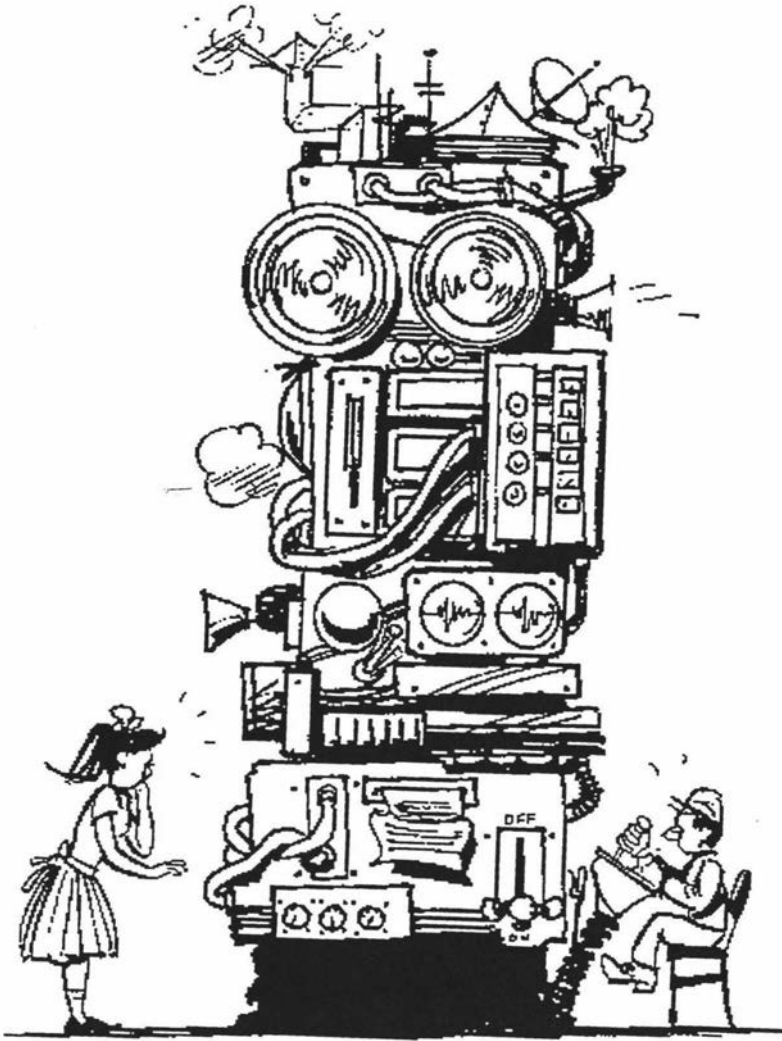


Figure 7.13 Wizard of Oz scheme

In the computer version, the Wizard responds in the same way that the program being simulated would do if operating correctly. The method has sometimes been used for evaluating interfaces. Here, it is applied to a complete system, to assess its feasibility and to detect any major flaws before a program is written.

Before such a scheme can be implemented, a number of obstacles have to be overcome. First, a method of communicating between terminals is needed. Secondly, if the user is to believe in the program, it must operate at a reasonable speed. This means, in practice, that much of the text to be presented to the user must be prepared beforehand. The Wizard should have a thorough knowledge of the domain and the scheme as proposed, and how it would operate. S/he also has to deal with any unexpected situations and dynamically generate responses to deal with these. Lastly, a complete record of the session needs to be kept for later analysis.

The Macintosh shareware package called Talk[®] (Lewis, 1993) has proved adequate for initial experiments. It allows text to be transmitted between Macintosh terminals using a split screen. Each person sends information via one of the window panes and receives responses in the other one. Crucially, it also allows text to be copied from a word processing package (in this case Microsoft[®] Word), pasted into the sender's window pane and delivered to the receiver's window pane. Figure 7.14 shows a sample screen during a dialogue as seen by the subject. The top pane contains the information presented by the Wizard and the bottom one shows the responses by the subject.

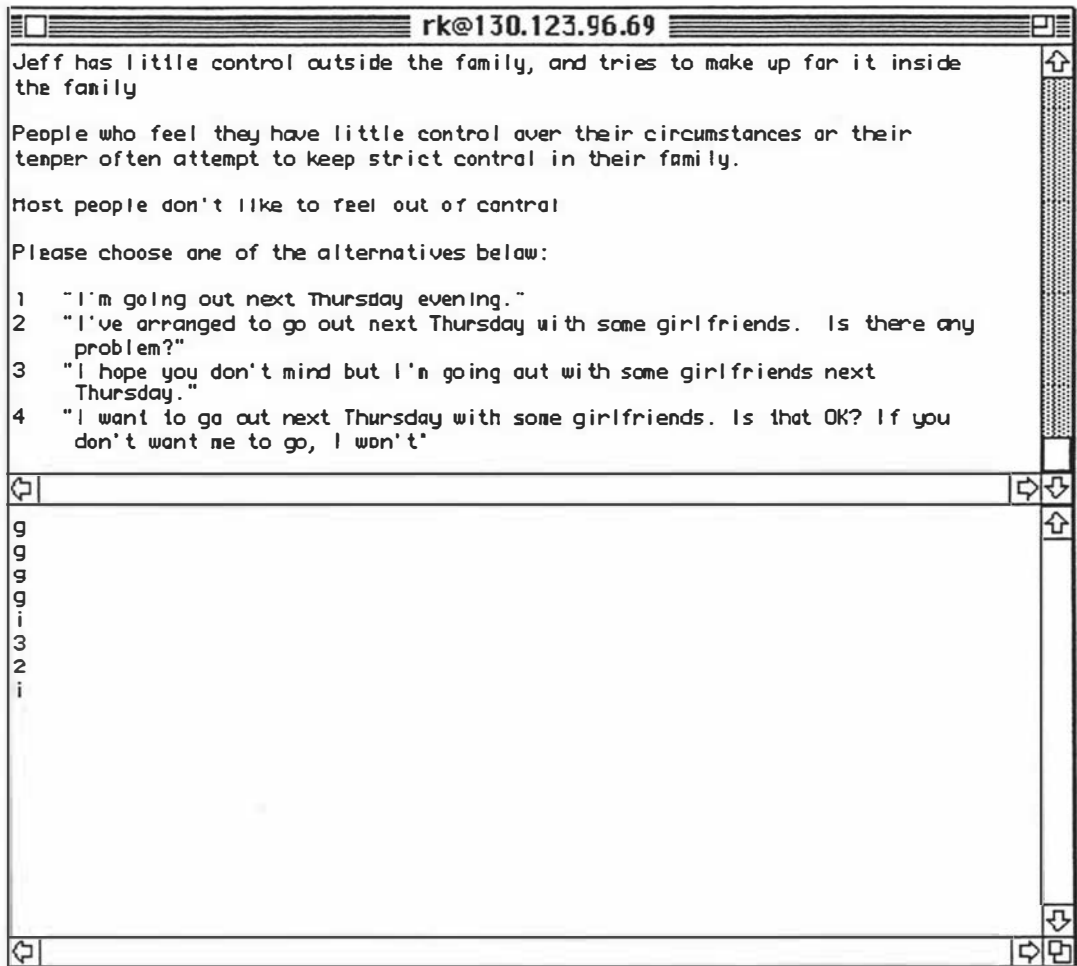


Figure 7.14 Screen from Wizard of Oz session

The Wizard of Oz method was used for the domestic dispute scenario with the author as the Wizard. The analysis developed in Section 7.3 and Appendix C was used as a foundation for driving the simulation. The likely behaviour of the actors was anticipated by examining their profiles, beliefs, goals, plans and actions within a specific framework.

Blocks of text were prepared beforehand and used as a basis for computer responses, but it was found that often they needed to be modified to deal with the specific actions

of the user, or, in some cases, new pieces of text had to be typed in and transmitted. Although the framework was found to be adequate, small but significant details had to be changed in each experiment to maintain the plausibility of the situations and occurrences. It is maintaining the correctness of these details that remains one of the main obstacles to producing a completely automated scenario generation scheme.

Details of the experiment set-up and results are given in Appendix D. As an initial feasibility study the system was tested out with 14 subjects, one at a time. They were each told that they were participating in a role playing experiment in which they were to play the part of a female who is having problems in her relationship with her husband. A sample page from a dialogue is given in Figure 7.15 (as usual, user input is in bold). A full session is shown in Section D.3.

Afterwards, subjects were given a short questionnaire to assess their reactions. The format and questions were adapted from the 'Software Usability Measurement Inventory' produced by the Human Factors Research Group at University College, Cork (HFRG, 1993). A summary of the results is given in Figure 7.16. As may be observed, subjects were generally very positive, which indicates that the basic approach is viable and that a fully implemented system would be of considerable interest and usefulness.

7.5 Summary

The knowledge-based simulation approach to modelling in ill-structured domains has been tested. The concepts introduced in Chapter 6 have been applied to specific situations. First, a framework for a police officer training system, which employs Lenat's method of event representation, has been outlined. Secondly, the scheme for behaviour representation described in Section 6.4.2 has been applied to the problem of implementing domestic dispute role playing scenarios.

It should be noted that both schemes are presented at a high level, and that suitable primitives could be incorporated into a design model. For example, the tables shown in Figures 7.5 to 7.7 and in Appendix C could be made the basis of input to an interpreter for scenarios. The table entries are in a form that domain experts could readily understand and that they could help to construct. Also, the transparent nature of the representation and its causal basis allows feedback to be given to the user. For example, explanations of actions, and goals and beliefs of protagonists can be readily provided.

Since, with the current state of associated technology, a full implementation for role playing seems infeasible, it was decided to use a Wizard of Oz method to simulate the operation of a program. The results from experiments indicate that the basic scheme is plausible and that subjects would benefit from using a package based on this approach.

You go out for the evening and have a good time, so good a time, in fact, that you don't arrive home until 2 a.m. You hope Jeff will have gone to bed but he is waiting for you. He's had a few drinks and is really mad. He wants to know exactly where you've been, who with, and what you've been doing.

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

i

He's not likely to believe you whatever you say
His pride has been hurt that you have disobeyed him.
He wants to punish you.

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

r

Please enter your comments, terminated by 'return'

I want to take an option between the first and second. I am aware of his need to control me and therefore want to passify him by letting him know that I have done nothing wrong rather than exacerbating the situation. But at the same time I want to retain my dignity and some control over the situation. I want to tell him that he has no right to demand to know my every move. We are equals in this relationship and as such should be treated as equals. At the moment he is treating me like a child. I do not accept that.

your remarks have been noted. please continue with the session

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

1

He gets madder and starts throwing things around the room, accusing you of being a whore.

- 1 You get out of the room and barricade yourself in the bedroom.
- 2 You try to appease him by telling him you won't do it again.
- 3 You tell him to grow up and stop behaving like a little boy.

i

Sorry. I don't know any more about this subject.

Figure 7.15 Sample dialogue from a role playing session

	disagree		
	undecided		
	agree		
	↓	↓	↓
	%	%	%
1. The system is easy to interact with.	100	0	0
2. It is easy to understand what to do.	86	14	0
3. The depiction of situations is realistic.	86	14	0
4. The computer responses are realistic.	79	21	0
5. The 'information' option is useful.	86	14	0
6. Using the system increased my understanding of domestic disputes.	86	14	0

Figure 7.16 Questionnaire results summary

The Wizard of Oz approach, itself, worked well and has obvious potential for prototyping schemes of various kinds: particularly those where a human agent is to be simulated on the computer, or where human knowledge is being modelled. For an experiment to be feasible, the Wizard would have to have sufficient knowledge of the subject area to be able to answer unexpected questions. If the program was an expert system, for example, then the Wizard should be a domain expert, and would also need to be able to keep track of the progress of the session.

Although the experiment was successful in demonstrating the basic feasibility of the approach and the potential educational benefits to users, it is instructive to consider what further work would need to be done to produce a fully-implemented system. First, data structures would need to be constructed to store information about the actors (for example, the kind of information presented in figures 7.5 to 7.7). This would be included within the agent models of Figure 7.4. The domain model would store static information about the situation as indicated in Figure 6.4, and the current scenario would keep track of the changes that occur during the running of the system. Like the Wizard, the interpreter would have to keep track of all this information in the light of actions selected by the user, but would also need to have access to a substantial knowledge base of real world facts for maintaining consistency.

Chapter 8

Conclusions and Further Work

8.1 Introduction

In this chapter, the research that has been carried out is summarized, its original contribution is assessed, and future lines of investigation are proposed.

8.2 Summary of research

A central goal of this project has been to examine ways in which the production of computer learning environments may be facilitated. It was envisaged that ideas from knowledge-based systems might provide the key to achieving this goal. As a prelude to the investigation, research into the role of knowledge in learning and how knowledge can be represented on the computer has been reviewed.

As a result of this review it was determined that faithful models of reality are highly desirable for producing environments from which the user could learn. More specifically, designers should aim for cognitive fidelity: taking into account not only the ability of the users to understand how the system operates, but trying to model closely their problem solving behaviour. Providing such a model simplifies the tasks of adapting the software to different requirements and including suitable feedback.

A further important consideration is that methods used for designing such systems should be accessible to non-computer specialists. Teaching and domain experts should be closely involved in the development of software and, to achieve this, it is desirable to have schemes that can be understood by them. At the same time, a degree of rigour is essential to remove ambiguity and to allow programs to be generated as directly as possible from the specifications.

Based on these ideas, a framework was outlined for using high-level primitives as building blocks for the development of interactive learning environments. The use of such primitives attempts to bridge the gap between the teacher and domain expert on one hand and the system developer on the other.

To determine whether such a framework was feasible, suitable primitives needed to be investigated. It was clear from the initial analysis that there would be no single tool that could be used in every kind of teaching situation, since domains and domain types vary greatly. The strategy was, therefore, adopted of considering two very different kinds of domains and determining appropriate techniques for system development for each.

8.2.1 Procedural tutors

Firstly, procedural domains were considered. In this kind of domain, the aim is to model the behaviour of devices and how people interact with them. Here, the user needs to acquire an understanding of how a device works and also how to carry out various tasks within the system.

It is important in domains of this type for users to be able to try different alternatives and see the impact. Obviously, the model must operate, as closely as possible, in the same way as the real device. Also, if something unexpected happens, the system should be able to help the user find out why.

In addition to exploration, such models can be used for setting tasks for learners. A goal may be achieved by trial and error but research suggests that constructive guidance can improve students' understanding of tasks and speed up their learning. Guidance is particularly beneficial if it includes feedback when users have gone wrong, pointing out their mistake and suggesting alternative steps that might be taken to rectify matters.

To be able to simulate the effect of users' actions in the domain, a state-space model was adopted. The use of transition nets was investigated for domain and task representation but was found to be inefficient and opaque. Alternative approaches based on planning formalisms used in artificial intelligence proved to be more efficient. Also, the suitability of the planning methods for denoting cause-effect relationships makes them much better for providing explanations and general guidance.

TWEAK, a Strips-style planner, provided some improvement over the transition net, but the most effective approach was found to be Tenenberg's more elaborate Strips

planning model. This separates out the static conditions of the system from the dynamic ones. Such a formalism adds both to efficiency (since fewer dynamic variables have to be manipulated) and to clarity (since we distinguish what is susceptible to change from what is not). Tenenberg's formulation has been shown to be theoretically sound and so there are no problems about possible inconsistencies arising.

These methods were adapted in order to design learning environments which could provide feedback. Programs written to simulate several systems have demonstrated the feasibility of these techniques. Systems described in the text have included VCR simulators and models for assembling and dismantling a gearing mechanism.

The planning methods investigated were found particularly suitable for separating out the domain and task considerations. Both for system development and for teaching purposes it is seen as desirable to be able to do this. Steps in each individual task can be represented as an overlay on the domain description. To specify these steps some further notation was needed. A suitable scheme, originally used by Mark Drummond as a controller for automated planning systems, was adapted for task description purposes. His concept of situated control rules allows guidance to be given to students at appropriate points in their attempts to achieve specific goals. Programs based on the modified Drummond scheme demonstrate the feasibility and appropriateness of the method both for the VCR and gear mechanism problems.

Although the primitives used in procedural domains were found to be appropriate for program development, they are not particularly suitable for high-level design or for communication to non-computing collaborators. A more transparent notation for communicating domain and task descriptions was considered desirable.

A pictorial notation called the plan net, based on a variant of the petri-net, is proposed for domain description. Using plan nets, cause-effect flow can be represented in a direct fashion. It is suitable for designing systems on paper, or straight onto the computer. The latter is possible since plan nets can be easily converted to the action tables used in Strips-like systems.

A design scheme that incorporates all of the above ideas has been proposed and has been elaborated in Figure 5.22. Complete programs that follow this scheme have not been written but examples of action tables, pictorial design elements and programs for all of the key components of the system have been produced.

8.2.2 Knowledge-based simulation in ill-structured domains

As a contrast to the procedural-style tutors, an examination was undertaken of ill-structured problems, ones where solutions are not clear-cut. Modelling of situations involving people was seen to be of particular concern. It is perceived that dealing with such domains is of increasing theoretical and practical significance.

Analysis has indicated that knowledge-based simulation seems to provide a suitable framework in this case. As before, it is important to attain a level of cognitive fidelity. In order to do this, it was proposed that more than a superficial knowledge of human behaviour patterns would need to be included.

As with procedural tutors, it was envisaged that suitable knowledge-level representations were probably already available. An attempt was made to use the CYC model for events, their causes and effects. This was found to have some value but to be generally insufficient to include information about people's motives, plans, goals and actions. An alternative notation, based on the techniques that Schank and Abelson developed for natural language understanding, was found to be appropriate for behaviour simulation.

These techniques have been tested for specific domains. The CYC approach was used in the design of POPIT, a scheme for demonstrating problem-oriented policing to novice police officers. The behaviour simulation method was applied to the problem of modelling domestic violence situations and the actions of the people involved.

Programming the domestic violence simulation is problematic since various other technical difficulties, such as the processing of natural language, would have to be considered. To avoid getting sidetracked into dealing with problems not directly related to the project goals, it was decided to test out the proposed scheme by using the Wizard of Oz approach. This involves the user interacting with the computer in the same way they might interact with a software package, but the processing of user input and the generation of responses is carried out by a human 'Wizard' located at another terminal. The Wizard follows as closely as possible the interactive scheme that has been produced, but may need to extemporize at various points. Results from this experiment suggest that the basic approach is sound and could be implemented using the techniques presented in Chapter 6.

8.3 Contributions of research

The original contributions of the research presented in this thesis are summarized below:

- The development of computer teaching systems has been reviewed from a knowledge-oriented perspective. The importance of knowledge representation and how the user becomes knowledgeable have been considered. The key role of modelling in the learning process has been emphasized.
- A scheme for distinguishing different kinds of system fidelity has been produced, by looking at the modelling of the domain itself and of the expert's understanding of it. Particular concepts of importance that have been highlighted include the actual functioning/behaviour of the domain and problem solving within the domain. Also of significance is the expert's knowledge of conceptual abstractions, causal relationships and his/her problem solving methods. These ideas have been used to distinguish conceptual simulation methods, qualitative reasoning models, expert systems and an idealized teaching system model.
- A framework for the development of interactive learning environments has been presented. It was originally envisaged that knowledge-based systems might provide tools that would aid in this process. In practice, no such techniques have been employed. Instead, significant features of the KBS approach (separation of declarative and procedural considerations, making concepts transparent, cause-effect reasoning) were used for guidance.
- The use of planning formalisms for modelling in procedural domains has been proposed and investigated. A Strips-like scheme has been shown to be appropriate for domain representation and has been adapted to provide user feedback.
- The use of a graphical notation has been proposed for designing systems of this kind. This notation enables a non-computer specialist to be involved in the development process. It is planned that the pictorial scheme should be implemented on the computer as a step in the automating of a methodology for creating learning environments.
- The separation of domain and task considerations in procedural tutor modelling has been advocated. To implement such a scheme, a method originally devised

for controlling an automated planner has been adapted for use in teaching systems. It provides a notation that is suitable for task specification within procedural domains, and allows task-oriented feedback to be distinguished from domain guidance.

- Graphical representation of tasks within a procedural tutor has been shown to be feasible. Again, the notation is such that a non-specialist should be able to help in the system design process. From the pictorial description, an interpretable textual task specification can be generated automatically. This specification is compatible with that used for the domain, and can be used in a similar fashion to provide appropriate feedback in a teaching situation. The pictorial representation used for this purpose is Drummond's projection graph.
- A methodology for developing a procedural tutor that employs plan nets and projection graphs has been presented.
- Techniques of knowledge-based simulation have been shown to be useful for teaching in ill-structured domains. An appraisal of other schemes, demonstrating the problems of using them as a basis for teaching, has been carried out.
- A framework for event representation in a knowledge-based simulation teaching scheme has been proposed. This includes provision of sub-event breakdown and cause-effect relationships that could be incorporated into a teaching system.
- Elements of this approach have been used in the POPIT project for designing a knowledge-based simulation system for teaching problem-oriented policing.
- Schank and Abelson's natural language understanding primitives have been used as the basis of a scheme for simulating human behaviour. A model that shows the interaction of situations, mental states and actions has been developed. A model of how profiles, goals, plans and actions can be assumed to interact has also been proposed.
- A scheme for teaching students about domestic violence, how it might occur and its repercussions, has been developed.
- The potential for applying the behavioural model in an educational setting has been demonstrated by using a Wizard of Oz simulation.

8.4 Future work

The next stage in the development of the procedural tutor scheme presented in this thesis is its implementation. As already noted, this is part of another project that is well under way. Foremost among the problems that need to be addressed is that of efficiency. An approach that is proving effective is the breaking down of domains into relatively independent subdomains. Work has been carried out both at a theoretical and practical level to show the feasibility of these methods (Smith and Kemp, 1995).

Further work needs to be done to provide structured feedback and binding it in with the Tenenberg formulation. If the hierarchical levels of predicates allowed in his scheme could be exploited then there could be an improvement in the guidance given. Users could be shown high-level, more long term goals, instead of, or in addition to the immediate ones.

The plan net scheme, at present, cannot deal with static axioms. These have to be represented separately. If input is to be entirely graphical, some method is needed to model these. If this can be done then a total graphical description of procedural tasks and domains can be envisaged.

Detection of logic errors in both domain and task specification could be improved if the graphical notation presented on the screen showed the state of the device at each stage, and the effect of each action.

An alternative approach to domain and task specification is their description in a textual form. A notation based on Prolog or a more declarative programming language such as Godel (Hill and Lloyd, 1994) which would allow a non-deterministic specification of tasks within a domain would appear to be feasible. This notation would be more succinct than a pictorial scheme for large domains and could be more easily translated into an efficient, runnable program.

The methods may be applied in a variety of domains. Problems to be considered in the near future include the setting up of an electron microscope and the choosing of chemical sprayers.

The knowledge-based simulation scheme for modelling human behaviour needs to be implemented. The overall design framework developed in POPIT can be combined with the detailed scheme used for simulating the domestic violence system. It has been demonstrated that the inclusion of information about the goals, plans and actions of the

agents can be used to provide a system from which subjects can learn. It is envisaged that such a scheme will be fairly brittle until much more general knowledge of the world can be incorporated. The inclusion of the CYC knowledge base with its huge store of real-world knowledge would make the scheme more tractable.

Finally, further work needs to be done to assess the suitability of languages for implementing this system. A logic programming language would be appropriate (although not essential, of course) for representing the deductive chains that need to be considered. The event and agent schemata could be conveniently stored in frames and so a language with object-oriented features, such as those available in some versions of Prolog would be suitable.

References

- Allport, G. W. (1961). *Pattern and Growth in Personality*. New York: Holt, Rinehart and Winston.
- Allport, G. W. and Odbert, H. S. (1936). *Trait-names: a Psycho-lexical Study*. Psychological Monographs.
- Alpert, D. (1975). The PLATO IV system in use: a progress report. In O. Lecarme and R. Lewis (Eds.), *Computers in Education* (pp. 181-185). Amsterdam: North-Holland.
- Anderson, J. R. (1976). *Language, Memory and Thought*. Hillsdale, NJ: Lawrence Erlbaum.
- Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.
- Anderson, J. R. (1984). Cognitive psychology and intelligent tutoring. *Proceedings of Cognitive Science Society Conference* (pp. 37-43), Boulder, CO.
- Anderson, J. R. (1987). Production systems, learning and tutoring. In D. Klahr, P. Langley and R. Neches (Eds.), *Production System Models of Learning and Development* (pp. 437-458). Cambridge, MA: MIT Press.
- Anderson, J. R. (1988). The expert module. In M. C. Polson and J. J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems* (pp. 21-53). Hillsdale, NJ: Lawrence Erlbaum.
- Anderson, J. R. (1989). Psychology and intelligent tutoring. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (p. 1). Amsterdam: IOS.

- Anderson, J. R. (1990). Analysis of student performance with the LISP tutor. In N. Frederiksen, R. Glaser, A. Lesgold and M. Shafto (Eds.), *Diagnostic Monitoring of Skill and Knowledge Acquisition* (pp. 27-50). Hillsdale, NJ: Lawrence Erlbaum.
- Anderson, J. R. and Bower, G. (1973). *Human Associative Memory*. Washington, DC: Winston-Wiley.
- Anderson, J. R., Boyle, C. F., Corbett, A. T. and Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42, 7-49.
- Anderson, J. R., Boyle, C. F., Farrell, R. and Reiser, B. J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), *Modelling Cognition* (pp. 93-133). Chichester: Wiley.
- Anderson, J. R., Boyle, C. F. and Yost, G. (1985). The Geometry Tutor. *Proceedings of 9th International Joint Conference on Artificial Intelligence* (pp. 1-7), Los Angeles.
- Anderson, J. R., Corbett, A. T., Fincham, J., Hoffman, D. and Pelletier, R. (1992). General principles for an intelligent tutoring architecture. In J. W. Regian and V. Shute (Eds.), *Cognitive Approaches to Automated Instruction* (pp. 81-106). Hillsdale, NJ: Lawrence Erlbaum.
- Andrew, I. and Ellis, S. (1994). Bringing virtual worlds to life. *AI Expert*, 9(5), 15-17.
- Barden, L. (1994). Chess. *Guardian Weekly* (2/10/94) p. 23.
- Barker, P. (1987). *Author Languages for CAL*. London: MacMillan.
- Barnett, J., Knight, K., Mani, I. and Rich, E. (1990). Knowledge and natural language processing. *Communications of the ACM*, 33(8), 50-71.
- Baum, L. F. (1969). *The Wizard of Oz* (Paperback ed.). London: Armada.
- Beeson, M. J. (1989). The user model in MATHPERT: an expert system for learning mathematics. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 9-14). Amsterdam: IOS.
- Benyon, D. and Murray, D. (1988). Experience with adaptive interfaces. *Computer Journal*, 31(5).
- Bloom, B. (1984). The 2 sigma problem. *Educational Researcher*, 13, 4-16.
- Bobrow, D. G. (1984). Special issue on qualitative reasoning about physical systems. *Artificial Intelligence*, 24(1-3).
- Boden, M. (1989). *Artificial Intelligence in Psychology*. Cambridge, MA: MIT Press.
- Bonar, J. and Cunningham, R. (1988). Bridge: an intelligent tutor for thinking about programming. In J. Self (Ed.), *Artificial Intelligence and Human Learning* (pp. 391-409). London: Chapman and Hall.

- Booch, G. (1991). *Object Oriented Design with Applications*. Redwood City, CA: Benjamin/Cummings.
- Bootzin, R. R., Bower, G. H., Zajonc, R. B. and Hall, E. (1986). *Psychology Today: an Introduction* (6th ed.). New York: Random House.
- Boud, D. and Feletti, G. (Eds.). (1991). *The Challenge of Problem-based Learning*. London: Kogan Page.
- Brecht, B. J., McCalla, G. I. and Greer, J. E. (1989). Planning the content of instruction. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 32-41). Amsterdam: IOS.
- Breuker, J., Wielinga, B., van Someren, M., de Hoog, R., Schreiber, G., de Greef, P., Bredeweg, B., Wielemaker, J., Billault, J.-P., Davoodi, M. and Hayward, S. (1987). *Model Driven Knowledge Acquisition: Interpretation Models*. Esprit Project P1098, Deliverable D1 (Task A1), University of Amsterdam and STL Ltd.
- Brown, J. S. (1990). Towards a new epistemology for learning. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Education* (pp. 266-282). Norwood, NJ: Ablex.
- Brown, J. S., Burton, R. R. and Bell, A. G. (1975). SOPHIE: a step toward creating a reactive learning environment. *International Journal of Man-Machine Studies*, 7, 675-696.
- Brown, J. S., Burton, R. R. and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 227-282). London: Academic Press.
- Brown, J. S., Collins, A. and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 18, 32-42.
- Brown, J. S. and VanLehn, K. (1980). Repair Theory: a generative theory of bugs in procedural skills. *Cognitive Science*, 4, 379-426.
- Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review*, 31(1), 21-32.
- Buchanan, B. G., Sutherland, G. and Feigenbaum, E. A. (1969). Heuristic Dendral: a program for generating explanatory hypotheses in organic chemistry. In B. Meltzer and D. Michie (Eds.), *Machine Intelligence* (pp. 209-254). Edinburgh: Edinburgh University Press.
- Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 157-183). London: Academic Press.

- Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 79-98). London: Academic Press.
- Carbonell, J. R. (1970). AI in CAI: an artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, MMS-11(4), 190-202.
- Carroll, J. M. and McKendree, J. (1987). Interface design issues for advice-giving expert systems. *Communications of the ACM*, 30, 14-31.
- Carter, P. (1993). *Computer-based Learning for Ill-defined Domains*. Dissertation in partial fulfilment of requirements for B.A., Massey University, Palmerston North.
- Cattell, R. B. (1964). *Personality and Social Psychology*. San Diego: Robert R Knapp.
- Champagne, A. B., Gunstone, R. G. and Klopfer, L. E. (1985a). Effecting changes in cognitive structures among physics students. In L. H. T. West and A. L. Pines (Eds.), *Cognitive Structure and Conceptual Change* (pp. 163-187). London: Academic Press.
- Champagne, A. B., Gunstone, R. G. and Klopfer, L. E. (1985b). Instructional consequences of students' knowledge about physical phenomena. In L. H. T. West and A. L. Pines (Eds.), *Cognitive Structure and Conceptual Change* (pp. 61-90). London: Academic Press.
- Champagne, A. B., Klopfer, L. E. and Gunstone, R. F. (1982). Cognitive research and the design of science instruction. *Educational Psychologist*, 17(1), 31-53.
- Chandrasekaran, B. and Mittal, S. (1983). Deep versus compiled approaches to diagnostic problem-solving. *International Journal of Man-Machine Studies*, 19, 425-436.
- Chang, M. K. (1990). Logo research in general and special education. In G. Schuyten and M. Valcke (Eds.), *Teaching and Learning in Logo-based Environments* (pp. 35-44). Amsterdam: IOS.
- Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32, 333-377.
- Chi, M. T. H., Feltovich, P. J. and Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121-152.
- Chin, D. (1984). An analysis of scripts generated in writing between users and computer consultants. D. J. Frailey (Ed.), *Proceedings of National Computer Conference*, 637-641, Las Vegas: AFIPS.
- Chipman, S. F., Segal, J. W. and Glaser (Eds.). (1985). *Thinking and Learning Skills: Research and Open Questions*. Hillsdale, NJ: Lawrence Erlbaum.

- Churchill, J. (1990). *Nissan Bluebird Owners Workshop Manual*. Newbury Park, CA: Haynes Publishing Group.
- Clancey, W. J. (1983). The epistemology of a rule-based expert system - a framework for explanation. *Artificial Intelligence*, 20, 215-251.
- Clancey, W. J. (1985). Heuristic classification. *Artificial Intelligence*, 27, 289-350.
- Clancey, W. J. (1987a). *Knowledge-based Tutoring: the Guidon Program*. Cambridge, MA: MIT Press.
- Clancey, W. J. (1987b). Methodology for building an intelligent tutoring system. In G. Kearsley (Ed.), *Artificial Intelligence and Instruction* (pp. 193-227). Reading, MA: Addison-Wesley.
- Clancey, W. J. (1988a). Acquiring, representing, and evaluating a competence model of diagnostic strategy. In M. T. H. Chi, R. Glaser and M. J. Farr (Eds.), *The Nature of Expertise* (pp. 343-418). Hillsdale, NJ: Lawrence Erlbaum.
- Clancey, W. J. (1988b). The knowledge engineer as student: metacognitive bases for asking good questions. In H. Mandl and A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 80-113). New York: Springer-Verlag.
- Clancey, W. J. (1988c). The role of qualitative models in instruction. In J. Self (Ed.), *Artificial Intelligence and Human Learning* (pp. 49-68). London: Chapman and Hall.
- Clancey, W. J. (1989). Viewing knowledge bases as qualitative models. *IEEE Expert* (Summer), 9-23.
- Clancey, W. J. (1991). Why today's computers don't learn the way people do. In P. A. Flach and R. A. Meersman (Eds.), *Future Directions in Artificial Intelligence* (pp. 54-62). New York: Elsevier.
- Clancey, W. J. (1992). Guidon-Manage revisited: a socio-technical systems approach. In C. Frasson, G. Gauthier and G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 21-36). Berlin: Springer-Verlag.
- Clancey, W. J. and Letsinger, R. (1981). NEOMCYCIN: reconfiguring a rule based expert system for application to teaching. *Proceedings of 7th International Joint Conference on Artificial Intelligence*, 2 (pp. 829-836), Vancouver.
- Coles, L. S. (1994). Computer chess: the drosophila of AI. *AI Expert*, 9(4), 25-31.
- Collins, A. (1977). Processes in acquiring knowledge. In R. C. Anderson, R. J. Spiro and W. E. Montague (Eds.), *Schooling and the Acquisition of Knowledge* (pp. 339-363). New York: Wiley.

Collins, A. and Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl and A. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 1-18). New York: Springer-Verlag.

Collins, A., Warnock, E. H. and Passafiume, J. (1975). Analysis and synthesis of tutorial dialogue. In G. Bower (Ed.), *The Psychology of Learning and Motivation* New York: Academic Press.

Corbett, A. T. and Anderson, J. R. (1989). Feedback timing and student control in the Lisp Intelligent Tutoring System. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 64-72). Amsterdam: IOS.

Corbett, A. T., Anderson, J. R. and Fincham, J. M. (1991). Menu selection versus typing: effects on learning in an intelligent programming tutor. *Proceedings of International Conference on the Learning Sciences* (pp. 107-112), Charlottesville, VA: Association for the Advancement of Computing in Education.

Corbett, A. T., Anderson, J. R. and Patterson, E. G. (1990). Student modeling and tutoring flexibility in the Lisp Intelligent Tutoring System. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Education* (pp. 83-106). Norwood, NJ: Ablex.

Craske, N. G., Richards, T. J. and Cumming, G. (1991). ExperTutor, a general purpose intelligent educational system. In R. Lewis and S. Otsuki (Eds.), *Advanced Research on Computers in Education* (pp. 301-306). Amsterdam: North-Holland.

CTGV (1993). Anchored instruction and situated cognition revisited. *Educational Technology* (March), 52-70.

Cullingford, R. E. (1981). SAM. In R. C. Schank and C. K. Reisbeck (Eds.), *Inside Computer Understanding: Five Programs Plus Miniatures* Hillsdale, NJ: Lawrence Erlbaum.

Cumming, G. and Self, J. (1989). Collaborative intelligent educational systems. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 73-80). Amsterdam: IOS.

Cumming, G. and Self, J. (1991). Learner modeling in collaborative intelligent educational systems. In P. Goodyear (Ed.), *Teaching Knowledge and Intelligent Tutoring* (pp. 85-104). Norwood, NJ: Ablex.

Davis, E. (1990). *Representations of Commonsense Knowledge*. San Francisco: Morgan Kaufmann.

Davis, L. H. (1979). *Theory of Action*. Englewood Cliffs, NJ: Prentice-Hall.

- de Kleer, J. and Brown, J. S. (1981). Mental models of physical mechanisms and their acquisition. In J. R. Anderson (Ed.), *Cognitive Skills and their Acquisition* (pp. 285-309). Hillsdale, NJ: Lawrence Erlbaum.
- de Kleer, J. and Brown, J. S. (1983). Assumptions and ambiguities in mechanistic mental models. In D. Gentner and A. L. Stevens (Eds.), *Mental Models* (pp. 155-190). Hillsdale, NJ: Lawrence Erlbaum.
- Dillenbourg, P. and Self, J. (1992a). A computational approach to socially distributed cognition. *European Journal of Psychology of Education*, 7(4), 353-372.
- Dillenbourg, P. and Self, J. (1992b). People power: a human-computer collaborative learning system. In C. Frasson, G. Gauthier and G. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 651-660). Berlin: Springer-Verlag.
- Doran, B. and Tate, G. (1972a). *An Approach to Structured Programming (Part I)*. Computer Science Publication No. 6. Massey University, Palmerston North.
- Doran, B. and Tate, G. (1972b). *An Approach to Structured Programming (Part II)*. Computer Science Publication No. 9. Massey University, Palmerston North.
- Drummond, M. (1985). Refining and extending the procedural net. *Proceedings of 9th International Joint Conference on Artificial Intelligence* (pp. 1110-1012), San Francisco: Morgan Kaufmann.
- Drummond, M. (1989). Situated control rules. In R. J. Brachman, H. J. Levesque and R. Reiter (Eds.), *Principles of Knowledge Representation and Reasoning* (pp. 103-113). San Francisco: Morgan Kaufmann.
- du Boulay, B. and Sothcott, C. (1987). Computers teaching programming: an introductory survey of the field. In R. W. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education (Vol 1): Learning Environments and Tutoring Systems* (pp. 345-372). Norwood, NJ: Ablex.
- Duchastel, P. (1989). Knowledge-based instructional gaming: GEO. *Journal of Educational Technology Systems*, 17(3), 189-203.
- Dugdale, S. and Kibbey, D. (1975). *Fractions Curriculum of the PLATO Elementary School Mathematics Project*. Computer-based Education Research Laboratory Technical Report, Urbana, Illinois.
- Dyer, M. G. (1983). *In-Depth Understanding: a Computer Model of Integrated Processing for Narrative Comprehension*. Cambridge, MA: MIT Press.
- Eck, J. E. and Spelman, W. (1987). *Problem Solving: Problem-oriented Policing in Newport News*. Police Executive Research Forum, U.S. Department of Justice.

- Elsom-Cook, M. (1990). Guided discovery tutoring. In M. Elsom-Cook (Ed.), *Guided Discovery Tutoring: a Framework for ICAI Research* (pp. 3-23). London: Paul Chapman.
- Ernst, G. W. and Newell, A. (1969). *GPS: a Case Study in Generality and Problem Solving*. New York: Academic Press.
- Eysenck, H. J. (1970). *The Structure of Human Personality*. London: Methuen.
- Fairweather, P., Gibbons, A., Rogers, D., Waki, R. and O'Neal, A. (1992). A model for computer-based training. *AI Expert*, 7(12), 30-35.
- Farrell, R. G., Anderson, J. R. and Reiser, B. J. (1984). An interactive computer-based tutor for LISP. *Proceedings of National Conference on Artificial Intelligence* (pp. 106-109), Austin, Texas.
- Fernandez-Valmayor, A. and Chamizo, C. F. (1992). Educational and research utilization of a dynamic knowledge base. In M. R. Kibby and J. R. Hartley (Eds.), *Computer Assisted Learning* (pp. 51-61). Oxford: Pergamon Press.
- Fikes, R. E. and Kehler, T. (1985). The role of frame-based representation in reasoning. *Communications of the ACM*, 28, 904-920.
- Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2, 189-208.
- Finkelstein, J. H. (1992). Strategies for developing a capacity to produce multimedia education and training materials in institutions of higher education. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 43-48), Brisbane: University of Queensland.
- Fishwick, P. and Luker, P. (1991). *Qualitative Simulation Modelling and Analysis*. New York: Springer-Verlag.
- Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, 24, 85-168.
- Forman, G. and Pufall, P. (Eds.). (1988). *Constructivism in the Computer Age*. Hillsdale, NJ: Lawrence Erlbaum.
- Fox, B. R. and Kempf, K. G. (1985). Opportunistic scheduling for robotic assembly. *Proceedings of IEEE Conference on Robotics and Automation* (pp. 880-889), St Louis, MO.
- Gagne, R. M. (1966). Varieties of learning and the concept of discovery. In L. S. Shulman and E. R. Keislar (Eds.), *Learning by Discovery: a Critical Appraisal* (pp. 135-150). Chicago: Rand McNally.

- Gagne, R. M. (1985). *The Conditions of Learning and Theory of Instruction* (4th ed.). New York: Holt, Rinehart and Winston.
- Gilb, T. (1988). *Principles of Software Engineering Management*. Wokingham: Addison-Wesley.
- Gilbert, W. S. (1992). *The Mikado*. New York: Dover Publications.
- Giles-Sims, J. (1983). *Wife Battering: a Systems Theory Approach*. New York: Guilford Press.
- Glaser, R. (1966). Variables in discovery learning. In L. S. Shulman and E. R. Keislar (Eds.), *Learning by Discovery: a Critical Appraisal* (pp. 13-26). Chicago: Rand McNally.
- Glaser, R. (1986). On the nature of expertise. In F. Klix and H. Hagendorf (Eds.), *Human Memory and Cognitive Capabilities* (pp. 915-928). New York: Elsevier.
- Glaser, R. (1987). Thoughts on expertise. In C. Schooler and W. Schaie (Eds.), *Cognitive Functioning and Social Structure over the Life Course* (pp. 81-94). Norwood, NJ: Ablex.
- Glaser, R. and Chi, M. T. H. (1988). Overview. In M. T. H. Chi, R. Glaser and M. J. Farr (Eds.), *The Nature of Expertise* (pp. xv-xxviii). Hillsdale, NJ: Lawrence Erlbaum.
- Glaser, R. (1989). Expertise and learning: how do we think about instructional processes now that we have discovered knowledge structures? In D. Klahr and K. Kotovsky (Eds.), *Complex Information Processing: the Impact of Herbert A Simon* (pp. 269-282). Hillsdale, NJ: Lawrence Erlbaum.
- Glaser, R., Lesgold, A., Lajoie, S., Eastman, R., Greenberg, L., Logan, D., Magone, M., Weiner, A., Wolf, R. and Yengo, L. (1985). *Cognitive Task Analysis to Enhance Technical Skills Training and Assessment*. Report Commissioned by Air Force Human Resource Laboratory, Learning Research and Development Center, University of Pittsburgh.
- Gobbo, C. and Chi, M. T. H. (1986). How knowledge is structured and used by expert and novice children. *Cognitive Development*, 1, 221-237.
- Goldstein, H. (1990). *Problem-Oriented Policing*. New York: McGraw-Hill.
- Goldstein, I. P. (1979). The genetic graph: a representation for the evolution of procedural knowledge. *International Journal for Man-Machine Studies*, 11, 51-77.
- Greer, J. E., Mark, M. A. and McCalla, G. I. (1989). Incorporating granularity-based recognition into SCENT. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 107-115). Amsterdam: IOS.

- Hammersley, J. M. and Handscombe, D. C. (1964). *Monte Carlo Methods*. London: Methuen.
- Hart, A. (1992). *Knowledge Acquisition for Expert Systems* (3rd ed.). New York: McGraw-Hill.
- Hart, P. E. (1982). Direction for AI in the eighties. *SIGART Newsletter*, 79, 79.
- Hartley, J. R., Ravenscroft, A. and Williams, R. J. (1992). CACTUS: Command and Control Training Using Knowledge-based Simulations. *Interactive Learning International*, 8, 127-136.
- Hartley, J. R. and Sleeman, D. (1973). Towards more intelligent teaching systems. *International Journal of Man-Machine Studies*, 5, 215-235.
- Hartog, R. (1989). Qualitative simulation and knowledge representation for intelligent tutoring. In H. Maurer (Ed.), *Computer Assisted Learning* (pp. 193-213). New York: Springer-Verlag.
- Hayes, P. J. (1985). Naive physics I: ontology for liquids. In J. Hobbs and R. C. Moore (Eds.), *Formal Theories of the Common Sense World* (pp. 71-107). Norwood, NJ: Ablex.
- Hayes-Roth, F. (1985). Rule-based systems. *Communications of the ACM*, 28, 921-932.
- Hemphill, L. (1973). *The Relationship of Language and Belief: with Special Emphasis on English 'for' Constructions*. PhD, Stanford.
- Henderson, J. (1992). Interactive Multimedia for experiential learning: a consideration of Dewey's educational philosophy. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 68-75), Brisbane: University of Queensland.
- Hewitt, C. (1969). PLANNER: a language for proving theorems in robots. *Proceedings of International Joint Conference on Artificial Intelligence* (pp. 295-301).
- HFRG (1993). *Software Usability Measurement Inventory*. University College, Cork.
- Hill, P. and Lloyd, J. (1994). *The Godel Programming Language*. Cambridge, MA: MIT Press.
- Hobbs, J. R. (1985). Granularity. *Proceedings of 9th International Joint Conference on Artificial Intelligence* (pp. 1-4), Los Angeles.
- Hollan, J. D., Hutchins, E. L. and Weitzman, L. M. (1984). STEAMER: an interactive, inspectable, simulation-based training system. *AI Magazine*, 5, 15-27.
- Holzl, A. and Robb, D. (1992). *Proceedings of Second International Conference on Information Technology for Training and Education*. Brisbane: University of Queensland.

- Jackson, P. (1990). *Introduction to Expert Systems* (Second ed.). Wokingham: Addison-Wesley.
- Judson, H. F. (1979). *The Eighth Day of Creation: Makers of the Revolution in Biology*. London: Jonathan Cape.
- Kahn, G. and McDermott, J. (1984). The MUD system. *Proceedings of First IEEE Conference on Artificial Intelligence Applications* (pp. 116-122), Denver: IEEE Computer Society.
- Kahn, K. M. (1979). *Director Guide*. AI Memo No. 482B. Artificial Intelligence Lab, MIT.
- Kalat, J. W. (1990). *Introduction to Psychology* (2nd ed.). Belmont, CA: Wadsworth Publishing Company.
- Karplus, W. J. (1977). The spectrum of mathematical modelling and systems simulation. *Mathematics and Computers in Simulation*, 19, 3-10.
- Kass, R. (1989). Student modeling in Intelligent Tutoring Systems - implications for user modeling. In A. Kobsa and W. Wahlster (Eds.), *User Models in Dialog Systems* (pp. 386-410). New York: Springer-Verlag.
- Katz, S., Lesgold, A., Eggan, G. and Gordin, M. (1992a). Modeling the student in Sherlock II. *Journal of Artificial Intelligence in Education*, 3, 495-518.
- Katz, S., Lesgold, A., Eggan, G., Gordin, M. and Greenberg, L. (1992b). Self-adjusting curriculum planning in Sherlock II. In I. Tomek (Ed.), *Computer Assisted Learning* (pp. 343-355). New York: Springer-Verlag.
- Kemp, R. H. (1990). Knowledge-based methods in teaching and training. *Proceedings of Fourth New Zealand Conference on Expert Systems*, Palmerston North: Massey University.
- Kemp, R. H. (1992). Intelligent Computer Assisted Instruction: a knowledge-based perspective. *Australian Computer Journal*, 24(3), 121-129.
- Kemp, R. H. (1993). Knowledge-based simulation for teaching. *Proceedings of IEEE International Conference on Artificial Neural Networks and Expert Systems* (pp. 192-195), Dunedin, New Zealand.
- Kemp, R. H. and Boorman, A. (1987). Using Wimps to beat the expert system blues. *Proceedings of Second New Zealand Expert Systems Conference* (pp. 69-75), Auckland.
- Kemp, R. H. and Burns, D. (1992). POPIT: an intelligent teaching package for police training. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 359-370), Brisbane: University of Queensland.

Kemp, R. H. and Carter, P. (1995). *Knowledge-based Simulation Applied to Domestic Violence Scenarios*. Mathematical and Information Sciences Report Series No. 95/1. Massey University, Palmerston North.

Kemp, R. H. and Smith, S. P. (1994a). Domain and task representation for tutorial process models. *International Journal of Human-Computer Studies*, 41, 363-383.

Kemp, R. H. and Smith, S. P. (1994b). *Facilitating Feedback in Discovery Learning Systems*. Information and Mathematical Sciences Report No. 94/19. Massey University, Palmerston North.

Khuwaja, R. A., Evens, M., Rovick, A. and Michael, J. A. (1992). Knowledge representation for an intelligent tutoring system based on a multilevel causal model. In C. Frasson, G. Gauthier and G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 217-224). Berlin: Springer-Verlag.

Klahr, P. and Faught, W. S. (1980). Knowledge-based simulation. *Proceedings of First Annual Conference of American Association for Artificial Intelligence* (pp. 181-183), Stanford, CA.

Koedinger, K. R. (1991). On the design of novel notations and actions to facilitate thinking and learning. *Proceedings of International Conference on the Learning Sciences*, Charlottesville, VA: Association for the Advancement of Computing in Education.

Koedinger, K. R. and Anderson, J. R. (1993). Reifying implicit planning in geometry: guidelines for model-based intelligent tutoring system design. In S. P. Lajoie and S. J. Derry (Eds.), *Computers as Cognitive Tools* (pp. 15-45). Hillsdale, NJ: Lawrence Erlbaum.

Kreutzer, W. (1986). *System Simulation: Programming Languages and Styles*. Reading, MA: Addison-Wesley.

Kreutzer, W. (1990). *Programming for Artificial Intelligence*. Reading, MA: Addison-Wesley.

Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence*, 29, 289-338.

Kulik, J. A., Kulik, C. and Cohen, P. A. (1980). Effectiveness of computer-based college teaching: a meta-analysis of findings. *Review of Educational Research*, 50, 524-544.

Kunz, J. C., Stelzner, M. J. and Williams, M. D. (1989). From classic expert systems to models: introduction to a methodology for building model-based systems. In G. Guida and C. Tasso (Eds.), *Topics in Expert Systems Design* (pp. 87-110). Amsterdam: North-Holland.

- Lajoie, S. P. and Lesgold, A. (1989). Apprenticeship training in the workplace: computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning*, 3, 7-28.
- Laurillard, D. (1990). Generative student models: the limits of diagnosis and remediation. In M. Elsom-Cook (Ed.), *Guided Discovery Tutoring: a Framework for ICAI Research* (pp. 42-53). London: Paul Chapman.
- Lenat, D. B. and Guha, R. V. (1990). *Building Large Knowledge-based Systems: Representations and Inference in the Cyc Project*. Reading, MA: Addison-Wesley.
- Lesgold, A., Eggan, G., Katz, S. and Rao, G. (1992a). Possibilities for assessment using computer-based apprenticeship environments. In J. W. Regian and V. Shute (Eds.), *Cognitive Approaches to Automated Instruction* (pp. 49-80). Hillsdale, NJ: Lawrence Erlbaum.
- Lesgold, A., Glaser, R., Robinson, H., Klopfer, D., Feltovich, P. and Wang, Y. (1988). Expertise in a complex skill: diagnosing X-ray pictures. In M. T. H. Chi, R. Glaser and M. J. Farr (Eds.), *The Nature of Expertise* (pp. 311-342). Hillsdale, NJ: Lawrence Erlbaum.
- Lesgold, A., Lajoie, S., Bunzo, M. and Eggan, G. (1992b). Sherlock: a coached practice environment for an electronics troubleshooting job. In J. H. Larkin, R. W. Chabay and C. Schefitic (Eds.), *Computer-assisted Instruction and Intelligent Tutoring Systems* (pp. 201-238). Hillsdale, NJ: Lawrence Erlbaum.
- Lesgold, A. M., Katz, S., Eggan, G. and Hughes, E. (1990). *Curriculum Content and Design Specifications for Sherlock II*. Technical Report No. 1990-042. Learning Research and Development Center, University of Pittsburgh.
- Lewis, M., Bishay, M. and McArthur, D. (1993). The macrostructure and microstructure of inquiry activities: evidence from students using a microworld for Mathematical discovery. P. Brna, S. Ohlsson and H. Pain (Eds.), *Proceedings of AI-Ed 93: World Conference on Artificial Intelligence in Education* (pp. 169-176), Edinburgh: Association for the Advancement of Computing in Education.
- Lewis, M. W., Milson, R. and Anderson, J. R. (1990). Designing an intelligent authoring system for high school mathematics. In G. Kearsley (Ed.), *Artificial Intelligence and Instruction: Applications and Methods* (pp. 269-301). Reading, MA: Addison-Wesley.
- Lewis, P. (1993). Talk: a shareware program for communicating between terminals, Peter Lewis, 10 Earlston Way, Booragoon, Perth.

- Lifschitz, V. (1987). On the semantics of STRIPS. M. Georgeff and A. L. Lansky (Eds.), *Proceedings of 1986 Workshop: Reasoning about Actions and Plans*: (pp. 1-9), San Francisco: Morgan Kaufmann.
- Livergood, N. D. (1993). Design issues in the simulation of beliefs in corporate intelligence systems: REALPOLITIK II. *International Journal of Man-Machine Studies*, 39, 99-112.
- LRDC (1986). *Centre for Study of Learning - Programs and Activities*.
- Magone, M. E. (1977). *Goal Analysis and Feature Detection as Processes in the Solution of an Insight Problem*. Masters thesis, University of Pittsburgh, Pittsburgh, PA.
- Malone, T. W. (1981). What makes computer games fun? *BYTE* (December).
- Mark, M. A. (1991). *The VCR Tutor: Design and Evaluation of an Intelligent Tutoring System*. Research Report No. 91-7. Department of Computational Science, University of Saskatchewan.
- Marsland, T. A. and Schaeffer, J. (Eds.). (1990). *Computers, Chess and Cognition*. New York: Springer-Verlag.
- Maulsby, D., Greenberg, S. and Mander, R. (1993). Prototyping an intelligent agent through Wizard of Oz. S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel and T. White (Eds.), *Proceedings of INTERCHI '93* (pp. 277-284), Amsterdam: ACM.
- McCalla, G. I. and Greer, J. E. (1990). SCENT-3: an architecture for intelligent advising in problem-solving domains. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Education* (pp. 140-161). Norwood, NJ: Ablex.
- McClure, J. (1991). *Explanations, Accounts, and Illusions: a Critical Analysis*. Cambridge: Cambridge University Press.
- McCrae, R. R. and Costa, P. (1987). Validation of the five-factor model of personality across instruments and observers. *Journal of Personality and Social Psychology*, 52, 81-90.
- McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction*, 5, 381-413.
- McKevitt, P. (1990). Acquiring user models for natural language dialogue systems through Wizard-of-Oz techniques. *Proceedings of Second International Workshop on User Modeling* (pp. 1-13), Honolulu, HI.
- Michie, D. (1982). High-road and low-road programs. *AI Magazine*, 3, 21-22.
- Miller, M. L. and Lucado, S. R. (1992). Integrating intelligent tutoring, computer-based training, and interactive video in a prototype maintenance trainer. In M. J. Farr and J.

- Psotka (Eds.), *Intelligent Instruction by Computer: Theory and Practice* (pp. 127-150). Washington: Taylor & Francis.
- Minsky, M. (1963). Steps toward artificial intelligence. In E. A. Feigenbaum and J. Feldman (Eds.), *Computers and Thought* (pp. 406-452). New York: McGraw-Hill.
- Mohan, P., Greer, J. E. and Jones, M. (1992). Enhancing the instructional capabilities of Intelligent Tutoring Systems. In C. Frasson, G. Gauthier and G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 531-538). Berlin: Springer-Verlag.
- Moyse, R. (1991). Multiple viewpoints imply knowledge negotiation. *Interactive Learning International*, 7, 21-37.
- Newell, A. and Simon, H. A. (1963). GPS, a program that simulates human thought. In E. A. Feigenbaum and J. Feldman (Eds.), *Computers and Thought* (pp. 279-298). New York: McGraw-Hill.
- Newman, D. (1989). Is a student model necessary? Apprenticeship as a model for ITS. In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 177-184). Amsterdam: IOS.
- Norman, D. A. (1983). Some observations on mental models. In D. Gentner and A. L. Stevens (Eds.), *Mental Models* (pp. 7-14). Hillsdale, NJ: Lawrence Erlbaum.
- O'Shea, T. and Self, J. (1983). *Learning and Teaching with Computers: Artificial Intelligence in Education*. Brighton: Harvester Press.
- Ohlsson, S. (1987). Some principles of intelligent tutoring. In R. W. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education (Vol 1): Learning Environments and Tutoring Systems* (pp. 203-237). Norwood, NJ: Ablex.
- Oresky, C. M. and Lenat, D. B. (1991). Strategic Automatic Discovery System. In P. A. Fishwick and R. B. Modjeski (Eds.), *Knowledge-based Simulation: Methodology and Application* (pp. 223-260). New York: Springer-Verlag.
- Palmiteer, S. and Elkerton, J. (1991). An evaluation of animated demonstrations for learning computer-based tasks. *Proceedings of Conference on Human Factors in Computing Systems* (pp. 257-263), New Orleans: ACM SIGCHI.
- Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.
- Parkes, A. and Self, J. (1988). Video-based intelligent tutoring of procedural skills. *Proceedings of ITS-88* (pp. 454-461), Montreal.
- Peachey, D. R. and McCalla, G. I. (1986). Using planning techniques in intelligent tutoring systems. *International Journal of Man-Machine Studies*, 24, 77-98.

Porter, D. (1992). *Towards the CommonKADS Method*. Esprit Project P5248 (KADS-II) T5.1.2. Touche Ross, London.

Powell, J. A. (1992). The use of multi-media to train intelligent command and control of major fire incidents. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 91-98), Brisbane: University of Queensland.

Quillian, M. R. (1969). The teachable language comprehender: a simulation program and theory of language. *Communications of the ACM*, 12, 459-476.

Rabinowitz, M. and Glaser, R. (1986). Cognitive structures and process in highly competent performance. In F. D. Horowitz and M. O'Brien (Eds.), *The Gifted and the Talented: Developmental Perspectives* (pp. 75-98). Washington, DC: American Psychological Association.

Raghavan, K., Schauble, L. and Glaser, R. (1991). A graphic notation to support reflection in scientific reasoning. L. Birnham (Ed.), *Proceedings of International Conference on the Learning Sciences* (pp. 370-374).

Ramsden, P. (1992). *Learning to Teach in Higher Education*. London: Routledge.

Reeves, T. C. (1992). Effective dimensions of interactive learning systems. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 99-115), Brisbane: University of Queensland.

Reimann, P., Raghavan, K. and Glaser, R. (1988). *REFRACT, a Discovery Environment for Geometrical Optics*. Technical Report No. 1988-040. Learning Research and Development Center, University of Pittsburgh.

Reiser, B. J., Anderson, J. R. and Farrell, R. G. (1985). Dynamic student modelling in an intelligent tutor for LISP Programming. *Proceedings of 9th International Joint Conference on Artificial Intelligence* (pp. 8-14), Los Angeles.

Reisig, W. (1985). *Petri Nets: an Introduction*. New York: Springer-Verlag.

Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelly and G. B. Bryan (Eds.), *Human Judgments and Optimality* (pp. 282-315). New York: Wiley.

Resnick, L. B. (1984). Comprehending and learning: implications for a cognitive theory of instruction. In H. Mandl, N. L. Stein and T. Trabasso (Eds.), *Learning and Comprehension of Text* (pp. 431-443). Hillsdale, NJ: Lawrence Erlbaum.

- Resnick, L. B. (1985a). Cognition and instruction: recent theories of human competence and how it is acquired. In B. L. Hammond (Ed.), *Psychology and Learning* (pp. 123-187). American Psychological Association.
- Resnick, L. B. (1985b). Cognitive science and instruction. *Proceedings of American Association for the Advancement of Science* (pp. 1-43), Washington, DC.
- Rheingold, H. (1991). *Virtual Reality*. New York: Summit Books.
- Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, 3, 329-354.
- Rich, E. (1989). Stereotypes and user modeling. In A. Kobsa and W. Wahlster (Eds.), *User Models in Dialog Systems* (pp. 35-51). New York: Springer-Verlag.
- Richards, T. J., Webb, G. I. and Craske, N. G. (1988). Object-oriented control for intelligent computer assisted learning systems. In P. Ercoli and R. Lewis (Eds.), *Artificial Intelligence Tools in Education* (pp. 213-219). Amsterdam: North-Holland.
- Rolfe, J. M. and Staples, K. J. (Eds.). (1986). *Flight Simulation*. Cambridge: CUP.
- Rosenfield, I. (1988). *The Invention of Memory: a New View of the Brain*. New York: Basic Books.
- Rothenberg, J. (1989). The nature of modeling. In L. E. Widman, K. A. Loparo and N. R. Nielsen (Eds.), *Artificial Intelligence, Simulation and Modeling* (pp. 75-92). New York: Wiley.
- Ryba, K. and Anderson, B. (1990). *Learning with Computers: Effective Teaching Strategies*. Eugene, OR: International Society for Technology in Education.
- Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2), 115-135.
- Sacerdoti, E. D. (1975). The non-linear nature of plans. *Proceedings of 4th International Joint Conference on Artificial Intelligence* (pp. 206-214), San Francisco: Morgan Kaufmann.
- Sanders, K. (1989). A logic for emotion. *Proceedings of Conference for Cognitive Science* (pp. 357-363), Ann Arbor, Michigan.
- Schank, R. C. (1975). *Conceptual Information Processing*. Amsterdam: North-Holland.
- Schank, R. C. (1982). Reminding and memory organizations: an introduction to MOPs. In W. Lehnert and M. Ringle (Eds.), *Strategies for Natural Language Processing* (pp. 455-493). Hillsdale, NJ: Lawrence Erlbaum.
- Schank, R. C. and Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum.

Schauble, L., Glaser, R., Raghavan, K. and Reiner, M. (1991). Causal models and experimentation strategies in scientific reasoning. *The Journal of the Learning Sciences*, 1, 201-238.

Sebrechts, M. M. and Marsh, R. L. (1989). Components of computer skill acquisition: some reservations about mental models and discovery learning. In G. Salvendy and M. J. Smith (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems* (pp. 168-173). Amsterdam: Elsevier.

Self, J. (1974). Student models in computer-aided instruction. *International Journal of Man-Machine Studies*, 6, 261-276.

Self, J. (1988). Student models: what use are they? In P. Ercoli and R. Lewis (Eds.), *Artificial Intelligence Tools in Education* (pp. 73-86). New York: North-Holland.

Self, J. (1989). The case for formalising student models (and intelligent tutoring systems generally). In D. Bierman, J. Breuker and J. Sandberg (Eds.), *Artificial Intelligence and Education* (pp. 244). Amsterdam: IOS.

Self, J. (1992). Cognitive diagnosis for tutoring systems. *Proceedings of 10th European Conference on Artificial Intelligence* (pp. 699-703), Vienna: Wiley.

Sherwood, C. (1990). Computers and higher order thinking skills. In A. McDougall and C. Dowling (Eds.), *Computers in Education* (pp. 1081-1086). Amsterdam: North-Holland.

Shneiderman, B. (1983). Direct manipulation: a step beyond programming languages. *IEEE Computer*, 26, 57-69.

Shortliffe, E. H. (1976). *Computer-based Medical Consultations: MYCIN*. New York: Elsevier.

Shulman, L. S. and Keislar, E. R. (1966). *Learning by Discovery: a Critical Appraisal*. Chicago: Rand McNally.

Shute, V. J. and Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world: Smithtown. *Interactive Learning Environments*, 1, 51-77.

Shute, V. J. and Glaser, R. (1991). An intelligent tutoring system for exploring principles of economics. In R. E. Snow and D. Wiley (Eds.), *Improving Inquiry in Social Science* (pp. 333-366). Hillsdale, NJ: Erlbaum.

Shute, V. J., Glaser, R. and Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P. L. Ackerman, R. J. Sternberg and R. Glaser (Eds.), *Learning and Individual Differences: Advances in Theory and Research* (pp. 279-326). New York: Freeman.

- Skinner, B. F. (1938). *The Behaviour of Organisms: an Experimental Analysis*. New York: Appleton-Century-Crofts.
- Sleeman, D. (1982). Introduction. In D. Sleeman and J. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 1-11). London: Academic Press.
- Sleeman, D. and Brown, J. S. (Eds.). (1982). *Intelligent Tutoring Systems*. London: Academic Press.
- Smith, R. B. (1986). The Alternate Reality Kit: an animated environment for creating interactive simulations. *Proceedings of IEEE Workshop on Visual Languages* (pp. 99-106), Dallas, Texas: IEEE Press.
- Smith, S. P. (1994a). Computer Program for Generating Situated Control Rules. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. (1994b). Computer Program for Procedural System Simulation using TWEAK. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. (1994c). Computer Program for Procedural System Simulation using Tenenberg. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. (1994d). Computer Program for Projection Graph Production. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. (1994e). Computer Program for Teaching Procedural Tasks. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. (1994f). PhD Project: Procedural Tutor Design and Implementation. Computer Science Department, Massey University, Palmerston North.
- Smith, S. P. and Kemp, R. H. (1995). Efficient modelling of domains for computer tutoring systems. *Proceedings of 18th Australasian Computer Science Conference*, Adelaide.
- Sowa, J. F. (1984). *Conceptual Structures*. Reading, MA: Addison-Wesley.
- Spada, H., Stumpf, M. and Opwis, K. (1989). The constructive process of knowledge acquisition: student modeling. In H. Maurer (Ed.), *Computer Assisted Learning* (pp. 486-499). New York: Springer-Verlag.
- Spensley, F., Elsom-Cook, M., Byerley, P., Brooks, P., Federici, M. and Scaroni, C. (1990a). Using multiple teaching strategies in an ITS. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Education* (pp. 188-205). Norwood, NJ: Ablex.
- Spensley, F., O'Shea, T., Singer, R., Hennessy, S., O'Malley, C. and Scanlon, E. (1990b). An "alternate realities" microworld for horizontal motion. In D. H. Norrie and H. W. Six (Eds.), *Computer Assisted Learning* (pp. 73-98). New York: Springer-Verlag.

- Spiro, R. J., Feltovich, P. J., Jacobson M. H. and Coulson, R. L. (1991). Cognitive flexibility, constructivism, and hypertext: random access instruction for advanced knowledge acquisition in ill-structured domains. *Educational Technology* (May), 24-33.
- Stasko, J. and Badre, A. (1993). Do algorithm animations assist learning? An empirical study and analysis. S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel and T. White (Eds.), *Proceedings of INTERCHI '93* (pp. 61-66), Amsterdam: ACM.
- Steels, L. (1991). Components of expertise (Tutorial T15). *12th International Joint Conference on Artificial Intelligence*, Sydney.
- Stevens, A. L. and Collins, A. (1977). The goal structure of a Socratic Tutor. *Proceedings of The National ACM Conference* (pp. 256-263), Seattle: ACM.
- Stevens, A. L., Collins, A. and Goldin, S. E. (1982). Misconceptions in students' understanding. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 13-24). London: Academic Press.
- Syms, M. (1992). *Tutoring and Game Playing using Computers*. Dissertation in partial fulfilment of requirements for Diploma in Social Science, Massey University, Palmerston North.
- Szlovits, P. and Pauker, S. G. (1978). Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11(1), 115-144.
- Tenenberg, J. D. M. (1991). Abstraction in planning. In J. F. Allen, H. A. Kautz, R. N. Pelavin and J. D. M. Tenenberg (Eds.), *Reasoning about Plans* (pp. 214-283). San Francisco: Morgan Kaufmann.
- Todd, R. R. (1988). Self-organized learning within an intelligent teaching system. In J. Self (Ed.), *Artificial Intelligence and Human Learning* London: Chapman and Hall.
- Towne, S. M., Munro, A., Pizzini, Q. A., Surmon, D. S., Collier, L. D. and Wogulis, J. L. (1990). Model-building tools for simulation-based training. *Interactive Learning Environments*, 1(1), 33-50.
- Tripp, S. D. (1993). Theories, traditions and situated learning. *Educational Technology* (March), 71-77.
- Try, K. (1992). Can we afford NOT to use technology for training? A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 127-133), Brisbane: University of Queensland.
- Tuthill, G. S. (1990). *Knowledge Engineering: Concepts and Practices for Knowledge-Based Systems*. Blue Ridge Summit, PA: TAB Professional and Reference Books.

- van der Veer, G., Tauber, M. J., Waern, Y. and Muylwijk, V. (1985). On the interaction between system and user characteristics. *Behaviour and Information Technology*, 4(4), 289-308.
- van Joolingen, W. R. and de Jong, T. (1992). Modelling domain knowledge for intelligent simulation learning environments. In M. R. Kibby and J. R. Hartley (Eds.), *Computer Assisted Learning* (pp. 29-37). Oxford: Pergamon Press.
- Wastell, D. (1991). Physiological measurement of cognitive load during interaction with process control displays. In H.-J. Bullinger (Ed.), *Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals* (pp. 228-232). Amsterdam: Elsevier.
- Watson, J. D. (1981). *The Double Helix (a Critical Edition edited by G S Stent)*. London: Weidenfeld and Nicolson.
- Webb, G. I. and Cumming, G. (1991). Educational evaluation of feature-based modelling in a problem solving domain. In R. Lewis and S. Otsuki (Eds.), *Advanced Research on Computers in Education* (pp. 101-108). Amsterdam: North-Holland.
- Weinstein, C. and Rogers, C. (1984). *Explaining Complex Engineering Devices*. Technical Report No. 548. Bolt, Beranek & Newman, Cambridge, MA.
- Weiser, M. and Shertz, J. (1983). Programming problem representation in novice and expert programming. *International Journal of Man-Machine Studies*, 14, 391-396.
- Weizenbaum, J. (1976). *Computer Power and Human Reason*. San Francisco: Freeman.
- Weld, D. S. and de Kleer, J. (Eds.). (1990). *Readings in Qualitative Reasoning about Physical Systems*. San Francisco: Morgan Kaufmann.
- Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. San Francisco: Morgan Kaufmann.
- White, B. Y. and Frederiksen, J. R. (1987). Qualitative models and intelligent learning environments. In R. W. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education (Vol 1): Learning Environments and Tutoring Systems* (pp. 281-305). Norwood, NJ: Ablex.
- White, B. Y. and Frederiksen, J. R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, 42, 99-157.
- Winkels, R. and Breuker, J. (1992). Modelling expertise for educational purposes. In C. Frasson, G. Gauthier and G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 633-641). Berlin: Springer-Verlag.
- Winograd, T. (1972). *Understanding Natural Language*. New York: Academic Press.

Winograd, T. and Flores, C. F. (1985). *Understanding Computers and Cognition: a New Foundation for Design*. Norwood, NJ: Ablex.

Woods, D. D. and Hollnagel, E. (1987). Mapping cognitive demands in complex problem-solving worlds. *International Journal of Man-Machine Studies*, 26, 257-275.

Woolf, B. P. (1988a). Intelligent Tutoring Systems: a survey. In H. Schrobe and AAAI (Eds.), *Exploring Artificial Intelligence* (pp. 1-43). San Francisco: Morgan Kaufmann.

Woolf, B. P. (1988b). Representing complex knowledge in an intelligent machine tutor. In J. Self (Ed.), *Artificial Intelligence and Human Learning* (pp. 3-27). London: Chapman and Hall.

Woolf, B. P. (1992). Building knowledge based tutors. In I. Tomek (Ed.), *Computer Assisted Learning* (pp. 46-60). New York: Springer-Verlag.

Woolf, B. P., Blegen, D., Jansen, J. H. and Verloop, A. (1987). Teaching a complex industrial process. In R. W. Lawler and M. Yazdani (Eds.), *Artificial Intelligence and Education (Vol 1): Learning Environments and Tutoring Systems* (pp. 413-427). Norwood, NJ: Ablex.

Woolf, B. P. and McDonald, D. D. (1984). Building a computer tutor: design issues. *IEEE Computer*, 17(9), 61-73.

Woolf, B. P., Murray, T., Suthers, D. and Schultz, K. (1988). Knowledge primitives for tutoring systems. *Proceedings of ITS-88* (pp. 491-498), Montreal.

Wright, I. A. (1992). System simulation development in the aviation industry. A. Holzl and D. Robb (Eds.), *Proceedings of Second International Conference on Information Technology for Training and Education* (pp. 134-141), Brisbane: University of Queensland.

Yob, G. (1975). Hunt the Wumpus. *Creative Computing*(September-October), 51-54.

Appendices

Appendix A

Gear Mechanism Description

A.1 Introduction

A gear mechanism, based on one used by Fox and Kempf (1985) for demonstrating robot scheduling, is here used as an example of how complex device assembly and dismantling problems can be described using the notations introduced in Chapter 4.

A.2 Gear mechanism representation

A sketch of the mechanism is shown in Figure A.1. Restrictions to building and dismantling are that only one component at a time may be added or removed, and that no component may be added or removed if there is another object in the way.

As a first step in the modelling, the device was considered as consisting of three subsystems, casing, link gearing and drive mechanism, as shown in Figure A.2.

The essential predicates denoting key conditions in the system were chosen as shown in Figure A.3. Inessential predicates are shown in Figure A.4. Possible actions for the system are listed in Figure A.5. Figure A.6 shows the static axioms produced, and Figures A.7 to A.9 give the action tables for each of the subsystems.

A.3 Summary

As may be seen from the figures a succinct representation of even a complex domain can be constructed using the action table notation with appropriate structuring and static axioms.

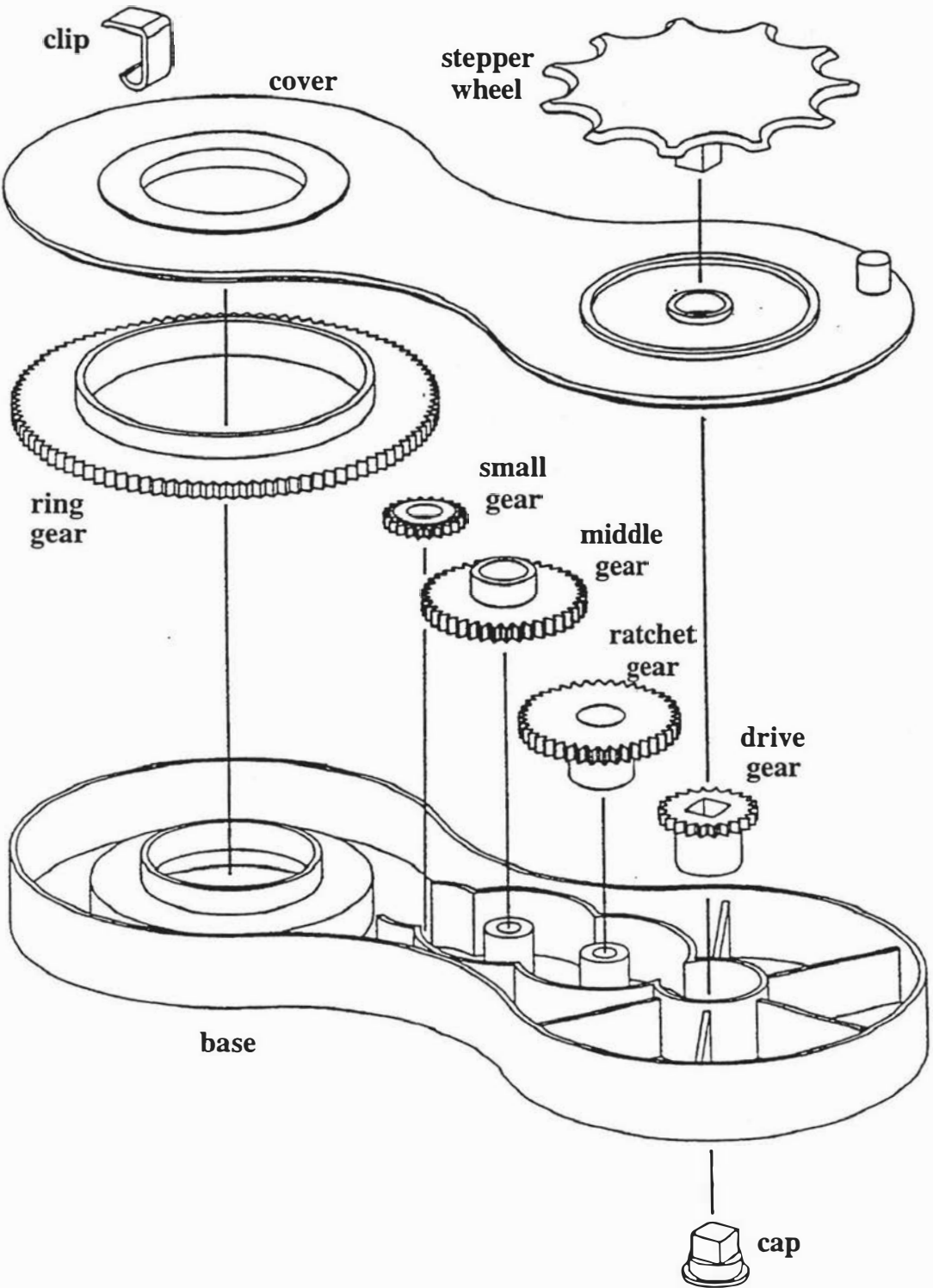


Figure A.1 Gear mechanism - Fox and Kempf (1985)

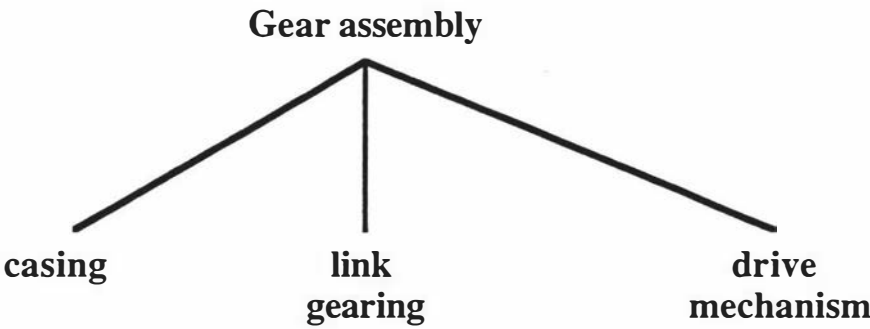


Figure A.2 Gear mechanism structure

predicate	parameter(s)	values
gear	G, P	G=ring, small, middle, ratchet, drive P=absent, present
casing	S	S=none, cover only, base only, mated, clipped
cap	S	S=present, absent
stepper wheel	S	S=present, absent

Figure A.3 Essential predicates for gear mechanism description

predicate	parameter(s)	parameter values
gear box open	-	
gearing	S	S=present, absent
base absent	-	-
cover absent	-	-

Figure A.4 Inessential predicates for gear mechanism description

casing	link gearing	drive mechanism
affix clip	add ring gear	add cap
remove clip	remove ring gear	remove cap
add base	add small gear	add stepper wheel
remove base	remove small gear	remove stepper wheel
add cover	add middle gear	add drive gear
remove cover	remove middle gear	remove drive gear
	add ratchet gear	
	remove ratchet gear	

Figure A.5 Actions for gear mechanism description

casing axioms	gearing axioms	misc axioms
casing(cover only) → gear box open	~gearing(present) → gearing(absent)	cap(S) → ~cap(T) S≠T
casing(base only) → gear box open	gear(ring, present) → gearing(present)	stepper wheel(S) → ~stepper wheel(T) S≠T
casing(cover only) → base absent	gear(middle, present) → gearing(present)	
casing(none) → base absent	gear(small, present) → gearing(present)	
casing(base only) → cover absent	gear(ratchet, present) → gearing(present)	
casing(none) → cover absent	gear(drive, present) → gearing(present)	
casing (S) → ~casing (T) S≠T	gear(G, P) → ~gear(G, R) P≠R	

Figure A.6 Static axioms

preconditions	operations	postconditions
casing(mated)	affix clip	casing(clipped)
casing(clipped)	remove clip	casing(mated)
casing(cover only) cap(absent)	add base [from bottom]	casing(mated)
casing(mated) cap(absent)	remove base [from bottom]	casing(cover only)
casing(none) cap(absent)	add base [from bottom]	casing(base only)
casing(base only) cap(absent)	remove base [from bottom]	casing(none)
casing(none) gearing (absent) stepper wheel (absent)	add base [from top, or from bottom if cap absent]	casing (base)
casing(base only) gearing (absent) stepper wheel (absent)	remove base [from top, or from bottom if cap absent]	casing(none)
casing (none) stepper wheel (absent)	add cover [from top, or from bottom if gears and cap are absent]	casing (cover only)
casing (cover only) stepper wheel (absent)	remove cover [from top, or from bottom if gears and cap are absent]	casing (none)
casing (base only) stepper wheel (absent)	add cover [from top]	casing (mated)
casing (mated) stepper wheel (absent)	remove cover [from top]	casing (base only)
casing (none) gearing (absent) cap (absent)	add cover [from bottom, or from top if stepper wheel absent]	casing (cover only)
casing (cover only) gearing (absent) cap (absent)	remove cover [from bottom if stepper wheel present, otherwise from either direction]	casing (none)

Figure A.7 Gear domain description: casing subsystem

preconditions	operations	postconditions
gear(ring, absent) gear box open	add ring gear	gear(ring, present)
gear(ring, present) gear box open	remove ring gear	gear(ring, absent)
gear(small, absent) base absent	add small gear [from bottom, or from top if middle gear and cover absent]	gear(small, present)
gear(small, present) base absent	remove small gear [from bottom, or from top if middle gear and cover absent]	gear(small, absent)
gear(small, absent) cover absent gear(middle, absent)	add small gear [from top, or from bottom if base absent]	gear(small, present)
gear(small, present) cover absent gear(middle, absent)	remove small gear [from top, or from bottom if base absent]	gear(small, absent)
gear(middle, absent) cover absent gear(ratchet, absent)	add middle gear [from top, or from bottom if small gear and base are absent]	gear(middle, present)
gear(middle, present) cover absent gear(ratchet, absent)	remove middle gear [from top, or from bottom if small gear and base are absent]	gear(middle, absent)
gear(middle, absent) base absent gear(small, absent)	add middle gear [from bottom, or from top if ratchet gear and cover are absent]	gear(middle, present)
gear(middle, present) base absent gear(small, absent)	remove middle gear [from bottom, or from top if ratchet gear and cover are absent]	gear(middle, absent)
gear(ratchet, absent) cover absent	add ratchet gear [from top, or from bottom if middle gear and base absent]	gear(ratchet, present)
gear(ratchet, present) cover absent	remove ratchet gear [from top, or from bottom if middle gear and base absent]	gear(ratchet, absent)
gear(ratchet, absent) base absent gear(middle, absent)	add ratchet gear [from bottom, or from top if cover absent]	gear(ratchet, present)
gear(ratchet, present) base absent gear(middle, absent)	remove ratchet gear [from bottom, or from top if cover absent]	gear(ratchet, absent)

Figure A.8 Gear domain description: link gearing subsystem

preconditions	operations	postconditions
cap(absent)	add cap	cap(present)
cap(present)	remove cap	cap(absent)
stepper wheel (absent)	add stepper wheel	stepper wheel (present)
stepper wheel (present)	remove stepper wheel	stepper wheel (absent)
gear (drive, absent) cover absent stepper wheel (absent)	add drive gear [from the top, or from bottom if cap and base absent]	gear(drive, present)
gear (drive, present) cover absent stepper wheel (absent)	remove drive gear [from the top, or from bottom if cap and base absent]	gear(drive, absent)
gear (drive, absent) cap (absent) base absent	add drive gear [from bottom, or from top if cover and stepper wheel absent]	gear (drive, present)
gear (drive, present) cap (absent) base absent	remove drive gear [from bottom, or from top if cover and stepper wheel absent]	gear (drive, absent)

Figure A.9 Gear domain description: drive mechanism subsystem

Appendix B

Domain Description using Plan Nets

B.1 Denoting VCR operation using plan nets

A diagrammatic description of the operation of a VCR using plan nets is given in Figures B.1 to B.5. Note that this is a slightly simplified model to that described in Section 5.2.2.

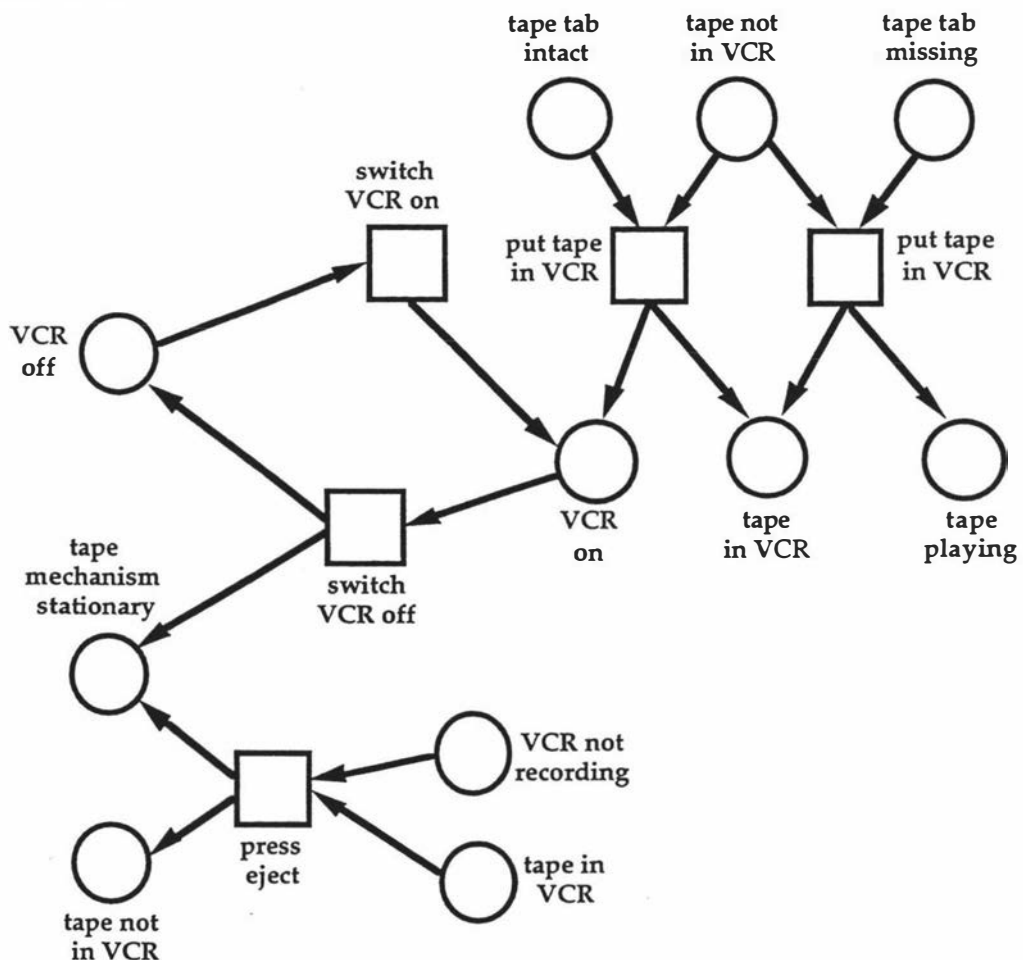


Figure B.1 VCR on/off and tape handling

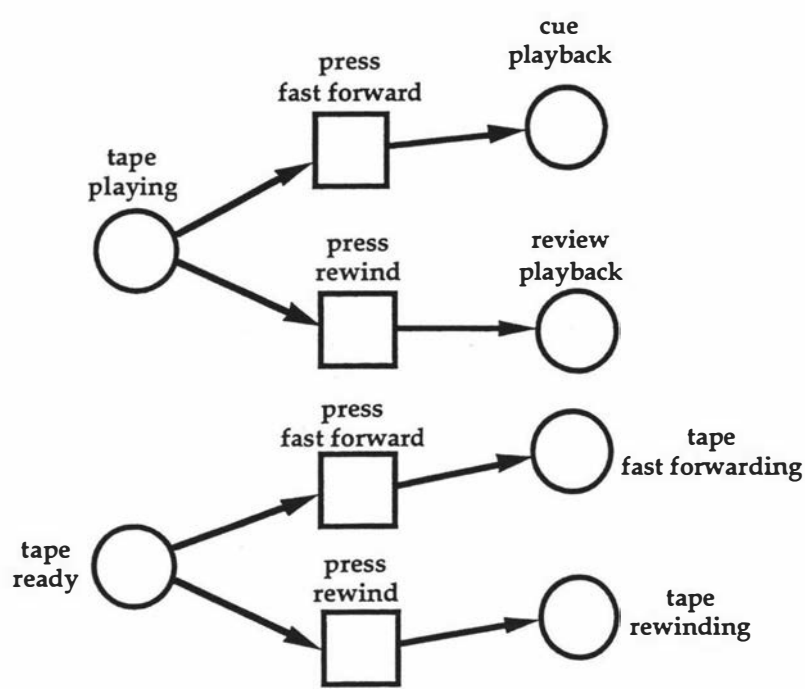


Figure B.2 Rewinding and fastforwarding

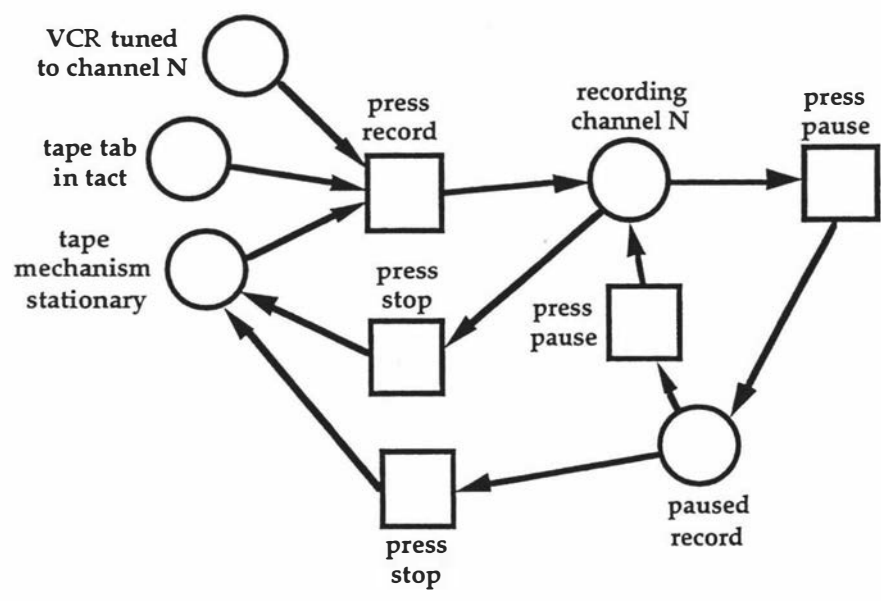


Figure B.3 Recording

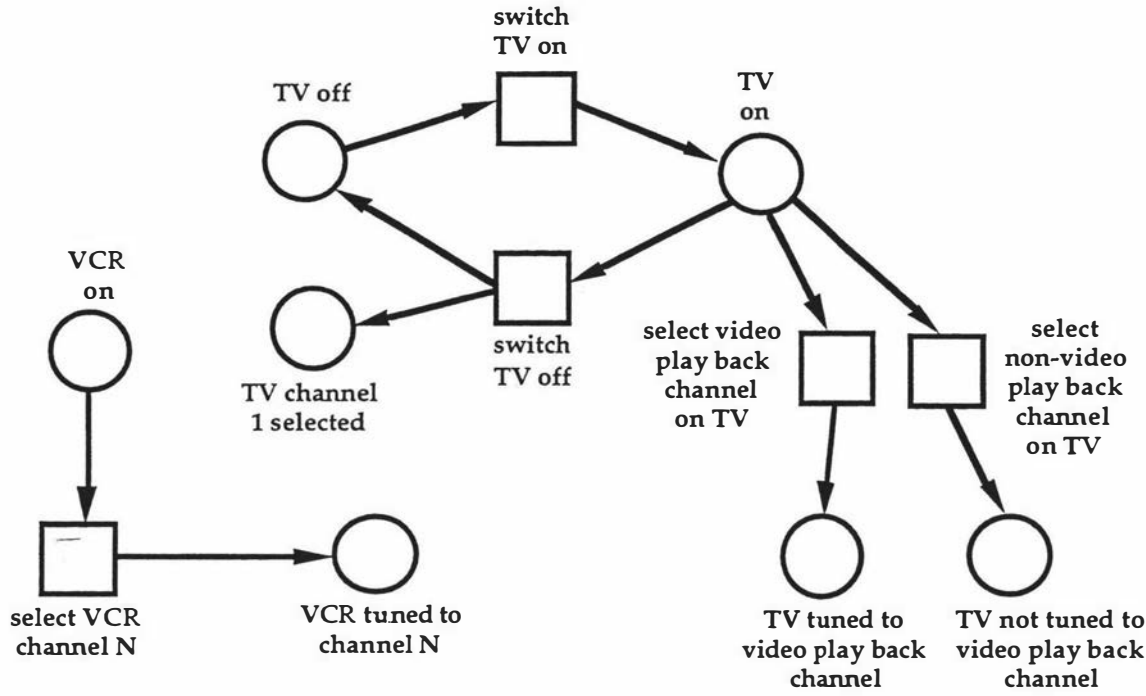


Figure B.4 Channel selection and television operation

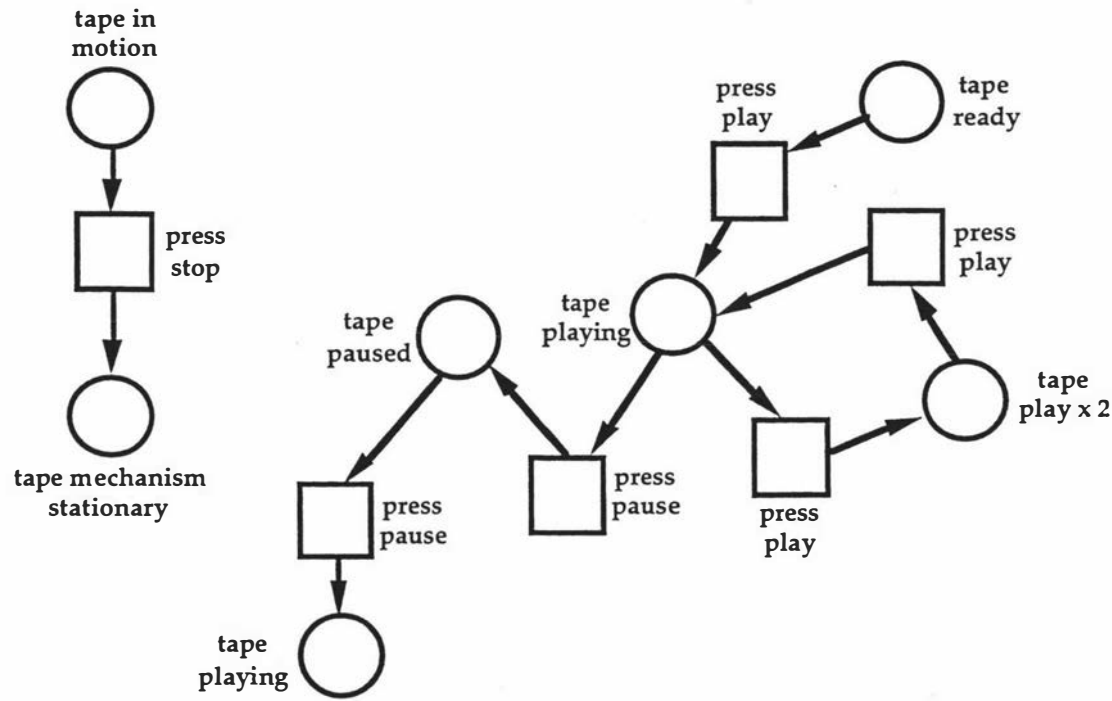


Figure B.5 Stopping and starting

Appendix C

Domestic Dispute Simulation

C.1 Introduction

An analysis is carried out of features that need to be taken into account to simulate a domestic dispute scenario. Information about domestic violence, its causes, manifestations and effects is analysed to provide the basis for scenario generation in an interactive learning environment.

C.2 Event breakdown

Event breakdown can be structured, as noted in Section 6.3, by using the organization outlined by Lenat and Guha (1990). The general hierarchy proposed in that section is reproduced as Figure C.1 for convenience.

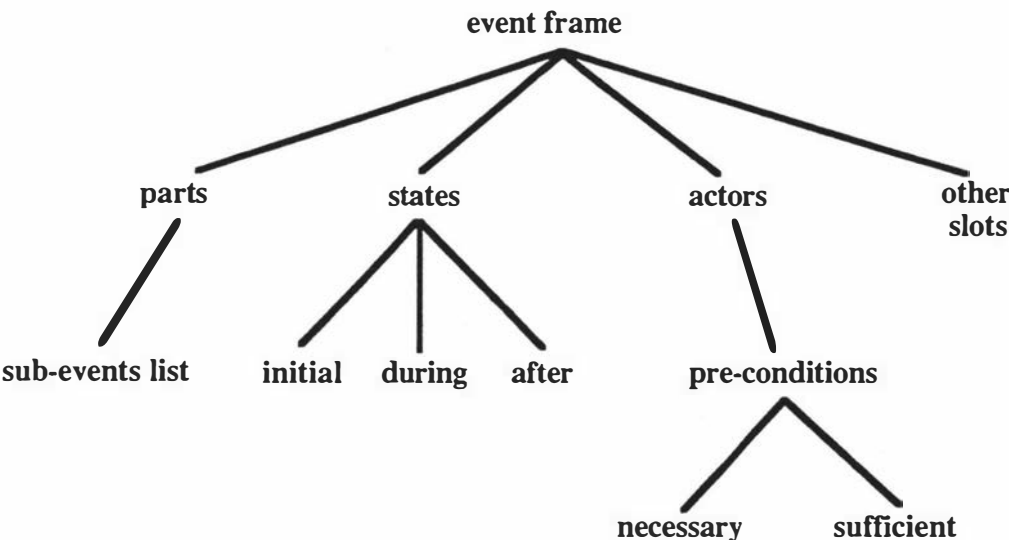


Figure C.1 Event breakdown in causal system

A typical sub-events structure for a domestic violence scheme is illustrated in Figure C.2. Such an event can be described in detail, or just summarised in terms of the changes that occur during the event. The sub-events can be structured in a similar fashion. This gives the possibility of what Lenat and Guha call a 'gray-box' approach, where any event can be summarised at a high level, or can be presented in more detail if necessary.

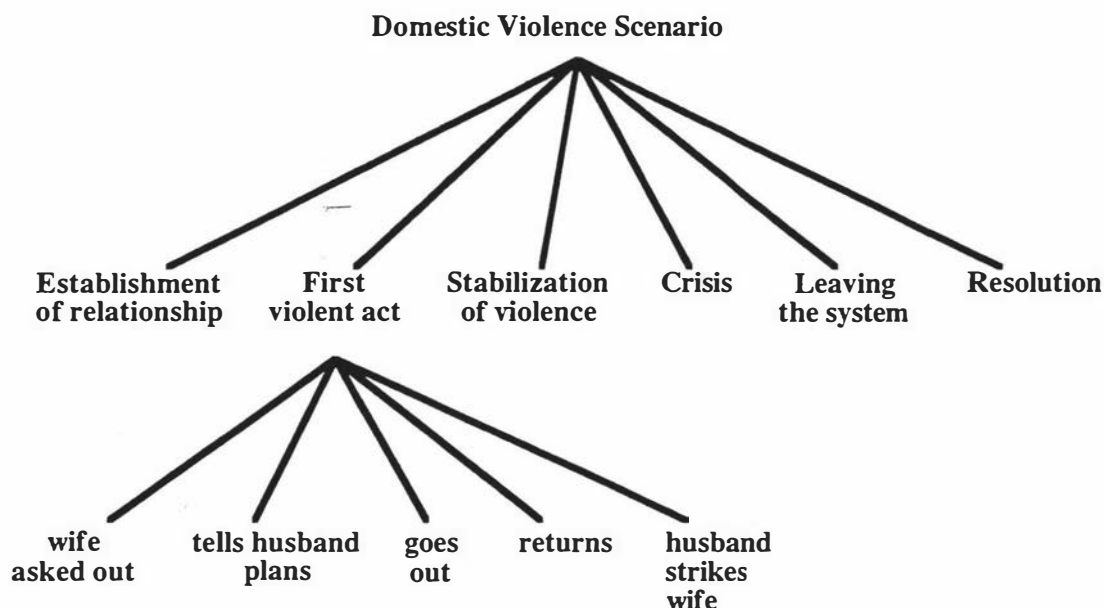


Figure C.2 Domestic violence events outline

As noted in Section 7.3, the structure is loosely based on the description given of the stages of the development of a violent relationship as depicted by Giles-Sims (1983). More details are given below, based on her work and that of other researchers. A fuller description of issues and factors in domestic violence can be found in a Massey University Mathematical and Information Sciences report (Kemp and Carter, 1995).

1. Establishment of relationship

before : couple not together

after : couple together

Three important questions need to be addressed when setting up this initial part of the system:

- (i) What external inputs influence initial system structure?
- (ii) How did the commitment by the parties occur and evolve?
- (iii) What rules about power relationships and violence were in place?

It is important to examine how patterns of the relationship are set up. Often, neither partner will have had previous personal experience of such a relationship and the guidelines will be taken from outside. They may be formed, for example, from a person's own home environment, a friend's experiences, flat life, films, tv, magazines or impressions from colleagues in the work place. If either has had a previous relationship then this may help form a basis. Social mores are particularly important: for example, the attitude in many countries that getting married is an important aim for a female, and that finding the right person or a career are only secondary. This attitude also increases the commitment to a relationship once it is in progress.

Much of what happens during this first period will either be regarded as unacceptable within the relationship or will become part of life. Of specific interest, as far as potential battering is concerned, are the power, control and violence aspects that are set in place. Experience of violence as a child seems to make acceptance of violence as an adult more likely. Also, reaction to violence may be similar. A person may become passive and dissociate themselves from what is happening, or may, perhaps, blame themselves. Approaches to dealing with stress generally can be learned from parents. These are key factors since surveys have shown the inherently stressful nature of family relationships.

So the ground rules of a relationship are set by the background of actors (eg violent upbringing), society attitudes (eg need for marriage, 'til death do us part', 'for better, for worse', 'a man's home is his castle'), attitudes of friends and relatives, expectations, goals, desires and beliefs, as well as behavioural predispositions. Each will have beliefs about relationship norms, conflict and violence, reaction to violence and dealing with stress. The female may be oblivious to signs of violence in her partner's past or may be unaware of these. In many cases, the husband may get sympathy for his unfortunate experiences. Another possible reaction is wariness.

The characteristics and predispositions of the actors are considered in more detail, later.

2. First violent act

before	: no violent act between couple
after	: one violent act between couple

Events that may precipitate the first violent act include moving to a new house or flat, a pregnancy or birth, jealousy of friends, and arguments over relatives.

There is a common pattern to many violent acts which can be used as the basis of an event script. The sequence is often: disagreement, quarrel, violence, appraisal (perhaps ignore partner), grief, contrition and forgiveness. In early versions of this sequence, all steps except the violence may be present. Note that husbands often deny there is a 'loving stage' after the incident although the wife often believes there is. The reaction of the male after an attack is often either not to talk, otherwise carrying on as if nothing has happened, or to leave the marital home. Remorse or contrition is more likely to occur early in the relationship. The husband often believes his action was provoked.

An early violent act is likely to be impulsive and used as a release from accumulated frustration and tension. What is crucial is not necessarily the violent act itself but the reaction of the participants. This, arguably, will determine whether future incidents occur. For example, is the violent person's goal satisfied? If so, then this is positive feedback and increases the likelihood of a repeat. In some cases, the violence may be coercive, attempting to ensure the spouse does not leave.

The reaction of the victim may be pacification, acceptance, rationalisation, fear, forgiveness or anger. She may try to trivialize the problem by regarding the violent act as an isolated incident, or may make excuses (such as 'he blacked out'). Alternatively, she may seek help. Many women feel angry at the time of the event but do not react in an angry, rejecting, or retaliatory fashion. The woman may feel guilt as a reaction to the first incident but is less likely to on subsequent occasions. Forgiveness also tends to reduce over a period of time.

Return to normal activity reinforces acceptance. Responses such as obtaining outside help or leaving the relationship give negative feedback to the action making it less likely to recur.

Critical branch point :

- either the event never occurs again
- or it recurs with some pattern of escalation

3. Stabilization of violence

- before : one act of violence
- after : regular acts of violence

Events precipitating violence may be similar to those cited for the first incident, but, as time goes by, there may be more and more flimsy excuses (overcooked food, the same

meal twice in one week, the prospect of inlaws coming) or even none. In some cases, violence will not be needed in order to maintain the *status quo*, just the threat of its use, cruelty towards family pets, or outbursts where household items are damaged.

Two common responses by the female at this stage are fighting back and withdrawal. Neither strategy seems to alleviate the situation.

The equilibrium of the system is maintained by several means. The male normally attempts to sustain a framework of dominance. The situation is further stabilized by the lack of outside influences. The female, in particular, often becomes isolated. She is unlikely to have a safe haven outside the house to which she can retreat. Lastly, a total commitment and investment in the relationship is often still shared by both partners. The female may stay because of this commitment or for many other reasons, including coercion by her partner and emotional blackmail (he may threaten suicide if she leaves). 'Normal' or contrite periods by the offender can encourage the woman to stay in the system even after violence has become routine.

Although there is a general stability to the situation, the level of violence will often increase over time. As the female reacts less to each level, the male steps up the intensity. A typical sequence is throwing soft items such as cushions, pushing, slapping, hitting, punching, and hitting with objects.

Often there is a polarization of attitudes and beliefs about what is happening and who is to blame. Family, friends, police, doctors and lawyers are often unhelpful (thus seeking help may produce negative feedback and is less likely to become part of the system). Sometimes there is an implicit alliance between the batterer and the police.

4. Crisis

before : regular acts of violence accepted
after : spouse attitude changes - she is prepared to leave

Violence has escalated to a level that is unacceptable either to the spouse or to outsiders. Events that often precipitate an escalation include the crisis triggers mentioned in stage 1 or the woman's desire to leave the relationship. Often, there is a new twist in the behaviour which may involve others. Children may be threatened or introduced into the violent situation, or outsiders may observe incidents or their consequences. This counteracts the effects of isolation such as the acceptance of high

and increasing levels of violence. The wife often feels ashamed at the possibility of discovery or fears for the safety of the children

At this point, the woman can change her internal goals from maintaining the family unit and complying with the partner to trying to change the system or to get out of it. If she does not do this then the pattern of violence will be further strengthened.

4a. Changing the system

At the stage when the violence has stabilised there may be an event (a crisis, for example) where the protagonists take stock. At this point, the abuser may attempt to improve the situation by counselling. He may then learn to control his anger and violent behaviour rather than attempting to control the partner. Such an attitude is highly likely to lead to subsequent non-violent behaviour.

5. Leaving the system

before : in marital home

after : left marital home

This is often decided upon some time after the critical incident. It is usually after the boundaries of the system have been broken. The old system has been de-stabilized, and a new one is being established.

The involvement of a confidante (someone outside system) can be crucial. This may be a friend who discusses the beatings, or someone else who has suffered batterings. Publicity (from a television programme, for example) may help. When a programme about wife battering was shown during 1994 on New Zealand television, many hundreds of calls were received from people who felt they were in a similar situation. After this initial lever, the battered person may become aware of other sources of information outside the system.

An opportunity to set up home is needed if the break is to be permanent. Unless there is an alternative system to go to then the break is unlikely to occur since there is still strong pressure from the abuser to maintain the violent one. Also, a belief by the female that the maintenance of the marital home is paramount will predispose her to return. She needs to have changed her overall goals in order to resist this.

6. Resolution

before : away from marital home but only short term aims

after : either returns to marital home or makes new life

Obviously, to be viable, the new system outside the relationship has to be stable. There may be violence and attempts at coercion which have to be dealt with.

If the spouse returns to the marital home then it may be under new rules. If there is some restructuring of the relationship (for example, the couple go to counselling together or the woman finds outside interests) then there may be a possibility of a new stable non-violent relationship. Alternatively, there may be a return home under the old regime (perhaps after a 'honeymoon' period). This last possibility often has the same effect as compliance with violence.

Whatever is decided, the wife is in shelter at this point and the negotiations can be carried out under more equal terms than when she is at home. Various factors make wives vulnerable to pleas from their husband: negative feedback from others of leaving the marital home, criticism, prejudice or little social support.

In order to simplify the design of a domestic violence simulation various assumptions need to be made. As already noted, it would be impossible to present a totally realistic picture of every feasible kind of situation. Instead, we provide a plausible structure with a limited amount of choice and variation. One simplification is to work within the confines of the above six point framework, although other possibilities have been observed. Violence may occur before the relationship is fully established, for example. In a minority of cases, it is perpetrated by the female upon the male. Also, homosexual relationships have not been covered. The aim is to consider the most likely scenarios for most commonly occurring relationships.

For each event, we can define sub-events. For example, in the third stage of the relationship we can postulate the following sequence occurring over and over again:

Tension building

There could be incidents over a short or long period of time. Tension often builds up over specific kinds of problem. These are, in order of occurrence, housekeeping (cooking, cleaning, repairing and so on), sex, social, money and children. These may appear to be superficial problems

(although they may not always be). Overall, the problem can be seen as one of who is in control, and this is what causes the tension. Stress levels also increase with change in circumstances: birth of a baby, change of home, change or loss of job.

Verbal foreplay

A critical incident

A specific occurrence may be arrival home drunk, burnt meal, etc.

A violent act

At the point where the violent action might occur the male may be able to control violence by walking away or by talking. The aim is to control emotions and violent behaviour, but to avoid controlling the woman.

The response

Crying out by the woman often aggravates the assault (screaming with pain or protesting, for example). Silence may stop the attack or, at least reduce its severity.

Withdrawal

Contrition

Forgiveness

Forgiveness or a 'loving stage' may be a perception of the female but not of the male. The man may be attempting to relieve his guilt.

This sequence can be broken at various points and alternative routes taken. For example, we may have a critical incident that produces an event of peak tension that has to be relieved in some way. This can be achieved by hitting the spouse, walking away or by talking. All would be possible within the domain constraints. Of these, walking away and talking might be recommended if the task was to reduce tension in a positive way.

The aftermath of a violent act is often that the husband carries on as if nothing has happened, or goes out. If there is any remorse or contrition by the male it is more likely to occur at an early stage of the relationship.

Men generally believe their action is provoked. Women often feel ashamed and do not want neighbours, friends or workmates to know. They often apologize to their husband.

C.3 Profiles

Profiles may be generated from the literature. Some aid can be obtained by using stock stereotypes and generating information from those. This tallies with Rich's work on stereotypes (Rich, 1979; Rich, 1989) where she assumes that various characteristics tend to be clustered together in people's personalities. For example, an aggressive person may tend to be egotistical and extrovert, whereas a meek person may be self-effacing and introspective.

Investigation of personality types has a long history in psychology (Kalat, 1990). A central concept for many researchers is that of the *trait*. Bootzin *et al.* (1986) define it as a predisposition to respond in a consistent way to many different situations. Allport and Odbert (1936) list almost 18,000 words that could be used to describe personality traits. Since then, Allport and others have tried to classify them in various ways. He distinguishes common traits (such as aggression) which are present to a degree in everyone, and individual ones or *personal dispositions* (Allport, 1961). Raymond Cattell (1964) has used factor analysis to study personality traits, which he divides into surface (clusters of behaviour that tend to go together) and source traits (the underlying causes of these clusters). Other researchers such as Eysenck (1970) and McCrae and Costa (1987) have used factor analysis to identify dimensions along which traits of personality can be measured.

C.3.1 Police officers

Common stereotypes for police officers have been noted by D. Burns (personal communication). These are depicted in Figure C.3, and typify attitudes. Figure C.3a shows the *problem solver*. S/he is thoughtful and professional but has a tendency to be an idealist and impractical. The *uniform carrier* (Figure C.3b) only does as much work as necessary, sidesteps problems where possible and looks forward to payday. Figure C.3c shows the *crime fighter* who takes the job very seriously, aiming to enforce the law as s/he sees it and preoccupied with catching criminals. The *negotiator* (Figure C.3d), perhaps the ideal for police officers to aim for, has a positive attitude to policing, is familiar with the law, but at the same time can relate to members of the public and can take account of individual circumstances in making sensible decisions. Personal qualities for these stereotypes are listed in Figure C.4.



(a) The problem solver



(b) The uniform carrier



(c) The crime fighter



(d) The negotiator

Figure C.3 Stereotypes of Police Officers

problem solver	uniform carrier	crime fighter	negotiator
idealistic intelligent conscientious thoughtful deliberate unrealistic independent reliable patient unemotional phlegmatic honest introspective painstaking curious slow	cynical rigid apathetic slipshod undisciplined unreliable subjective dissatisfied extrapunitive wily lethargic genial smug uninquiring placatory judgmental	adventurous surly hostile punitive suspicious persevering impulsive subjective excitable impatient loyal assertive headstrong tough mulish intuitive energetic self-confident authoritarian	cooperative open adaptable optimistic conscientious reflective flexible reliable patient honest calm sensitive gregarious understanding self-confident alert placatory non-judgmental

Figure C.4 Personal qualities of different types of police officers

Corresponding to individual qualities or groups of qualities, general and specific goals can be postulated. From these goals and plans, specific actions can be predicted. Some examples for the uniform carrier are given in Figure C.5.

life theme component	general goal	plans	actions
placatory genial wily uninquiring	keep a low profile	avoid making an arrest	persuade wife not to make complaint
subjective judgmental wily	follow own judgment, bend rules as necessary	take husband's side, pretend to be sympathetic to wife	be conciliatory to husband, calm down wife

Figure C.5 Characteristics, aims and actions for uniform carrier

Each of the lifestyle components for police officers can be itemized and corresponding goals, plans and action lists constructed. Some of these for the crimefighter are listed in Figure C.6. As can be seen, the crimefighter is not, in general, interested in domestic

disputes since s/he regards these as trivial compared with his/her real mission in life, which is to get crime off the streets.

Individual beliefs (which may well depend upon personality) can be itemised for police officers. Some of these, and the repercussions, are itemised in Figure C.7.

life theme component	specific feature	goal	plan	action
ambitions	to rid streets of criminals, to be feared by law breakers	to spend as much time as possible hunting down criminals	to resolve domestic dispute quickly	brusque and dismissive attitude to protagonists
lifestyle	fitness enthusiast Spartan			
political attitude	right wing			
approval	of peers			

Figure C.6 Life theme components, goals, plans and actions for the crime fighter

C.3.2 The Batterer

According to much of the literature and many workers in the area (Kemp and Carter, 1995), consistent personality traits and patterns of behaviour are shared by the vast majority of battering males. They are often in their late 20s or early 30s, have been socialized in a patriarchal system, have witnessed or suffered from battering themselves as a child and have a low self-image.

A typical life theme profile built up from the information in the report by Kemp and Carter is shown in Figure C.8. From individual and groups of these traits, corresponding goals, plans and actions can be surmised. Examples are shown in Figure C.9.

Many beliefs are often shared by batterers, which again affect their attitudes, goals and actions. Some of these, together with reasons, are depicted in Figure C.10.

belief	reason for belief	goal	actions
if women misbehave they deserve to be punished	patriarchal upbringing		avoid arresting husband for violence if officer believes justified
alcohol is major cause of domestic violence	husband is often drunk when called to domestic dispute		tell wife to allow husband to sleep off drink and to try to get him to cut down on drinking
nagging wife brings violence upon herself	hearsay, own experience and reports from male friends		tell wife to stop harassing husband
men should be in control in home	own experience, patriarchal upbringing	maintain control structure in home	tell wife to obey husband
domestic incidents are unimportant	hearsay, attitude of peers and superiors	avoid domestics, treat lightly	deal with domestic situations quickly and superficially
women are free to leave marital home	lack of consideration of issues involved		treat wife with disdain
arresting batterer has no effect	own experience, lack of support from superiors and courts	avoid arresting batterer	try to get protagonists to make peace
violence is an acceptable way to resolve some problems	own experience, societal influences		tolerate reported violence of man against wife
main cause of battering is unemployment	own observation		tell couple that it is not a problem that police can help with
woman has exaggerated incident	police often arrive after things have quietened down		be sceptical about woman's account
domestic disputes should be dealt with by social services	limited view of police role	refer cases to social services	defuse situation and leave

Figure C.7 Typical beliefs, reasons, goals and actions for police officer

personal quality	moody, insecure, aggressive, suspicious, passive, self-pitying, possessive, withdrawn, changeable, hostile, dependent, jealous
ambition	to feel safe and secure, to feel in control
lifestyle	self-gratification, pleasure from indulgence, patriarchal, traditional view of sex roles
political attitude	
approval	to be respected (or feared) by friends and wife
physical sensations	getting drunk

Figure C.8 Typical life theme profile for batterer

life theme component	general goal	plan	actions
possessive, suspicious, aggressive, hostile	ensure no one else gets close to spouse	sever links between spouse and friends	stop spouse going out, stop friends coming to house, threaten friends of spouse
reticent	avoid communicating feelings to others		don't explain feelings to wife

Figure C.9 Examples of life theme, goal, plan and actions for batterer

Beliefs	Reasons	Goals	Actions
respect from others is important	lack of self-esteem	attempt to achieve respect from others	attempts to impress others
it is important to win arguments	thinks that winning arguments maintains superiority	aim to win arguments	wins arguments by any means possible, including force
domination is a good method of resolving conflict	has worked previously	attempt to dominate family	gets family to carry out orders
if others fear you then they respect you	equates fear with respect, desires respect to bolster self-image	attempt to instil fear in others	behaves in way that will make others fearful
if you feel slighted or undermined it is acceptable to hit out	low self-esteem makes husband feel threatened and liable to react violently		hit out if criticized
it is important to attain and maintain control of your environment	lack of self-control increases desire to control others	attempt to control family	vet's what everyone in family does
what you do in your home is your own affair	a desire to have a place where dominant, worried about outside criticism	achieve aims in home without constraint of outside influences	takes any action necessary to achieve aims in home
a wife may be regarded as personal property	a desire to have someone to control, possessive nature	control what wife does	keeps close watch on wife's movements
he has to be 'norm enforcer' acting in righteous and appropriate manner	belief that his version of normality has to be obeyed by all in the family	get wife to behave in what he believes is normal fashion	is highly critical of 'abnormal' behaviour
violence is an acceptable way of achieving ends	has learned through modelling that violence is appropriate way to deal with feelings	use violence if it will help achieve ends	is violent towards wife to achieve aims

Figure C.10 Some beliefs, reasons, goals and actions for batterer

C.3.3 The Battered Wife

Like the batterer, the spouse can be ascribed typical characteristics and beliefs. Often these are not dissimilar to those of the husband. She often has low self-esteem, is highly volatile and may be quite aggressive, although the aggression usually manifests itself verbally rather than physically. She may also play into the aggressor's hands by believing that he has a right to punish her, and by harbouring excessive feelings of guilt. A profile for a typical battered wife is shown in Figure C.11.

personal quality	voluble, timid, withdrawn, friendly, submissive, low self-esteem, self-effacing, excitable, self-critical, scared, complaining, critical
ambition	to have peaceful life, to be happy
lifestyle	isolation, pre-occupation with house and children
political attitude	
approval	to be loved by husband and children, to be respected by outsiders
physical sensations	

Figure C.11 Typical life theme profile for battered wife

The wife and batterer may also have some overlap of beliefs although these may change rather more significantly for the wife over a period of time (with age, experience and disillusionment). Her typical beliefs, reasons, goals and actions during the pre-marriage and the early stages of violence are shown in Figure C.12.

The wife often tolerates what may be regarded by others as an unendurable situation for several reasons. She may believe she can help the husband combat his problem, or she may be socially isolated, not having family or friends to talk to. Often she will be financially dependent upon the husband. Lastly, if she has had unsuccessful relationships before, she may be determined to ensure the current one does not fail.

Beliefs	Reasons	Goals	Actions
marriage (or bond with male) is a highly desirable state	upbringing, attitude of peers	get married or live with male, stay married	takes actions to get married and to maintain marriage
husband should be allowed to hit her in extenuating circumstances	learned from childhood that it is OK to show love by violence		accepts violence from husband
children are highly desirable	upbringing, attitude of peers	have children	persuades husband to have children
family and friends give good advice	credulous nature	listen to advice	carries out advice of family and friends
conflict is bad	natural desire for peaceful life	avoid conflict	attempts to avoid arguments, and mollify husband
men have a need to be admired	upbringing, societal attitudes	show admiration for men in whom you are interested	agrees with everything husband says
a violent man has had extenuating circumstances in the past that won't arise in the new relationship	credulous nature, belief in husband's account		ignores signs of violent behaviour
divorce is bad, children need 'stable' home with father, children from broken home carry stigma, children's schooling and friendships shouldn't be disrupted	attitudes of peers and family	avoid divorce	endures increasingly brutal treatment in order to maintain home and marriage
there is no realistic alternative to staying with husband	lack of knowledge of alternatives, doubt in ability to get along alone, fear of losing custody of children, economic reliance	stay with husband	does not consider leaving husband as option
situation is not province of police	experience, low self-esteem, husband's influence, shame	avoid involving police	does not call police if hit by husband
nagging at husband will get him to change ways	advice from friends	get husband to change	carps and criticizes husband when given opportunity

Figure C.12 Beliefs, reasons, goals and actions for battered wife

C.3.4 Other actors

Obviously, as many actors and profiles as desirable can be included, with a corresponding increase in complexity. As noted in Section 6.4.2, we can combine black box modelling with glass box modelling by considering various events as external to the model and not explainable within the system and here the choice is with the developer as to which actors are to be considered in each category. Others who might be relevant are the neighbours, children, parents, friends, social workers, judiciary and so on.

A few beliefs for some of these categories are included in Figure C.13 to show the kinds of opinions that may be held and used as a basis for a more intricate model.

family court judge	neighbours	husband's family	women's refuge worker
unless the injuries to the woman are serious there is no problem women who deviate from standard role of homemaker may be punished by husband women who challenge domination by husband may be punished by husband	what others do in their home is their business anyone who nags husband deserves to be hit wife enjoys being hit	wife is exaggerating accounts of violence violence is an acceptable way of getting wife to toe line wife should take better care of husband	priority must be given to stopping violence rather than educating batterers violence is used for power and control women are afraid to give details of violent incidents

Figure C.13 Beliefs for other possible actors in wife battering scheme

C.4 A Sample scenario

A sample scenario based on the material and observations of the previous sections will now be developed. It is taken from step 2 of the developing relationship as defined in Section C.2: the events that may lead up to the first violent incident.

The first scene occurs at the marital home when the husband is out for the evening. The wife receives a phone call (an external event as depicted in Figure 6.7). It is from a girlfriend from whom she has not heard for a long time. The girlfriend says that a group of them are going out the following Thursday for a meal and asks if she would like to join them. The wife has no other commitments for the night in question and it is an evening when the husband nearly always stays in.

The main question arising from this event is: does the wife want to go out? If, from her character description, she enjoys company then the answer is likely to be yes. This may not be her response, however. Possible responses with reasons are shown in Figure C.14.

Response	Reason	Underlying belief or life theme
Definitely	Decides to go even though husband is likely to object	Defiant personality
Probably	Will discuss matter with husband since he likes to know what is going on	Best to discuss plans before finalising decisions
Possibly	Will ask husband, but unlikely to agree	Husband has final decision on movements
No	Husband will not agree	Pessimistic and knows husband will not agree

Figure C.14 Responses to initial request to go out

Obviously, choice points will occur at various times during the developing story. It is important to realise that these will not be hardwired in via some algorithmic framework but are driven by conditions that exist or are put in place, actions that may occur as a result of conditions, and changes in conditions as a result of actions. Rather than enumerate all of these possible alternatives we shall concentrate on one path through the system, highlighting reasons why the specific subevents occur.

To continue the story: the scene now moves to a discussion between the husband and wife of the friend's telephone call. At this stage, a condition will have been put in place from the phone call, noting the state of things as far as the wife is concerned. Dependent upon this, she may make various choices. She may decide not to tell the husband about the phone call, for whatever reason (for example, because she is scared of what he might say or do) or might discuss the phone call. The nature of any discussion that ensues will depend upon her previous action. We consider the case when she has not made a final decision but is going to discuss the matter with her husband.

The husband does not want her to go out since he will be at home that evening and says he likes her company. At this stage she has the choice of accepting his decision, refusing to do so, or appearing to accept but deciding to go out anyway.

Reasons for the various arguments and counter-arguments can be generated from the tables of beliefs and attitudes. The wife is sociable and wishes to go out to enjoy herself. She, therefore, formulates the plan of arranging to go out with friends. Part of this plan involves seeking the agreement of the husband since she has been conditioned to believe that he is head of the household and therefore should be consulted.

The husband's attitude to the suggestion is very negative. This arises for several reasons. His jealous, possessive and suspicious nature leads him to believe that the wife cannot be trusted to go out without him. His weak nature pre-disposes him to attempt to exert control whenever possible and this is one case where he believes he can maintain his authority and have an impact on events. When the wife refuses to accept his 'ruling' he becomes more suspicious and demands to look in her handbag for incriminating evidence. Again, this is motivated by the desire to display authority.

She may not comply with this request, or accede. In either case, the husband may grab the handbag and look inside. This is a further action in his goal of maintaining domination over the wife. A slanging match ensues and both parties retreat in a bad mood. Later, he comes to apologize and admits he gets jealous but that is because he loves her. She accepts this since she has a need to be loved. He then appeals to her maternal nature by attempting to evoke sympathy. He says that he gets depressed being left in the house alone and this stems from memories of childhood when his parents left him at home alone. She accepts this argument and feels sorry for the husband.

In the next scene, he states that he intends to go out on the night in question anyway since it is a friend's birthday and there is a celebration at the local bar. Her sense of fairness leads her to demand the right to go out on the same night. His excuse of not wanting to be home alone no longer applies but he still seeks to assert his authority. He equates strength of character with decisiveness and rigidity. He refuses. She sees no future in arguing.

On the night when she was due to go out, the husband goes out as arranged. The wife also decides to go out, believing that she can return before the husband and that he will not know.

In the next scene of this melodrama the wife returns home to discover that the husband is already back. He is furious that his authority has been undermined. She is defensive, believing that she has broken the rules by disobeying her husband. Her apologies for her actions just consolidate his belief that he has been badly treated. He needs to step up the punishment level and, having a violent character, he smashes a favourite vase of

hers. She feels the need to reciprocate the punishment. She is physically weaker and so uses verbal means, taunting him about his failure to keep jobs and his general fecklessness. These jibes hit home and he becomes angrier and angrier. He is not articulate and so seeks an alternative means to break the increasing tension. This he does by slapping the wife across the face. From his point of view, this act has a successful outcome. The woman withdraws in distress and he becomes melancholic, gratifying his tendency towards self-pity.

In the last scene, the husband returns home from work the following evening. The wife is still distressed but he is contrite. He has bought her flowers and apologizes for his action of the previous night. He dismisses it as a momentary lapse due to excessive jealousy and begs forgiveness. Both find this action acceptable. The husband still feels in control since he has brought a peace offering and is dictating events. She is flattered by his unusual attentiveness and accepts his apology without reservation.

C.5 Summary

The beliefs, attitudes, goals, plans and actions of participants in a domestic violence scenario have been described. It has been shown how an analysis of this information can be used as the basis of a computerized system driven by these factors. This approach enables feedback to be incorporated in a direct and helpful way.

Appendix D

Administration of Wizard of Oz Experiment and Results

D.1 Introduction

In this appendix, the way that the Wizard of Oz experiment was set up is described, and the results from the experiment are summarized. The basic goals were to see how role playing might work in a computerized environment, whether a plausible interaction is possible and whether subjects can benefit from taking part in such an exercise.

D.2 Experimental set-up

First, the style of interaction had to be determined. Ideally, the subject should interact with the computer in as natural way as possible. Arguably, this might be achieved by conversing in natural language, either through written text or by spoken interaction. This form could quite easily have been incorporated into the Wizard of Oz experiment, since the computer responses can be generated dynamically by a human. This possibility was rejected, however, since any person who was at all computer literate would realize that such a style of intercourse is beyond the technology currently available. This could affect the credibility of the experiment and invalidate the results. Also, if textual input was used, the Wizard would need to be a very fast, accurate typist and have extremely quick reactions to be able to respond in a reasonable time frame.

The alternative chosen was to set up situations and to provide multiple choices to be selected from by the user. This could be augmented by allowing the user to include remarks and to ask for advice.

Most of the text for setting out scenarios and presenting alternatives had to be generated prior to the experiment to maintain a reasonable response time. These were created by using the principles and scenarios described in Chapters 6 and 7, and in Appendix C. In particular, the extra information that could be requested by subjects to aid them in their decision making was based on the information that would be used in an event-driven simulation for determining what might happen in practice.

If all such information could be anticipated and every response by the user catered for then, of course, a computer program could be written and the Wizard would not be needed. However, in the kinds of situation being simulated it is important that all details are correct. As noted in Chapter 7, even a small deviation from reality could reduce the subject's faith in such a system. If two people are in separate rooms and the computer says that one of them hits the other then the user is going to think this is rather odd. The main purpose of the Wizard is to ensure that small but significant details such as this remain consistent. Also, if the interaction takes an unanticipated turn then the Wizard has to be able to cope with this possibility.

In view of the above considerations, it was decided to store much of the descriptive text and questions in a word processor file, in this case, Microsoft® Word. If necessary, changes would have to be made to the file during the session to maintain consistency. In addition, commonly occurring messages were included such as 'Thinking, please wait . . .' and 'This is not an allowable alternative, please choose a number in the given range'.

A method for communicating between terminals was needed. Talk© (Lewis, 1993) was readily available and has all the facilities required for a simple simulation. Two way interaction between subject and Wizard is achieved in Talk by using a split screen as shown in Figure D.1 (reproduced from Figure 7.14 for convenience). Text could be copied from a Word file and transmitted to the subject via the output pane. The transmission rate was slow (about 30 characters per second) but this was actually advantageous since it inured the subjects to the slow response rate for question answering. If, for any reason, text had to be generated dynamically it was first typed into a Word file, selected, copied and then transmitted as a block to assure uniformity of presentation rate.

In preliminary experiments it was found that even the time taken to select blocks of text using a mouse slowed the process down significantly and was error-prone. To improve this, the blocks of text were stored in Word table cells that could be selected by a single operation (option-click).

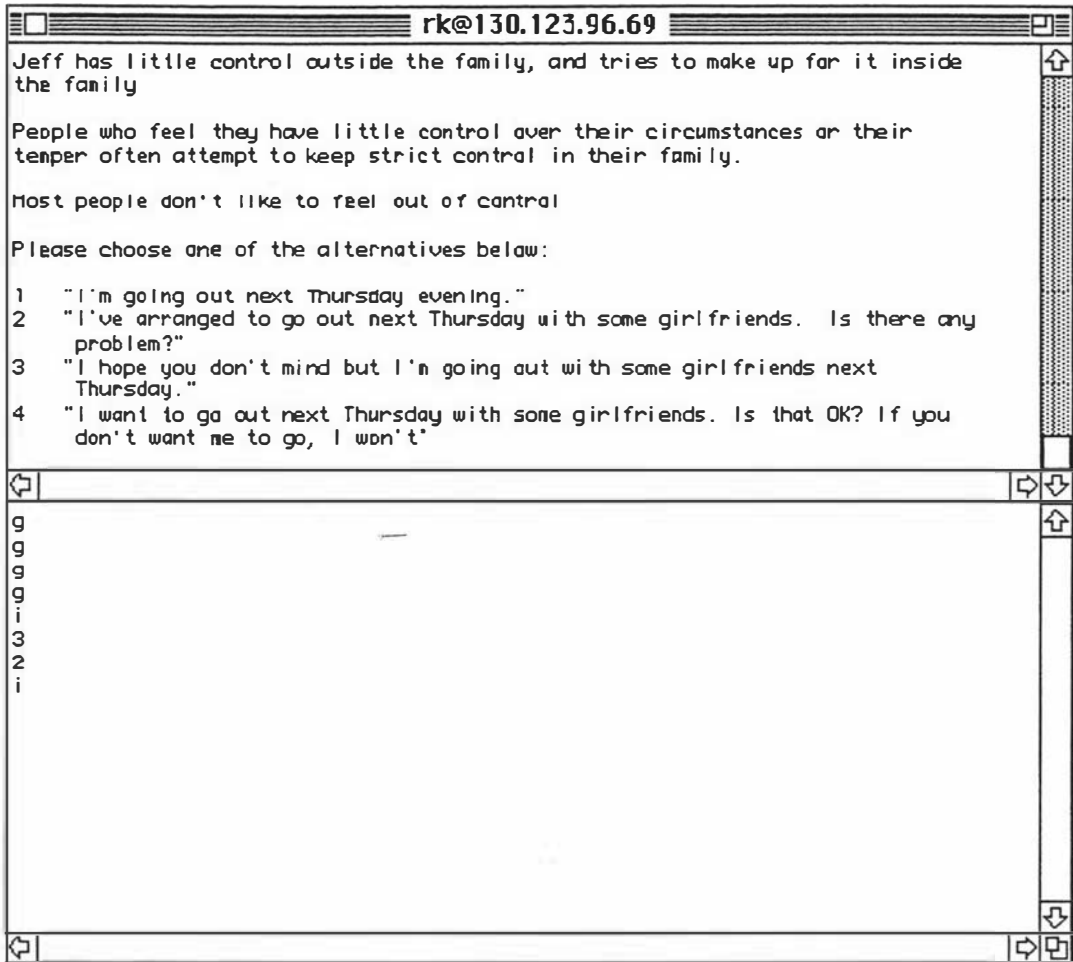


Figure D.1 Screen from Wizard of Oz session

It was considered crucial for the Wizard to know what the subject was seeing and when, so that s/he could fully appreciate the significance of each response and could estimate when it was likely to occur. This second point was particularly important in view of the slow transmission rate. One way of achieving this level of observation would be to physically link up two monitors to the same computer. This possibility was rejected on practical grounds since the Wizard and subject may be large distances apart, and so trailing a long wire between two rooms would be inconvenient. Instead, a third computer was included in the system. This was linked to the Wizard's main machine via Talk and its monitor was placed on the same desk, so the Wizard could see its screen. Responses could be transmitted first to this computer and then, immediately afterwards, to the subject's. Thus the Wizard could see what was being presented to the subject and estimate the rate at which the user would see it. The configuration is depicted in Figure D.2.

This scheme was found to have an additional benefit. No one is infallible, not even a Wizard, so if it became clear that the wrong piece of text had been transmitted to his/her

other computer then the Wizard could quickly rectify the situation before transmitting to the subject. In effect, the extra computer provided a preview mechanism that reduced the likelihood of error.

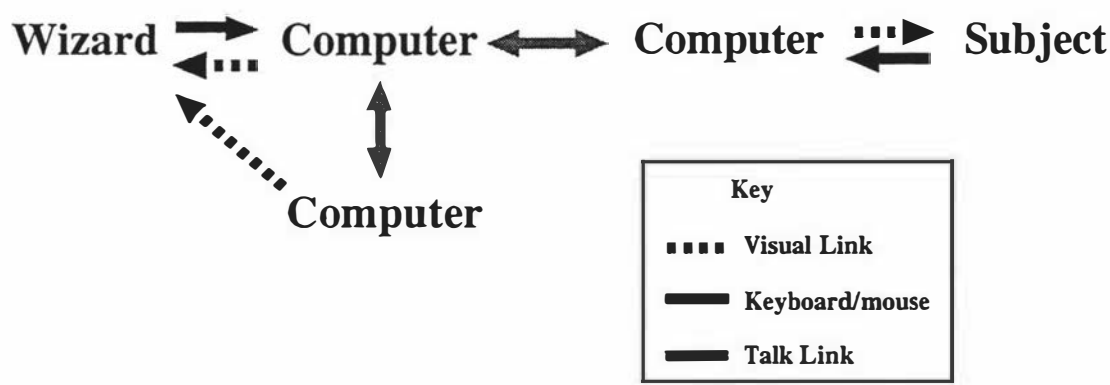


Figure D.2 Wizard of Oz configuration

In view of the inbuilt blocks to speedy interaction it was important to streamline the rest of the system as much as possible. A Power Macintosh 7100/66AV was used by the Wizard, being the fastest machine available for the purpose. Three windows had to be available on the screen: Talk windows to the two other computers, and a Word file window. To ensure each window could be of an appropriate size a 16" screen was found to be adequate.

D.3 Sessions, questionnaire and results

In this preliminary study there were 14 subjects reflecting a diverse range of backgrounds and experiences. Predominantly, they were female (two males were included for comparison purposes). There was a wide age range: 5 subjects were aged between 20 and 25, 3 between 25 and 40, and 6 subjects between 40 and 50. Lastly, the academic backgrounds were also varied: 5 subjects had no higher qualifications, 7 had a first degree, and 2 had a higher degree. Subjects with degrees had obtained them in various disciplines including English, Information Systems, Philosophy, Computer Science, Soil Science and Sociology. All but two of the users believed that they were interacting with a computer program. Even the most computer literate subjects had no inkling that a human was providing the responses.

The experiments went surprisingly smoothly (apart from one occasion when a change in network configuration led to a system crash in the middle of a session). The Wizard was installed two rooms away from the subject. An assistant was available in the same room as the subject in case of problems but did not watch the session, in case this

inhibited the user. The assistant was only very occasionally needed when the 'program' responded erratically (for example, during one experiment the Wizard lost track of where the user was in the session). Sessions lasted anything between 15 and 35 minutes depending upon the subjects' choices, the amount of information they asked for, and the extra remarks they included.

Preliminary screens explain how to use the system and give appropriate examples. The initial screens presented to the user are shown in Figures D.3 to D.5. Note that, in practice the amount that could be presented to the user at any one time was limited by the size of the Talk window panes, and, for convenience the contents of several screens have been condensed into these figures.

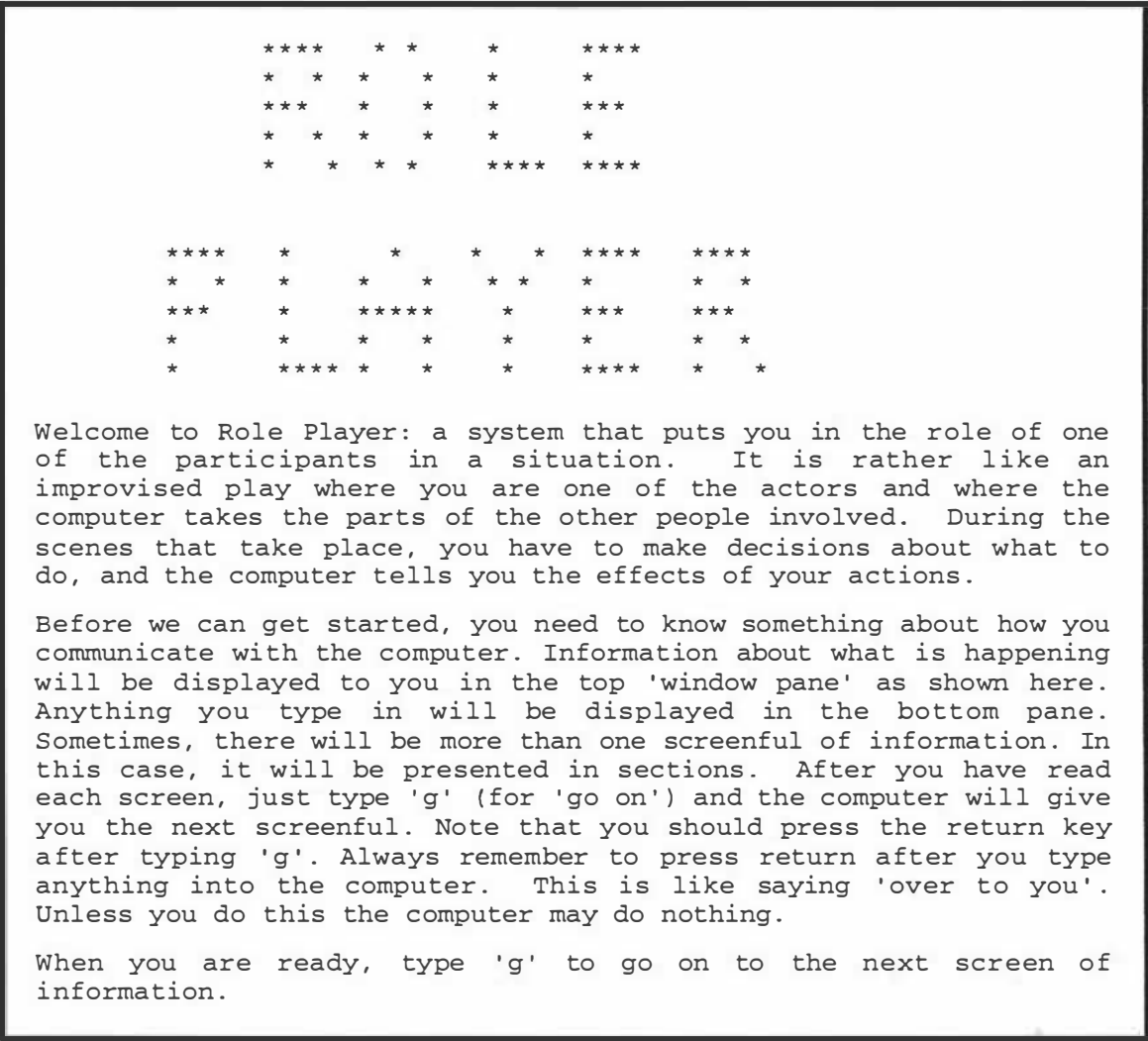


Figure D.3 Initial role playing system screens (1)

At various times, you will be able to make comments, ask questions or request further information.

If you are asked a question where alternative responses are listed then type in the number corresponding to your answer. For example, given the question:

Do you prefer

- 1 cornflakes
- 2 weetbix
- 3 coco-pops
- 4 none of these

You might select 3 to tell the computer you like coco-pops best. However, you must remember to press return after your answer.

At any stage, you can type in the letter 'i' which tells the computer you want background information to help you make your decision. If, for example, you pressed 'i' instead of 1, 2, 3 or 4 then the computer might respond:

cornflakes are crunchy but tasteless
 weetbix get soggy in milk
 coco-pops are very sweet

You would then be presented with the original question and again asked to select one of 1 to 4. If you still wanted further guidance then you could type 'i' a second time, to ask for more information.

This time the computer might respond:

cereals are good for you

You can keep pressing 'i' as many times as you like. Eventually, the computer will run out of answers and reply:

Sorry. I don't know any more about this subject

As a last alternative, you can type an 'r' allowing you to make a remark of your own. The computer responds with: 'please enter your remarks'. You then type in your comments and finish by pressing return. So, in the above dialogue, after getting some background information, you might observe:

I don't think cereals are always good for you. Particularly this load of junk.

The computer won't be able to understand what you say, but the information will be stored away and will help us to improve the system and give us more insight into what you are thinking.

In summary, at various stages you can:

- press a number (to select an option)
- press 'i' to get information
- press 'r' to make a remark
- press 'g' to go on to the next screen of information.

Figure D.4 Initial role playing system screens (2)

WARNING! This is an early version of the system and so may contain errors. If the computer appears to take an illogical step or ask an irrelevant question then just deal with it as best you can. You can make a note of the fact later in the debriefing questionnaire.

If the system seems to have crashed then you may need help to get it started again. Sometimes the computer will be very slooooow to respond, particularly if you make unusual choices. Just try to be patient.

OK. Time to make a start. Remember, you should put yourself in the position of the person you are playing, perhaps reflecting upon what their circumstances might be rather than just giving your own 'knee-jerk' reaction. The purpose of the experiment is to increase your understanding of the situation that is being depicted: the personalities of the people involved, their goals, their actions and any consequences of decisions. In order to get the most out of the exercise you should try to understand what motivates each of the participants. The computer can help by providing background information on what people are trying to achieve and why. Use the 'i' option freely as you work through the scenes.

Sometimes you may be asked to explain your decisions. Please, just type in an answer. These cannot be understood by the computer but the information will be useful for future reference.

Figure D.5 Initial role playing system screens (3)

Output from the main part of a session with a female, post-graduate sociology student is shown in Figures D.6 to D.12. Subject responses are in bold. Typically, as can be judged from this example, subjects became highly absorbed in the interaction. Note that this particular subject, possibly because of her academic background, was able to make useful suggestions of ways in which the system could be amended to introduce further alternatives that Anne could try.

Your name is Anne. You feel trapped in a marriage that has gone sour. You have two children, Jane and David, who are 4 and 2 years old respectively. Jeff, your husband, has a demanding job but you, yourself, have been unable to find any suitable part time employment. Anyway, Jeff isn't keen on you getting a job. You are expected to keep the house tidy, provide meals and look after the children. You have no enthusiasm for the role of housewife and would rather be out working or enjoying yourself.

Jeff is worried about being made redundant and, as a result, has felt very stressed recently. He complains about your meals, the state of the house, and the way the children are dressed. He goes out drinking regularly but doesn't like you to go out. Invariably you end up watching television and painting watercolours which is the one creative pleasure you still have. Your only real contact with other people is with your mother who you generally see a couple of times a week. Jeff has recently started complaining about this and suggests you see her too often.

Jeff is not a heavy drinker although he has been drinking more frequently recently. However, when he has had a few whiskies, he gets aggressive and critical. He has never hit you (he feels strongly that women should be respected and protected) but he is verbally very abusive.

One evening, when Jeff is out, you get a phone call . . .

Kay: "Hi, Anne. It's Kay, here. How are you? Haven't seen you around for ages"

You: "Hello, Kay. It's nice to hear from you. No, I haven't been out much recently. Don't seem to have the energy in the evenings after looking after the house and kids all day"

Kay: "You should take a break. Listen, me and some of the other girls are going out next Thursday to celebrate my new job. How do you fancy coming? We'll go for a few drinks and then perhaps go on to that new nightclub in town".

Choose your response.

- 1 "OK. I'd really enjoy that. See you next Thursday"
- 2 "Yes. I'd really like to go, but I'll just have to check with Jeff to make sure he hasn't got any other plans. If I don't phone, assume I'm coming"
- 3 "Maybe. I'll have to make sure there is no problem with Jeff. I'll get back to you"
- 4 "Sorry, I can't go next Thursday. Jeff stays in that night and he likes company".

Select your option (remember you can choose 'i' if you want more background information).

i

Figure D.6 Sample role playing session (1)

You like to go out.

Jeff likes you to ask him before making decisions.

Jeff doesn't like you to go out without him.

1 "OK. I'd really enjoy that. See you next Thursday"

2 "Yes. I'd really like to go, but I'll just have to check with Jeff to make sure he hasn't got any other plans. If I don't phone, assume I'm coming"

3 "Maybe. I'll have to make sure there is no problem with Jeff. I'll get back to you"

4 "Sorry, I can't go next Thursday. Jeff stays in that night and he likes company".

2

When are you going to discuss the matter with Jeff?

1 As soon as possible.

2 When he appears to be in a good mood.

3 On the night of the outing.

i

Jeff doesn't like surprises

He can be quite amenable when in a good mood

He often changes his mind

When are you going to discuss the matter with Jeff?

1 As soon as possible.

2 When he appears to be in a good mood.

3 On the night of the outing.

Jeff says "There is no way you are going out next Thursday. You know it's the night I like to stay in with you so we can be together".

2

How do you react? Do you

1 try to badger him into agreeing?

2 explain quietly that you rarely go out but are keen to see friends occasionally?

3 tell him how much you love him, but would really like the chance to see some other friends if possible?

4 give up and decide not to go?

5 leave the room and sulk, determined to go anyway?

i

Figure D.7 Sample role playing session (2)

Jeff doesn't like to feel he's been bullied or cajoled into a decision.

you like to get your own way and usually try to do so by being persistent.

How do you react? Do you

- 1 try to badger him into agreeing?
- 2 explain quietly that you rarely go out but are keen to see friends occasionally?
- 3 tell him how much you love him, but would really like the chance to see some other friends if possible?
- 4 give up and decide not to go?
- 5 leave the room and sulk, determined to go anyway?

2

Jeff interrogates you to make sure that you are only going out with females. He then agrees, saying:

"OK but be sure to be back before midnight, Cinderella!"

You are concerned that Jeff might change his mind about letting you go out so, a few days later, you raise the subject again and he gets suspicious. He demands to look through your handbag.

You respond by:

- 1 Letting him look.
- 2 Saying that you don't want him to look but if it will relieve his suspicions then he can.
- 3 Explaining that your own possessions are private.
- 4 Telling him to go to hell.

i

Your request to go out has made him generally suspicious

You have nothing incriminating in your handbag

do you think he should be allowed to examine your private possessions?

You respond by:

- 1 Letting him look.
- 2 Saying that you don't want him to look but if it will relieve his suspicions then he can.
- 3 Explaining that your own possessions are private.
- 4 Telling him to go to hell.

2

Figure D.8 Sample role playing session (3)

You give him the handbag and he looks through it

- 1 You feel cheapened by this exhibition and tell him never to try it again.
- 2 You are frightened by his behaviour and decide to call off your outing.
- 3 You sulk, and feel more determined than ever to go out.
- 4 You accept his right to look through your belongings - you have nothing to hide.

1

Later on, Jeff comes to apologize for his actions. He says he realizes that he is sometimes unreasonable but this is because he is so fond of you and gets very jealous. He tells you how his mother ignored him and used to bring other men to the house while his father was away.

- 1 You feel sorry for him and forgive him. He obviously has problems.
- 2 You tell him that you understand his problems but that he just can't behave in this way.
- 3 You tell him he is just making pathetic excuses for inexcusable behaviour.
- 4 You tell him you've heard it all before and he is just a weak-kneed wimp.

2

On the following Thursday, the day of the proposed outing, Jeff comes home and says that it is his mate's birthday and he's decided to go out that evening. Since the children have to be looked after, you won't be able to go out.

- 1 You accept his decision.
- 2 You tell him he's a louse and go away to sulk.
- 3 You tell him you'll get your mother to babysit and go out anyway.

i

Basically he's trying to stop you going out

He wants to exert control over all your actions and to isolate you from others

He believes it's important to maintain control in his own house

- 1 You accept his decision.
- 2 You tell him he's a louse and go away to sulk.
- 3 You tell him you'll get your mother to babysit and go out anyway.

3

Figure D.9 Sample role playing session (4)

You go out for the evening and have a good time, so good a time, in fact, that you don't arrive home until 2 a.m. You hope Jeff will have gone to bed but he is waiting for you. He's had a few drinks and is really mad. He wants to know exactly where you've been, who with, and what you've been doing.

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

i

He's not likely to believe you whatever you say
His pride has been hurt that you have disobeyed him.
He wants to punish you.

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

r

Please enter your comments, terminated by 'return'

I want to take an option between the first and second. I am aware of his need to control me and therefore want to passify him by letting him know that I have done nothing wrong rather than exacerbating the situation. But at the same time I want to retain my dignity and some control over the situation. I want to tell him that he has no right to demand to know my every move. We are equals in this relationship and as such should be treated as equals. At the momment he is treating me like a child. I do not accept that.

your remarks have been noted. please continue with the session

- 1 You give him all the details.
- 2 You tell him it's none of his business.
- 3 You embellish the story to make him jealous.

1

He gets madder and starts throwing things around the room, accusing you of being a whore.

- 1 You get out of the room and barricade yourself in the bedroom.
- 2 You try to appease him by telling him you won't do it again.
- 3 You tell him to grow up and stop behaving like a little boy.

i

Sorry. I don't know any more about this subject.

Figure D.10 Sample role playing session (5)

- 1 You get out of the room and barricade yourself in the bedroom.
- 2 You try to appease him by telling him you won't do it again.
- 3 You tell him to grow up and stop behaving like a little boy.

1

He pushes the door to the bedroom open to get in.

He is still not satisfied and tells you that not only will you not be allowed to go out again but that you will stop seeing your mother. He gives you a push to emphasize his physical superiority.

- 1 You push him back.
- 2 You lie on the bed and start crying.
- 3 You tell him to stop it or you'll call the police

r

Please enter your comments, terminated by 'return'

Again - I am in a position of wanting to take two lines of action. On one level I am aware that in order to diffuse the situation right now I need to take the least 'provocative' action - i.e. - 2. But at the same time I am aware that if I let him take this level of control over me at this point I may not be able to regain control over my own life when the current situation has been diffused. With this in mind I want to ring the police to let him know that there are limits to what he can do. In order to make this decision I need to know more about what has happened in the past - have I tried conciliatory moves before and he has just continued to push harder or have I made a stand and he's backed down at this point?

your remarks have been noted. please continue with the session

- 1 You push him back.
- 2 You lie on the bed and start crying.
- 3 You tell him to stop it or you'll call the police

i

He can't subdue you verbally and so seeks to do so physically.

He was brought up in a violent family and believes that it is a legitimate way of dealing with a situation.

- 1 You push him back.
- 2 You lie on the bed and start crying.
- 3 You tell him to stop it or you'll call the police

2

He feels threatened and gets even more furious. He slaps you hard across the face and then goes out and locks himself in the spare room. You collapse exhausted in bed.

Figure D.11 Sample role playing session (6)

Next day, he's gone to work when you get up. You still feel stunned. This is the first time he's ever hit you.

- 1 You try to forget what has happened and just get on with your jobs.
- 2 You take the kids and go to your mothers. You are not going back until something has been sorted out.
- 3 You decide you are going to discuss the matter seriously with Jeff when he gets in.
- 4 You feel embarrassed and slightly guilty about the whole affair. You intend to be reserved when he gets back to show that you're hurt but secretly you blame yourself.

2

He tracks you down at your mothers. He has a lovely bunch of flowers and presents them to you, saying how sorry he is and asking you to forgive him.

- 1 You forgive him, and tell him you were partly to blame.
- 2 You forgive him but tell him in no uncertain terms that you are leaving him if he hits you again.
- 3 You throw the flowers in the trash and tell him to get lost.

i

Sorry. I don't know any more about this subject.

r

Please enter your comments, terminated by 'return'

Again I would take a different option to those offered. I would not go back to him on his promise. Having done it once and given his family background I do not beleive that he will not hit me again - I would raise the possibility of counselling but refuse to return to him until we had undergone several sessions. I would also ask him to join a group like Men Against Violence so that he could explore his own violent tendencies. I would not place myself in the situation of possible violence again. But I also beleive that people can change and as such would not cut him short without the possibility of exploring how far he was prepared to go to make a change.

your remarks have been noted. please continue with the session

- 1 You forgive him, and tell him you were partly to blame.
- 2 You forgive him but tell him in no uncertain terms that you are leaving him if he hits you again.
- 3 You throw the flowers in the trash and tell him to get lost.

2

The crisis has led to a resolution, but this is only temporary. The underlying problems are still present. The only thing that has changed is that there is now no psychological barrier to the use of violence. It has been used and has been perceived by Jeff (and possibly by you as well) as a means of providing some respite.

THE END

Figure D.12 Sample role playing session (7)

Debriefing the subjects after using the system is obviously crucial. In order to ensure the kinds of question and the answer format were appropriate, the format and questions were adapted from the 'Software Usability Measurement Inventory' produced by the Human Factors Research Group at University College, Cork (HFRG, 1993). A copy of the questionnaire is shown in Figure D.13.

Role Playing System Questionnaire			
<p>This questionnaire contains 6 statements. Against each one there are 3 boxes. You should mark the first box if you generally AGREE with the statement. Mark the central box if you are UNDECIDED or if you can't make up your mind. Mark the right box if you generally DISAGREE with the statement.</p> <p>In marking the left or right box you are not necessarily indicating strong agreement or disagreement but just your general feeling most of the time.</p>			
	agree	undecided	disagree
	↓	↓	↓
1. The system is easy to interact with.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. It is easy to understand what to do.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. The depiction of situations is realistic.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. The computer responses are realistic.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. The 'information' option is useful.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. Using the system increased my understanding of domestic disputes.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Add any further comments below:			

Figure D.13 Copy of the role playing system questionnaire

Results from the survey were very encouraging. An analysis is given in Figure D.14. As can be seen, subjects generally found the system easy to use and beneficial to their understanding of domestic disputes. Of particular interest were the responses to questions 5 and 6. In both cases, 3 of the 4 'undecided' subjects had not used the

information option at all, and the other one had used it minimally. All those who made extensive use of the 'i' option believed it was useful and that their understanding of the subject area had been increased.

	<div>agree ↓</div>	<div>undecided ↓</div>	<div>disagree ↓</div>
1. The system is easy to interact with.	14	0	0
2. It is easy to understand what to do.	12	2	0
3. The depiction of situations is realistic.	12	2	0
4. The computer responses are realistic.	11	3	0
5. The 'information' option is useful.	10	4	0
6. Using the system increased my understanding of domestic disputes.	10	4	0

Figure D.14 Summary of results from role playing system questionnaire

The comments of subjects (excluding those that relate to system performance such as speed of response) are listed below:

- 'After about five minutes into the session I was Anne, and stressing out deciding what to do.'
- 'As less information became available, ie towards the end of the scenario, it became increasingly difficult to "role play" and therefore more likely that the user would fall back on their own feelings. Still good fun!' [sic]
- 'Easy to relate to Anne.'
- 'Enjoyed playing - would like to play several times and try out different outcomes, ie play the roles of different people, ie it was satisfying to use.'
- 'I do not know a lot about role-playing. I could see that I could reply in several ways and possibly would have met with violence sooner.'
- 'I found the info option very useful but it seemed to be very directive in terms of which option to choose. Many of the options left no middle ground - responses were at extremes when I think, in some circumstances, less extreme options would often be selected.'

'I'd have liked to be able to mix and match responses.'

'If I had access to a system like this when I had domestic problems things might have turned out different.'

'If your aim is #6, I am not sure that any program would necessarily increase my understanding of domestic disputes. So much seems to depend on the personalities involved and their upbringing (social environment).'

'It helped me feel some of the trauma and "no win" sense that exists in that situation.'

'More choices for some of the questions in the system would have been helpful.'

D.4 Conclusion

The use of the Wizard of Oz method for simulating program behaviour in an interactive role playing system worked well. Even though the technology for handling complex situations is not yet available, it was useful to be able to test whether such a system would be of interest to subjects and whether they would be able to gain any insight from it. The answer to both questions appears, for most subjects, to be yes. Obviously, the survey carried out was not large scale, nor was it ever intended to be, but the generally enthusiastic response shows that further research into the area should be most fruitful.