Loughborough University

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

Attribution-NonCommercial-NoDerivs 2.5

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

No Derivative Works. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

LOUGHBOROUGH
UNIVERSITY OF TECHNOLOGY
LIBRARY

AUTHOR/FILING TITLE

CALLEAR, D.

ACCESSION/COPY NO.

040121925

VOL. NO.

CLASS MARK

LOAN COPY

0401219259

A Course-oriented Intelligent Tutoring System

with Probability Assessment

by

David Henry Callear

A doctoral thesis

submitted in partial fulfilment of the requirements

for the award of

Doctor of Philosophy of the Loughborough University of Technology

1989

© D. H. Callear, 1989

Loughborough University
of Technology Library
Date May 96
Class
Acc. No. 040121925
q 645 1465

This is to certify that the work presented in this thesis is original work undertaken by the author himself in the Department of Computer Studies, Loughborough University of Technology, along with the acknowledged references.

The work described has not been submitted in part or in full to this or any other institution for a higher degree award.

Abstract

Most Intelligent Tutoring Systems (ITSs) in the past have concentrated on small domains and have been topic-oriented. They have tended to be non-extendable prototypes and have neglected the expertise of human teachers. It is argued here that a promising approach at this time is to design course-oriented ITS shells which are based on the human teacher. Courses using such shells could be used to take some of the load of first-time delivery and assessment from teachers and lecturers, and leave them more time for individual tutoring.

Such a system was designed and built along these lines. The basic three-component structure of a number of previous topic-oriented ITSs was used, with the addition of an environment module and an interface module, and the system formed a declarative front-end using a videodisc to cover the larger knowledge domain of a whole course. Existing expertise from the fields of ITS, Expert Systems, Computer Assisted Learning and Interactive Video was built upon. Techniques of inexact reasoning used in expert systems were applied to the assessment of students to form a student model such as a teacher builds up. Hypotheses about the student were tested by questioning, and probabilities of the truth of the hypotheses were determined using a technique involving Bayes' Theorem. The resulting method of assessment, subject to further investigation, promises to be more efficient than conventional assessment, clearer about what is being assessed, and could enable several student abilities to be measured at the same time. It needs to be stressed, however, that many of the doubts regarding the independence of evidence in expert systems will apply here, unless the method is used with caution.

The system was evaluated according to principles of Action Research, and several of the arguments presented in the thesis were reinforced, including that such transferable shells can be built and extended beyond the prototype stage; that the human teacher makes a useful model on which to base a course-oriented system; that the probability assessment method compared satisfactorily with other methods of assessment; and that such a course-oriented approach might make a promising general direction for other research in ITS.

Acknowledgements

My thanks are due to Professor Ernest Edmonds, who made it possible for me to carry out this project and gave invaluable help; Dr. Steve Scrivener, who showed great skill and patience in pointing out and helping to correct numerous problems; and Dr. Robert Goodwin, who showed great interest and helped from afar in Australia.

I should also like to thank, for their help at various stages, Tony Clarke, Chris Hinde, and Linda Candy at Loughborough University; Mike Gardner, now with British Telecom; Peter Messer and Professor Derek Teather at Leicester Polytechnic; and Stuart Gill, Stuart Ward and Christine Mehta at Gateway School.

David Callear

Contents

Figures included with the text

Tables included with the text

Chapter 1

A Course-Oriented Intelligent Tutoring System

1.1 Intelligent Tutoring Systems (ITS)

The field of Artificial Intelligence (AI) has enjoyed some successes in recent years, notably in the field of Expert Systems (ES). The branch of AI which covers educational applications has come to be known as Intelligent Tutoring Systems (ITS), but this field is little known outside of AI. (See Sleeman and Brown, 1982, Lawler and Yazdani, 1987, Polson and Richardson, 1988, or Psotka, Massey and Mutter, 1988 for reviews of the field).

Appendix 1.1 describes selected ITSs which are mentioned in this thesis, and the variety will be apparent. There is little evidence of a unified approach. Most of the effort in ITS has been, and still is, in the USA. Several ITSs are in use with students in the centres where they were developed, such as Anderson's LISP tutor and Joobbani and Talukdar's CIRCUIT TUTOR, both at Carnegie-Mellon University. Others have been successful as prototypes in demonstrating certain techniques, such as BUGGY, SOPHIE, LMS and SITS. (See Appendix 1.1.)

There are now certain features and techniques which have come to be accepted as characteristic of 'intelligence' in ITSs. The components of a typical ITS are shown in Figure 1.1. One of the first to identify the three types of expertise required by an ITS as knowledge of what is to be taught, knowledge of how to teach, and knowledge of who is being taught was Self (1974), and this 'standard form' of ITS has been described by several writers, for example Clancey (1979), and Roberts and Park (1983). Such a model consists of a tutoring module, a knowledge base and a model of the student. The concept of a student model and the nature of the techniques used in the tutoring module are the aspects which distinguish an ITS from more conventional CAL systems or programs. Burns and Capps (1988) describe a more complex five component system, shown in Figure 1.2. This will be used in Chapter 4.

Some other features which can form part of an ITS are a natural language interface with the student, a rule based structure for the tutoring module, and a flexible 'interactive environment' in which the student can work. Certain languages tend to be used for programming AI and ITSs, such as LISP (mostly in the USA) or PROLOG. All these features were included in the project described in this thesis.

On the whole ITSs appear to have failed to impress outside the AI community, and are little known. The UK was not doing enough a few years ago, according to Heaford (1983): "At present there is no evidence to suggest that the UK is attempting to lead the world in the production of Intelligent Tutoring Systems, in spite of the fact that this country boasts more tools for building expert systems than any other in the world."

It is instructive to investigate why ITSs have not become widely used, or well-known, in the way that, for example, expert systems have. Some of the criticisms that have been levelled at the field of ITS will be examined, and possible responses to them will be discussed to illustrate the intentions behind this research.

1.2 Some Criticisms of ITS

1.2.1 Concentration on small domains in ITS

A feature of most ITSs, which was clearly identified by Sleeman and Brown (1982), is their concentration on small domains. They say: "ITS has clearly abandoned one of CAL's early objectives, namely that of providing total courses, and has concentrated on building systems which provide supportive environments for more limited topics." (Sleeman and Brown, 1982.)

Self also comments: "Present ICAI [ITS] systems are implemented as stand-alone packages which are not integrated directly with the wider curriculum." (Self, 1987.)

As recently as 1988, Sleeman and Ward were saying: "Until now, CBL and ITS have offered software for teaching single isolated topics rather than complete courses." (Sleeman and Ward, 1988.) Perhaps the time has now come to depart from 'supportive environments for limited topics' and attempt to tackle the problems of whole courses.

Figure 1.1

The three component ITS model (based on Self, 1974)

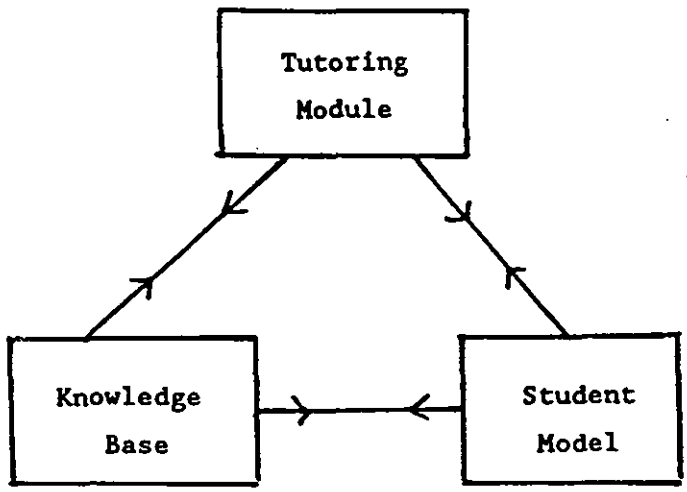
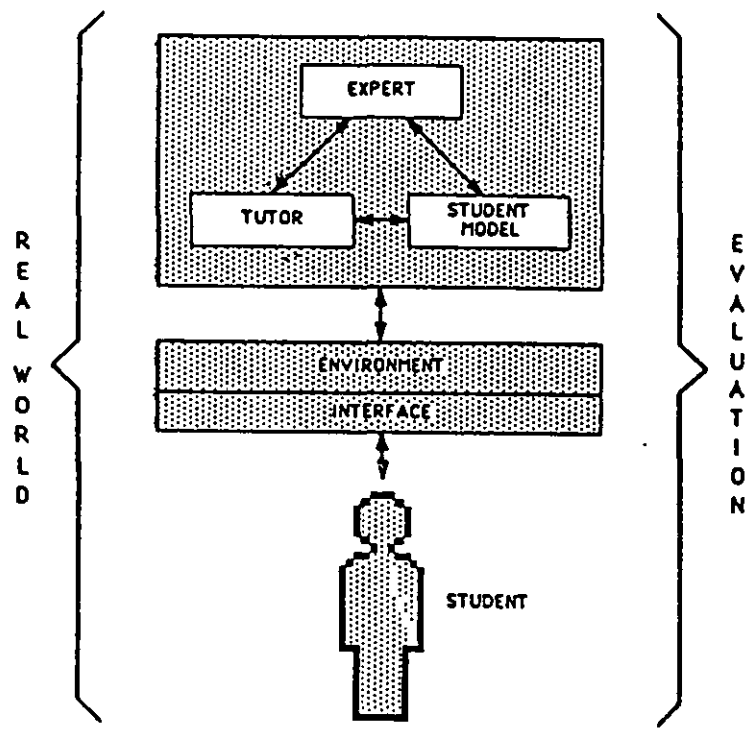


Figure 1.2

The five component ITS model (Burns and Capps, 1988)



1.2.2 Limited prototype systems in ITS

Roberts and Park (1983) warn: "Many of the write-ups in the literature discuss a prototype system, the problems encountered, and the recommendations for how those problems might be solved. This is very different from describing a system that has actually solved these problems."

Similar sentiments have been expressed by Self (1985), who is doubtful about the promise of existing systems: "... most [ITSs] are incomplete demonstrations and the others show that we lack the knowledge necessary to carry out complete implementations ... The implication remains ... that design strategies which it has not been possible to carry out for toy domains will be appropriate for real ones."

Yazdani comments: "Most ITSs ... have been designed and remain as prototypes." He quotes as a notable exception SOPHIE, sponsored and developed by the US Department of Defence, and now, after limited use, no longer maintained (Yazdani, 1988, p.190).

Miller, reported by Psotka, Massey and Mutter (1988, p.407), comments: "Experience with toy systems or academic prototypes does not necessarily scale up easily into economically viable tutors ... if the changes are not maintained, the ITS can suffer from 'extreme bitrot' and will not function on any viable computer. This is as true for academic prototypes as it is for commercial products."

1.2.3 ITSs as practical, usable systems

Systems to date have rarely been practical systems capable of being used with students, except in an evaluation situation.

Ross gives a list of ITSs, reproduced in Figure 1.3, and says of the systems in it: "Most of these do a lot less than the topic indication suggests and are only experimental ... hardly any have been properly tested on more than a very few people." (Ross, 1987.)

Psotka, Massey and Mutter (1988), reporting Miller (p. 406), advise: " Prepare to deliver the tutor on PCs. ... The ITS must interface with the real-world environments: main-frames, line editors, absence of graphics,

compilers, job performance aids, and so on." Anderson concludes a paper by saying: " It would be profitable to see what would happen if we made the practical compromises necessary to implement an intelligent tutorial system in an actual classroom." (Anderson, 1988.)

ITSs have been mostly run on expensive computers. Sleeman and Brown (1982) observe: "If ITSs are to have any practical importance, the subject areas need to be chosen with considerable care as each system represents a major investment of resources ... there has always been the nagging question of when such systems would become cost-effective?" Roberts and Park (1983) also repeat this question, which they point out is often asked: "When will ICAI systems become readily available in the marketplace? Unfortunately, this is not a simple question to answer."

Psotka, Massey and Mutter (1988) comment: "... intelligent training systems must provide some explicit benefit ... Benefits must be clear. They must be expressed in terms of effectiveness, material covered, time and costs."

1.2.4 A partial solution

The field of ITS has been criticised for concentrating on small domains or limited topics, for producing prototype systems which are not capable of further extension, and for not producing practical, usable systems. There are other criticisms as well, some of which will be mentioned later.

It is possible that the aims of the field of ITS have been over-ambitious, and that there has been an acceptance of goals that would not be achieved for a long time to come, except in a limited way within the constraints indicated by these criticisms.

A course-oriented approach, rather than a topic-oriented approach, in which the aim was to design a system to teach a number of topics making up all or part of a whole course, would commit the system to tackling many of the criticisms above, and might lead to a partial solution.

1.3 A Course-oriented Approach

1.3.1 Course-oriented approaches in ITS to date

Recently Sleeman and Ward have voiced the opinion:

"At present most Intelligent Tutoring Systems (ITS) and Computer-Based Learning (CBL) programs tend to be topic-oriented rather than course-oriented: they teach topics such as linear equation solving rather than courses in mathematics. We believe that this limits the acceptability of computer-based learning and training, which is unlikely to become fully accepted until there is considerably more breadth to the instructional materials that currently exist." (Sleeman and Ward, 1988.)

Blaine and Smith (1977), who worked on the system EXCHECK originated by Suppes, have also argued in favour of the whole-course approach in ITS, pointing out that solving the new problems raised could benefit AI research. With a few exceptions such as this, a course-oriented approach was almost unknown in 1984 when this research was started.

Sleeman, a prominent researcher in ITS, now of Aberdeen University, is one of the few who have consistently advocated a course-oriented approach. In 1973 Sleeman, along with Hartley, described a general decision-making scheme for a teaching program, which is closely similar to the scheme decided upon for the prototype system described later in this thesis (Hartley and Sleeman, 1973). This will be returned to later. It would not have been possible through technical limitations to build this scheme into a readily available system in 1973, and in addition reaction to the TICCIT and PLATO projects at that time was causing some disillusionment with the notion of whole courses taught by computer. In 1982 Sleeman, along with Brown, as editors of the influential book 'Intelligent Tutoring Systems', made the comment quoted above that ITSs had 'abandoned total courses' in favour of 'limited topics'. Recently Sleeman has been working on PIXIE, a course-oriented expert system shell for ITSs. In 1988, at the British Computer Society Expert Systems Conference, Sleeman, along with Ward, presented the paper from which the above quotation is taken.

Figure 1.3

Some Examples of ITSs and Environments

(Ross, 1987)

Name	Date	Topic
ACE	1982	Interpretation of NMR spectra
ALGEBRA TUTOR	1983	Solving simple linear equation systems
ALGEBRALAND	1985	Algebraic proofs tool
BANDAID	1978	Introductory *BASIC* programming
BIP	1976	*BASIC* programming
BUGGY/DEBUGGY/IDEBUGGY	1978/82	Diagnosis in basic arithmetic
EUROHELP	1987	Tutorial help, initially for UNIX mail
EXCHECK	1983	Simple logic and set theory
FGA	1985	Basic French grammar
FLOW	1977	Flow computer language
GEOMETRY TUTOR	1985	Geometric proofs tool
GERMAN LANGUAGE TUTOR	1978	Simple German syntax/vocabulary
GUIDON (& -DEBUG, -WATCH)	1979/86	Medical diagnosis
INTEGRATION TUTOR	1976	Basic integral calculus
LISP-ITS	1985	*LISP* programming
U of T LISP TUTOR	1985	*LISP* programming
MACSYMA ADVISOR	1979	Use of *MACSYMA*
MALT	1973	Basic machine language programming
MENO-II	1983	Very simple *PASCAL* programming
NEOMYCIN	1981/84	Medical diagnosis
PIXIE	1983	Algebra equation solving
PROUST	1985	*PASCAL* programming
QUADRATIC TUTOR	1975	Solving quadratic equations
QUEST	1984/87	Simple automotive electrics
SCHOLAR	1973	Facts of South American geography
SIERRA	1983/87	Learning arithmetic procedures
SOPHIE-I,II,III	1976/82	Electronic trouble shooting
SPADE	1982	Simple *LOGO* programming
SPIRIT	1984	Probability theory
STEAMER	1982/87	Marine steam propulsion plant
TALUS	1985	Basic *LISP* programming
THEVENIN	1985	Simple electrical circuits
TRILL	1983	Basic concepts of *LISP*
TUTOR	1985	British Highway Code
VP2	1985	Basic English grammar for Spanish
WEST	1982	Simple arithmetic skills
WHY	1982	Basic processes in meteorology
WUSOR	1982	Expertise in a maze game

It is clear that Sleeman, a major figure in the field of ITS, has felt no need to revise his early opinion that a course-oriented approach is desirable; in fact, this opinion has clearly strengthened over the years.

1.3.2 The course-oriented approach and usability

An ITS which had a course-oriented approach would be more likely to be attractive to those in education who have to decide whether to use such systems, as educational institutions think in terms of teaching courses rather than single topics.

The notion of designing a prototype which might be extended to a larger domain later, by someone else, would have to give way to addressing the problems of a larger domain directly. If the aim of producing whole courses increased the probability of ITSs being used more widely, there would be pressure to move away from designing prototypes towards designing working systems.

In recent years there have been great advances in computer speed and storage capabilities, and also in programming techniques. There are many computer systems in other AI areas which are practical, are in use, are well out of the prototype stage and are usable and cost-effective. Commercially available expert systems are a case in point. Improvements in software techniques have made it possible to produce programs which are portable between computers. People's attitudes to computers have changed, and they now expect systems to be practical and easily usable. There is no real reason why an ITS should not now conform to such criteria.

In short, usability is closely bound up with a course-oriented approach.

1.3.3 The objectives of ITS

The field of ITS has had certain 'traditional' objectives. For example, O'Shea (1982) indicates that the most important objectives of ITS should be the development of natural language, development of the ability of the

computer to 'understand' what is being taught, the 'understanding' of students' misconceptions, and the ability for ITSs to learn from experience.

These are ambitious aims, arguably over-ambitious, though in a practical sense some of them are achievable now. For example, a number of natural language parsers have been developed, simple but often adequate, and in practice natural language is not always the best way to communicate with a student in an ITS anyway. Miller says briefly: "Natural language interfaces are a diversion." (Psotka, Massey and Mutter, 1988, p.407.) Although progress has been made, something of an impasse seems to have been reached on such objectives.

In future, in the short term, it might be profitable to give priority to objectives relating to usability, rather than the traditional objectives, for the following reasons:

(a) Classroom ITSs are now technically feasible.

(b) Currently, because ITSs are little used in practice, little empirical data exists to test the relevance of theoretical ideas. Some of the theoretical objectives might be supported, or otherwise, by the results of experiment and observation.

The adoption of an approach based on usability is likely to lead to a course-oriented approach. ITSs are unlikely to be widely used if they deal with minute areas of knowledge. Certainly a course-oriented approach is more likely to lead to ITSs that can be realistically employed to teach.

There are signs that in America an approach to ITSs based on usability is indeed being accepted by researchers, even if not spelt out explicitly. Some of the quotations above bear this out. In their forward to a book based on an ITS conference sponsored by the U.S.Army, Psotka, Massey and Mutter comment: "In one sentence the Army expects intelligent training systems to improve our ability to train soldiers today and in the future 1988.)

1.3.4 The potential of course-oriented ITSs

Computer Assisted Learning (CAL) workers in the sixties and seventies had the declared aim of providing whole courses, TICCIT being the best example. This was considered to be such a worthwhile project that the National Science Foundation of America invested several million dollars in it over five years. The arguments used to justify TICCIT then, one being that CAL might provide "better instruction at less cost than traditional instruction", are no less valid today. (See O'Shea and Self, 1983, p.86.)

CAL has by now proved its worth in education, and there are many reasons why CAL programs and systems are potentially of great practical and theoretical usefulness. The student can work at his or her own speed, can have individual attention, can receive immediate responses, and so on. Some arguments for using CAL are given in Appendix 1.3. These and other reasons apply to a whole course ITS as to many other CAL projects. Some of the arguments against CAL, compared with human tutors, are given in Appendix 1.4.

There are problems in educational establishments that might be overcome by the use of efficient ITSs to cover whole or substantial parts of courses. One problem is that of students who miss parts of courses through illness or late arrival, and at present have no way to catch up except by private study. Another is the problem of covering courses in some subjects where teachers are scarce, such as in maths and science subjects. In many subjects, notably science, syllabuses are frequently so crowded that teachers have difficulty covering them, and assistance from ITSs in supplementing the teacher could effectively increase teaching time for the student. Another problem is that of the individual student who is simply not receptive to human teachers, and plays truant or 'switches off'. Many such students respond to computers, offering privacy and individual attention, in a way they do not to other forms of individual study.

There is a potential 'spin-off' value in a course-oriented approach, predicted by Blaine and Smith at a time when the available technology made

it little more than a possibility: "There is significant further progress to be made in AI by systematically dealing with an entire curriculum within the CAI paradigm ... and solving whatever information processing problems are present within those curricula". (Blaine and Smith, 1977.) In the

course-oriented ITS project described in this thesis, a probability method of assessing students was devised. This can be considered independently of the ITS, and bears out the suggestion that useful spin-offs can be expected from such an approach.

1.4 An ITS Shell

1.4.1 Extendability of previous ITSs

ITSs have tended to concentrate on certain subject areas, and have not usually been extendable to other areas. In the past few researchers have attempted to produce extendable, generalisable systems, though this has changed somewhat in recent years (see below). It is argued here that it is highly desirable for an ITS to be extendable, and that this is quite compatible with a course-oriented approach.

ITSs have mostly dealt with mathematical or computer language domains. The techniques demonstrated are usually unlikely to be applicable to other subjects. Most systems have appeared to be exercises by mathematicians and computer specialists in their own subjects. Self (1987) comments: "... ICAI (ITS) workers have retreated from building systems which deal with traditional classroom subjects ... to building systems which help learners master some computer-based skill, such as programming or using an operating system ..."

Ross provides a representative list of ITSs and their topics, (see Figure 1.3) and comments that it "shows that the AI world is fairly inward looking when it comes to picking experimental domains." (Ross,1987.) In this list, 71% relate to the computer-associated topics of programming, mathematics or electricity and electronics.

Roberts and Park (1983) also point out that "most ICAI [ITS] systems have been restricted to the highly-structured content areas like mathematics, electronics, and games ... the wide applicability of ICAI systems and models needs to be verified in other content domains as well."

1.4.2 Expert System shells

The field of Expert Systems, probably the most successful field of AI, has brought the notion of a program which is a 'shell' into common use.

Yazdani defines expert systems concisely: "Expert systems are such knowledge based systems which use inference to apply knowledge to perform a task." One of the reasons for the success of expert systems in the AI field is that it has proved possible to produce them in the form of a 'shell', whereby the expert can be made to operate upon different domains of knowledge.

If an expert system is regarded as being composed of an expertise mechanism (or inference engine), a knowledge base and an interface with the user as in Figure 1.4, it can be seen that it is a short step to replace the knowledge base with one for a completely different domain, and thus have an expert in, for example, car maintainance rather than medicine. One of the first expert systems was the medical system MYCIN, in which it proved possible to separate the 'inference engine' from the knowledge base of diseases and symptoms to form EMYCIN, 'empty MYCIN'. (See Shortliffe, 1976.) A more elaborate structure for ESs is shown in Figure 1.5.

There are strong structural similarities between ITSs and expert systems, which can be seen by comparing Figures 1.1, 1.2, 1.4 and 1.5. It was thus reasonable to attempt to convert expert system shells into tutoring systems.

1.4.3 Converting Expert Systems to ITSs

There have been attempts to convert Expert System shells directly into ITSs by substituting a subject knowledge base and seemingly suitable tutoring expertise. In the UK, Fisher and Howe (1982) investigated the potential of expert system training aids, while in America Clancey adapted the medical expert system MYCIN (see Clancey, 1979) to create a tutorial system, GUIDON. Such attempts have been followed by the realisation that the techniques do not transfer as readily as was thought.

The techniques used in GUIDON have been criticised heavily by Ford (1984), Soloway and VanLehn (1985), and Self, who commented in 1985: "... there is an emerging concensus on the methodology of ICAL design [i.e. basing it on expert systems] ... this methodology is mistaken." Two years later Self was able to say: "In fact, the expert systems-based approach to ICAI, which to many outsiders is by definition the only approach to ICAI (perhaps because of the high profile of expert systems themselves), has been largely rejected by ICAI researchers ..." (Self, 1988).

The main problem with expert systems as teaching aids is that 'expert' knowledge is very different from student knowledge, and from knowledge presented in a form suitable for a student. An expert system uses its knowledge base to reason and reach a valid 'expert' conclusion, such as what a patient is suffering from. An ITS is not so much concerned with reasoning or reaching conclusions from the knowledge base as with presenting the knowledge to the student in such a manner that the student can understand and absorb it.

1.4.4 Other ITS shell projects

Although the direct transference of expert system shells to ITSs has not been outstandingly successful, the lesson that it should be possible to design more generally applicable ITS systems has been learned. Several projects have recently attempted to provide ITS 'toolkits' for applying ITS methods to different topics.

Sleeman has implemented "a data-driven ITS shell which attempts to diagnose student errors within particular knowledge domains by finding models which represent those errors", called PIXIE (see Sleeman, 1987) and has recently applied a commercially available expert system shell, S.1, to enable a student to diagnose and rectify simulated faults on an oil platform (see Sleeman and Ward, 1988).

Anderson has produced a toolkit system called PTA, which stands for PUPS Tutoring Architecture. PUPS is a flexible production rule system, and the toolkit written in it enables different skills to be tutored using 'model-tracing'. A model of problem-solving is followed through as the student

tackles the problem, and is compared with the student's activity. When he or she deviates, action is taken. The system is most suitable for teaching programming, and has been applied to several languages. (See Anderson and Skwarecki, 1986.)

Another generalised application of ITS expertise is the 'Bite-Sized Tutor' of Bonar and others, which is an authoring language to enable knowledge to be tutored in 'bite-sized' chunks. It is under development and is aiming to generalise tutoring methods so that a minimum of domain-specific tutoring will be required. (See Bonar, Cunningham and Schultz, 1986.)

The work presently being done on extendable ITSs was largely of an embryonic nature in 1984 when the present project was started, so the approach here tends to adopt an independent approach rather than building on such work.

1.4.5 Extendability in a course-oriented ITS

An ITS which could deal with the domain of a whole course would be more likely to be extendable to other domains, or to contain features which could be extended. The problems of broadening the domain would have been at least partially solved.

Because expert systems do not transfer directly to ITSs, this need not mean that the expert system notion of a shell cannot be used in the field of ITS, or that techniques used in the field of Expert Systems, such as inexact reasoning, cannot be used usefully in ITSs. It should be possible to produce a whole-course ITS along the lines of a transferable shell, which could have new knowledge bases slotted in, and be used to teach different courses.

1.5 A Teacher-based ITS

1.5.1 Tutoring principles in ITSs

Previous ITSs seem to have neglected the expertise avilable in the field of education, and also that available from studying the methods of practising

Figure 1.4

Three Component Expert System Model (Bratko, 1986)

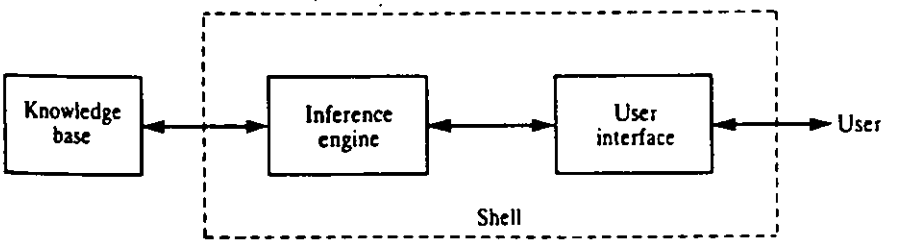
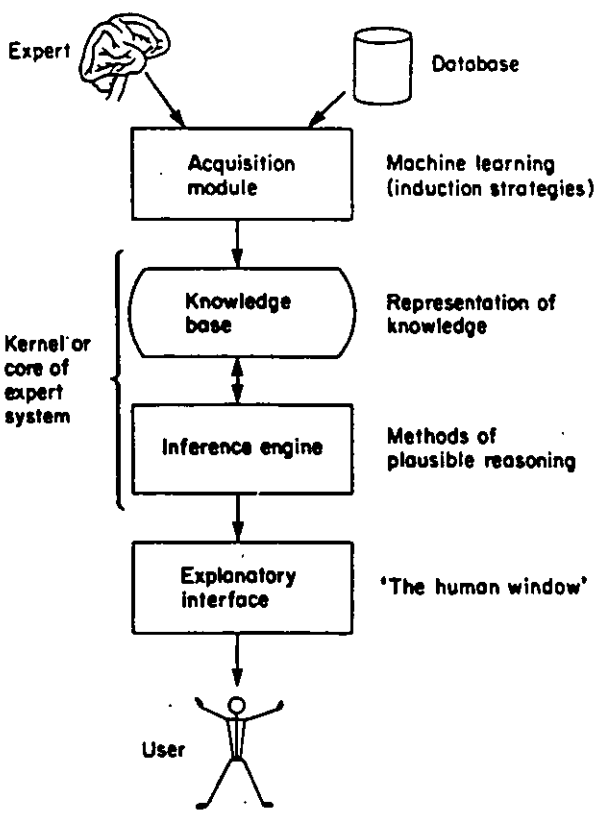


Figure 1.5

Four Component Expert System Model (Forsyth, 1984)



teachers. The tutoring principles used daily by teachers, although not always explicitly stated, could yield useful guidelines on which to construct ITSs. In particular, teachers have been teaching whole courses for many years, and if course-oriented ITSs are built their expertise should be taken into account.

There is talk in the ITS literature about determining valid tutoring principles on which to base systems. Self says: "... there is a serious desire in ICAI [ITS] to develop a computational formalism for expressing general tutoring principles." (Self, 1987.)

In fact the field of ITS has not been conspicuously successful in its aim of determining tutoring principles. Sleeman and Brown (1982) were critical of progress to date, pointing out that "the tutoring theory embedded within these benchmark programs for conveying this expertise is elementary". They go on to add that each system "has tended to emphasise some aspects of an overall coaching system and neglect others. Thus it is not surprising that the designers of these systems are dissatisfied with their system's overall performance."

Ross has commented on the subject of ITSs: "The educational approach is still, in some cases, very simplistic - little more than 'present it, test it, assess it, maybe reteach it' ... it will be a good while before the teaching rises above the purely methodical towards the inspirational, too." (Ross,1987.)

Anderson has been optimistic about progress with his ACT* (Adaptive Control of Thought) theory, which applies principles from cognitive psychology to his Geometry Tutor and LISP Tutor, saying in 1985 that: "These obstacles to past efforts at ICAI are now being overcome." (Anderson, Boyle and Reiser, 1985.) In 1988 he was saying that there has been "... dramatic progress in our understanding of how to build the expert module for a tutoring system." (Anderson, 1988.)

However, Anderson admits that "... we need a great deal more basic research before construction of expert modules can progress as an engineering enterprise ... theories of learning, in contrast to theories of performance, have yet to be integrated into tutoring systems." (Anderson, 1988.) His theories have not met with universal acclaim, though Yazdani (1988) suggests that they "are offering a strong hint of a breakthrough."

1.5.2 Neglect of teaching expertise in ITSs

If there has not been conspicuous success in determining tutoring principles, it seems that there is little respect for educational expertise in the field of ITS. Some seem to feel that ITS has more to offer education than vice versa. Richardson comments: "Spinoffs from ITS research have enriched several other fields. One of the most important spin-offs is the application of the concepts and philosophies, if not the methods, of ITSs to traditional instruction ... the importance of conceptual understanding, the role of preconceptions, the need to connect in-school and out-of-school learning, the importance of self-monitoring and self-management techniques, and the vision of lifelong learning." (Richardson, 1988, p.251.) In so far as these concepts have been adopted in the UK, educationists would be surprised to learn that they owe them to ITS.

In the UK Self also talks disparagingly of the field of education: "...educational theory seems (to an outsider, at least) to be in some disarray ..." He goes on: "... ICAI [ITS] is right to ignore demands that it should be shown to work in present classrooms. In terms of the history of education, these are transient structures and manifestly not optimised for learning. Of course, neither is an ICAI system ..." (Self, 1987.)

Self (1987) also comments: "... ICAI [ITS] workers have retreated from building systems which deal with traditional classroom subjects ... this has had the undesirable effect of isolating ICAI researchers and encouraging them to ignore educational inputs."

Richardson mentions educational research, and almost immediately dismisses it, concluding a section with: "However, the educational literature says much more about classroom interaction, questioning strategies, and teaching methods in formal instructional situations than about tutoring." (Richardson, 1988, p.246.) It seems strange that he rejects so summarily a huge body of research, apparently on the grounds that there is no parallel between teaching in the formal instructional situation and teaching the same things using individual tutoring.

In reading such works as Sleeman and Brown (1982), Polson and Richardson (1988), and Psotka, Massey and Mutter (1988) one is struck by the absence

of references to any of the theories of educationalists, and the absence of any empirical work examining how human teachers teach or how human tutors tutor.

1.5.3 Using teaching expertise in ITSs

It is rarely acknowledged by those in the ITS field that others in the field of educational research are also pursuing such tutoring principles (and have been doing so for much longer). Perhaps more to the point, practising teachers daily make use of such principles, which they have arrived at of necessity by trial and error.

While work is progressing on such theories, a useful short-term approach might be to model an ITS on the human teacher, who has solved many of the problems in a practical sense. It is arguably a more viable approach for a course-oriented ITS than for a topic-oriented ITS.

There is, in fact, considerable uniformity in education on some things. The way teachers organise the knowledge they teach into courses; the ways they use to present it to students; and the way they obtain information about students to build up a type of 'student model'; these are all things which many teachers would agree upon, and which must also be the concern of ITSs.

1.6 Summary and Outline of the Thesis

The argument outlined in this chapter may be summarised as follows:

1. In the past ITSs have tended to concentrate on small domains and have been topic-oriented. Objectives have tended to be theoretical rather than practical, and to date have only partially been realised. This has led to impractical systems which are not widely used. An alternative approach is to investigate the possibility of classroom exploitable ITSs, built to teach courses rather than fragments of a course. This will provide empirical data to test and support theory.

2. Previous ITSs have tended to produce non-extendable prototype systems in computer-related domains. An effort should be made, and is now being made

in some quarters, to build extendable ITS shells of greater generality. This has been achieved with expert systems. However, expert system shells do not convert directly to tutoring systems.

3. The field of ITS has in the past neglected educational and teaching expertise. This has much to offer in a course-oriented system, as teachers teach whole courses and have solved many of the problems. Course-oriented ITSs can usefully be regarded as Expert Systems based on the teacher as an expert.

The thesis is organised as follows. The first four chapters deal with a literature survey and with the development of the argument of the thesis.

Chapters 2 and 3 are used to survey fields which impact upon the development of ITSs. Chapter 2 looks at some intelligent systems, examining briefly some ideas from the field of Artificial Intelligence (AI), focussing on Intelligent Tutoring Systems (ITS) and Expert Systems (ES), and identifying successes and failures. Chapter 3 looks at the field of Computer Assisted Learning (CAL), where it is found that past experience in the area of whole courses is illuminating. Present programs and approaches are also examined. This chapter also goes on to look at the field of Interactive Video (IV), a recent and successful area of CAL which, it is argued, offers a practical solution to the problem of storing and presenting the greater amount of knowledge in a course-oriented ITS. A pilot project which was carried out in 1985 is described. Chapter 4 brings together what has been learned from this survey, and makes some proposals for the design of a course-oriented ITS.

The last four chapters deal with the building of a prototype system, its evaluation and the conclusions drawn from the exercise.

Chapters 5 and 6 describe a course-oriented ITS which was built along the lines argued in the first part of the thesis, WITS (a Whole-course Intelligent Teaching System). Chapter 5 covers the system generally, and Chapter 6 the student model and the intelligent assessment method which was developed. Chapter 7 deals with the evaluation of WITS with school students, according to 'action research' principles. Finally some conclusions are drawn from the project in Chapter 8.

Chapter 2

A Survey of Intelligent Systems

2.1 Introduction

In the last chapter the field of ITS was introduced, and some general criticisms were explained which led to the present discussion of a course-oriented approach.

To develop further the notion of a course-oriented approach, the wider context of Artificial Intelligence will be examined in this chapter. The field of ITS will also be examined more closely, as will the relationship between the fields of ITS and Expert Systems.

2.2 Artificial Intelligence (AI)

2.2.1 Defining AI

The field of Artificial Intelligence, that is, the programming of computers to exhibit characteristics of human intelligence, however defined, is now a valid and respectable area of study within the field of computing. Bramer (1984) comments: "In a short period the problems of the British AI community, and particularly those involved in Expert Systems development, have changed from being those of a small group regarded with suspicion to being those of a rapidly growing community in high fashion."

There is little doubt that within some definitions, computers can exhibit forms of intelligence. As Simons (1984) says: " ... if an ape or a dolphin were able to compute a complex payroll or actuarial table, we would quickly see such behaviour as evidence for intelligence ..." Simons goes on to say: "An animal that could do differential equations would be deemed intelligent: a similarly skilled computer would not."

The reservations that remain hinge around a number of features of machine intelligence, listed and answered by Turing (1950) in a classic paper. Two main objections to the notion of machine intelligence may be mentioned here. First, it is restricted to small 'domains', and although some forms are extensible to other areas, machine intelligence is a long way from matching the versatility and generality of human intelligence. Second, the

working of machine intelligence can be seen in operation and understood, since it was programmed into the machine by humans, and the machine is therefore considered only to be doing what it was programmed to do. Turing (1950) called this 'Lady Lovelace's objection'. People who mindlessly do as they are told are considered unintelligent, but this contention that only divergent behaviour should be regarded as intelligent would would be difficult to justify.

It is relevant to examine such basic ideas, if only briefly, because it is necessary to determine what characteristics of a course-oriented CAL system might identify it as an 'Intelligent' Tutoring System. Inevitably it will be restricted to the domain of teaching, but it will be more convincing to describe it as intelligent if it can be shown to be extensible to several teaching domains. Also inevitably, the intelligence will be programmed and 'explainable', but the system will be more convincingly intelligent if it can be made to teach students in ways which cannot be predicted beforehand.

2.2.2 Features of AI systems

The characteristics of Intelligent Knowledge-Based Systems (IKBS), which might be described as the applied form of Artificial Intelligence, are identified by Sloman (1983) in what he describes as 'a tentative overview'. Sloman points out that "although the list may seem very ambitious, it actually reflects the spread of research which is already in progress", but that "intelligent systems designed in the next decade will at best exhibit only a subset of [these characteristics]". He also points out that "it must not be assumed that all AI workers would agree with this list."

Sloman's list was one of the sources used, at the start of this project, to determine characteristics that might be required in a course-oriented ITS. It is reproduced here in modified form so as to include some relevant comments, as follows.

1. IKBS can have a general range of abilities, including:

   (a) the ability to cope with varied objects, such as stories, images, scenes - here varied teaching sequences could be added.

   (b) the ability to cope with a variety of domains, such as, in this case, a variety of taught subjects.

   (c) the ability to perform a variety of tasks in relation to an object, for example, seek out an object that is a teaching sequence, present it to a student, offer tests on it, record the assessment of it, etc.

   (d) the ability to recognise which sub-ability to use.

2. IKBS can include various forms of discovery, learning or self-improvement (on the part of the machine). There have been several intelligent tutoring systems which have attempted to improve their own ability to teach, notably O'Shea's system (1982) which teaches itself how to teach quadratic equations. Such systems form an important category on their own. An important feature of ITSs in general is that they should learn about the student, and be able to adapt to him or her.

3. IKBS can perform inferences, including not only logical deductions but also reasoning under conditions of uncertainty, non-monotonic reasoning and reasoning with non-logical representations, e.g.maps, diagrams and networks. Clearly an intelligent feature that an ITS should possess is the ability to reason about the student from uncertain or incomplete information. This important feature will be returned to later.

4. IKBS are able to communicate and co-operate with other intelligent systems, especially human beings. The communication of an ITS is primarily with one human being, the student, and also more intermittently with the student's teacher. The communication should be as fluid as possible, using perhaps 'natural language' but also devices such as commands and menus which computer users have become familiar with. This area of the 'human-computer interface' will be discussed further later.

5. IKBS can co-ordinate and control a variety of sensors and manipulators in achieving a task involving physical movement or manipulation. The field of intelligence in manipulating physical systems or intelligent robotics is a separate and specialised field of AI.

6. IKBS can cope flexibly with an environment which is not only complex and messy, but also partly unpredictable, partly friendly, partly unfriendly and often fast moving. This includes the ability to interrupt actions and abandon or modify plans when necessary. In the case of a course-oriented ITS, the complex and messy environment would be the student's learning. The system would need to provide flexible guidance, and would need to modify its approach and even abandon it as required with a particular student.

7. IKBS can possess self-awareness, including the ability to reflect on and communicate about at least some of one's own internal processes. This includes the ability to explain one's actions. This characteristic, often referred to as 'transparency' in intelligent systems, was seen to be important in a course-oriented ITS. Too often CAL systems teach and assess the student while keeping him or her in the dark about progress made. In particular, the system should keep the student aware of his or her progress at all times, and make accessible the student model that is being built up.

8. IKBS can cope with a multiplicity of "motivators", i.e. goals, general principles, preferences, constraints, etc. which may not all be totally consistent in all possible circumstances. In this respect, a course-oriented ITS must reconcile the need to cover the required subject matter, the need to cover it at the right speed for the student, the need to accommodate the student's preferences regarding, for example, the ordering of the material, the need to assess the student, the need to keep him or her informed, and others besides. On the subject of multiple goals, it would be useful if the system could test, not just general student ability, but a number of different abilites, such as recall, understanding, and others.

Sloman also points out that "the notion of intelligence is bound up not only with what can be done but also with how it is done (i.e. the style, or manner). Some of the examples he gives are as follows:

1. An intelligent system should never 'crash' or reject a problem when the information is insufficient, but should degrade gracefully. It goes without saying that a course-oriented ITS should, in such circumstances, respond sensibly and politely to the student and ask for further information, or if appropriate offer an alternative course of action.

2. It should use **insight and understanding**, rather than brute force or blind and mechanical execution of rules to solve problems. This is the approach of the chess player, who needs insight because there are so many possible consequences of a move that they cannot all be checked beforehand. A course-oriented ITS will be concerned not so much with the solution of complex problems, but with the manipulation of complex knowledge. This is returned to in 3 below.

3. An intelligent system should be able to use **inference** to answer hypothetical questions. A central question in a whole course ITS would be: How much has the student learned, and what is the level of his or her ability at the moment? An unfortunate feature of many CAL courses is that, to obtain a 'model' of the student of sufficient accuracy, he or she has to be tested with large numbers of detailed questions. A human teacher, by contrast, tends to set or ask relatively few questions, and use inference (or insight and understanding, as in 2 above) to build up a 'model' of the student. 'Fuzzy reasoning' or similar techniques could be used to simulate the human teacher in this way.

4. **Conflicting goals** should not be dealt with simply by means of a pre-assigned set of priority measures, but for example by analysis of the reasons for the conflict and making inferences about the consequences of alternative choices of compromises. 'Traditional' CAL programs have usually had pre-assigned priorities; for example, the priorities with a topic might be (1) to instruct the student and (2) to test him or her with, say, ten questions. It should be possible for the student to determine whether and when a topic was learned, and also to determine whether and when he or she would be tested. The student should thus help to determine the goals pursued at any moment.

Sloman goes on to suggest what features need to be built into intelligent systems. One such feature is 'rich stores of domain-specific knowledge', which would be required for a course-oriented ITS. This requirement could be covered, as explained elsewhere, by using interactive video.

2.3 A Brief Survey of ITS

2.3.1 The field of ITS

Intelligent Computer Aided Instruction was identified in the Alvey program for action on Intelligent Knowledge Based Systems (IKBS) as an important field with commercial possibilities, and potential mass markets (SERC/DOI, 1983). It is thus accepted as an area particularly well worth pursuing in the attempt to make the UK competitive in information technology.

Anderson, Boyle and Reiser (1985) describe 'intelligent' CAL systems as those "that simulate understanding of the domain they teach and that can respond specifically to the student's problem-solving strategies."

A course-oriented ITS should thus be able to put the large amount of knowledge in its domain into an intelligible order, and be able to present it to the student according to his or her learning strategy. However, it is not usual for a tutor to allow the student to determine completely the learning strategy, or for the student to want to. It would be desirable for the ITS to limit the student's course of action in some ways, as a human tutor would.

The application of Artificial Intelligence (AI) techniques to Computer Assisted Learning (CAL), like other computing and educational fields, has begun to accumulate acronyms. The term Intelligent Tutoring Systems (ITS) became popular around the time of publication of a key work in the field by Sleeman and Brown (1982), and this will be prefered here. In the USA the term Intelligent Computer Aided Instruction (ICAI) is common. Some writers (e.g. Self, 1985) use the term 'Intelligent Computer Assisted Learning' or ICAL.

Sleeman and Brown (1982), in their introduction to what was for several years the most comprehensive book in this field, point out that intelligent tutoring systems had their roots in the early 'generative' CAL programs, which instead of storing questions for the student generated them as required, in the subject domain of arithmetic (see Uhr, 1969). These gave

way to 'adaptive' programs which in addition adapted the generated questions to the ability of the student (see Suppes, 1967). A successful adaptive system was produced by Pask (see Lewis and Pask, 1964), to teach keyboard skills. Adaptive systems, if not generative systems, might fall within the above definition of intelligent CAL systems. Since the early programs, there has been a gradual evolution of features which can be described as 'intelligent' in teaching or tutoring systems, often derived from developments in other areas of artificial intelligence.

2.3.2 Classifying ITSs

The field of ITS has a relatively long history, for a computing field. Its development is largely characterised by a number of tutoring prototypes which are largely unrelated and concentrate on one or just a few aspects of tutoring. Because of the general nature of a course-oriented ITS, many of these prototype systems are of interest, and a selection are listed in Appendix 1.1, with some explanatory notes. This appendix also contains some AI, ES and IV systems which are of relevance to this research. The systems referred to by name in this thesis will all be found in this appendix.

There is no agreed way of classifying intelligent tutoring systems. There seem to be almost as many approaches as there are workers in the field. Sleeman and Brown's book (1982) is divided into four parts: protocol analysis, computer-based coaches, artificial intelligence techniques and self-improving teaching systems. The authors pick out three areas of concern (p.4): habitable (friendly) natural language systems; student modelling and concept formation; and special-purpose deduction techniques. Looking at some other classifications, Goldstein and Carr (1977) divided up computer assisted learning prior to 1977 into primitive, classical, romantic and modern periods. Ford (1984) concentrates on three main ITS philosophies: the expert model (e.g. GUIDON, which will be discussed in the next section), the information processing model (e.g. SPADE), and the genetic model (e.g. WUSOR). Wenger (1987) discusses ITSs in terms of "people, ideas, and systems".

As mentioned in Section 1.1, Self (1974) is associated with the identification of three main parts of most ITSs: (a) a knowledge of how to teach, (b) a knowledge of what is being taught and (c) a knowledge of who

is being taught. (See Figure 1.1.) ITSs can be divided up according to their degree of concentration on each of these three areas. The last of these parts of a system has become known as a 'student model'. Roberts and Park (1983) also point out that there are three main components of an ITS, and it is striking that they correspond closely to the three parts of a standard expert system (described later):

An expertise module (knowledge base).
A student module (user interface).
A tutoring module (knowledge manager).

This grouping should be as useful for course-oriented ITSs as for topic-oriented ones. A more detailed one, which will be used in Chapters 5 and 6, is shown in Figure 1.2.

It will be useful to look at some of the themes tackled by ITSs, as typified by selected prototype systems. So as to introduce some order, these will be described in the three categories indicated above, under the headings: knowledge organisation, tutoring strategies, and student modelling. The groups will overlap to some extent, and some systems will occur in more than one group. Expert system based approaches, as in GUIDON, are left to Section 2.4. Natural language in ITSs will be left to Chapter 4, Section 4.6.6.

2.3.3 Knowledge organisation

In his pioneering system SCHOLAR, Carbonell (1970) used a 'semantic network' for the first time in a teaching system, though the idea was originally devised by Quillian (1966). In a semantic network, words forming nodes are linked by relationships, so that a topic is 'understood' in the sense of relating it to other topics, and in a dialogue progress can be made from one topic to another.

In SCHOLAR the subject was South American geography, and the way the semantic network was arranged is indicated in Appendix 2.1. This organisation of knowledge may be appropriate for some topics, and may be akin to the way the human brain works in some instances, but O'Shea and

Self (1983) comment: "SCHOLAR's semantic network contains little information about desirable orders of presentation of topics." It will be essential in a whole course ITS to decide what is a 'desirable order of presentation of topics'.

Schank (see Schank and Abelson, 1977) developed a system of 'conceptual dependency' to express verbal concepts by a method supposedly independent of language, and extended this to the idea of scripts to describe sequences of actions. Some of the flavour of the method is conveyed by the restaurant example Schank gave in his original paper (see Appendix 2.2). Scripts in a modified form were used in WHY, a system to teach rainfall originated by Stevens and Collins (1977). Carbonell's topic nodes were replaced with verbal formulations in a more hierachical structure. Parkes (1987) is now developing a system which will 'understand' educational films using scripts.

Another suggestion for organising knowledge is that of using 'frames', associated with Minsky (1975) and used, for example, by Bobrow and others (1977) in a non-educational AI system GUS for taking airline flight bookings. Knowledge is grouped in collections of information called frames, which have slots into which new information can be placed by the system as it is obtained from the user (or student). There is a similarity here with videodisc frames to represent educational knowledge, which are also collections of information, though it is information that cannot be directly modified. A frame system could be a useful method of organisation for the student model in a course-oriented ITS.

In a system called WUSOR, Goldstein and others introduced the notion of the 'genetic graph', not unlike the semantic network but with rules as the nodes, their interrelationships being represented by links. (See Goldstein and Carr, 1977.) WUSOR was in fact a system to teach the game 'Hunt the Wumpus'. Goldstein (1982) comments that the 'coach' (called a tutor in the UK) "has evolved from an unordered skill set to a genetic graph of skills linked by their evolutionary relationships", and it seems that this kind of network approach to knowledge organisation is most suitable with knowledge possessing a very loose, almost random, structure. A whole course ITS would normally deal with subject knowledge with a definite and fairly rigid hierarchical structure.

It is noticeable that knowledge domains used in ITSs so far have tended to be in the fields of either mathematics, or computer languages. Out of 28 ITSs listed in Appendix 1.1, 19 are in this category, and 4 more deal with largely mathematical problem-solving, in electronics, spectroscopy, engineering and circuitry. This comprises 82% of the total. The remaining 5 systems deal with domains that have clearly been chosen for their suitability for the techniques adopted: geography in SCHOLAR to illustrate semantic networks; rainfall in WHY to illustrate script-based dialogue; medicine and the highway code in GUIDON and TUTOR to illustrate large knowledge base manipulation; and an exploration game in WUSOR to illustrate the genetic graph.

Many of the domains used in ITSs seem to be almost the only ones that could have been used in that type of system. It was hoped here to design a whole course ITS which might achieve a greater degree of universality and 'portability'.

2.3.4 Tutoring strategies

There are at least six strategies that have preoccupied ITS workers. These are the game strategy; the environment strategy; the monitor strategy; the debugging strategy; the machine-learning strategy; and the theory of learning strategy. These will be looked at in turn.

The system WUSOR, due to Goldstein and others (Stansfield et al, 1976; Goldstein, 1982) proposed the notion that the learning of an exploration game, 'Hunt the Wumpus', could be used as an analogy for educational learning. Another system, WEST, taught arithmetic through a game, 'How the West was Won' (Burton and Brown, 1982). This 'game' approach to CAL will be discussed in a later chapter, as it has useful things to say for a course-oriented ITS.

Some systems have aimed at creating a learning or reactive 'environment' rather than direct tutoring. The philosophy behind this approach comes from CAL programs such as LOGO (Papert, 1980) and SMALLTALK (Goldberg and Ross, 1981), which do not claim to be intelligent or even tutors. The BASIC environment called BIP, due to Atkinson (1975) and others does make this claim, maintaining a student model and posing tasks for the student on the basis of a tutorial strategy. SOPHIE (Brown and Burton, 1975) is

another reactive environment, using a natural language parser. There is no student model; the student finds a fault in an electronic circuit by question and answer dialogue. Other systems using an environment approach are SPADE (Miller, 1982), ACE (Sleeman and Hendley, 1982) and MENO-II (Soloway et al, 1983).

Ferguson (1984) has examined the advantages and disadvantages of the exploratory environment and the structured approach, and his findings are reproduced in Figure 2.1, (i) and (ii). Perhaps a course-oriented ITS aiming to cover a large amount of material with a variety of students should possess features of both these approaches, the reactive environment and the structured approach, with possibly a means of combining the two as well. If the system is to be extensible to other knowledge domains, one approach would be more appropriate for some, another for others.

Some systems which do their tutoring more overtly are described as 'monitors', which do not interfere with the student until he or she goes wrong, when they step in with advice. ACE, for example, (Sleeman and Hendley, 1982) which teaches spectroscopy, is called a 'problem-solving monitor' or PSM. This monitor type of tutorial strategy is popular because it is "possible to avoid some of the more complicated problems of user modelling", (Ross et al, 1986). A more recent system, SPIRIT (Barzilay and Pople, 1984), also adopts this policy. The 'monitor' approach was seen as potentially useful for a whole course ITS, but the system would probably need to 'take charge' for a large part of the time when introducing new knowledge.

If a system is to intercede when the student goes wrong, it needs to detect 'bugs' or faults in his or her knowledge or reasoning. WHY (Stevens and Collins, 1977) investigated bugs in students' understanding, and this was pursued in more detail in BUGGY, a diagnostic system which detected student bugs in subtraction, originated by Brown and Burton (1978). In DEBUGGY the bugs diagnosed by BUGGY could be corrected (Burton, 1982), using a module called interactive DEBUGGY or IDEBUGGY. Another system which concentrates on a debugging approach is MENO-II (Soloway et al, 1983), which has a database of 18 common bugs in PASCAL as templates and attempts to find a mismatch with the student's efforts. Another 'debugging' ITS is Anderson's Geometry Tutor (see Anderson, Boyle and Yost, 1985)..

Figure 2.1

Tutoring Strategies (Ferguson, 1984)

(i) The exploratory environment

Advantages of the environment approach:

1. Allows for the integration of new knowledge into existing structures in a "natural" manner.

2. Allows for learning far beyond that which can be explicitly identified.

Some concerns:

1. There is no assistance for the student who gets hopelessly lost in the free environment.

2. It is not easy to construct new models, since the essential parameters and their relationships to each other are often not clearly specified.

3. It is not easy to bridge the gap between the intuitive aspects of a subject that may be acquired through exploration and the more formal aspects of the subject.

Figure 2.1

Tutoring Strategies (Ferguson, 1984)

(ii) The structured approach (Ferguson, 1984)

Some advantages of more structured tutoring:

1. It is often easier to provide guidance for the student.
2. It may be easier to examine models to gain insights for constructing new models.
3. The relationship between intuitive and formal aspects of a subject can be more clearly delineated.

Some concerns:

1. There is some question as to the completeness of explicit models.
2. Explicit models are often regimented, and may fail to allow for many of the individual differences in structuring knowledge.

An AI concept often used is that of a 'plan', or hypothetical sequence of actions, devised so as to work toward a 'goal'. In his MACSYMA ADVISOR, Genesereth (1982) tries to identify the plan the student is working to in solving a mathematical problem. The system does this by attempting to recognise plan fragments from the 'bottom up', and also recognise the goal the student is working towards, from the 'top down'. This is basically a 'debugging' strategy which aims to correct flaws in the student's approach.

The 'debugging' approach, if it can be perfected, will form a highly effective tutoring method in the mathematical fields where it is most actively pursued. However, it is highly domain-specific and unlikely to be portable to many other subjects. In dealing with coarse-grain knowledge as used in a whole course ITS, the average human teacher (rather than tutor) will not always attempt to identify individual misconceptions in great detail. The solution is usually to give the student certain information that he or she has not been given before, or failed to take in when it was given. In practice, the main concern is not to identify the 'bug' or misconception precisely, but to correct it. The right standard response from the teacher will frequently remove a large number of student difficulties.

Another theme in ITS is that of self-improving teaching systems. Notions of machine learning have been applied to ITS in several systems, notably by Kimball (1982) and O'Shea (1982). O'Shea's tutor teaches itself how best to teach quadratic equation solving by inspection, modifying its strategy as it learns from the results of successive students. More recently Self (1985) has proposed an approach to ITS based on machine learning in which the computer initially has no knowledge of the subject, only knowledge of how to learn, so that the student and computer learn side by side from a database of subject knowledge. Self's approach is discussed again in Section 2.6.

A problem that has slowed down work in ITS is the lack of an accepted and complete theory of learning. Anderson (1983) and his co-workers claim to have solved this problem with their Advanced Computer Tutoring (ACT) theory of cognition, which "falls at the intersection between cognitive psychology and artificial intelligence." ACT appears to be a compendium of various ideas current in artificial intelligence and the techniques embodied in logic programming, along with some common-sense teaching principles.

"Knowledge is divided into two categories: declarative and procedural. The declarative knowledge is represented in a propositional network, similar to other semantic network representations ... ACT can learn both by adding propositions to its database and by adding production rules." (Anderson and Kline, 1979). ACT also uses forward inference towards goals, backward inference from goals, instruction in a problem-solving context, and immediate, 'debugging' feedback about errors. Anderson and his co-workers suggest that "the learning principles derived from these theories provide the direction needed in the design of instructional software." (Anderson, Boyle and Reiser, 1985).

A recent development considered highly relevant to the tutoring or teaching module in the present whole course ITS approach is evidence of a return to traditional ideas of curriculum. This is dealt with in Chapter 5.

2.3.5 Student modelling

The student model has now become the central feature of many intelligent tutoring systems, virtually distinguishing such systems from other CAL systems which are either not intelligent, or not tutors. Soloway and VanLehn (1985) identify three types of ITS: the ITS with student modelling and tutoring; the reactive environment with tutoring but no student modelling; and the diagnostic system with student modelling but no tutoring.

Several systems have investigated the construction of a student model without attempting to do any tutoring. BUGGY was such a system. Another, the Leeds Modelling System (LMS), uses a production rule representation for rules and mal-rules the student uses, (Sleeman, 1982). "LMS is considered to have succeeded [in modelling the student] when the inferred model gives the same answers as the student on the problem set."

Work which is useful in considering student modelling has been, and is being, carried out in the field of the human-computer interface under the umbrella of user modelling. Benyon (1986) reports on MONITOR, a self-adaptive user modelling system, drawing on frames corresponding to stereotype users which are fed parameters for an individual user. His

prototype system deals with the field of CAL. Ross and others (1986) are also working on a user model for students learning the language UNIX, so as to tell them when they go wrong. This is similar in intention to SPADE and other ITSs described here and in Appendix 1.1.

Ross and others (1986) divide user-modelling generally (closely related to student modelling) into two approaches, predictive (e.g. BUGGY, LMS and WUSOR) or analytic (e.g. WEST, MACSYMA ADVISOR and GRUNDY). Ross comments: "All the existing models are, to some degree, OVERLAY models" involving matching with expert, or ideal student, stereotypes. The student's knowledge overlays an ideal knowledge structure, such as a genetic graph, and discrepancies are identified as areas where the student needs further study or practice.

A useful notion in designing a user model is that of the 'stereotype', used in the GRUNDY system for advising library users by Rich (1979). Stereotypes (or models) of readers are made up of a hierachy of frames which the system matches to the reader it is dealing with. The system can modify its standard stereotypes to match real users more exactly.

Another system, OPM by Hayes-Roth (1985), uses a 'blackboard' for organising its knowledge and reasoning about its own control. This envisages different hypotheses about the user entered two-dimensionally on a blackboard, the user model being a subset of these hypotheses. Thus different user models can be tried out and the best used in given circumstances. Each can be modified in the light of experience.

An ITS which is course-oriented will almost certainly require a student model, and its characteristics will be determined, as in other systems, by the aims behind the system. For example, it will probably need a knowledge component recording what the student has covered, and an ability profile recording aspects of student attainment. The student model may not need to be complex. It may be unnecessary that it should be based on stereotypes, for example, as in Rich's system, or that it should form elaborate hypotheses about the student on a blackooard as in Hayes-Roth's system. A student model consisting of a large, single frame might suffice, with student data slotted in as it is obtained.

2.4 Expert Systems (ES) and ITS

2.4.1 Brief overview of ES

Expert systems represent one of the most successful areas of the applied side of AI work. While A.I. attempts to fathom and reproduce human intelligence in some or all of its aspects, expert systems accept the limitation of relatively small knowledge and inference domains, and simulate specific intelligent tasks carried out by human experts. Classic examples are MYCIN, an expert system for medical diagnosis (see Shortliffe,1976), DENDRAL, which identifies molecular structures in chemistry (see Feigenbaum, 1971), PROSPECTOR, which assists geologists in finding mineral deposits (see Duda, Gaschnig and Hart, 1980), and MACSYMA, which assists in the solution of mathematical problems (see Genesereth, 1982).

Expert systems typically consist of three main parts: a knowledge base, a knowledge manager or 'inference engine', and a user interface (the term used by Bratko, 1986). This is shown in Figure 1.4. Forsyth (1984) distinguishes input from output, identifying an acquisition module and an explanatory interface, shown in Figure 1.5.

It was realised early on that the knowledge manager might be applicable to different knowledge bases, forming a 'shell' into which kowledge could be fitted. Such a shell was EMYCIN, derived from the medical system MYCIN, and meaning 'empty MYCIN'. Several expert system shells are now available commercially, such as Micro Expert or Expert Ease. An expert system shell has been published in BASIC which can be typed in and run on microcomputers (Naylor, 1983).

The standard type of expert system, which simulates the thinking of an expert in solving a specific problem with many variables, often uses some kind of inexact reasoning. Forsyth (1984, Chapter 5) distinguishes between the use of fuzzy logic, certainty factors and Bayesian logic. He points out that all these have "proved their worth in serious applications", but "have their critics too". Bramer (1984) is critical of methods of dealing with uncertainty (as he is of other aspects of expert systems) but he observes that they work in practice, and there is no reason to abandon work on them, "just the reverse".

Another central feature of the 'standard' expert system is the use of rules. A typical rule is of the type: "IF the patient is sneezing AND the patient's nose is running AND the patient feels ill THEN the patient has a cold with certainty 0.9." Such rules can be collected together in a rule base, and the program searches through them as required to find which ones are true in a particular case. The assumption is made that such rules can be handled independently of each other (not always justified, see Bramer, 1984). Rules can be added as required to allow for further contingencies. In some systems the program can add rules itself as it goes along, and this is the basis of one form of machine learning. At the end of the search the rules which apply are combined using inexact reasoning to reach a final solution to the problem.

2.4.2 ITSs as expert systems

It would be possible to plan an ITS as an expert system which simulates the human tutor, the three parts being clearly delineated. The knowledge base consists largely of subject knowledge, plus additional knowledge such as questions. The knowledge manager handles the knowledge according to the expertise of a teacher. The user interface consists of data that is built up relating to a particular student.

There is a type of ITS that is derived directly from the standard expert system. MYCIN, which carried out medical diagnoses, offered the possibility of a teaching system for medical students, who could take case study symptoms and compare their own diagnoses with that of the system. Clancey (1979) added two levels to the original MYCIN to form an intelligent teaching system, GUIDON: a 'support level' to justify individual rules, and an 'abstraction level' to organise rules into patterns. Clancey comments that "it is desirable to add teaching expertise and other levels of domain knowledge to MYCIN-like expert programs if they are to be used as educational programs. Furthermore, it is advantageous to provide a flexible framework for experimenting with teaching strategies, for we do not know the best methods for presenting MYCIN-like rule bases to a student."

Self (1985) is critical of systems of this type, on several grounds. First, they concentrate on the wrong subject matter for education, i.e. the domain

of an expert. Second, "the proposed methodology focuses first on expertise and only as an afterthought on what learners actually do and know." Third, they are rule-based, and the rules which work well for an expert system do not explain satisfactorily to a student what is happening, and need to be restructured for teaching. Fourth, such systems are designed with an emphasis on performance, with no attempt to simulate how experts think. Ford (1984) also criticises GUIDON, saying: "The teaching strategy is ... regimented to accord with the expert's view of the subject ... it is not uncommon for subject experts to be poor teachers."

Soloway and VanLehn (1985) are even more critical, citing this sort of system as an example of "How NOT to build an ITS". The strategy NOT to follow is: "1. Get an ordinary expert system for the domain. 2. Wrap an expert tutoring system round it. Result: Mediocre tutoring. Worse: Can't improve tutoring without changing the expert. The Lesson: Expertise must be EXPLICIT!"

In a detailed report, Fisher and Howe (1982) examine the potential of expert system based training aids. The authors are from the Department of AI, University of Edinburgh, where much of the pioneer work was carried out on expert systems, and their analysis is derived from MYCIN-like rule-based systems. They recognise that a teaching system needs to go some way beyond the three component standard expert system model, and some idea of their thinking can be obtained from their diagram of a hypothetical expert training system structure, given in Figure 2.2. There is a suggestion of a course-oriented approach here. Fisher and Howe identify five potential roles for expert training systems, and these are reproduced in Figure 2.3. A whole course ITS needs to fulfill most if not all of these functions, particularly those of adviser, examiner and tutor.

It is interesting that Fisher and Howe (p.81) conclude that the highly interactive nature of GUIDON is what makes it most problematic as a tutoring system, not the expert nature of its knowledge, as do the critics already cited. They recommend a Demonstrator expert training system characterised as follows: "This type of system incorporates the more passive features of the GUIDON system and omits the student modelling and tutorial aspects. Further, the system solves the problem (in the Demonstrator system), rather then the student (as in the GUIDON system)." Fisher and Howe (p.81) estimate 4 man-years to construct this Demonstrator system. They comment that "a more sophisticated system, such as Clancey's

GUIDON, would be a preferable alternative ... Unfortunately, much research remains before such systems will be successful."

It is also interesting that the Demonstrator system described by Fisher and Howe resembles the prototype system which resulted from the present project in several respects, and that the time estimated for its construction approximates to the time which has been spent on this project. In the case of this system, additional time has been spent on preliminary investigative work and on gaining expertise, but on the other hand time has been saved by using a ready-made videodisc containing much of the expert knowledge.

In short, ITSs derived directly from rule-based expert systems are inherently difficult to adapt as tutoring systems, and would seem to be unsuitable for teaching new knowledge, which is one of the things a whole course system needs to do. They are most likely to be of use for reinforcement exercises when a student has reached a level of ability comparable to that of an expert. This is not to say, of course, that techniques used in such systems (such as the use of rules and reasoning about uncertainty) would not be of use in a whole course ITS.

2.4.3 Expert system features usable in ITSs

One contribution of the field of Expert Systems which is applicable to ITSs is the notion of a transferable shell. This has been expanded upon in Section 1.4 as a desirable feature of a course-oriented system.

Most descriptions of expert systems stress that they are only applicable to a certain type of problem. Yazdani (1984) says: "Currently, any diagnostic application which depends on the knowledge of a very narrow domain is a 'good' application, while anything depending on creative and commonsense reasoning in a wide-ranging domain is a 'bad' one." More specifically, the type of problem expert systems apply to is one where a single problem is to be solved, and the solution depends on the relevance of a large number of symptoms with no clear relation to each other, as in MYCIN.

Figure 2.2

Hypothetical Expert System training structure
(Fisher and Howe, 1982, p.72)

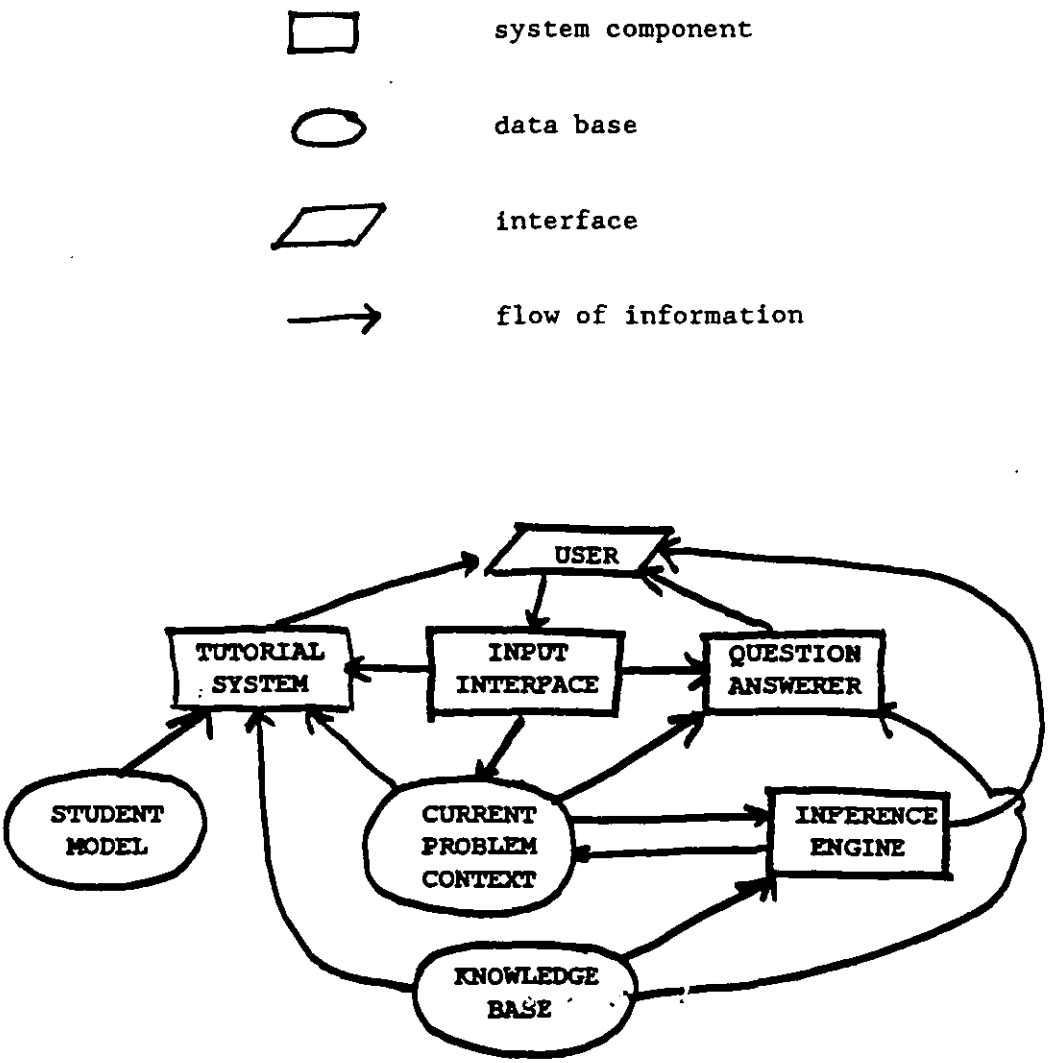


Figure 2.3

Potential roles for expert training systems
(Fisher and Howe, 1982)

What roles might potential expert training systems have? The roles are:

Adviser - the system provides advice to the user who is attempting to solve a problem. This is a fairly minimal training system. In that the training comes by way of the experience gained in the course of solving actual problems. The burden upon the system is substantial.

Demonstrator - this system demonstrates the solution of typical problems, in order to give the student some conception of the use of domain methods.

Examiner - the system evaluates the student's problem solving or decision making capabilities. This probably entails the training system presenting the student with a problem, soliciting a response, and critiquing the student's solution.

Exerciser - the focus of this system is upon exercising the user's existing skills - so as to help him/her gain experience with actual problem solving and understanding of the relevance of factual knowledge.

Tutor - this system focuses upon the complete educational task - which includes the teaching of both the factual information and the conceptual relationships needed to relate the facts to their use. This role is the most difficult and requires the most "human-like" qualities - understanding of the subject and the student, and how best to teach the material.

Figure 2.3 (contd.)

We now list four types of structures which can contain elements of the above roles.

1. **Review/refresh** - to exercise and remind the trained user of appropriate information.

2. **Drill device** - to provide practice in the knowledge and reasoning methods taught in the course work.

3. **Skill enhancement** - to augment the student's basic understanding with the expert's conception of the rules used in the domain.

4. **Experimental laboratory** - to provide a framework supporting problem solving through simulation of specific example systems.

There is a problem of this type which will figure in the design of a course-oriented ITS, namely, to determine the ability of the student at a given moment in specified areas. This is a problem capable of a rule-based solution, the rules being of the type: "IF the student answers B rather than A, C, D or E for Question 37 THEN there is a probability that the student has proficiency of recall 0.6 and proficiency of understanding 0.3." A large collection of such rules could be expected to yield a profile of the student's ability.

2.5 Declarative and Procedural Knowledge

2.5.1 Imparting new knowledge in ITSs

One feature of the ITS field seems to stand out. ITSs have 'traditionally' been concerned almost exclusively with interactive applications which rehearse skills already encountered, and have neglected the teaching of new knowledge.

There seems to have been an acceptance that ITSs would not instruct, or teach, or impart knowledge didactically, but would interact, or test, or diagnose, or impart knowledge heuristically. It is likely that in a course-oriented ITS there will have to be some didactic teaching as well as interaction. This could be an area where the approach required in a whole course ITS will differ from that used in previous ITSs.

Miller says of didactic systems: "They can direct attention away from bogus issues previously thought crucial ... didactic systems with limited objectives can lay the foundations for later work on more sophisticated and complete tutors. They can drive theory evolution, and suggest alternative applications." (Miller, 1982.)

Fisher and Howe were also recommending investment in a 'demonstrator' training system in 1982 in which "the student would be able to select among available topics, request greater detail, ask some simple questions or merely passively follow the presentation." (Fisher and Howe, 1982.)

There is much discussion of declarative and procedural knowledge in ITSs. The first-time teaching of factual, declarative knowledge may become a goal of research in the future.

2.5.2 Declarative and procedural knowledge defined

There is some preoccupation in ITS with what are referred to as declarative and procedural knowledge. (A third type mentioned is causal knowledge, which will not be considered here. Richardson (1988) comments that this "... is so new to artificial intelligence that ITSs for this type have not

been implemented.") The notions of declarative and procedural knowledge, thus called, derive from Anderson's ACT* theory of knowledge, described in Anderson, 1983.

Anderson defines declarative and procedural knowledge thus: "There are domains like calculus problem solving where the main knowledge to be communicated is procedural, that is, knowledge about how to perform a task. There are domains like geography, where the tutorial goal is to convey declarative knowledge in the form of a set of facts ... " (Anderson, 1988, p.34.) In spite of such examples, it is clear that all subjects to be taught will contain elements of both declarative and procedural knowledge.

The terms are similar to those programmers use when discussing languages such as Pascal, which is a procedural language (programs consisting of a list of pre-determined instructions to solve a problem), and Prolog, which is a declarative language (consisting of a database of facts and rules which can be interrogated to solve problems). It seems that in the teaching context the terms refer to background facts taught for the first time (declarative knowledge) and manipulative or problem-solving skills (procedural knowledge). The first can be taught to the student as a one-way, didactive process, the second is best taught by a two-way, interactive method.

VanLehn offers the warning: "The distinction between procedural and declarative knowledge is notorious in artificial intelligence as a fuzzy, seldom useful differentiation." However, he goes on to use it, and it seems a useful one here.

Anderson acknowledges that there cannot be procedural knowledge without some acquisition of declarative knowledge: "... it is part of our general theory of knowledge acquisition (Anderson, 1983) that knowledge must start in a declarative way before becoming proceduralised." (Anderson, 1988, p.40.) Thus one cannot execute a skill without at least some basic factual knowledge of the quantities involved.

2.5.3 A declarative front-end for ITSs

The traditional method of teaching is to put across declarative knowledge initially using a didactic presentation, then to put across procedural knowledge by the student practising the skills involved.

Most ITSs have been preoccupied with teaching procedural skills, in line with the common belief that computers are good at interaction, but not good at presenting first time knowledge. As an extension of this belief (which will later be shown to be no longer true, with interactive video available) interaction has been seen as the field of ITS, with the presentation of first time knowledge left to conventional CAL via branching tutorials (or to books and handouts).

VanLehn (1988) classifies several ITSs as declarative, namely GUIDON, SCHOLAR, WHY, MENO and PROUST, but this classification seems debatable. It is true that GUIDON, the expert system based medical ITS, deals with declarative knowledge of diseases and symptoms, but this knowledge is assumed to be known, and the aim of the system is to tutor a skill, medical diagnosis. Likewise, SOPHIE taught the skill of electronic trouble-shooting or problem solving.

It is now being realised that the teaching of declarative knowledge is an essential and worthy aim of ITS. Anderson comments: "Another need for declarative tutoring is illustrated in our LISP tutor, for which we have created a special textbook (Anderson, Corbett and Reiser, 1986) for teaching the declarative underpinnings of the procedural knowledge the LISP tutor teaches. It clearly would have been better to have extended the LISP tutor to cover what is in the textbook." (Anderson, 1988, p.40.)

Elsewhere Richardson says: "... there is absolutely no reason why some initial effort cannot be made in developing ITSs that formally represent and teach both the declarative and procedural aspects of a domain. One approach would be to take an existing ITS that is procedural and augment it with declarative knowledge." (Richardson,1988, p.244.) It seems that the field of ITS in the US is moving towards the concept of teaching all parts of a course, both 'declarative' and 'procedural', and it also seems from Section 1.4.4 that some are moving towards the idea of an ITS as a transferable shell.

Another frequently used way of looking at expert system shells is to regard them as 'front ends' that can be 'bolted on' to the front of a knowledge base. A declarative front end for procedural ITSs is what Richardson is describing above.

Much of this thinking, which has emerged in print only in the last twelve months, was adopted in this research five years ago. In an ITS adopting a course-oriented approach, the problems of presenting new knowledge need to be addressed directly, as they are an essential part of a complete course.

2.6 An Appraisal

Considerable achievements have been made in the past, but there is evidence now that several observers feel there is need for a new approach. This country in particular is not being radical enough, according to Heaford (1983), quoted in Section 1.1. Heaford continues, on the subject of ITS: "Radical thinking is now a vital requirement if Educational Technology is to advance beyond mere attempts to enhance today's learning systems."

As mentioned in Section 1.2.1, most systems to date have dealt with small curriculum areas. They have been run on expensive computers, and have not been practical or usable. The reader is referred back to the comments on the limitations of ITSs by several writers, quoted in Chapter 1.

There seems to be some consensus of opinion that a fresh direction is required for work in ITS, but less consensus on what it should be. There has been considerable interest in the possiblity of adapting the successful techniques of expert systems to CAL, in such systems as GUIDON (see Clancey, 1979). This has been dampened by the realisation that the techniques do not transfer easily, as discussed in Section 2.4.

Self (1985) proposes a machine learning approach to ITS, suggesting that this might yield more than an expert systems approach. Future ICAI systems should be focussed on the learner, with a student model as the central component. The student would learn along with an intelligent learning

machine, an expert not in the subject matter but in the process of learning. However, attempts to design machines to teach how to learn, when there is so little consensus of opinion on how human learning takes place, might be over-ambitious. In addition, the field of machine learning is barely well enough developed to tackle such a task. There is also some doubt whether students, faced with the choice of a tutor who is expert in the subject and a tutor who is simply an expert at learning, would fare better with, or prefer, the latter.

Several possible approaches are suggested by the examination of the fields of ITS contained in this and the preceding chapter. One possibility for a new approach, on which this research is largely based, is that of substituting a course-oriented approach for the hitherto predominant topic-oriented approach.

An approach now sometimes adopted, derived from the field of Expert Systems, is to develop transferable ITS shells, as discussed in Section 1.4. (This was by no means a clear and distinct approach when this research was commenced in 1984.) Another possible approach argued in Section 1.5, and not usually followed at all, is a teacher-oriented approach, using established classroom teaching expertise.

In this chapter it has been argued that although expert systems used directly as ITSs have been disappointing, techniques used in expert systems can be used also to advantage in an ITS, such as production rules and inexact reasoning. It has also been argued that the teaching of first-time declarative knowledge has been neglected in the ITS field, and that there is a need, as Richardson (1988) says, for declarative 'front-ends' for the more usual interactive and procedural ITSs.

Chapter 3

A Survey of Computer Assisted Learning

3.1 Introduction

This survey of the field of Computer Assisted Learning (CAL) is required because a course-oriented system will have its roots in traditional CAL. There is no real precedent in the field of intelligent CAL. The chapter will contain a brief review of this very large field, but clearly cannot be exhaustive. Intelligent applications have been dealt with in Chapters 1 and 2. An area of CAL of particular interest is the field of Interactive Video, which opens up new possibilities.

This survey is also necessary because there has been considerable progress in software outside of the ITS field. Students are used to using software which is now often of high quality, and ITS software, if it is to be used widely, will have to emulate this high quality.

Since the mid-seventies computer technology has advanced with dramatic speed. It is now common for an individual to own a computer as powerful as the ones on which TICCIT and PLATO (described later) were developed. The keyboard and television monitor have replaced the old punched cards, and programs now use a variety of types of text, sound output and excellent colour graphics. Developments like windows, icons, the mouse and pointers (so-called WIMP environments) mean that, as O'Shea and Self (1983, p.98) comment: "the ground rules of this particular game have changed considerably ..."

Computers in schools and CAL are regarded as highly important in the UK, by among others no less an authority than the government itself. In 1979 the Microelectronics Education program (MEP) was set up, with a budget of £12.5 million (later cut to £9 million), among other things "to use the computer to enrich the study of traditional subjects" (DES, 1980). At about the same time a separate Micros in Schools Scheme was set up by the DoI to put a microcomputer in every secondary school by 1982. This was successful, but O'Shea and Self (1983, p.261) point out that the MEP has suffered from having no clear objective, and the Micros in Schools scheme has caused "effective standardisation on two or three British microcomputers centred on BASIC and based on increasingly outdated technology."

Software has advanced also, though inevitably lagging behind the hardware. Perhaps because it is difficult to protect good software from copying, big

companies have been less interested in its production, and have seemed to encourage the production of large quantities of cheap software on a so-called 'cottage industry' basis to support the sales of hardware. Not surprisingly, while there is a substantial quantity of CAL software available, it is of great variety and variable quality.

Previous ITSs have exclusively used digital computer storage methods for storing knowledge. This can make it difficult to store and retrieve large quantities of educational information, though large quantities can be stored in present systems if necessary. More importantly, it makes it a laborious task to input information, which normally has to be typed in and formatted correctly, usually requiring an authoring system. Even more important is the fact that certain types of knowledge, for example how to carry out a skilled operation, cannot be clearly demonstrated and stored in digital form.

The problem of knowledge storage is perhaps the most important reason why many ITSs have seemed incapable of extension beyond the small 'domains' they have dealt with, and why 'ITS has clearly abandoned ... providing total courses' (Sleeman and Brown, 1982). If an ITS deals with a small topic, perhaps one per cent of a whole course, and uses up most of the storage capacity of a mainframe computer, the problems of covering a whole course are daunting ones.

A solution lies in using analog rather than digital storage, in the form of optical videodisc. This relatively recent technology, under its name of 'interactive video', has been described as having "tremendous potential for information storage ... a versatility unequalled in any other medium." (Parsloe, 1983.) Apart from greater information storage, such an analog approach makes the input of information in some ways faster and easier, as direct camera-type techniques can be used. The feature which gives the analog approach the deciding advantage, however, is that complex educational techniques such as the demonstration of skills can be presented (even social skills), virtually impossible with digital techniques alone.

Some dismiss videodisc technology as just another peripheral, like the disc drive or printer. (O'Shea and Self, 1983, discuss it under 'peripherals'.) It does appear, however, to be somewhat more: like the television and the computer, of which it is a hybrid, it requires its own specialised 'software' or 'courseware'. As such it provokes new thinking, and

generates its own brand of creativity. The BBC Domesday project has demonstrated that it can open up new horizons, and Laurillard and others have shown that new approaches to teaching are possible.

3.2 Computer Assisted Learning (CAL)

3.2.1 The field of CAL

There are many acronyms in the field, and the main ones are given in Appendix 3.1. Romiszowski (1986, p.267) comments that CBL, CBI, CBT, CAL and CAI all mean much the same to most authors, but are "endowed by others with slight shades of special meaning." The general term of computer assisted learning (CAL) will be prefered here, because this is the term with the longest pedigree. A journal in the field which began in March/April 1985 has been called the Journal of Computer Assisted Learning (Lewis, 1985).

Romiszowski divides the field of 'computers in education' into three in a diagram reproduced in Figure 3.1: the computer as a tool for educational management (CML), as a tool for the teacher (CAI), and as a tool for the learner. Strictly speaking, the last of these should be covered by CAL, but Romiszowski acknowledges that CAL has become an all-embracing generic term (p.268). All these three aspects of the computer are relevant in this project.

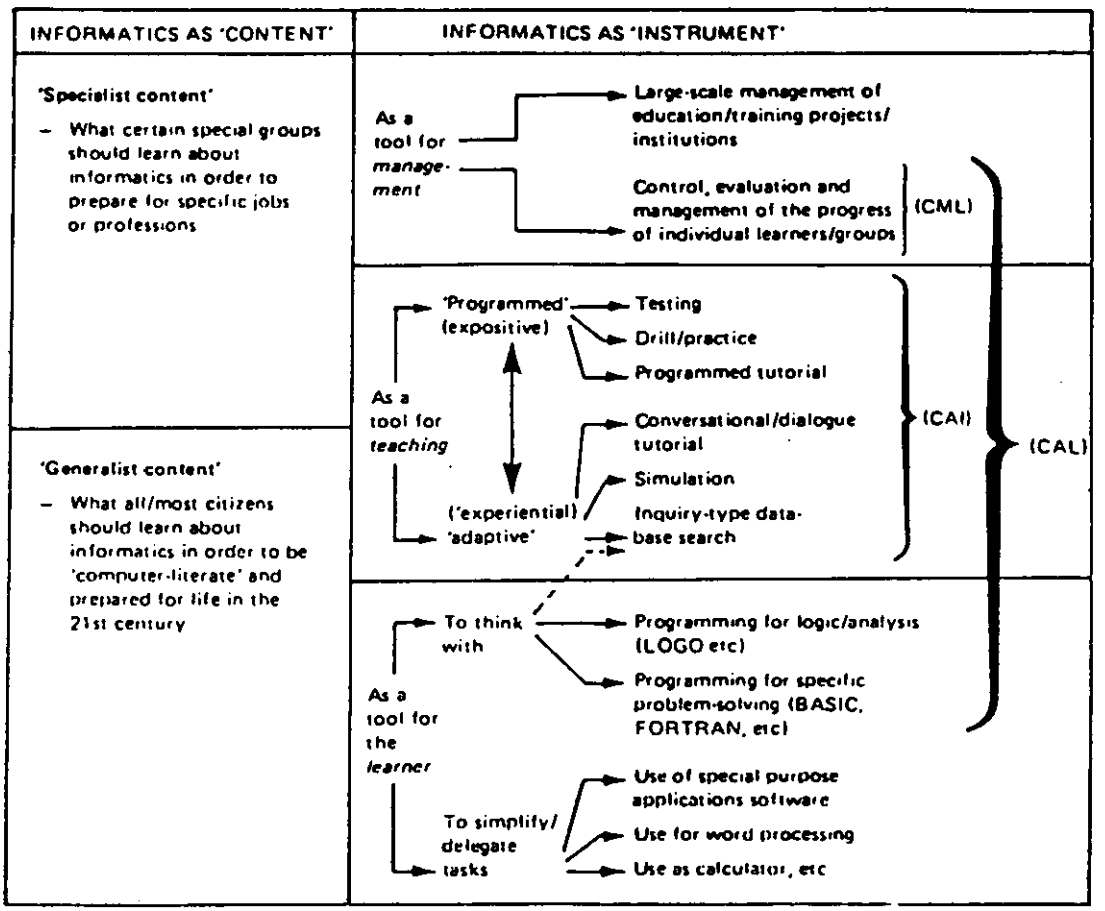
It is worth noting that the word 'training' is normally used in industrial applications where 'teaching' or 'learning' would be used in academic circles. Indeed, as Heaford (1983, p.2) points out, the term computer based training, CBT, "has become the industry 'standard' for all aspects of using the computer in training and education."

3.2.2 Pre-microcomputer CAL

The simplest, earliest form of CAL program was one in which the student was asked successive questions, and moved on to others as he or she gave

Figure 3.1

The field of computers in education (Romiszowski, 1986, p.268)



correct answers. The questions were structured to assist the student, and wrong answers were virtually ignored. This was called 'linear' teaching, based on Skinner's (1954) mechanical 'teaching machines', using operant conditioning. Following linear CAL came branching CAL, which responded in different ways depending on the student's answer. On giving a wrong answer, the student was given hints and another chance to find the right one. Such programs provided 'feedback' for the student, and were 'adaptive' to some extent to the student's needs.

Intrinsic programming, a type of branching program proposed by Crowder (1959), used multiple choice questions with several partially correct answers, branching accordingly. This was a foretaste of today's menus, discussed later in Chapter 5, and as a testing technique has recently been lost sight of. Multiple choice testing is used a great deal nowadays, but with only one correct answer, the others being 'thrown away' when given. This is in itself a mechanical, machine-like approach which a human teacher would not use in verbal questioning, and will be returned to later in Chapter 7 as it offers an opportunity for 'intelligent' testing.

The early CAL work in the sixties culminated in two major American projects, discussed at length by O'Shea and Self (1983). They were begun in 1971 and carried out over five years, funded by the National Science Foundation of America (NSF).

One of these projects was called Time-shared Interactive Computer Controlled Information Television, or TICCIT. It was designed to be a whole course system, with large quantities of course material produced by teams of experts in a variety of subjects, using the best of branching programming and colour videotapes as backup. The other project was called Programmed Logic for Automatic Teaching Operation, or PLATO, and had a markedly different approach. It was much more loosely organised, with various authors putting original teaching material into the system using a special authoring language, TUTOR. Student participation was voluntary. While TICCIT was intended to perform a substantial amount of the essential coursework of two community colleges, PLATO was intended to feed into a large network, with terminals in a large number of colleges, which would draw on the facilities of the system as required.

Neither of these two systems has been universally or even widely adopted, and therefore neither has been outstandingly successful. However, TICCIT

continued to be used at the pilot colleges and software from both projects is still being sold, now on microcomputer.

3.2.3 Why early whole courses failed

It is perhaps unfair to say that early CAL was unsuccessful, though this is generally thought to be the case. It failed mainly in the ambitious objectives which it set itself. As a new field, it continued to grow, and it may yet conceivably achieve those objectives. But TICCIT did not succeed at the time in producing converts to the idea of total courses on computer, and PLATO did not succeed in inspiring a network of educational computer terminals across America. Ford (1984) comments: "In spite of encouraging results, TICCIT has not been widely adopted." O'Shea and Self have an explanation:

"By an unfortunate accident of history, computers were becoming widely used at the time when the 'teaching machine' bandwagon began rolling. Inevitably, people began to use computers to do their linear programming. The poverty of linear programming is so manifest that the technique has long been extinct in computer assisted learning."

Skinner (1954) originally saw the teaching machine as a mechanical device with a knob turned by the student, and the early computer programs were not much more advanced, even when they started to use branching. This labelling of the computer as a 'teaching machine' has had long-lasting consequences. Heaford (1983) rather guardedly called his book 'Myth of the Teaching Machine', and a television series in which O'Shea (1985) appeared was pointedly called 'The Learning Machine'.

Systems such as TICCIT and PLATO were implemented on expensive computers and it became apparent that they would not be cost-effective, particularly PLATO. The organisers of TICCIT, a television concern called the Mitre Corporation, adopted a hard-sell approach with staff of the colleges and declared their aim of trying to 'create a market'. It seems that the poor initial climate worsened into antagonism in some quarters. (See Morrison, 1975.)

This might have been overcome had the computers been as reliable and sophisticated as modern ones. With hindsight it can be seen what was

lacking, in TICCIT especially. The system used just upper case text, and simple graphics. In the days of punched cards, TICCIT had a specially designed keyboard, of which the students tended to use only one of the keys. The language and approach of the TICCIT software seems 'unfriendly' by today's standards. (See the example in O'Shea and Self, 1983, p.89.)

It is important here to examine why TICCIT 'failed', as it was the most ambitious whole course project ever attempted. O'Shea and Self appear critical of the course preparation, (p.87), mentioning the 'factory-like preparation of course material', but they ultimately lay the blame not so much on the basic educational aproach as on other factors: the aggressive, business-like methods, the inadequacy of the 'human-computer interface' in the form of the specially designed keyboard, staff opposition and arguments, and underestimation of software development time (p.91).

Thus the main reasons TICCIT is regarded to have failed were probably the limitations of the computers and software of that time, and the absence of favourable attitudes, both on the part of the organisers and in the educational community. There is reason to believe that these problems would be greatly reduced today. Hardware and software have improved beyond recognition, and computers are now accepted in education as useful tools with the potential of contributing considerably more.

3.3 Approaches to CAL Today

3.3.1 The computer as a tool for management

Maddison (1982, p.66) identifies six ways of classifying CAL: by subject, by mode of presentation, by programming technique, by educational paradigm, by psychological theory, or by clarity of structure. Both Romiszowski, and O'Shea and Self, list different types of CAL program, and these are given in Figures 3.1 and 3.2. Some types of program will be looked at in more detail in the next section, using a combination of these groupings. Dialogue systems are discussed under natural language in Section 5.8.

Figure 3.2

Types of CAL program listed by O'Shea and Self (1983) p.293

Approach	*Distinguishing characteristics*	*Illustration*
Linear programs	Derivation from behaviourism; systematic presentation; reinforcement and self-pacing.	Last (1979)
Branching programs	Corrective feedback; adaptive to student response; tutorial dialogues; use of author languages.	Ayscough (1977)
Generative computer-assisted learning	Drill-and-practice; use of task difficulty measures; answering student questions.	Palmer and Oldehoeft (1975)
Mathematical models of learning	Use of statistical learning theories of limited applicability; response-sensitivity.	Laubsch and Chiang (1974)
TICCIT	Team production of courseware: 'mainline' lessons; use of television and minicomputers; learner control.	Mitre Corporation (1976)
PLATO	Multi-terminal interactive system; visual displays; 'open shop' approach; concern over cost.	Bitzer (1976)
Simulation	Computer as laboratory; interactive graphics; typically small programs.	McKenzie (1977)
Games	Intrinsically motivating; audio-visual effects; often lacking educational aims.	Malone (1980)
Problem-solving	Computer as milieu; programming by children; derivation from Piaget's theory and artificial intelligence.	Papert (1973)
Emancipatory modes	Computer as labour-saving device; task-oriented; use of microcomputers and public information systems.	Lewis and Tagg (1981)
Dialogue systems	Tutorial strategies; use of natural language; mixed initiative; use of complex knowledge representations.	Carbonell (1970)

There is a large-scale approach to the use of computers in schools and educational institutions which resembles their use in business: holding staff and student office records, storing data relating to equipment and buildings, word processing in the office, and so on (see Figure 3.1). This approach is only indirectly related to the education of students.

Another approach is to use computers to help in classroom management, in helping the teacher to organise such things as students' learning, equipment requirements, assessment results and reports. Programs are available to enable a teacher to save and process marks for a class, and a project called WRAPP offers assistance in compiling student records and profiles (Humphries, 1988). Another long term project funded by the Scottish Education Department offers teachers computer assistance with testing students in different areas of the syllabus, recording and analysing the results and preparing detailed student profiles (Mitchell et al, 1985).

It is necessary to recognise the importance of management of the student's learning and assessment in a course-oriented ITS. This is returned to in Chapter 5.

3.3.2 The computer as a tool for teaching

PROGRAMMED TUTORIALS

The modern version of the programmed learning type of program includes linear and branching techniques, in what Romiszowski (1986, p.298) calls the 'programmed tutorial mode'. It is possible to buy simple programmed learning CAL programs for microcomputers, but there is a tendency now to provide authoring languages instead, so that the teacher can write his or her own (e.g. Microtext or IVL). Books are appearing to help teachers do this (Heaford, 1983; Romiszowski, 1986).

Romiszowski (1986, p.299) lists three basic principles of instructional programming, which clearly need to be borne in mind in a whole course as in a limited tutorial:

* Active participation by learners in the learning process.
* Immediate knowledge of results and corrective feedback.
* Avoidance of excessive errors on the part of the learner.

The programmed tutorial offers considerable possibilities of extension in a whole course approach, but care needs to be taken. Pure programmed learning has a bad name educationally, as Maddison (1982) points out: "Programmed instruction, through texts and teaching machines, has been tried, and, in Britain, has failed to achieve widespread use in schools. This may be partly due to ... reaction against initial overselling." At present the programmed tutorial approach, to give it its more sympathetic name, is regarded with suspicion by educators, but it is starting to be taken up by industry as a cost-effective way of training staff. Good graphics and techniques such as windows, icons, mouses and pointers (WIMP) and videodisc have made radical improvements on the early programs.

DRILL-AND-PRACTICE

This is a slightly derogatory term for the basic techniques of assessment by computer. Questioning in drill-and-practice differs from questioning in a programmed tutorial in that the student's score is kept, rather than (or as well as) the answer being used to determine his or her subsequent route. If the score is kept from the student at the time, the technique is testing pure and simple. If the student is given his or her score at the time, and perhaps given help and another try before being told the correct answer, the process involves teaching as well as testing.

Drill-and practice can be quite sophisticated. In its simplest form a store of questions is kept and worked through by the program, but with suitable subjects, like mathematics, it is possible to generate questions of a standard form. Thus the student gets different data each time, but the program is much more economical. Some programs include a 'task difficulty model', so that the degree of difficulty is tailored to the student, and as his or her skill increases, so does the difficulty of the questions.

An example of a successful drill-and-practice program is SAKI, a keyboard instructor for typists developed by Pask (1960). Pure drill and practice seems most applicable to developing mathematical, pyschomotor or perceptual

skills. A task difficulty model is close to the ITS notion of a student model, and generative programs of this type are precursors of 'intelligent' programs. Techniques of this kind are transferable to a whole course ITS.

SIMULATIONS

Some educational topics are taught very effectively using a computer model or simulation. For example, the UK Atomic Energy Authority and the Computers in the Curriculum Project (1985) have produced a simulation of a nuclear reactor, a topic which by any other treatment is bound to be highly theoretical. Other subjets can be taught very much more easily and graphically using a simulation; programs are available in Physics to simulate radioactive decay, planetary motion, satellite orbits and other subjects.

Romiszowski (1986, p.307) lists several reasons why simulations are the fastest growing type of educational program: for example, they are sometimes the only way of developing certain types of learning experience; they use the speed and storage capacity of the computer to best advantage; they are often easy to plan, if not to program; and they are strictly in the 'teacher's help' category, and do not alarm teachers. Romiszowsky also lists four types of simulation: the system facsimile, the apparatus simulation, the decision-making exercise and the process model. Clearly, a facility to call up simulations to cover certain topics would be a useful one in a whole course system, but producing a simulation is usually a substantial programming project in itself.

It is worth noting that advanced flight simulators can be used to train pilots in the complete absence of real aeroplanes or human instructors, until the final stages of instruction, and can provide something like a whole course. In one sense, a whole course ITS system might usefully be regarded as a complex decision-making simulation of the teacher.

3.3.3 The computer as a tool for the learner

PROBLEM SOLVING ENVIRONMENTS

These represent an approach in which the student explores and finds out how to do things and solve problems. Several computer programming languages with 'interactive interfaces' could be described thus, such as interpreted BASIC or PROLOG.

The approach is closely associated with Papert (1980) and the language or environment of LOGO. This consists essentially of a system enabling children to build up functional groups of computer instructions, identified by words, with which they proceed to build up more and more complex systems. It is aimed at quite young children, and has achieved some success with them, though what precisely is being learned is hard to assess. More recently the language or 'series of software systems' called SMALLTALK has attempted to provide an environment at the same time simple enough for children to use, and sophisticated enough for adult programming. (See Goldberg and Ross, 1981.)

It could be advantageous to look upon a course-oriented teaching system, not as a course to be worked through, but as an environment to be entered and interacted with.

INQUIRY PROGRAMS

This is the term used by Romiszowsi (1986, p.312) for programs that support a large knowledge base, which a student can either browse through as 'serendipity learning' (Rushby, 1979), or search through in a structured manner. There is a resemblance here to a problem solving environment, but the aim is to enable the student to acquire knowledge, not develop problem-solving skills. O'Shea and Self include such use of databases as an emancipatory mode (1983, p.113) but it seems to merit a separate classification here.

Databases on micros have until recently been constrained by RAM and disc storage capacities, but with the latest 16-bit micros with hard disc drives

this is no longer a problem. More important, videodisc now offers the possibility of a database consisting of over 50,000 frames of information, each immediately accessible. The BBC has recently produced a 'Domesday Disc' containing a vast number of geographical frames on Britain (see Linderholm, 1987). Its main use will be as a database for both serendipity learning and guided investigation.

Romiszowsi (1986, p.313) says of 'heuristic databases': "This vision, to become a functioning reality, will need a lot of organisation of the information of these 'great libraries' in such a manner that the not-too-skilled potential user can find his way about and locate what he wants, or what he might want if he only knew that it existed. The task of organising such immense databases is not all that well understood."

In a course-oriented ITS which uses videodisc for its knowledge store, a facility for heuristic or discovery learning by the student will be desirable as well as structured instruction. This raises the question of the degree of choice to be given to the student, or the balance between choice and guidance, and this will be discussed further in Chapter 4.

EMANCIPATORY PROGRAMS

Referred to thus by O'Shea and Self (1983, p.113), this is a type of program which performs the mundane or difficult parts of a task for a student, releasing him or her to concentrate on the interesting or salient points without distraction.

Calculators perform this function in mathematics, and computers, as well as performing calculations, can help with experiments in science, for example by drawing a graph of the results. The degree of help students should be given is a matter of eucational judgement. It depends whether the student should be learning the subsidiary skill as well as the main skill, for example how to draw graphs as well as how to do the experiment. In practice 'emancipatory' programs are liable to be used to save time and get through the syllabus, without much regard for educational considerations.

An emancipatory aspect of a course-oriented ITS is that it should, like the human teacher, organise the student's study and learning route through the material of the course, and leave the student free to learn.

GAMES

Games, although rarely educational in themselves and regarded with suspicion by teachers, are inextricably linked to computers and education. Parents buy microcomputers with education in mind; children want their parents to buy them with games in mind. Perhaps the main point is that games have made computers popular, and this has opened the way to their use in education. An important feature of the modern microcomputer, although it is still evolving, is that it has already proved itself with students. They will sit for hours with an educational program, and the association with games has a lot to do with this.

Computer games started in the amusement arcades, and arcade games are now using videodisc. Fox (1983) sees videodisc games, like computer games, finding their way into homes: " ... any home with a computer-videodisc system bought for games playing will also unwittingly have bought the hardware necessary for running educational programmes."

Apart from this indirect influence, many educators and psychologists think that lessons from the study of games could be applied directly to educational programs. O'Shea and Self (1983, p.103) refer to research by Banet (1979) into features of successful computer games, and research by Malone (1981) into characteristics of intrinsically motivating environments. For high motivation, computer programs should have the following features:

1. They should use audio and visual effects well, particularly to reward success.

2. They should be adaptable in difficulty, so as not to be too hard or too simple.

3. They should keep the user aware of his or her progress, i.e. give scores.

4. They should present the user with a challenge to be overcome, or a goal to satisfy.

5. There should be elements of curiosity - the user should not be told everything.

6. They should include elements of fantasy.

Most of these features (except perhaps the last) could be borne in mind for a course-oriented ITS.

3.3.4 An appraisal of CAL today

Perhaps the widest generalisation that can be made about the diverse types of educational computer program available today is that nearly all are in the category of 'teacher's help', or 'tools' to give assistance. The whole course approach, since TICCIT, seems not to be contemplated.

CAL software for education, though not for industry, is mostly limited to small topics. This is mainly because the structure of schools and many colleges precludes the use of micros by students for long periods, and allows them to be used only on an auxiliary basis for short periods in class. Students are taught as classes, not as individuals, and computers are still a comparatively scarce resource. Also, teachers may feel threatened by the suggestion that a large part of a course could be taught by a computer, and react against such a notion. The influential Papert (1984) has said: "I do not believe there will be schools in 20 years time, perhaps only 10 years ..."

In 1983, IBM started a project comparable in scale to TICCIT and PLATO, spending $8 million in 101 educational institutions to which were donated some 1,500 IBM PCs complete with software, "to develop and refine a model for the effective use of computers in secondary schools". There was no mention of computers contributing substantially to whole courses, however. The aim was to "encourage the use of word processing in English classes, database management in science and social studies classes, electronic spreadsheets in business education classes, and graphics in art and vocational education classes." (Cline and Schneiderman, 1986.)

The recommendations of the IBM project, called the Electronic Schoolhouse, are either obvious or rather vague. They include "develop a written

plan..." without recommendations as to what the plan should be, "use a shared planning approach ..." without saying what the approach should be, "teach computer programming effectively ..." without saying how to do this, "select reliable hardware ..." and so on. A rather surprising recommendation is "recognise that using computers ... will create additional work for teachers." (Cline and Schneiderman, 1986; see also Taylor, 1987.) Perhaps the aim of enabling the computer to take a substantial amount of the work off the teacher's shoulders would have given the project more sense of direction.

After the reaction against early CAL, software needs to be original, of the highest quality and educationally unimpeachable. Maddison (1982, p.67) states categorically what few would disagree with: "The computer should only be introduced as a teaching aid if it is likely to improve the quality of teaching." There is a great determination to insist on the highest of standards, at a time when innovations are sometimes made in other areas of education for fairly slender reasons. This is praise-worthy, but it probably helps to deter investment in major software projects.

O'Shea and Self (1983, p.120) sum up the present attitude when they say: "Approaches derived from programmed learning are unfortunately too easy to implement on a computer. However ... there are alternatives and these, on the whole, demonstrate a trend from a behaviouristic to a cognitive approach to teaching and learning in that they view computers as devices for implementing not rigid, mechanistic, statistically-based teaching systems, but ones which treat the student as a thinking, understanding and contributing individual."

3.4 Interactive Video (IV)

3.4.1 The medium of IV

The usefulness to teacher and to student of moving photographic-type images has always been recognised in the use of film and television. What is new about 'interactive video' is that these photographic-type images, both moving and still, can now through the intermediary of a computer be presented to and controlled by a student with great precision and efficiency.

An optical 'laservision' videodisc can store some 36 minutes of moving video sequences, which can be divided up, stopped, run backwards, run slowly, run fast, accessed immediately, and accompanied by either or both of two sound tracks, all under the control of a user. Perhaps more importantly, if the whole of the videodisc is used for information in the form of stills rather than moving video, over 50,000 frames of photgraphic-type (analog) pictures or text can be stored and accessed easily and immediately. The still picture has no flicker and is of higher quality than normal television, depending on the production process, and it can be left on one frame as long as required.

There is no wear at all on the disc, as it is 'touched' only by laser light, and even when not in a player the discs are robust and unaffected by small scratches. (For the technical side of videodisc, see Kohler, 1977, or the books by Duke, 1983, and Parsloe, 1983.)

Videodisc applications have been classified in Levels (see Figure 3.3). Videodiscs are rarely used for Level 0 applications in preference to videotape players, and the videodisc player containing its own 'computer' involves such complexities of production, and inflexibility, that it is rarely used. Leveridge (1983) in a comparative evaluation of Level 2 and Level 3 comes down heavily in favour of Level 3.

3.4.2 The potential of IV

When coupled with a computer the video sequences or stills on the disc can be woven into a CAL program, with text, icons or graphics superimposed on them as required. Videodisc, at least the laservision version which has now practically won the battle over competing systems (see Clemens, 1982, for a comparison of the systems) thus has great potential for automated instruction. It has been received in some quarters with great enthusiasm. Some comments expressing this enthusiasm are quoted in Appendix . (For review articles on IV, see Fox, 1982 and 1984, Laurillard, 1984, Clark, 1984, and Doulton, 1988. For comprehensive surveys, see Duke, 1983, and Parsloe, 1983.)

Some comments on the potential of IV are given in Appendix 3.2. After the initial enthusiasm for interactive video, it has been recognised that adoption is likely to be slow, for several reasons:

1. Videodiscs cannot be made by the amateur, or by the teacher, as videotapes can, but only using a special process by manufacturers like Philips. This virtually cuts out experimentation with the video material, and makes the 'cutting' of a disc a once-only, do-or-die operation. On the other hand, discs once made cannot be copied or 'pirated'.

2. The production of a videodisc is comparable to the production of a television programme, but more complex. The density of information is such that it has been likened rather to half an hour of advertisement time on television, in costs as well as production effort. Moreover, as well as production experts, computing and programming experts are required, and for a CAL project education experts as well.

3. There are thus high production costs. In addition, 'pre-mastering' the material onto a broadcast quality tape system, then 'mastering' the videodisc, costs several thousand pounds per side; then there are the costs of cutting as many discs as are required. Also the videodisc players are expensive, and so is the computer that is needed. An interface may be required to couple the two, and other expensive peripheral equipment, including a special 'fast-blanking' monitor. The cost of a complete teaching system, including a fairly high price for the videodisc and software to recoup production costs, is likely to be anything from £5,000 upwards at the time of writing. Videodisc is thus an expensive business all round, and the market is limited.

4. There are some technical restrictions. The short playing time for moving video (36 minutes maximum) can be a problem. Material is best shot on a high quality tape system. Preparing high quality stills can be a problem. Systems are usually confined to one type of computer, which can limit the market. American videodiscs on the NTSC television standard may not operate on British PAL systems, and vice versa, though the latest videodisc players will adapt to either system.

Figure 3.3

Videodisc operational levels

Disc systems are usually classified according to the level of control that is possible as follows:

Level 0 This is a player of the simplest type used on its own simply for playback. Such players are still available, but have no advantage over videotape.

Level 1 A player again used on its own, but with all the special features of the videoplayer, such as fast and slow motion, freeze frame, two audio channels, rapid search for individual frames, and chaptering, where whole video sequences can be located.

Level 2 A player with its own built in micro-processor, so that the player can be used on its own but the material on the disc can be organised as an interactive program. Different programs can be plugged in as ROM chips, or it is possible to "dump" a program on the disc itself along with the video material.

Level 3 A player connected to an external computer, with great versatility, depending on the sophistication of the computer and software.

Development of IV teaching systems is held up because teaching systems are too expensive for education or the home market at present, and so are not bought in large quantities, and until they are bought in large quantities they will not become cheaper. As Fox (1982) says: "When there are enough video-disc players in our homes there will be a market for more esoteric educational material."

This is similar to the situation that held with computers some years ago, but videodisc technology may not become very much cheaper, as did computer technology, because the greater part of the cost of a videodisc is the production cost. However, videodisc 'training' systems are achieving success in industry, where clients can afford them, and they may find their way slowly into education.

Many have commented on the potential, but it has proved harder to realise than might have been thought. As one expert in the field commented: "We have an aeroplane, but nobody knows how to fly it." (Michael Grove, videodisc adviser to Acorn Computers, in a telephone conversation.)

3.4.3 Some approaches to IV

It is useful to examine some of the experiments that have been carried out with IV, so as to determine pointers for its use in a course-oriented ITS.

In 1984 Philips, the manufacturers of laservision videodisc players, tried to launch videodisc onto the general public by issuing a large number of feature films and other video material on disc. The launch failed, by general agreement largely because there is no facility for the owner to record with a videodisc player, so that it is less useful than a videotape player. "Videodiscs may not be a success as carriers of programmes such as feature films", says Fox (1982), "but the future of videodisc does not lie in feature films".

Gradually it is being acknowledged that videodisc is most likely to be useful in the long term as a high-density storage medium, not as a store of moving video sequences. "Commercially, perhaps the most exciting prospect is the video encyclopaedia ... It is worth noting that the 43 million words and 24,000 pictures in the Encyclopeadia Britannica fit easily into 30 minutes of videodisc space." (Clarke, 1984).

Mention has already been made of the Domesday discs, made by the BBC to commemorate the 900th anniversary of the Domesday Book in 1986. These discs, a National disc and a Community disc, form a modern electronic 'book' which is a catalogue of nearly all aspects of modern Britain, designed to use a new videodisc player which has its operating system stored on a ROM chip and can read optically stored digital data from the videodisc. It thus uses a combination of Level 2 and Level 3 operation. Many thousands of maps and photographs are stored on the videodisc in analog form, as well as 325 Mbytes of digital data, which can be searched with computer assistance using a selection from 270,000 keywords. These discs, which can be bought in a complete system with educational discount, are the most ambitious and technically advanced use of videodisc so far, and are likely to have a considerable though unpredictable impact on education. Linderholm (1987) remarks: "The Domesday Machine is perhaps most significant for what it heralds, rather than for what it is."

The BBC has now made other 'AIV' (Advanced Interactive Video) discs, one of which is the 'Ecodisc'. This "brings together a vast collection of information in the form of photographs, film sequences, graphic displays and data all under the control of a computer program ... The objective is to simulate a real place [Slapton Ley in South Devon] with all its activities and conflicting demands. The users of the videodisc take on the role of Nature Reserve Manager..." (BBC, 1988). This is a divergent and extremely useful 'teacher's help', though project-oriented rather than course-oriented.

In a course developed by the Minnesota Educational Computing Consortium, videodisc is used in a complementary rather than a central role to give teacher-independent tuition in Economics. The course is designed to fill a gap where teachers are scarce. The authors ask: "Should students be deprived of an enriched education because they live in a sparsely populated area or attend a small school?" (Kehrberg and Pollack, 1982.) The course is of interest in that it aims at teaching a whole course, but the CAL is only a supplement to ordinary textbook learning. The course was made cheaply, using mostly existing film, and the computer it was designed on was primitive by today's standards. "Initial reactions from students and teachers are favourable and suggest that the use of microcomputer and videodisc technology will play a significant role in the future of instruction," say the authors.

The Cardio-Pulmonary Resuscitation (CPR) project, originated by the Advanced Technology Development Section of the American Heart Association (Hon, 1982) aimed to replace human instructors of CPR by a computer system, so as to make the teaching of CPR more cost effective, enable more people to be taught, and to save more lives. Hon claims that the system actually teaches better than a human teacher. "One of the reasons for the system's success becomes apparent only to someone who watches the system in use. Reacting to data supplied by the sensors in the mannikin, the system gives far more precise and immediate coaching about the student's position and the depth and rhythm of compression than any human instructor can provide ..."

In addition to these experimental approaches, a large number of 'training' discs have now been made for use in industry. In fact this area represents the major effort in the field of videodisc production in this country. The National Interactive Videodisc Centre (NIVC) provides a list of firms specialising in videodisc production, which numbered about 30 in 1985 (see Bayard-White, 1985). Two examples which are of interest in being more ambitious than most, and are also aimed more at education rather than training, are the discs produced by Epic Productions for the Electrical Engineers, Technicians and Plumbers Union (EETPU), on Solid State Electronics and on Digital Electronics. These discs are well-produced and well-structured, if unadventurous, and were designed for use at retraining summer schools. They are meant to be used with laboratory equipment, but large parts of them form a 'stand-alone' course, and one was chosen for the prototype system described later in this thesis (see Lea, 1988).

3.4.4 Criticism of IV

It is relevant to assess negative attitudes to interactive video, because extravagant claims have been made for its potential, and if they are not realised there could be a reaction. If interactive video were to be given a bad name, as programmed learning was in the sixties and seventies, it could fail for many years to come. Any course-oriented systems attempted, it is argued here, are likely to be very largely dependent on the use of interactive video.

Rockman (1983) criticises interactive video on grounds of the cost of equipment and production, the shortage of capable people to produce materials, the non-availability of equipment in schools, teacher conservatism which will prevent its success, and the unfavourable command structure in schools: "People who want to use high technology to solve educational problems are not always those who have the responsibiity for getting the job done."

Whitten (1982) dwells on the commercial realities: "In Europe the videorecorder is already well-entrenched. With no overwhelming price or quality advantage and two major disadvantages - lack of flexibility and time shift - I have always been doubtful about the prospects for videodisc. A market will certainly exist, but I believe there will be little demand for a product totally dependent on pre-recorded software." Kewney (1981) is cynical: "Interactive video is being set up as the ideal teaching machine, not because that is what interactive video is good at, but because mindless teaching is very easy to mechanise."

O'Shea and Self (1983, p.253) are outspoken in their criticism of interactive video. They say: "The videodisc has in fact been identified as 'the single development that is most likely to have a profound effect, both on educational produts and on systems and organisations for delivery education' (Luehrman, 1977). We do not think so, and it may be worth saying why ... we doubt that its influence on the quality of education will be thought beneficial ..." Their criticism is based on the premise that "the videodisc encourages the freezing of chunks of teaching material and a reversal to modes of teaching which have not been found effective."

However, the Open University, where O'Shea teaches, has been freezing chunks of well-prepared and well-presented teaching material for some time, and far from being criticised for it, is considered a success. Textbooks freeze chunks of material, so do human lecturers, and so does almost every other medium of instruction. A mode of teaching that has not been found effective the first time may be effective the second or third time, if it simply went too fast or seemed too complex the first time. If this remark is meant in a more general sense, and refers to a reversal to programmed learning techniques, it is premature to say that because such techniques 'failed' on primitive computers they will also 'fail' with the advantages of modern micros and interactive video.

In fact at the same time as they criticise videodisc, O'Shea and Self are unable to disguise a certain approval: "The videodisc could therefore replace conventional videoplayers in education ... videodisc technology is very appealing... The videodisc does provide a finesse to the problem of creating graphics and animation sequences for use in CAL."

3.4.5 Research into IV

There has been a great deal of experimentation with interactive video, but relatively little of it has been carried out systematically enough to count as research.

In a Level 1 project in junior schools, Mably (1984) investigated the use by four primary school teachers of three Thorn EMI VHD videodiscs on junior science topics, and found them favourable and enthusiastic. Four uses of videodiscs in schools were identified, similar to uses of computers: as a demonstration medium for class use, as an aid to group work, as an individual reference medium located in the library, and as a training resource for teachers themselves.

The Centre for Educational Technology at University College, Cardiff has made its own experimental videodisc. This consisted of four segments. The first was an instructional sequence dealing with literature searches by library users; the second, a collection of stills on zoology; the third, a science film explaining the nature of sound and its synthesis; and the fourth, a sequence of a teacher teaching a class, for use in training teachers. Some of these experimented with 16mm film, 35mm film and U-matic videorecording, and Roach (1984) concludes: "disc production has not proved difficult." The most successful of the sections seems to have been the second, indicating that the most promising use of videodisc lies in storage of still images. Clark is an advocate of this view (see Clark, 1984).

At the University of California, Henderson and others (1983) studied the teaching of mathematics to 36 control and 43 experimental students, using a microcomputer linked to a videotape recorder. The project was prompted by concern about the steadily falling standard in mathematics of college entrance students in the USA, and the authors conclude: "The results of the field trials showed that the computer-video instructional modules were

effective in teaching or reteaching mathematical skills to secondary school students." In another project at the University of Arkansas, in which "a total of 32 white subjects, 16 males and 16 females, was randomly assigned to treatment and comparison groups after a pre-treatment phase," it was found that students taught by an interactive video-computer system passed a post-test "with a significantly higher score than the traditional group." (Boen, 1983).

In a review of an American IV course from JAM Productions, 'Introduction to Computer Literacy', Huntley and Alessi (1985) are critical of several techniques used, such as not using fully the interactive capabilities of the computer, extensively using an on-camera narrator instead of demonstrations and graphics, and a heavy orientation towards teaching vocabulary. The course comes with an authoring system, DiscWriter, and Huntley and Alessi draw a distinction between an authoring system, which is easy to use but can be expected to have limitations, and an authoring language, which involves computer programming but can be expected to offer greater flexibility. DiscWriter falls between two stools, in calling itself an authoring system but involving programming. In a whole course ITS it is likely that, to obtain the flexibility required, neither an authoring system nor an authoring language would be suitable. A computer language of an appropriate kind would be required; such a language is Prolog, discussed further in Chapter 6.

A project in the UK concerning student use of interactive video was carried out at the Open University (OU) by Laurillard (1985). This study was based on the so-called 'Teddy Bears' Disc' (Williams, 1984). A course on Engineering Materials was supplemented at a summer school by an optional videodisc system, which showed a number of topics taught contributing to the resolution of a court case centred around the fracture of plastic teddy bear eyes. "The litigation problem posed required students to piece together evidence presented in court, and concepts taught in the course, to arrive at a judgement on the reasons for failure." The disc is thus the basis of a testing technique, rather than a teaching technique. The study came to numerous conclusions about student behaviour with interactive video, which are so relevant to the design of whole course CAL systems that they are reproduced in full in Figure 3.4. These conclusions form a checklist of good practice in designing IV teaching systems, but it has to be remembered that they arose from research with mature OU students, not school students.

Figure 3.4

Findings of Laurillard (1985) [slightly shortened]

1. Students find continual interrogation and varied presentation a good way to learn.

2. These features keep students working in a concentrated way for long periods.

3. The most educationally successful forms of interaction are information testing, hypothesis framing, and simulations.

4. Observation by students and testing of students takes the most student time, the latter because it often promotes discussion.

5. Student-constructed input, with an answer-matching algorithm, sometimes makes it difficult for students to know what kind of answer is expected, and discussion is about this, rather than the substantive issue.

6. Students need some control over the presentation of information.

7. A 'skip' option needs to be backed up with information on how to use it and what its effect is.

8. A 'contents' list should be accessible at any time and should be detailed enough for a student to be able to make sense of it.

In another project involving an interactive videotape system to teach 'sound' to 22 students, Laurillard (1984 [1]) studied student control, and found: "The overwhelming conclusion from this study is that students can make full use of most aspects of control, and moreover make use of it in such a variety of ways that it becomes clear that program control must seriously constrain the individual preferences of students." This study also was carried out with mature OU students and not school students.

3.5 A Preliminary IV/CAL Project

3.5.1 Description of the project

Facilities for producing experimental interactive CAL programs, using a tape system with the Interactive Video Learning (IVL) sytem from Dalroth Computers, were available in the Educational Technology Unit at Leicester Polytechnic. To investigate some of the factors involved in producing a whole course system (though not an 'intelligent' system), the experimental project described in this section was carried out, with the following objectives:

1. To test the proposition that interactive video technology can support an extended topic and cover it adequately.

2. To examine the constraints imposed by a purchased authoring system, the IVL authoring system.

3. To identify some features that might be desirable in a course-oriented ITS.

4. To examine the possibilities and limitations of a tape system, compared with purely computer based systems and with systems using videodisc.

The equipment of this preliminary project is shown in Figure 3.5, and a brief description is given in Appendix 3.3. The IVL system contains a program to assist the teacher to develop a course, and an example of the use of this is given in Appendix 3.4. The layout of the course devised, on Radioactivity, is shown in Appendix 3.5. The 'blocks' and 'segments' of the course, as listed for the IVL system, are given in Appendix 3.6. The video 'scenes' are shown in Appendix 3.7.

It was found possible to write even an extensive course of this kind relatively quickly using the IVL authoring system. The whole project took just a few weeks for a single worker (with the help of a presenter to link sequences), including the shooting and editing of the tape material. The only systematic evaluation of the course that was carried out was for an experienced teacher of Physics to work through the course. Some 'bugs' in the course were discovered, and some in the IVL system.

3.5.2 Points arising from the project

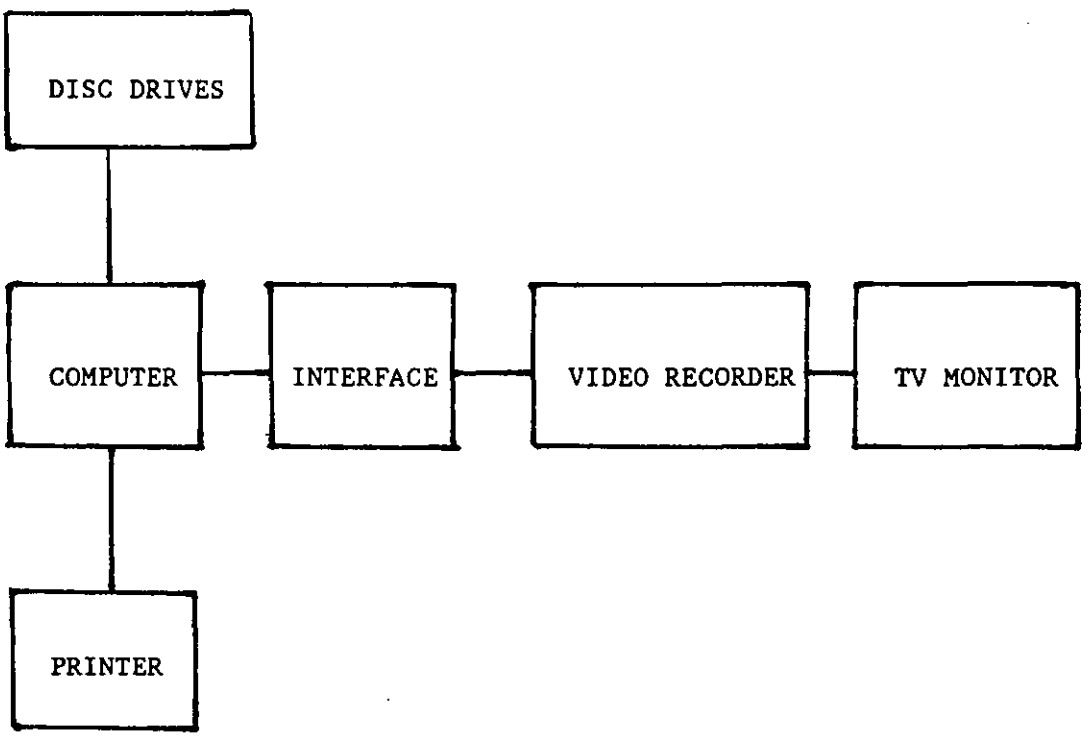
An initial problem was that there was so much equipment on the desk top that the monitor was on a stand about 0.5m above the computer keyboard. This was found to be much too high for comfort, and the stand was lowered, highlighting the problems of ergonomic design of the workstation. The complexity of equipment can be daunting to a student, and as a general principle it might be best to keep visible equipment to a minimum.

It would have been a useful facility for the student to be able to obtain printouts of some of the screens in the course, which was not possible. It would also have been useful to him or her to obtain a printout of test results at the end, but although the teacher could obtain these results using an extra control program, it was not possible for the student to obtain a printout.

In the IVL system, passing a pre-test bars a student from working through a block, and passing a post-test bars the student from returning to it. Failing a post-test involves repeating the block and the test, forever if necessary, until it is passed. A teacher testing the program is obliged to fail a test repeatedly to gain access to a block. The student is not told his or her marks for pre- and post-tests, though they are important in determining the subsequent route. This requirement of 'blind obedience' on

Figure 3.5

Equipment for the preliminary IV/CAL project



the part of the student is dubious educationally. It is contrary to findings on desirable motivational factors (see Malone, 1981), and is not in keeping with aims of 'transparency' in AI and expert systems (see Chapter 4). It would be preferable to provide the student with information and frequent choices.

Of the 86 segments in this program, 29 were tests. It is a characteristic of CAL programs that testing of the student is bound to take up a large proportion of the program. It seems that with interactive video the basic formula of VIDEO SEQUENCE - TEST is hard to escape from, and virtually impossible with a prescribed authoring system.

Problems were encountered in predicting all possible right answers by the student. Laurillard (1985) also found problems with students' ambiguous responses (see Section 3.4.5.) She comments that an open-ended question format is "not without its problems". CAL questions need to be concise and as free from ambiguity regarding the answer required as possible.

Questions also need to be brief and quick to answer, if the student is not to feel that all the time is being spent on tests. True/false and missing word questions would have been used more extensively for this reason, except that the IVL system was found to place a severe restriction on them in requiring questions to be shorter than a single line of text. It was found extremely difficult to word questions clearly and unambiguously using only 35 characters (including the correct answer), as many contained words like radioactivity (13 characters), ionisation (10) and measurements (12).

It would be preferable to allow the student to refuse, or at least delay, questions. It might be better to aim at fewer formal tests with a number of questions, and more use of single questions with the answer analysed in greater detail to determine the student's future direction. In short, a more efficient system of assessment than the usual ones is required.

3.5.3 Lessons learned from the project

Referring back to the objectives for this project, and also drawing on the prior examination of the field of interactive video, the following conclusions may be drawn:

1. Although the experimental program had many deficiencies, it was felt to demonstrate the possibility of presenting a substantial topic using this type of system. It was clear that a whole course could be covered if required, using a series of units with a videotape and a floppy disc for each.

2. Some of the constraints imposed by the IVL authoring system are indicated above. As a basis for an experimental program it was much too rigid, though for producing simple courses quickly it could be effective. There are certainly more flexible authoring systems than IVL, for example Microtext, used in a project carried out at ULAVC (Callear, 1984), but it was clear from this exercise that no authoring system would be flexible enough to use for an ITS. In order to include 'intelligent' features in the system it would have to be programmed in a computer programming language.

3. Some features which might be considered important in a course-oriented ITS, and which would seem to arise out of this project and the preceding analysis, are as follows. A CAL or IV system should aim at a single screen system, with the minimum complexity of equipment. It should aim to use sequences of video stills for most instruction, with short moving sequences where movement is essential to the explanation. Any rigid structure of blocks, pretests and post-tests, as in the IVL system, should be avoided. Knowledge within the subject can be organised hierachically, with topics at the same level, whose order is not important, offered to the student to choose from. The student should have the greatest possible degree of choice and control over his or her learning, in keeping with the findings of Laurillard in Figure 3.4. An intelligent system should aim to make assessment as unobtrusive and as unobjectionable as possible to the student by using fewer questions with the answers analysed in greater detail. Questions to the student should expect unambiguous, selected answers. The problem of analysing unanticipated responses is unsolved as yet.

4. Compared with a purely computer based system, an interactive video system has the advantage that it can put

across topics which need to be delivered didactically in a clear and interesting way. In short, it can teach, while the computer is best at simulations and interactive techniques. Also, it can put across large volumes of information, visual as well as textual. It was evident from this project that if whole courses are to be taught by computers, it is highly desirable, and probably essential, to use interactive video techniques.

As to comparing tape with disc, one becomes used to the leisurely pace of a tape system, and tape has two advantages: longer sequences can be used, up to a possible total of 4 hours, and original material can be recorded and included relatively easily. However, the inability of a tape system to display still frames for any length of time was seen as a major disadvantage compared with videodisc. Also, the system used was not accurate. Sequences were liable to start either before or after the correct start frame, an effect which could destroy a student's confidence and in some cases might make a short sequence unintelligible. A system has recently been developed at Brighton Polytechnic (1989) which claims to have overcome the problems of inaccuracy and poor still frame quality, while using an ordinary domestic videotape recorder.

This project confirmed the belief that interactive video would be highly desirable for a course-oriented ITS project. It also established that it would be better to use videodisc than videotape. This was the case in 1985, though a tape system might be more viable now than it was then. A comparison of disc and tape systems is given in Figure 3.6. However, the project also made evident the large amount of development time and effort needed to produce video material, even by the economical methods used here, and this, considered along with the cost of videodisc production, might make it impractical to contemplate producing a videodisc specially. The alternative is to find a commercially produced disc that is suitable to use.

Figure 3.6

Disc and tape interactive video systems compared

(i) Disc Systems

ADVANTAGES	DISADVANTAGES
Up to 50,000 still frames available; any one can be held all day. (With this many units of knowledge available, knowledge based approach possible). Easily linked directly to any computer with RS232 interface. Easily controlled by direct reference to frames. No coding necessary. Fast random access, precise to one frame (Max. 4 secs access VP835) Total control of pictures - slow, fast, still, reverse, etc, inc. looping. Text over picture facility with the VP835.	Cannot do own recording. Mastering of discs costly and complicated. Only 36 mins per side, though various compression techniques make this a longer time.

Figure 3.6 (contd)

(ii) Tape Systems

ADVANTAGES	DISADVANTAGES
Schools and institutions can do their own recording.	Slow serial searching (a wait of 1 min. per hour of tape.)
Can employ more and longer video sequences. (Several hours total, but in practice limited by slow access times.)	Inaccurate control - to 5 frames at best in the IVL system (see Chapter 3)
Production and playback equipment is much less expensive.	Difficult and costly to interface to computers. Interface costs over £1,000, tape needs to be coded.
	No text over picture facility without more elaborate interfacing.
	No slow, fast or reverse.
	Still frames not available for practical purposes. They flicker, cut out after a few minutes, cannot be located precisely.

3.6 Conclusions

CAL programs are produced today on a piecemeal basis with many different approaches and objectives. There appears to be no clear consensus as to how their use should be co-ordinated in schools, but they nearly always seem to be regarded as 'tools' to assist the human teacher. A great deal of inventiveness and educational technique has been developed in the field of CAL which could be made use of in the design of a whole course ITS.

As well as being a tool for the teacher and the learner, a course-oriented ITS must manage the student's learning. The part that will be played by the system and the part that will be played by the teacher will probably need to be determined, in the end, in the educational environment where it is used. It should use the most modern technology available, bearing in mind that inadequate technology contributed to failures in the past. Similarly it should be realistic in its objectives, and aim at a high standard of software.

Course-oriented ITSs should avoid comparison with programmed learning as practised in the early days of CAL, but should use the best aspects of the modern programmed tutorial, including active learner participation, immediate, corrective feedback, and avoidance of excessive learner errors. There are several relevant features of game-playing programs which might be included where possible, such as providing the student with a goal to aim for, and information on his or her progress towards it. A whole course system should include facilities for both inquiry learning and for direct teaching, perhaps as alternative student 'environments'.

In the early days of CAL, computers were felt to offer the promise of teaching substantial parts of whole courses, to the extent that some large scale projects to do this were tried. They are considered to have failed, probably because the technology was not then adequate and people's attitudes were not yet ready to accept large scale use of computers. There is reason to think that this situation may have changed today, and that education may now be ready for the whole course approach again, although there still appears to be a reluctance to contemplate it in some circles, perhaps as a residue from the early unsuccessful experiments.

There is no such reluctance in industry, where interactive video is being used as a cost-effective way of providing whole training courses. This new technology offers the possibility of storing and presenting large quantities of knowledge in educational programs. It makes possible the notion of teaching a whole course by a virtually stand-alone system, with tutorial help and supervision from a human teacher.

It would be possible to approach building a course-oriented ITS as a type of intelligent videodisc 'user interface' or 'front-end'. Laurillard (1985) comments: "The precise format of helpful user-interface features ... is an issue worth considerable attention if interactive video is to become a user-friendly medium for learning."

Chapter 4

The Design of a Course-oriented ITS

4.1 Introduction

4.1.1 A recap of arguments so far

In this chapter the design of a course-oriented ITS is discussed, drawing together points made in previous chapters.

It has been argued in other chapters that there are certain desirable features of such a system, if it is to be usable with students. The system might well be designed as an extendable shell, a feature borrowed from expert systems. Other possible borrowed features are a rule-based structure and techniques of inexact reasoning, as explained in Section 2.4. How these can be built into an actual system will be shown in later chapters.

The system should aim to teach declarative as well as procedural knowledge, as argued in Section 2.5. Perhaps the most useful feature of all in designing the system will be that it should be teacher-based, as argued in Section 1.5.

There are of course many ways in which the design of a course-oriented system might be approached. To take an existing expert system as a starting point would not be promising. It is now felt that expert systems in their usual standard form offer only restricted possibilities in education, as explained in Section 2.4.2.

4.1.2 The overall design of a course-oriented ITS

As explained elsewhere, there is some agreement as to the components which make up an intelligent tutoring system. A 'standard form' of ITS has been described by several writers, as shown in Figure 1.1. It consists of a knowledge base, a tutoring module and a model of the student. This is likely to be a viable model on which to base a course-oriented system as well as topic-oriented systems. Some ways in which these components will need to differ in a course-oriented ITS, almost by definition, are as follows:

(a) The knowledge base will contain more knowledge.

(b) The tutoring module will need to handle and 'manage' this larger quantity of knowledge in different sized 'chunks'. It will also be necessary to put across substantial quantities of new knowledge. Ideally the tutoring module should perform all the functions of the human teacher as in Section 5.2.1.

(c) The student model will need to achieve a broader and more detailed overall assessment of the student, capable of being presented to the student in an understandable form.

Some writers in the field of ITS identify other components in addition to the three mentioned so far. Burns and Capps (1988) distinguish also the instructional environment created, consisting of "those elements of an ITS that support what the learner is doing. Situations, activities and tools provided by the system to facilitate learning ..." They also distinguish the human computer interface, consisting of such user-friendly devices as natural language interfaces and speech processing. They mention videodisc as an interfacing tool. The Burns and Capps model is shown in Figure 1.2.

Using this five component model, some requirements of a teacher-based course-oriented ITS will now be dealt with in detail, mentioning some points raised in previous chapters.

4.2 The teaching module

4.2.1 The role of the teacher

The part of a course-oriented ITS which can most profitably be based on the human teacher is the teaching module, and it will be useful to examine what the teacher's role consists of. The following analysis is based largely on the writer's experience in teaching, and leans towards science teaching in a secondary school, though it is more generally applicable. Much of the analysis will be found to be borne out in works such as Sutton and Haysom (1974), Sutton (1981), and Sands and Hull (1985).

1. The teacher clarifies at the outset the overall aims. High amongst these, though it is not always stated, is often the preparation of the student for an external examination.

2. He or she organises the material according to the syllabus required but in an optimum order.

3. The material is divided up and delivered in digestible 'chunks' or lessons.

4. The teacher organises the student's time and resources, making sure information is to hand when required.

5. Throughout the course the teacher tries to spend some time in dialogue with the individual student, to offer the student a degree of choice, and to adapt continuously to his or her needs, though these things may be difficult in a class situation.

6. The teacher adds other material for relevance and interest value. He or she incorporates, in addition to the syllabus material and requirements, a teaching of such things as enjoyment of the subject and a knowledge of its value. The teacher asks questions, injects humour, and asserts his or her own personality in order to motivate the students.

This is frequently the main contribution of the teacher, which elevates classroom learning above private study in the eyes of the student.

7. The teacher sets tests and out-of-class work at intervals, usually after each unit, to test the progress of the students. He or she attempts to identify and correct individual errors made by students. Throughout the course the teacher assesses each student with regard to ability, application, interest, and specific abilities such as understanding, problem solving ability and others.

8. On the basis of the tests, the teacher adjusts the general pace, and gives extra help to slow students and extra work to fast students.

9. The student is given feedback to enable him or her to improve his or her own performance.

10. Towards the end of the course the teacher revises work covered.

11. At the end of the course the teacher may be required to organise an examination, or there may be an external one.

12. Usually at the end of the course the teacher completes a report on the student (usually a short one) for the benefit of the student, the parents, the housemaster, prospective employers, and others concerned. A modern approach to student assessment involves the teacher in compiling a 'profile' of the student during the course, which is given to the student at the end of the course as a more detailed form of report.

Some of these aspects of the teacher's role, such as 6, a computer-based system may not be able to do as well as a good teacher, though in an IV system using television techniques with a presenter it is possible to include such things, even humour and a personal element. Other aspects, such as 5 and 9, a computer-based system may be able to do somewhat better,

as a result of its greater speed of operation and ability to give individual attention.

This list represents a markedly different approach to instruction from that embodied in most ITSs to date. It is evident from the list that there is a large element of management in the teacher's role.

4.2.2 Individual attention

The ability of a teacher to give individual attention to students determines the tutoring component of his or her teaching. As such this is relevant to the design of a course-oriented ITS.

The treatment of students of different abilities has always been a major preoccupation in education. This has become more important since comprehensive schools have placed teachers in the position of needing to cope with 'mixed ability' classes containing students of more widely different abilities than before. Many teachers have found coping difficult.

In one analysis, based on a report by HMI inspectors, Stott (1979) says: "Only the most highly skilled and committed teachers did justice to the whole range of capabilities of the children in their classes. The average did so for only a section of them; the weak [teachers] failed miserably." Stott proposes a number of principles for effective mixed ability teaching, and many of these contain suggestions particularly applicable to CAL systems, such as letting pupils see the results of their work immediately, providing enjoyable competition without too much stress, continuously checking the achievement of every pupil, and replacing classes with informal individual (and group) work. It is striking that a number of these recommendations coincide with similar ones arising from research into CAL and IV by an independent route, as described in previous chapters. (See, for example, Malone, reported in Section 3.3.3 and Laurillard in Section 3.4.5.)

Stott's last principle is that teachers will have to cease to be teachers in the old sense of the word, and become organisers of learning programmes.

Again there is a suggestion that a teacher, or a course-oriented ITS, might best serve the student's needs by organising and managing his or her learning.

4.2.3 How the teaching module will be different

The tutoring module of a course-oriented system, like a teacher, needs to handle and manage a large quantity of knowledge, putting across substantial quantities of new knowledge. A large part of a teacher's time is spent in putting across new knowledge, rather than rehearsing the manipulation of knowledge already known, as most ITSs have tended to do.

As explained in Section 2.5, it appears that there is a need for a declarative shell or 'front-end' for other ITSs, and this function could well be compatible with the notion of course-oriented, teacher-based ITS shells that is being built up here.

The dichotomy between declarative and procedural knowledge is similar to that between didactic instruction and interaction in the education field. A practical tutoring strategy for a whole course system will efficiently and effectively organise and present new declarative knowledge, as well as enabling the student to practise procedural skills already learned. Richardson (1988) says: "There is absolutely no reason why some initial effort cannot be made in developing ITSs that formally represent and teach both the declarative and procedural aspects of a domain."

A teacher-based, course-oriented ITS is likely to differ from other ITSs as follows. The tutoring modules and strategies of previous 'restricted topic' ITS systems have usually had the following general characteristics:

(a) They have been modelled on a human 'tutor', who ideally gives the student individual attention in minute detail.

(b) They have been largely preoccupied with the details of the cognitive process that goes on either in the mind of a student learning alone, or between tutor and student.

(c) They have been concerned almost exclusively with 'interactive' methods, used to practise principles and test knowledge already learned, and have rarely been concerned with imparting new knowledge.

A whole course ITS system is likely to require a different approach, as explained above. It may be summarised as follows:

(a) It should take rather as its model the human teacher, who deals with knowledge on a broader scale, and often gives less attention to minute detail.

(b) The teacher is concerned with cognitive processes, but is also largely concerned with the distribution of effort between different parts of a syllabus and with the management of information.

(c) In addition to interacting with the student, the teacher has to be concerned for a large part of the time with the imparting of new knowledge.

Some workers in ITS, who see the main aim of intelligent tutoring as providing detailed individual interaction, might see this change of emphasis as an abandonment of their aim. It is not seen thus here, but as a practical approach to the process of education, such as teachers have to adopt daily in order to carry out their job of work. It might be said that the aim of ITS is to improve on the human teacher, who works under difficult constraints, in giving more detailed attention to the student, but this is a long term aim, and it might be best first to try to equal some of a teacher's skills.

4.3 The knowledge base

4.3.1 The role of Interactive Video

Computer storage capacities have increased considerably in recent years, but to contain a whole course within the memory of a computer, and handle these large quantities of knowledge, would clearly stretch any system. Psotka, Massey and Mutter (1988) comment: "Working ITSs will demand all the

power anyone can conceivably provide; they will continue to grow as quickly as the underlying computational machinery will allow."

Fortunately a way has appeared to bypass some of the problems of storage in an instructional course. This is 'interactive video', (IV) described in Chapter 3. Large quantities of declarative knowledge can be captured on an analogue, optical videodisc and presented as television-type sequences or stills. The medium is more efficient than digital storage, and can also present some kinds of instruction, such as demonstration of complex skills, which cannot be stored at all in a computer.

The potential of IV is not appreciated by those in the ITS field. O'Shea and Self are unenthusiastic, discussing it simply as a 'peripheral' and commenting: "... the videodisc encourages the the freezing of chunks of teaching material and a reversal to modes of teaching that have not been found effective." Miller remarks: "There is no strong theory to guide and motivate the use of video as part of interfaces and ITSs. " (Miller, 1988.) Videodiscs are successful, however, in industrial training, and video techniques could offer much in a course-oriented approach.

4.3.2 Organisations of knowledge

The knowledge base in a course-oriented system needs to be larger, accommodating knowledge in a structure containing units of different sizes. The way teachers organise and present knowledge in a course can be useful in designing this knowledge base. It seems reasonable to organise the knowledge along lines tried and tested and normally used by teachers and others in the field of education.

Other systems, within their narrower domains, have sometimes tried to reorganise knowledge. It is necessary to consider whether a whole course ITS should attempt to reorganise its broader field, and whether one type of organisation is preferable over another.

There have been a number of attempts in education recently to improve upon traditional organisations, which may in some cases have been simply attempts to break them down for non-educational reasons. Ingle and Jennings (1985) identify a 'curriculum development era' from 1960 to 1980. As an example, the Schools Council Integrated Science Project (SCISP) takes

selected material from the traditional subjects of Physics, Chemistry, Biology and Astronomy and organises it in a course called 'Patterns', using loosely structured 'concepts' of which the fundamental ones are building blocks, energy and interactions (Hall, 1973). As another example, the Nuffield A-level Physics course reorganises its subject area, traditionally arranged in energy groupings of Mechanics, Heat, Light, Sound and Electricity, into three main areas of Fields and Action at a Distance; the Nature of Matter and of Atoms; and Motion and the Analysis of Change (Nuffield Foundation, 1971). The Nuffield and Schools Council courses have exercised a definite influence: Ingle and Jennings (1981) point out that school science has become more strongly practical, and that assessment methods, textbooks and collaboration between teachers have changed.

These variations on traditional groupings of knowledge have not been universally adopted and have often been criticised. Ingle and Jennings say the Nuffield courses looked and were very difficult, and Shayer (1972; see also Shayer and Adey, 1981) has illustrated clearly that many topics were conceptually beyond pupils of the age they were intended for. Ingle and Jennings also say that the courses did not go far in illustrating the social relevance of science, and were highly specialised. In some cases teachers have found the courses difficult to organise and teach, and suitable for only the most able pupils. (See Galton and Eggleston, 1979.) It is interesting that while reorganising the knowledge within their subject areas, the Nuffield courses did not attempt to look beyond them, and accepted their subject boundaries as fixed.

Curriculum development is still going on, but the emphasis has moved away from new courses. In recent years there has been concern that the curriculum has been reorganised too much, and with too little coordination. A pupil can now be taught very different things depending on which institution he or she attends. The Department of Education and Science has published a number of documents encouraging schools to coordinate their 'core curriculum' (DES, 1980 [1]; DES, 1980 [2] and DES, 1982), and the Secretary of State for Education has now assumed the power to influence by law the basics that are taught in all schools.

These brief comments may suffice to show that the choice and organisation of a knowledge domain for a whole course system is not easy, and the changing nature of much of the curriculum may be one factor that has discouraged large-scale investment in teaching whole courses by computer.

It would seem inadvisable for a designer of an ITS system to attempt to propose a radically new way of organising knowledge.

4.3.3 The traditional organisation of knowledge

A traditional organisation of knowledge has grown up in most taught subjects which is now well established, and is reflected in the classification systems used in libraries and textbooks. It is also reflected in the structure of professional organisations and learned societies in the outside world, a fact which has led some to suggest that the present organisation tends to preserve the status quo and to make knowledge inaccessible to many (see Young, 1971). This debate will not be entered into here, but it does seem that a logical structuring of knowledge exists in an accepted form, and to acknowledge this can only help the student to understand it, although different structures are possible and different students may prefer different routes into the structure.

There is evidence of a return to 'traditional' curriculum-based methods in ITS systems. Blaine and Smith (1977), in describing their EXCHECK mathematical set theory system, have the following to say: "There is much implicit information about human learning contained within classical curricula, and some of this information can be applied to the creation of computational models of knowledge, representation and influence ..." More recently, Soloway and VanLehn (1985) have described their ideas of curriculum in VanLehn's 'Step Theory' under the heading "What to do next", commenting: "Teachers use a curriculum to teach, with good reason."
Step theory also has useful ideas to offer on what might be called the 'grain size' of the knowledge handled by the system, mentioned already in connection with the attention span of students. VanLehn, working in the field of ITS, has proposed that knowledge is imparted "one simple step at a time", and is based on 'felicity conditions'. The four of these that VanLehn gives (1983) are reproduced in Figure 4.1. The felicity conditions are interesting in that they will come as no surprise to a practising teacher. VanLehn has built an ITS to teach arithmetic, SIERRA, based on his Step Theory and on Brown's 'Repair Theory', which explains how students cope with getting stuck with a problem, by such 'repair' tactics as skipping a step, or going back and taking another path. (See Brown and VanLehn, 1980.)

If a videodisc is used for the knowledge base of an ITS, it is possible to organise the knowledge on the videodisc in a variety of ways, regardless of the way the videodisc was meant to be used. The size of the smallest knowledge steps or chunks, however, will be largely determined by the videodisc chosen for the system.

It has been argued that a course-oriented ITS should be extendable to other knowledge domains. A method of organisation is thus required which can be applied to most knowledge domains, and is not subject to the debates and criticisms levelled at 'new' types of organisation. Such a 'traditional' organisation of subject sized knowledge domains exists in most subjects.

The 'traditional' way of organising a subject-sized knowledge domain for teaching purposes can be summarised as follows:

1. Knowledge is divided into a number of module or 'part' groupings, then units or chapters within these, and topics or sections within these. Further sub-division beyond three levels is not usually required.

2. Modules, units and topics which need to be understood for the understanding of others precede those others.

3. Modules,units and topics which are self-contained and do not affect others are arranged in ascending order of difficulty for the student.
4. Modules, units and topics which are self-contained and are of roughly the same difficulty can be presented to the student in any order, perhaps as options.

This approach to organising subject knowledge arises naturally when tackling the problem of designing a whole course. It is adopted by the writers of most textbooks, by the designers of most teaching syllabuses, and by most teachers.

Figure 4.1

'Felicity Conditions' in Step Theory (VanLehn, 1983)

1. Students expect a lesson to introduce at most one new "chunk" of procedure. Such chunks are called subprocedures.

2. Students add their new subprocedures to their current procedure rather than replacing large parts of it. That is, they expect the lesson to augment their procedure rather than making parts of it obsolete.

3. Students induce their new subprocedure from examples and exercises. That is, students expect the lesson's material to correctly exemplify the lesson's target subprocedure.

4. The students expect the lesson to "show all the work" of the target procedure. Even if the target subprocedure will ultimately involve holding some intermediate result mentally, the first lesson will write the intermediate result down. In a later lesson, the student is taught to omit the extra writing by holding the intermediate result mentally.

4.4 The student model

4.4.1 Probability assessment

A consequence of covering a whole course is that a different kind of analysis of the student is required. Instead of identifying student learning difficulties in great detail, as some ITSs have attempted to do, the system needs to generalise about the student as a teacher does in an end of term report.

At the same time, it would be desirable to make this analysis without the need to set the student a very large number of conventional, right or wrong questions. It is best if the student does not have to spend most of the time being assessed. A human tutor might ask a student very few questions, but from the answers received the teacher is be able to make wide generalisations about the student.

Expert systems bypass the need to deal with impossibly large sets of information by using inexact or 'fuzzy' reasoning. In a course-oriented ITS inexact reasoning could be applied to student assessment to build up a broader generalisation using less information. This generalisation about the student, a type of student profile, would form the student model.

This suggests a form of probability assessment in which each answer from the student is analysed in terms of several variables about the student. Such an assessment model is described in detail in Chapter 7, and will not be discussed further here.

4.4.2 Criterion referencing and profiling

One of the ideas current in education at the present time which was considered to be of use in WITS is the notion of criterion-referenced testing rather than norm-based testing, which is almost universal in external school examinations. (For a discussion of this, see Brown,1980.)

Norm-based testing, which involves allowing a pre-determined proportion of students to pass the test, has been criticised for putting the emphasis on

a narrow form of achievement, and for automatically barring a large proportion of students from this achievement. Hodson and Brewster comment:

"Testing procedures tend to concentrate on academic, cognitive matters and non-cognitive aspects of the curriculum are almost totally ignored, so that a balanced picture of each child's particular strengths and weaknesses cannot be obtained. In the drive for fairness through uniformity, objectivity and anonymity, a bureaucratic and impersonal climate of assessment is created which makes little provision for individuality, creativity and interpersonal skills." (Hodson and Brewster, 1985.)

Criterion-referenced testing involves allowing any proportion of students to pass provided they individually satisfy predetermined criteria. It is argued that criterion-referenced testing, while maintaining similar standards, will remove the competitive element to a large extent and allow students to achieve results by their efforts, independently of others. Examples of criterion-referenced tests presently in operation are the driving test and piano-playing grades. Testing by computer, completely objective and without reference to a large sample of similar students, lends itself to criterion-referenced testing.

As the Schools Council comments: "The difficulty [of automatically failing a large proportion of students] would in part be solved if styles of assessment which distinguish between levels of attainment could give way to those which differentiate kinds of competence." (Schools Council, 1975, quoted by Hodson and Brewster, 1985.)

There has thus been interest recently in the use of student profiles, rather than examination marks, to describe a student's all-round ability and achievement in a subject. This is a technique given prominence in the recently introduced GCSE examination. It would be useful, in a course-oriented system, to summarise the student's achievement at the end of the course in a written report, describing his or her efforts under certain main headings. Percentages and marks would be given where relevant. The assessment in WITS uses multiple choice testing in an unconventional way to compile the student model, which is available to the student at any time during the course as a 'profile of student progress'.

A further comment may be made. As Hodson and Brewster (1985) point out: "The design of assessment strategies to achieve all this [i.e. variety of

assessment in a profile] will require a major change in attitude on the part of many teachers and a heavy investment of time, energy, commitment and resources. But the rewards are immense." Hard-pressed teachers have heard many exhortations of this kind in recent years, and it is doubtful whether they can respond any more, but the computer in a teaching system will, of course, work tirelessly, consistently and accurately to compile such a profile, once suitably programmed.

4.4.3 The student model in a course-oriented ITS

The student model in an ITS has a counterpart in the picture or profile a teacher builds up of a student via assessment. In a teacher-based system the student model can be built up using similar methods to those used by teachers. A good reason for adopting a student model along the lines a teacher uses is that if a student is being taught a whole course by an ITS, assessment will be expected throughout the course in terms the student himself or herself can understand and is used to. Such a model is readily translated into student information.

A student model designed along the lines teachers use in a course-oriented ITS would lend itself to criterion referenced assessment, in which different student abilities are assessed rather than a blanket, overall skill. It would also lend itself to profiling, in which the assessment takes the form of a description of the student rather than an overall mark.

The teacher's picture of a student is rather different from the student model aimed at in some ITSs, which picture the student in terms of a match with an ideal student model, the nature of the mistakes made, or the knowledge gained. The teacher assesses the student in terms of such things as understanding, problem-solving ability, ability to remember facts, creative insight in the subject, and other abilities, some subject specific.

The teacher is able to assess such obscure and largely undefined abilities because he or she possesses expertise to do so. A model which attempts to capture this expertise is described in Chapter 7, using the multiple choice testing used by teachers in an unconventional way. The method of assessment described in Chapter 7 builds up a more detailed profile of the student using less questions than in conventional CAL, the end result being a model

of the student on the lines of the picture a teacher might have in his or her mind's eye when writing an end of term report on the student.

4.5 The user environment

4.5.1 Student-centred CAL

There has been a movement in recent years in education, as in CAL, to make learning 'student-centred', so that knowledge is organised and presented in terms to suit the student, rather than the teacher, the school, an examination board, or some other body. This is a sentiment with which most would agree in principle, but it has not always proved possible to know what approach does in fact suit the student best, the problem being that students vary greatly in their requirements.

The role of the student can vary between two extremes. On the one hand he or she can be almost totally passive, simply handing in written answers and assignments from time to time to the teacher or lecturer, who takes complete charge. On the other hand he or she can be left entirely to his or her own devices, learning by active inquiries and investigations from whatever sources are available, such as books, video material, computer databases, experiments and other sources. The ideal of most students is somewhere between.

In CAL, the extremes of student participation have their counterparts in the early linear CAL programs in which the student has little choice in what happens next, and the LOGO type of environment in which the student explores freely, learning by his or her own inquiries (Papert, 1980). The database inquiry program forms an environment for the student in which he or she can search out knowledge, an example being the BBC Domesday IV system (see Linderholm, 1987).

What students themselves prefer is not necessarily a good guide to efficiency, though it may assist motivation. Atkinson (1976, quoted in O'Shea and Self, 1983) found that "the learner is not a particularly effective decision-maker in guiding the learning process". Steinberg (1977, quoted in O'Shea and Self, 1983) found "inconclusive but suggestive

evidence that learner control did increase motivation but might also decrease learning efficiency".

4.5.2 Control and choice in CAL

The degree of program control given to the student is a major factor determining the role of the student. In teaching that is highly structured by the teacher, student control is virtually zero. In a learning environment, the student can have a great deal of control. It is not possible to say categorically how much student control is 'best', as it will depend on the student and the subject matter. Laurillard (1985) found in her work with mature Open University students that they liked a great degree of control, and used all avenues open to them. Other students might prefer more guidance, particularly where the subject matter is complex and a knowledge of the underlying structure can assist understanding.

Another major factor in determining the student's role is the amount of choice given to the student. Choice in computer programs is usually achieved by presenting the student with screen menus. Some students prefer a large amount of choice, usually mature students. Some prefer to be spoonfed, usually less able, or young, or insecure students. Clark (1986) says: "Students may prefer to work from the general to the particular, or from the particular to the general ... Many students prefer to be taken along pre-programmed paths."

The attitude of a human teacher to student choice would usually be to give it where possible, but to guide the student where necessary. For example, freedom to choose topics at will can result in students trying to study advanced topics before introductory ones, which is clearly not in their best interests.

4.5.3 Control and choice in education

Student-centred approaches are also a major preoccupation in education, and attempts are made to give students as much choice and control of their learning as possible. Heuristic or discovery method, in which students find out about a subject for themselves by trial and error, has been an important factor in education this century. An early champion of the

approach in science education was H.E.Armstrong (see Brock, 1973) and it has greatly influenced the Nuffield Foundation courses in science (Nuffield Organisation, 1966-73).

Such an approach has strengths and weaknesses, and the subject is too large to go into here in detail. If carried out well, it is reputed to train better research scientists, but clearly pupils who have to find out many things for themselves will need more time. Many teachers have found it impossible to cover sufficient ground in the time available, and many consider extreme forms of the method unsuitable for all but the most able pupils in situations where they do not have the pressure of exams forcing them to master a crowded syllabus.

However, most teachers now include elements of discovery method in their teaching, especially in science where discovery is built into the philosophy of the subject, but also in other subjects, for example in English, where pupils are encouraged to find out about little-known topics for themselves in the library. Indeed, there is a view of education in general which proposes that, in a world where society is changing rapidly, pupils should be taught not established knowledge, but techniques for finding things out for themselves.

The idea of a student-based computer assisted learning course in which a student discovers rules and relationships for himself by trial and error, extracting information and choosing topics from a large database of knowledge, is an attractive one in theory. However, teachers know that most students need, and prefer, to be taught.

4.5.4 The environment in a course-oriented ITS

A whole course aiming to cater for a variety of students needs to allow for either a high degree of control and choice, or very little. Where choice is given, it may have to be guided, or restricted to choices that make sense. Self (1985) has described a proposed guided choice system based on machine learning. 'Monitor' ITS systems, mentioned in Chapter 3, have a guided choice philosophy, guiding the student when he or she goes wrong (see Sleeman and Hendley, 1982). It should be possible to:

1. Lead some students through the subject matter in a highly structured way.

2. Give some students a great deal of control and choice.

3. Give other students 'guided choice' of subject matter and treatment, depending on their ability and preferences.

Most teaching systems have tended to provide either 1 or 2 only. An ITS teaching a whole course should ideally provide both, and 3 as well.

Ferguson (1984) has examined the advantages of the exploratory environment and the structured approach, and his findings are reproduced in Figure 2.4. This analysis, and the experience of the teacher, suggest that over the sustained effort of a whole course both types of environment are likely to be required to achieve the best performance by all pupils.

In particular, many pupils who prefer a structured form of teaching while going through the course for the first time might find it useful to revise at the end of the course by exploring the material in an open-ended way. It would seem that an important feature of a whole course, arising from its extensive coverage of a variety of topics, will be to give the student a choice of learning environment. A structured environment should be available, and so should an open one with an efficient search mechanism to allow the student to explore the course material. A method of accommodating this need is described in the next chapter.

4.6 The student interface

4.6.1 The importance of the student interface

As computers and computer software have become more sophisticated, it has become more difficult for non-technical users to communicate with them. The subject of communications between humans and computers has become a major field in itself, referred to as the human-computer interface (HCI) or

man-machine interface (MMI). (See Coombs and Alty, 1983.) This goes beyond simply designing programs to be as 'user-friendly' as possible. Mention has already been made of intelligent user-interface systems such as those of Ross (1986) and Benyon(1986) which are similar to ITSs in that they aim to instruct the user in the use of the computer.

The student module in an ITS is sometimes referred to as the student interface (see Roberts and Park, 1983) and the interface with the user ('the human window') is sometimes included as a major part of an expert system (see Forsyth, 1984, p. 11, and Figure 1.5). User modelling has similarities to student modelling (see Ross, 1986).

Edmonds (1983, [1]) has proposed design features for man-computer interfaces which adapt to the user, identifying the user, identifying the functions of the interface as passing information to and from the programme, but intercepting it first and in some cases consulting the user further before communicating with the program. A development which may become universal is the use of user cards which store data relating to the user. These can be inserted into the computer during use and are updated as the user becomes more proficient.

Opinions sometimes differ on suitable designs for user interfaces. Anderson , for example, points out the problems students have with holding data in their 'working memory', and attempts to help in this "by having the tutor encode on the computer screen much of the information a student is likely to forget." (See Anderson, Boyle and Yost, 1985.) Against this one needs to balance recommendations from other sources, for example Alderson and DeWolf (1984): "In general, display the minimum amount of information (both in text and graphics) necessary to achieve the purpose of the display." Maddison (1982) comments: "The computer screen is not a good medium for displaying text."

The importance of an efficient and attractive interface in a CAL system should not be under-estimated. Many commercial and games programs solve the problems well, and students and teachers now expect programs to have good interfaces. It is possible that the importance of this has indeed been underestimated in many ITSs, which have not aimed at being usable systems, developed beyond the prototype stage.

Some relevant areas of student-computer interface design are now examined, with comments relevant to a whole course system. For a general discussion, see Miller (1988).

4.6.2 Menus

Clark (1986) comments: "Menus are vehicles for student control within a course. The greater the degree of control, the more the student is motivated into an exploratory style of learning." These "choice structures" (Clark's term) are thus integrally bound up with student choice and control. A great deal of effort has gone into the design of menus, which are one of the most basic tools for communication with a computer.

A menu normally consists of a list, from which the user chooses by pressing a key number or letter, or typing in a key word. Snowberry and others (1983) have found that menus more than seven deep are difficult to scan and select from. Clark points out that key numbers are easier to locate on the keyboard than key letters. Either are apt to be confused with certain types of data. Key words are less ambiguous but more laborious, and lazy students might opt for the shortest. Clark points out that numbers or letters on the right-hand side of the screen leave the left-hand side free for the eye to scan the menu items, though usually the reverse system is used to give the numbers or letters prominence.

Feature menus are menus present at all times on the screen, containing options the user might require at any time. In this respect they are similar to a command system. Such menus frequently include a "help" option which the user selects when in trouble. Edmonds (1983, [2]) comments, "The simple possibility of escaping from any state of the program to a static state (e.g. elementary help) and then returning to the original state, can be important in giving a student the chance for relief and escape from situations which they find difficult."

The first menu might call up another 'nested' menu. Billingsley (1982) found nesting of menus of more than two levels difficult for users to keep track of, requiring a diagram of the nesting structure.

4.6.3 Commands

An alternative to menus, similar to using a keyword menu, is the command system, where a user is required to type in commands to make choices. These might have to be remembered, or might be displayed continuously. A command system is useful where a small number of the same choices are used frequently. It is less useful where there are many choices to be made. In a wide-ranging and versatile whole course system a combination of commands and menus might well be best.

4.6.4 WIMP environments

The more recent 16-bit micros with their larger memories have made possible interesting variations on the menu theme. Pull down or drop menus and pull out menus are now a feature of some operating systems, as are pointers.

Seidner (1984) found that inexperienced users preferred the pointer to be an arrow, while experienced users preferred it to be a 'reverse video bar'. Icons, small pictures representing objects, usually along the top of the screen, are also now common. They can be selected using a 'mouse' to move a pointer on the screen.

Such systems usually make use also of 'windows', rectangles on the screen containing information that can be called up by the mouse and pointer system. The whole is known as a WIMP (windows, icons, mouse, pointer) system. Telford (1986) distinguishes three levels of windows: static, movable and overlapping. He also identifies "three levels of sophistication with icons": those which remain on view, those which appear depending on the current window, and those that are movable. Programs or 'environments' such as as Microsoft Windows and GEM enable WIMP systems to be programmed easily, and there have been a number of papers in the computer press recently which describe how to program them on micros (see, for example, Telford, 1986 and Pountain, 1986). Some attribute the principles used in WIMP environments like Windows and GEM to the Xerox PARC Smalltalk project, via the LISA interface adapted for the Apple Macintosh computer (see McKinnell, 1987).

Using the mouse and pointer system with a menu the user does not need to use the keyboard, and using icons the user does not even need to be able to read. However, the mouse should not be regarded as the solution to all input problems, as Freeman (1986) concludes; "It does simplify operations which involve jumping around the screen or between windows, but ... it was quicker to keep my hands on the keyboard." It needs to be noted that if a system uses natural language input then the keyboard is required and the mouse is superfluous.

4.6.5 Touch screens

An interface method usually marketed along with videodisc systems is a television with a touch screen, with which a user can select by touching a menu item, bypassing the keyboard. This would not seem to have any marked advantage over the mouse system, and certain disadvantages: for example, soiling of the screen with sticky fingerprints, and more importantly the necessity for the student to sit within arm's length of a large screen, with the consequent risk of eyestrain. As with the mouse, a natural language system requires keyboard input and would make the touch screen redundant.

4.6.6 Natural language

Probably the first system to apply work on natural language dialogue to a teaching system was the SCHOLAR system of Carbonell (1970). This system carried on what was refered to as a 'pragmatic' type of natural language dialogue with the student on South American geography.

The dialogue method was carried further in the more sophisticated system SOPHIE, by Brown and Burton (1975, and see Brown et al, 1982) which dealt with the more complex environment of electronics. Natural language has since become a regular, almost distinguishing, feature of AI and ITSs. The systems WHY and SOPHIE, for example, and the aircraft bookings system GUS (see Appendix 1.1) have natural language dialogue as an integral part of their operation, bound up with the 'scripts' on which they are based.

An interesting approach to dialogues, which has perhaps been dismissed too readily by serious AI workers, is the 'semantic-trickery' approach of

Weisenbaum (1966) in his ELIZA program. There are now similar programs available for micros. In one such program (Hartnell, 1984) a conversation is kept going by techniques such as throwing back at the user what he has just typed in; reacting to 'trigger' topics mentioned; giving him random, standard responses; or recalling something that has been mentioned earlier in the conversation. The effect is very realistic. These techniques are in fact used by humans in conversation when they are unable to respond logically or sequentially, and if a 'personal', human effect is required in a human-computer interface such techniques can be effective.

Most natural language applications in ITS have been within the dialogue mode, consisting of alternate sentences from the student and the computer. It is not clear whether natural language will ultimately be most useful in this form in CAL systems, when perfected, though such work is paving the way for speech input systems. In an educational program, the student does not want to spend time typing in whole sentences if the interaction can be carried on more simply and quickly by means of menus or single keywords, unless the aim of the system is to teach English language, which present techniques are short of achieving. VanLehn (1985) comments: "Sometimes natural language is essential; sometimes impossible."

In a whole course system which contains an open-ended learning environment, a facility for natural language input would be an advantage, but its degree of complexity will depend on the application. It is likely that for most purposes a simple version, analysing input sentences and extracting keywords with which to search the knowledge base, will be sufficient.

4.6.7 Screen layout

Standard ways of setting out information on a computer screen have begun to evolve, and briefly the following guidelines are usually adhered to:

1. Status information such as the mode the student is in, and help information such as the commands available, are usually placed at the top of the screen.

2. The main educational information or message to the student is placed in the middle.

3. Prompts requiring responses, or instructions to the student, are placed at the bottom.

Alderson and DeWolf (1984) list a large number of guidelines for writing well-presented educational programs.

4.6.8 The interface in a course-oriented ITS

A whole course system covering a variety of topics with a variety of student types is likely to require a complex interface using a variety of methods. A mixture of menus and commands is likely to be found best, making use of windows and the now conventional system of status and proceed information.

A choice will probably need to be made between a WIMP system, a touch screen, and natural language input, as they are incompatible and changing from one to the other frequently would be distracting. Natural language input is likely to be useful in a whole course system incorporating a free search discovery environment.

4.7 Conclusions

A course-oriented ITS cannot be modelled on any one previous ITS, but the general form of ITS systems described by Self (1976) and later in more detail by Burns and Capps (1988) can be used as a basis for the design of such a system. Help is also to be found in research on previous CAL, ITS, AI and IV systems. The main source of guidance in the design of such a system might be found in teaching expertise.

In the tutoring module of the system the role of the teacher in organising and managing the student's learning may be taken as a basis. The module will need to expend more effort on putting across first-time declarative knowledge than a topic-oriented system. The knowledge base of the system will do well to use interactive video techniques to store and present a larger volume of more 'coarse-grain' knowledge. Traditional methods of organising knowledge, as used by teachers, are probably the most suitable to use.

The student model of the system might be along the lines of the report or profile a teacher builds up of a student. Assessment techniques used by teachers can be used as a basis for constructing this student model, but the inexact reasoning techniques used in expert systems might be adopted to assess students more efficiently.

The environment module of the system in a whole course system which aims to teach a variety of students over an extended period will need to provide a range of environments, from structured teaching to open-ended exploratory learning. The human-computer interface of the system needs to be based on existing designs which are efficient and attractive.

A course-oriented ITS was designed on the lines described in this chapter, and subsequently built and tested. This system is described in the next two chapters.

Chapter 5

WITS - a working course-oriented ITS

5.1 General Description

5.1.1 Background

A system is described in this chapter which was built over a period of some three to four years as a prototype to teach Electronics, a Whole-course Intelligent Teaching System (WITS). It was based on the arguments put forward in previous chapters. The main intelligent feature of WITS, a probability assessment method used to compile the student model, is described in the next chapter. Much of this chapter will inevitably be descriptive, and a poor substitute for using the system itself.

Research for WITS began in 1984 in the Human Computer Interface Research Unit (HCIRU) at Leicester Polytechnic. Programming began towards the end of 1985, by which time the research unit had moved to Loughborough, to become the Loughborough University of Technology Computer Human Interface (LUTCHI) unit. During the period of construction of WITS changes were made to the original design conception to keep abreast of advances in hardware and software.

5.1.2 Software aspects of WITS

There was a possibility at the start of the project that a videodisc authoring system could be used for WITS. However, the pilot project described in Chapter 3 demonstrated that authoring systems were too inflexible, certainly affordable ones at that time. Huntley and Alessi(1985) are not complimentary about authoring systems, pointing out that the more flexible ones have features of computer languages and need to be learned like a language.

The choice was essentially between the procedural language PASCAL, flexible and popular in higher education institutions so that expertise was readily available, and one of the languages popular for Artificial Intelligence, namely PROLOG.

PROLOG is a declarative language consisting of a knowledge base of facts and rules. Problems are solved in response to appropriate queries, treated as goals, which initiate a search for the relevant facts and rules. The language thus embodies many of the ideas pioneered before its inception in the field of AI. The program envisaged for this project was one which continually consulted a bank of educational knowledge in response to student input, and presented the knowledge to a student in a suitable manner, subsequently building up a store of facts and an analysis about the student. It appeared that a knowledge handling program of this type would be an ideal application for PROLOG, and so it was chosen for the project. Experience has borne this out.

The programming began in 1985 using Prolog 1 from Expert systems, a version with all the Clocksin and Mellish features of PROLOG which have virtually become the standard (see Clocksin and Mellish, 1981). Versions with more features were very much more expensive and required more expensive computers. Prolog 1 was used in spite of certain limitations, notably its lack of colour graphics and its slowness, and its choice determined the choice of hardware, described later.

In October 1986 Borland brought out a version of PROLOG called Turboprolog, available at a low price, with colour graphics, windows and other features which are coming to be expected by computer users, and fifty times faster than Prolog 1. Turboprolog lacked some of the Clocksin and Mellish features, but these were advanced programming features mainly used for machine learning programs in which new rules are added to the database during execution, and were not required. The program was converted to Turboprolog, requiring hardware changes described in the next section.

WITS was written entirely in PROLOG, and a number of points arose from the size of the program.

1. PROLOG was found to be quite suitable for a large program, its fact and rule structure lending itself readily to a modular approach.

2. Although PROLOG is considered slow for some applications because of the continual searching of the database, Turboprolog was found to be quite fast enough for this real-time application. With Prolog 1 the early versions

gave a wait of a few seconds between student input and program response, but with Turboprolog the wait was imperceptible.

3. The problems that arose were through stack failure due to iterative constructions using recursion. These were solved eventually by using the program control mechanisms of 'cut' and 'fail' to give iteration by back-tracking. A great deal of time was spent on discovering such methods, as the PROLOG textbooks describe the cut mechanism only briefly and do not mention the problems and their solutions. There is a serious anomaly here, in that recursion is the standard method for iteration in PROLOG, because it is a logical construction and PROLOG is based on logic, but the method only works in small programs. With a large program a 'crash' is inevitable sooner or later due to stack failure unless iteration is achieved using non-logical computer-dependent methods involving the cut, regarded by PROLOG purists as tricks. At least two students at Leicester Polytechnic, graduating from small experimental programs in PROLOG to using it in substantial projects, had the same experience. Perhaps PROLOG purists should acknowledge the problem, and describe the appropriate 'tricks' in the text books, as PROLOG is liable to get a bad name as an accident prone language.

5.1.3 The hardware of WITS

The preliminary IV/CAL project carried out using a videotape system in the Educational Technology Unit of Leicester Polytechnic (described in Chapter 3) had indicated the following hardware requirements:

1. Videodisc was preferable to videotape, to give the required speed and accuracy of operation.

2. It was not practical to make a videodisc specially for this project, so that a suitable existing disc would be required.

3. A single screen system was preferable to a two screen system.

The videodisc player decided upon was the Philips Laservision Professional VP835, the best available in 1985. At that time there was still a choice of competing videodisc systems, and Thorn EMI were marketing their VHD system. (See Clemens, 1982, for a discussion of different systems, and Mably, 1984, for a report on the Thorn EMI discs.) However, it was decided that the Philips Laservision system was superior in terms of both operating characteristics and robustness, although more expensive, and the VHD system has now been withdrawn.

Fortunately an existing videodisc of suitable (though not ideal) format was available in the Solid State Electronics disc made by Epic Industrial for the EETPU. (See Lea, 1988). This disc contained a whole course for City and Guilds Solid State Electronics (SSE), made up of separate short sequences divided into units and modules. This was the traditional organisation of knowledge used in textbooks, syllabuses and teachers, and comprised the type of knowledge base considered suitable for a system based on the teacher's expertise. (See Section 4.3.)

To meet the single screen requirement a teletext system was originally chosen. In this system the videodisc output was displayed on a television equipped to receive teletext. Computer output was sent to the videodisc player as a stream of ASCII codes, and the player converted these to teletext messages superimposed on the television picture, in a special teletext encoder fitted to the player as an optional extra. This is shown in Figure 5.1 (a).

The alternative system was one which used a special 'fast-blanking' monitor to display the videodisc output. Computer output was mixed with this using a Videologic MIC interface board in an IBM PC computer. This was a superior system to teletext, as the full range of computer graphics could be mixed with the videodisc pictures, but it was not adopted at the start of the project for the following reasons:

1. Prolog 1 from Expert Systems, the language chosen to program the system, did not have colour or graphics facilities. The teletext system would provide these.

Teletext commands could be programmed by directing the computer to address the videodisc player as a printer.

2. The MIC system was expensive, as was the IBM PC in 1985. The computer chosen for the project was the Research Machines Nimbus, a powerful and inexpensive micro which seemed likely to become popular in education, but this would not take the MIC interface card.

The teletext system was thus in 1985 a relatively inexpensive way of obtaining a single screen system with attractive computer output in colour programmed in Prolog. During the course of the project things changed: IBM compatibles became available at low prices (lower than the Nimbus) and Philips began to phase out the teletext option on all their videodisc players but the most expensive ones. It became apparent that teletext systems for interactive video were going out of fashion. Also Turboprolog came onto the market, and was adopted for reasons described in the previous section. This would not run on the Nimbus. It seemed worthwhile to change to an MIC system run on a different computer, and the one chosen was the Opus PC II, an IBM PC in all but name, but twice as fast at half the price.

The initial and final systems are shown in Figure 5.1, (a), (b), (c) and (d). The change over from one to the other in the middle of the project entailed considerable modification of the code already written and re-writing of the teletext output rules as MIC rules. Along with obtaining and setting up the new hardware, this took several months.

5.1.4 System Details of WITS

The program was written in Turboprolog, which enabled it to be compiled, in its final form, to executable code. This final form occupied about 340k, excluding data files containing screen messages, videodisc data and multiple choice questions. This would just fit onto a 360k 6 1/4 inch floppy disc. Sizes of all the files comprising the trial version of WITS are shown in Figure 5.2.

A portable form of the program was prepared consisting of three floppy discs: a system disc containing the executable WITS program (the teaching module), a data disc containing files (the knowledge base), and a student disc which was left in the floppy disc drive of the computer while the program was running, and on which data for the student model was stored in a student file (the student model).

In fact it was possible to put the student file onto the data disc, contracting the three disc system to two for test and demonstration purposes. This form of the program, developed on the Opus PC II, ran on any IBM PC compatible and was demonstrated several times on IBM PCs and Olivetti M24s, but it would not run on the non-compatible RM Nimbus. Because of the size of the program and data files, a PC of at least 512k RAM was required. Were the three disc version to be used with a number of students, it would be convenient to use a different student disc for each student.

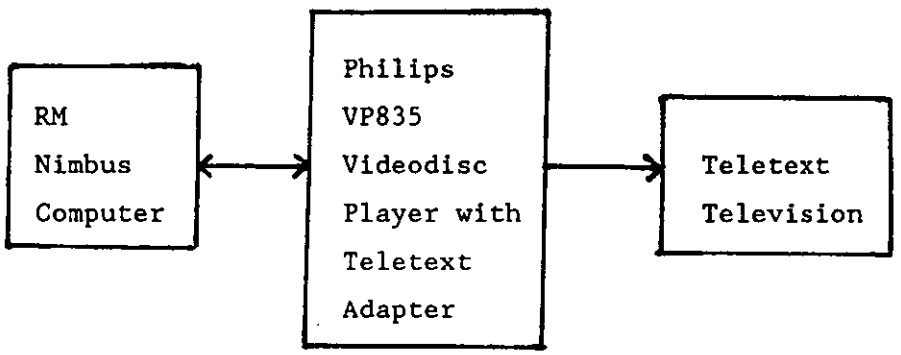
Reading in a program of 340k and data files totalling 150k takes several minutes on a PC running at the standard 4.5 MHz, and rather less on one running at 9.7 MHz like the Opus, so for the trials the program was run from a hard disc installed in the Opus with much faster access. However, to prevent the students having access to their files when evaluating the system, the system was configured to boot or start up from a floppy disc, which was then taken away. This meant that if a student ended the WITS program, or re-booted the system in the usual PC manner using the CTRL-ALT-DEL keys, he or she could not then re-load the operating system to have access to the files. Thus there was no possibility that the student files did not represent true student data. A backup file was stored at the end of each session on a student floppy disc.

The system was started in the same way with both the floppy based version and the hard disc based version used in the trials, by switching on with the system floppy disc in situ. An initial screen with the name of the system appeared, shown in Figure 5.3, (a). When the data files were loaded a further instruction to insert the student disc appeared. When this was done the program started.

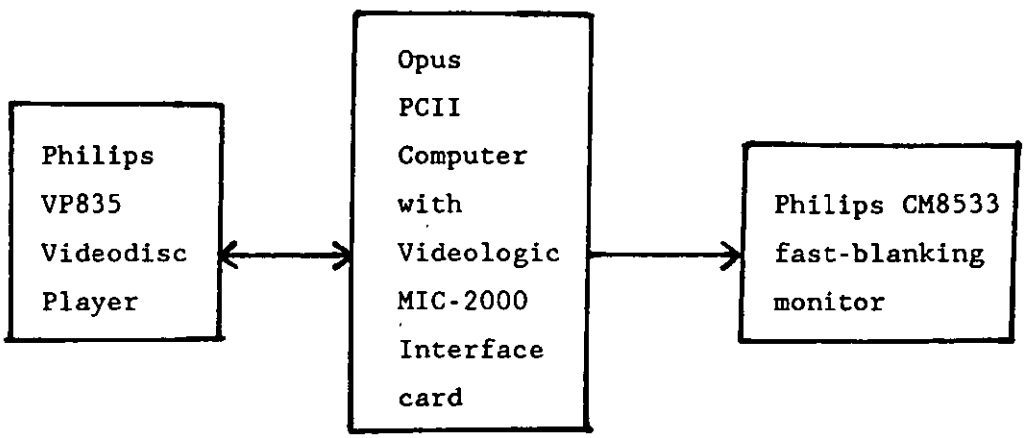
Figure 5.1

Hardware of the WITS System

(a) Diagram of initial hardware system of WITS (1985)

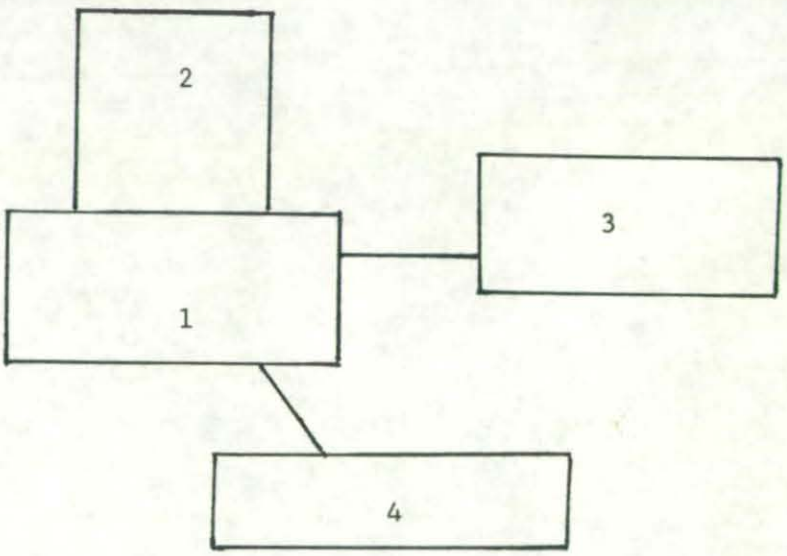


(b) Diagram of final hardware system of WITS (1988)



(c) Some screens of the initial teletext system.

(d) Final hardware of WITS



1. Opus PC II Computer
2. Philips CM8533 Monitor
3. Philips VP835 Videodisc player
4. Keyboard

Figure 5.2

Files to run the WITS program

(Videologic MIC system files and WITS source files not included).

	bytes
WITS .EXE	346613
AUTOEXEC.BAT	197
DISCDATA.DBA	20029
PROGDATA.DBA	21392
QUESDAT1.DBA	17095
QUESDAT2.DBA	10299
QUESDAT3.DBA	11946

5.2 The Student Interface

5.2.1 General

This component of the program is explained first so as to provide an introduction to the system as it appears to a student. The section will describe the ways in which a student can input his or her intentions into the system, and the ways information and instructions are output to the student.

There are three ways the student can spend time:

(a) Interacting with the system generally.
(b) Viewing videodisc sequences.
(c) Answering questions.

When the program proper starts, after the initial loading of data files and recording of the student's name, the student is asked how much he or she knows about the subject, and is then, depending on the reply, placed in one of the three learning environment modes, described in a later section. The student is then presented with the main interaction screen, shown in Figure 5.3 (b).

5.2.2 Normal interaction

Screen output in WITS follows broadly what have now become conventions, outlined in the last chapter (Section 4.6).

The windows of the main interaction screen are shown in diagram form in Figure 5.4, numbered as they are in the program code. Status and command information is at the top of the screen, in Windows 2, 3, and 4. Text messages are in Window 5 in the centre of the screen, but to the right, to leave a space on the left of the screen through which the last videodisc picture to be seen normally shows. This was a useful device, as most users of the system agreed, for reminding the student of what was done last. The instructions for continuing, and the space where student input is typed in, are in Windows 6 and 7 at the bottom of the screen.

Status information provided is the present environment mode (INSTRUCT, CHOICE or REVISE) in Window 3, the present module and unit, and the number of topics covered so far out of the total of the whole course, in Window 4.

The commands available, shown in Window 2, are as follows:

- I obtain information
- S search for a topic on the videodisc
- M miss out a topic
- C change mode
- Q ask for a question
- P see a profile of student progress
- R print out a student record
- F finish the session

Information available (on entering I) is presented in a menu of three types:

A System information

B Mode information

C Question information

This information, as coded in the prototype program, is shown in Appendix 5.1. See also Figure 5.9. This was considered adequate for the trials when the author was available to give help. The way the system responds to other commands is described in subsequent sections.

Perhaps the most important command is entered by pressing the space bar, which always in INSTRUCT mode, and in CHOICE mode except when a menu to choose from is on the screen, results in simply continuing with the course. The space bar is thus equivalent to 'continue'. It is possible in INSTRUCT mode to go through the whole course simply by repeatedly pressing the space bar, apart from breaks to watch the videodisc or answer questions.

All commands except 'continue' require <ENTER> to be pressed after the letter, and as such are really single letter key words rather than single key press commands. Use of the function keys for these commands might be better, but this might make the system less transferable to other computers. The program in fact treats the commands as keywords. This is because at any point the student can also enter whole sentences, and this would not be possible if the letters listed above initiated action by the system after a single key press at the start of a sentence. All the commands can be replaced by a variety of alternative key words. For example, 'continue', 'go' or just 'g' can be substituted for pressing the space bar. 'C', 'change' or 'mode' all achieve a change of mode. The PROLOG facts containing the data for these command key words are shown in Appendix 5.2.

A continuation message at the bottom of the screen always tells the student what to do to move on, in Window 6. This changes according to whether the response needs to be YES or NO, to press a key to make a menu choice, to press the space bar as the only option, and so on. Normally in INSTRUCT or CHOICE mode the message is "Space to continue, or a message" and in REVISE mode, which does not respond to a 'continue' message, "Type a sentence, word or letter". If the student enters a word or sentence, it appears as it is being typed in in the text input window, Window 7. The way words and sentences are analysed after being typed in is described in Section 5.5.5.

Figure 5.3

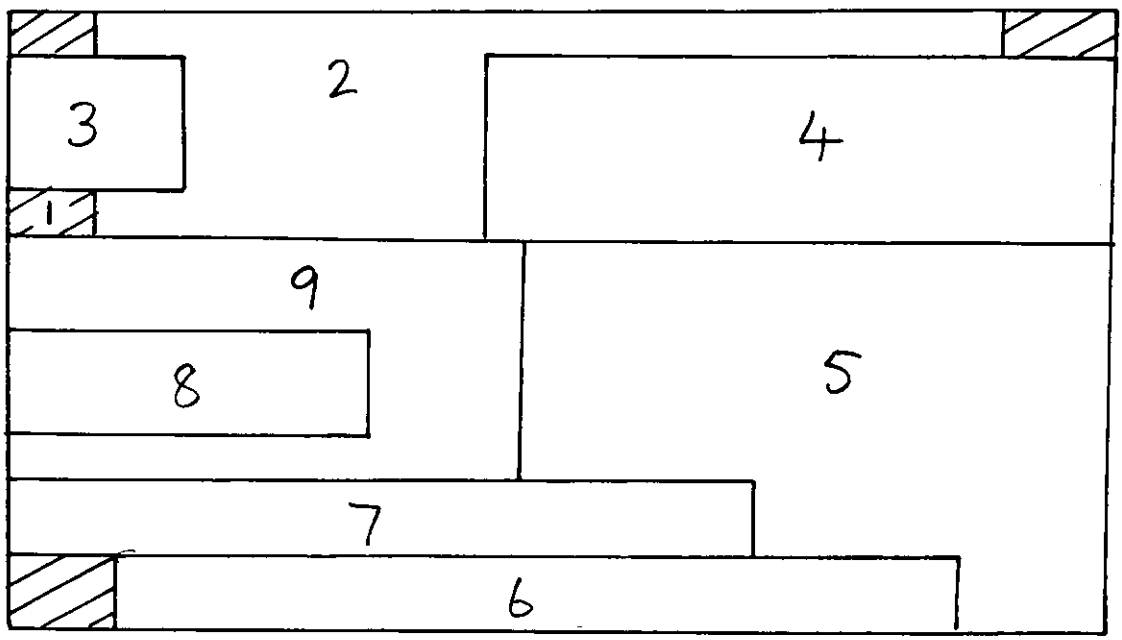
Some WITS Screens

(a) The initial screen of WITS

(b) The main interaction screen of WITS

Figure 5.4

Diagram of windows used in the WITS program



1. Full-screen background window
2. Command window
3. Environmental mode status window
4. Module, unit and topic status window
5. Main message window
6. Continue message window
7. Student input window
8. Error message window
9. Hint, message and answer for questions window

WITS thus normally accepts a space to continue the course, one of certain commands, or a typed in word or sentence forming a simple type of natural language input. If the student has temporarily diverged from the course, for example to search for a particular topic, request information or change mode, the input that is acceptable is necessarily different. A menu may be presented, or the only option may be to continue to the next screen using the space bar.

The main PROLOG rules for the screen layout in the three modes are shown in Appendix 5.3, and the rules for collecting different types of student input are shown in Appendix 5.4. Relevant subordinate screen layout rules are shown in Appendix 5.5.

5.2.3 Viewing sequences

When viewing a videodisc sequence, only the continuation window, Window 6, remains on the screen, with information about the single key press commands to move on, move back, pause, or end, leaving the rest of the screen free for the video picture. Examples are shown in Figure 5.5, (a) and (b).

The videodisc controlling rules are shown in Appendix 5.6 and represent, at the present time, possibly the only videodisc controlling code in existence written in PROLOG. The original versions using recursion had to be rewritten to achieve iteration using back tracking, to avoid computer stack failure. To explain the code in detail is not feasible here.

Some of the videodisc data facts are shown in Appendix 5.7. It will be seen that there are four types of videodisc sequence:

1. A moving video sequence (second argument 'vid').
2. A sequence of consecutive stills ('con').
3. A sequence of random stills from all parts of the videodisc ('ran').
4. A sequence consisting of one still frame ('one').

While in a video sequence the student can pause the video, start it again, return to the start, or end. While in either of the types of still sequence the student can move one forward or one back, return to the first one, or end. When viewing a single still sequence the only option is to leave it by pressing space.

Some sequences on the Solid State Electronics disc are sequences of still strung together as video with a sound track. These are treated by WITS for practical purposes as ordinary video sequences. In some cases one sequence has two alternative sound tracks, using the two stereo sound channels. These are treated by WITS as separate video sequences. Either or both sound tracks can be made part of a sequence by suitably encoding the data for the sequence.

There are two versions of the rule to show each type of sequence depending on whether the sequence is part of a chain of sequences or not. On the Solid State Electronics disc there are some sequences which end in a question, and depending on the student's answer it is necessary to branch to the correct one of several other sequences, as shown in Figure 5.5 (b). Some of these branching chains go through several layers of question and answer, as shown in Figure 5.5 (b).

Thus features of branching CAL, or 'programmed tutorials', can be included for a suitably designed videodisc. WITS can accommodate branching chains of videodisc sequences of mixed types up to a considerable length which is ultimately determined by the computer stack size, as the coding uses recursion. This is a rigid, small-scale branching mechanism to accommodate groups of linked sequences mastered onto the videodisc. The main branching according to student choice in WITS is on a more flexible basis as described later, and is not constrained by computer considerations.

5.2.4 Answering questions

When doing questions, the Status Window 3 contains the question number, and Window 2 the multiple choice question. Window 5 contains the five possible answer choices. Window 6 as usual contains continuation information, here simply 'press the appropriate key'. Window 7 contains a message saying "QUESTION TIME" and Window 4 is not used. An additional window is used for hints and model answers, Window 9. This uses the left-hand side of the screen where the last video picture normally shows. Video is switched off during questions in case it provides help. This will be returned to in Chapter 6 where the assessment process is described. Figure 6.5 shows a sequence which might occur as a student answers a question.

5.3 The Knowledge Base

The knowledge of Solid State Electronics (SSE) which is to be taught to the student is contained on an optical videodisc. The same knowledge stored in digital computer form would take up an enormous amount of memory, and some sequences could not be stored digitally at all. For example, a sequence to illustrate positive holes moving in the opposite direction to negative electrons shows people in a dentist's waiting room moving along a row of chairs, and the vacant seat moving the opposite way. This classic video sequence, shown in Figure 5.6, neatly illustrates a point which would be impossible to illustrate as clearly with a computer program alone. Numerous other sequences on the videodisc are similar.

All the information on the disc is contained in short teaching, testing or revision sequences. Each of these corresponds to a topic to be made available to the student. Topics are grouped in larger units of knowledge, and units into a small number of major modules of knowledge.

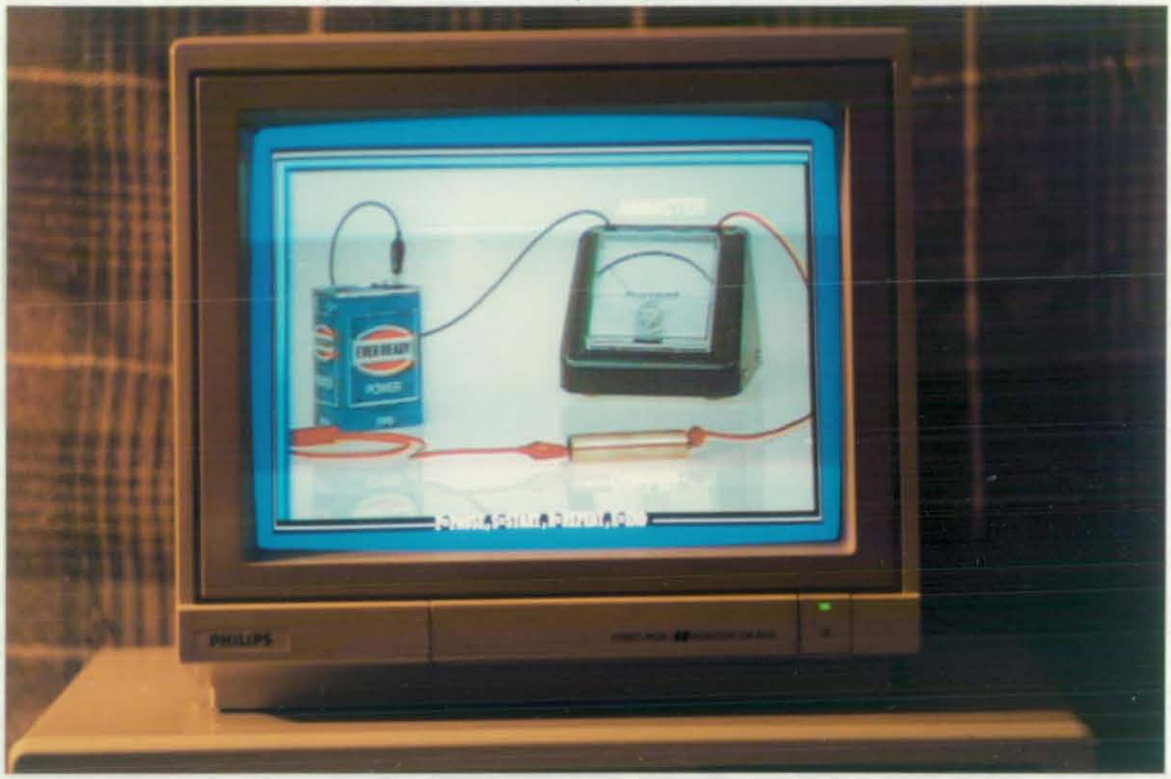
The grouping of topics into units and modules by the producers of the videodisc, as provided in a rather crude, typed document and in the 'contents page' on the disc, were found to be illogical and inconvenient. For example, some units contained as many as fourteen topic sequences, while a number of other units consisted of a single topic sequence in themselves. This was felt to be confusing for students, apart from causing screen layout problems, and the topics were rearranged within unit groupings so that no unit contained less than three or more than eight topics. The units within modules were rearranged so that no module contained less than three or more than six units.

After rearrangement, the modules could be thought of as parts or sections into which a book on Electronics might be divided, the units as chapters within each part, and the topics as sections within the chapters. This is a common way of arranging textbook knowledge in science and technical subjects. The modules, units and some of the 121 topics of the course are shown as represented in the coding as PROLOG facts in Appendix 5.8. The title of the module, unit or topic appears as a PROLOG list of atoms between square brackets. (Note that the title of the course is stored as the title of Module 0, and that there are also, for programming purposes only, a Unit 0 and a Topic 0.)

Figure 5.5

Videodisc viewing screens of the Electronics course

(a) Conductivity experiment



(b) Zener diode question

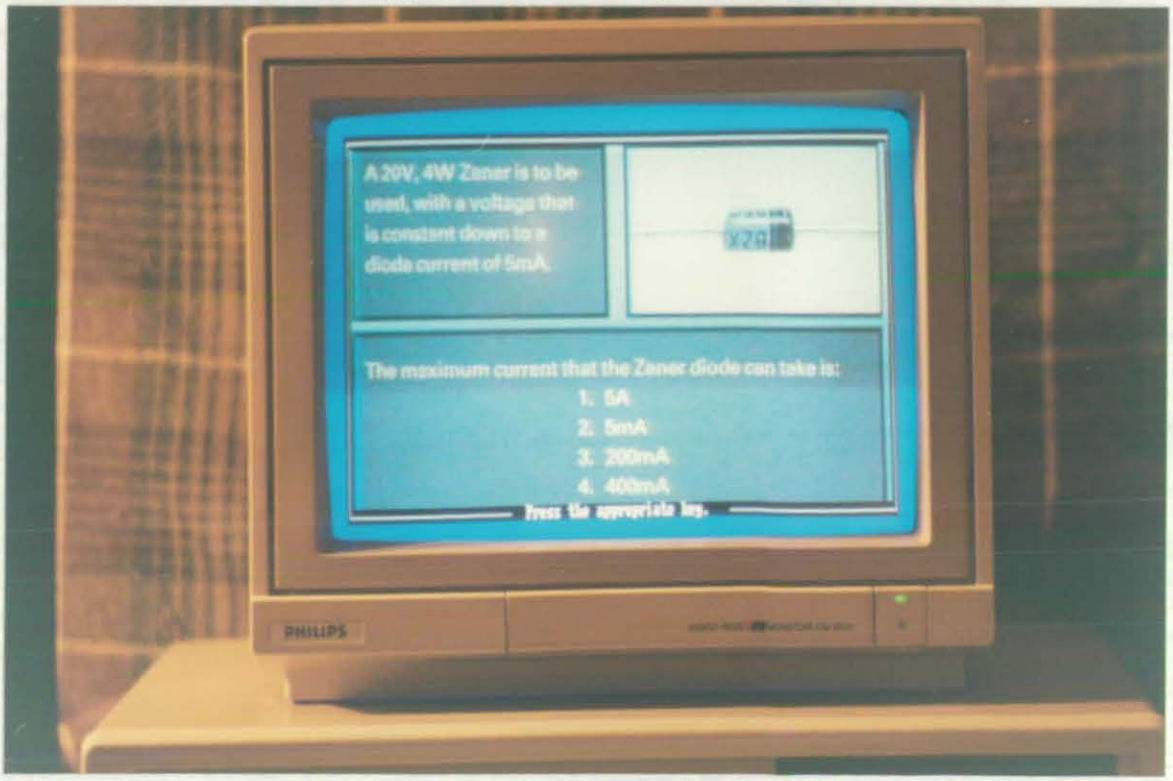


Figure 5.6

Screen from a sequence to show flow of holes during conduction

Corresponding to each topic there is also a PROLOG fact giving the relevant videodisc data of the sequence. Some of these sequence facts are shown in Appendix 5.7. In a video sequence, identified by the PROLOG atom 'vid' as the second argument, the third argument is a code for the audio track (1 or 2, or 0 for both), and the fourth and fifth are the start and end frame numbers for the sequence.

In a sequence of consecutive stills, identified by 'con', the third argument is the number of stills and the fourth argument the start frame number. In a sequence of random stills, identified by 'ran', the last argument is a list of all the frame numbers of stills forming the sequence, in order. For a sequence consisting of a single still, identified by 'one', the fourth argument is the frame number of the still.

For the Electronics disc there was another dimension to the knowledge base, a store of multiple choice questions used in assessment, stored as conventional computer text files. These questions, and their storage and presentation, will be described in Chapter 6.

5.4 The Teaching Module in WITS

This part of the program simulates the main expertise of the teacher, and as such attends to the following tasks:

1. It presents the knowledge contained in the course to the student in the right order and with a degree of flexibility, thus organising the student's learning.

2. It organises assessment of the student and compiles data for the student model.

3. It uses the student model to adjust to the student in certain ways, discussed later.

4. It provides information and feedback as needed, including a written report if required.

Essentially, after the initial entry process, the WITS program is a loop which repeats itself until the student chooses to end the program. The loop can be summarised:

1. Get an input from the student.
2. Analyse the input.
3. Take action on the input.
4. Repeat.

A recursive form of PROLOG code to achieve a loop of this kind is shown in Figure 5.7 (a), and this illustrates the basic structure of the WITS program. However, such a form of code would cause a program failure in practice after just a few repetitions, and a rather more convoluted form which was found to be necessary, achieving the loop using back tracking caused by a cut-fail mechanism, is shown in Figure 5.7 (b). This is an example of the practice of PROLOG differing substantially from the theory, a matter which may not be addressed sufficiently by proponents of PROLOG.

The action taken by WITS after analysing the student's input is to search a list of 34 main rules and act on one which is appropriate. The rules are reproduced in Appendix 5.10. They have been slightly simplified for greater clarity by removing some of the complexities necessary in Turboprolog to speed execution. The rules may be summarised as follows:

1. After a certain number of topics has been seen by the student, the number depending on the student's ability, a flag is set to signal to the teaching module that question time has been reached. If this is the case, the student is presented with some questions, the number depending on the student's ability. (Rules 1 to 4.) This is explained further in Chapter 7.

2. If the student has used the command M to miss out or skip a topic, this request is dealt with according to the circumstances. (Rules 5 to 8.) This is explained further in the next section on the WITS environment.

3. If the student has pressed 'space' to continue with the course, this is dealt with according to the present mode and the stages reached. (Rules 9 to 21.) Rule 11 is a

key rule which shows the next topic if it is time to do so. Rules 12 to 15 deal with INSTRUCT mode, and rules 14 to 21 with CHOICE mode. The way these rules allow progress through the course in the different modes is explained in the next section.

4. If the student has used one of the other commands, it is dealt with suitably. (Rules 22 to 28.) Rule 22 presents a menu of types of information available, then shows appropriate information screens, Rule 28 checks that the student wants to end the session, and if so, ends it. Rules 24 and 25 enable the student to change mode or to commence a search for a topic, and are dealt with further in the next section. Rule 23 presents the student with a progress profile, Rule 26 presents questions on request, and Rule 27 gives a printed report. These are dealt with further in Chapter 7.

5. If a key word has been detected in the student input, and linked to a particular module, unit or topic, this is dealt with according to the environment mode the student is in. (Rules 29 to 33.) This is dealt with further in the next section.

6. If the student input has not yielded anything on which to take action so far, a 'catch-all' message is output apologising for not being able to understand. (Rule 34.) As with some other messages used frequently, a random choice from several messages of similar meaning is output to give variety.

As well as running the main loop, the Teaching Module needs to deal with the reading of data files and, in the case of a student's first session, collecting the student's name, establishing how much of the course is known and allocating the student to an environment mode. Screens for this are shown in Figure 5.8. The rules which do this are shown in Appendix 5.11.

5.5 The Environment Module

5.5.1 INSTRUCT mode

This part of the program handles the flexibility made available to the student which might be said to constitute 'intelligence'. Three different environments are available, presented to the student as 'modes', a word students are familiar with from using other programs such as word processors. In general terms, the modes give differing degrees of choice, from virtually none in INSTRUCT mode, through guided choice in CHOICE mode, to total freedom to explore the course in REVISE mode. The difference between the modes can be seen summarised in Figure 5.9.

The instructional strategy in INSTRUCT mode is straightforward, as is the programming of it. In the knowledge base the topics of the course are labelled with ascending numbers which represent an optimum order in which to present them. The program records the last topic the student saw, and when 'space' is pressed to continue presents the next one. No choice of units or modules is allowed.

A student is placed in INSTRUCT mode if he or she says none of the course was known. The assumption is made in this mode that the student has no prior knowledge, and it is based on programmed learning methods. Thus if the student finds a topic to study via the search option or by using a key word, he or she is not allowed to see it out of order unless it has already been covered in the course or seen before in another mode, i.e. free searching can only be used to look back over the course, not forward. The student is not allowed to miss out topics, though of course it is possible to view them briefly, using 'end' almost as soon as they have been started. When questions are to be presented to the student for assessment, he or she is not given any option to refuse them.

5.5.2 CHOICE mode

The instructional strategy in CHOICE mode, as the name implies, is to give the student choice and some freedom of action, while at the same time leading him or her through the course. The student is assumed to know enough about the subject to make decisions for him or herself.

Figure 5.7

Main program loop in PROLOG

(a) Recursive version

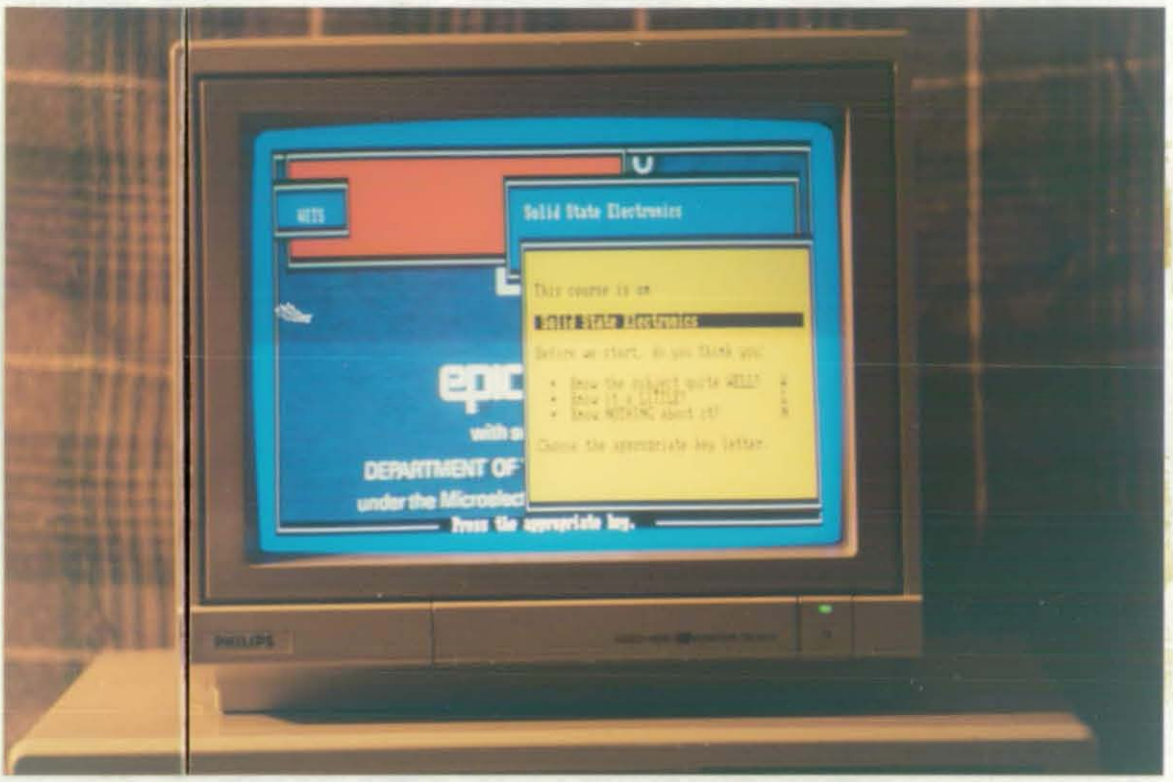
```
goal tutoring.
tutoring if initialise and interacting.
initialise if ...
interacting if get_input(I),analyse(I,X),rule(X),interacting.
```

(b) Non-recursive version as used

```
goal tutoring.
tutoring if init(start_up), started.
tutoring if continuing.
started if !,init(initialise).
init(start_up).
init(initialise):- ... !,fail.
continuing if repeat, interacting.
interacting if get_input(I),analyse(I,X),rule(X),!,fail.
```

Figure 5.8

Some of the screens for collecting initial information



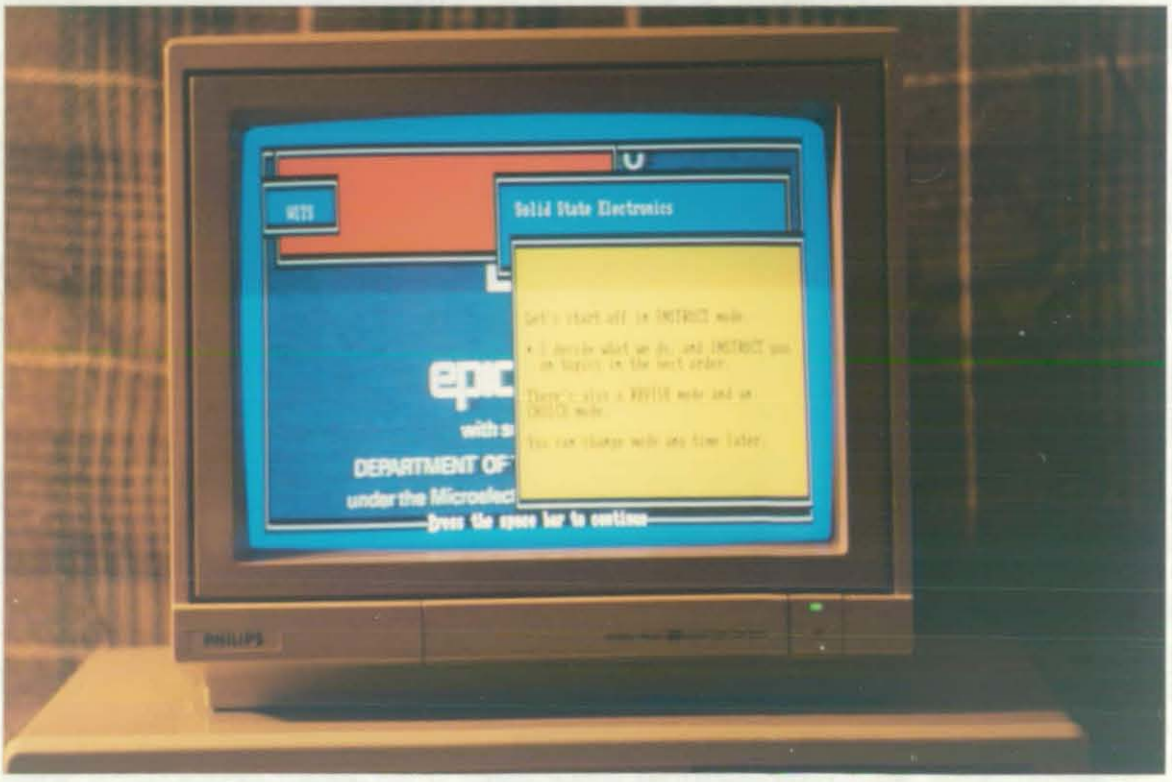


Figure 5.9

Differences between the WITS environmental modes

MODE	INSTRUCT	CHOICE	REVISE
Learning route	Predetermined	Individual	None
Choice of unit/module/topic	None	Modules & units where possible	Any shown
'Skip a topic' option	No	Yes	Not applicable
Topics found via info	Only if previously seen	Only if seen or known	At any time
Topics requested by keyname	Only if previously seen	Only if seen or known	Any shown
Assessment	No choice	Up to three refusals	Optional but prompted
Help, problems, change mode, questions, information, finish available	Yes	Yes	Yes

At the start of the course it is possible that the student already knows some of the material, or for his or her own reasons wishes to miss out some of it. The student is asked whether this is the case, and if so is led through a special sub-routine which lists the modules and units of the course and asks which are to be missed out. The student can thus design his or her own learning route through the material of the course. Some screens which show what happens when designing a learning route are shown in Figure 5.10. The rules of this special sub-routine are given in Appendix 5.12.

The facility for the student to design his or her own course is also available when changing mode into CHOICE mode. It is possible to change mode at any time by using the 'c' command.

When the modules and units which are to make up the course are presented to the student, a choice is given wherever possible. With the SSE course, a module tends to contain an introductory unit which needs to be studied first, then the order in which other units are studied is not important. This is probably the case with much science and technology material. A diagram can be constructed for the modules of the course, and for the units within each module, showing the possible next choices from each module or unit. Such diagrams for the Electronics course are shown in Figure 5.11, (i) to (vi).

A similar choice of topics within the units was considered unnecessary, and might lead to time-wasting complexity for the student. Topics are presented consecutively within the unit as in INSTRUCT mode. However, in CHOICE mode the student has the option of missing out or skipping topics, which is not allowed in INSTRUCT mode.

These diagrams each represent a networks or net, with modules or units at the nodes and choices as links between them. They are not semantic nets, as the links are of the same type and are not relationships between the nodes. They do however resemble possible transitions in a search space, another AI concept. (For an explanation, see for example Winston, 1977.)

The information to enable WITS to navigate the nets according to the student's choices is contained in the knowledge base. Referring to Appendix 5.8, the fourth argument for each module fact is a list of the possible modules to go on to after that one, when learning the module. The

fifth argument for each module fact is a list of the possible units to begin with in the module when beginning it. Similarly, the last argument for each unit fact is a list of possible units to go on to when learning the unit.

The navigation of the nets so as to construct a new list of choices to go on to for the student, on completion of a module or unit, is more complex than may be thought. If the first unit in a module is completed and none of the others has been seen or is known, the choice list to go on to is straightforward, and is simply the list which is the last argument of the fact for the completed unit. If some units have been seen or are known, however, it is best not to present them again to the student. Indeed, it is part of the route planning 'contract' with the student not to present them. Simply to delete such items from the choice list is not sufficient, as a deleted unit might be the only route to another unit, which would then never be seen. What is required is to delete seen or known units, but to substitute for them their own last argument choice lists. Any duplicated units then need to be deleted so that each possible unit occurs only once in the choice list for the student.

This type of search space treatment, which allows for any possible sequence of student choices, and eliminates modules and units as they are seen, is not found in conventional CAL programs, and can be described as an intelligent feature. The program code rules handling this feature are shown in Appendix 5.13.

There are other ways in which the student is given choice or responsibility in CHOICE mode. Topics requested by key name or found by the search option can be seen if they have been declared as known at the start of the course, or have already been seen as in INSTRUCT mode. Thus these mechanisms are intended for revision here as well. When it is time for questions in CHOICE mode, the student is allowed to refuse questions up to three times. After this, if the student refuses to accept a question, he or she is transferred to REVISE mode.

5.5.3 REVISE mode

No attempt is made to lead the student through the material of the course in REVISE mode. Total freedom is allowed. Topics found by the search

option or by key word can be seen without restriction. Questions are offered at regular intervals, depending on student ability as before, but they can be refused. If the student reaches the end of the course without having been assessed, he or she simply has no progress profile to examine and cannot obtain a printed report. It is then possible, however, to do questions as an end of course test.

The only way the student can study the course in REVISE mode is to use the search and key word mechanisms, which will now be described.

5.5.4 The search option

On using the command 'S' for search the student is given a menu of all the modules of the course, lettered from 'A' onwards in optimum study order, with 'Z' to return to the main interaction screen. On choosing a module the student is given a menu of all the units in that module. On choosing a unit he or she is given a menu of topics in that unit. On choosing a topic, in REVISE mode the topic is immediately shown. In INSTRUCT mode it is only shown if already seen, and in CHOICE mode if already seen or declared as known. After seeing the topic the student is returned to the topic menu, and so can easily work through or explore the topics of a unit, in any order. On pressing 'Z' in the topic menu, the student is returned to the unit menu for the module, and can thus work through the units of a particular module.

Screens that appear in the search option are shown in Figure 5.12, and the rules for it are given in Appendix 5.14. The search option represents a powerful exploratory facility, not unlike searching the videodisc manually, but easier to use and with the advantage of showing the structure of the subject more clearly. (To explore the Electronics disc manually a list of frame numbers is required, and these have to be tapped out on a hand-held control.)

5.5.5 Natural language in WITS

If the student chooses to input a natural language sentence, the input is converted by WITS into a PROLOG list of atoms. This is first tested to see whether it contains one of the commands, or a word corresponding to a

command. (See Appendix 5.2.) In particular, 'space' is a command to continue the course. If no command has been entered, the input list is tested for keywords identifying a module, unit or topic. Some screens showing keywords that have been entered, and the response, can be seen in Figure 8.1.

The first test is to see whether the actual keyname of a module, unit or topic has been entered, i.e. its name exactly as it is used in the course. (See Appendix 5.8.) This might be the case if a keyword has resulted in a module or unit being detected, in which case a list of units or topics respectively will have been presented to the student, from which the student can choose by entering a keyname exactly as it appears in the list. If one of the keynames representing a module, unit or topic of the course is detected as a sublist of the student's input list, the identifying parameters of the module, unit or topic are returned to the teaching module.

If no keyname is present, the input sentence is tested to see whether certain identifying keywords attached certain modules, units and topics are present. Some of these keywords are shown in Appendix 5.15.

When testing, each atom of the input list of atoms is matched against the keyword atoms. This is a lengthy process, if a keyword is not found to be present early in the search. Thus to cut down searching time certain words, mostly definite and indefinite articles and prepositions, which do not occur as keywords, are removed from the input list at the start. Also, to avoid inclusion of plurals for most words in the keyword lists, and what would effectively be a search for each word twice (once for its singular and once for its plural) no words are allowed to end in 's'. The final 's' is removed from any in the input list which have one, and no keywords are entered with a final 's'. Thus, words such as 'bias' and 'series' have to be entered as 'bia' and 'serie' and a word such as 'supply' with an irregular plural has to be entered as both 'supply' and 'supplie'.

Any number of identifying keywords can be included in a list, but even so it is not possible to distinguish between all topics adequately with single keywords in a complex course. For example, on the SSE disc there are sequences on 'The Transistor', 'Transistor Circuits', 'Uses of Transistors', 'Testing Transistors', and 'The Transistor as a Switch'. Thus in WITS a system of linked lists is used. The way the data is

Figure 5.10

Some screens seen when designing a learning route through the course

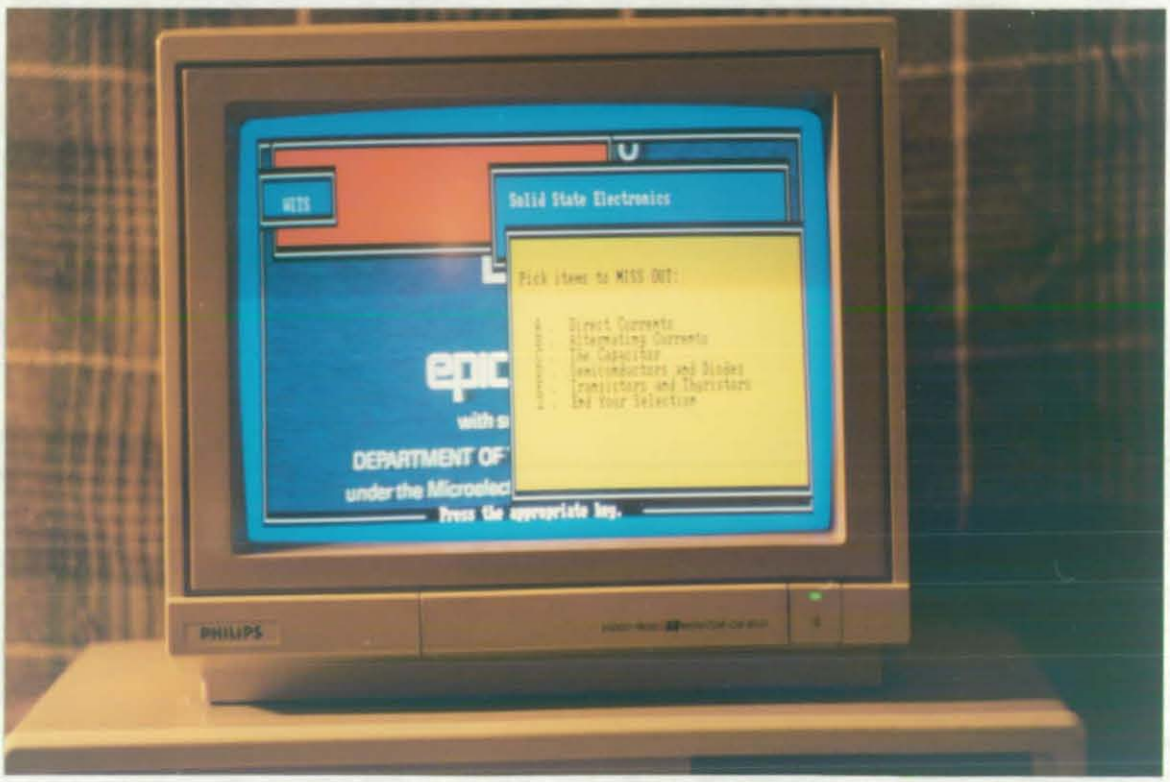
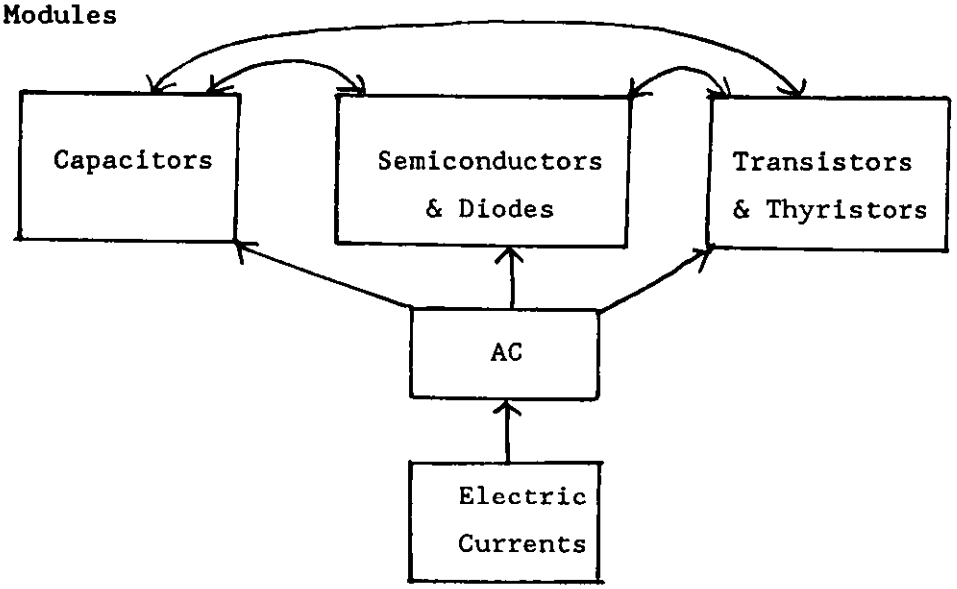


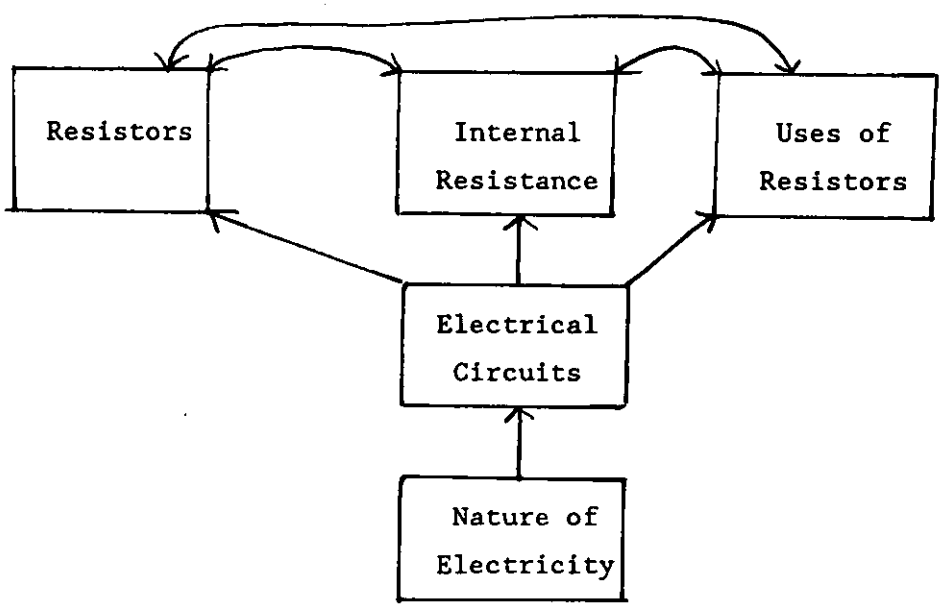
Figure 5.11

Transition nets for the modules and units of the Electronics course: (i) Modules, (ii) to (vi) Units within Modules

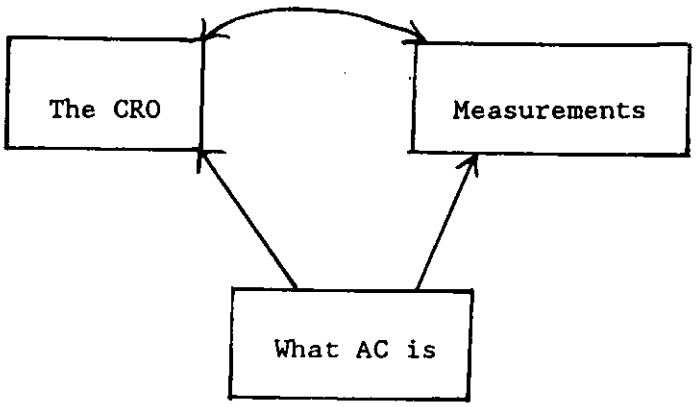
(i) Modules



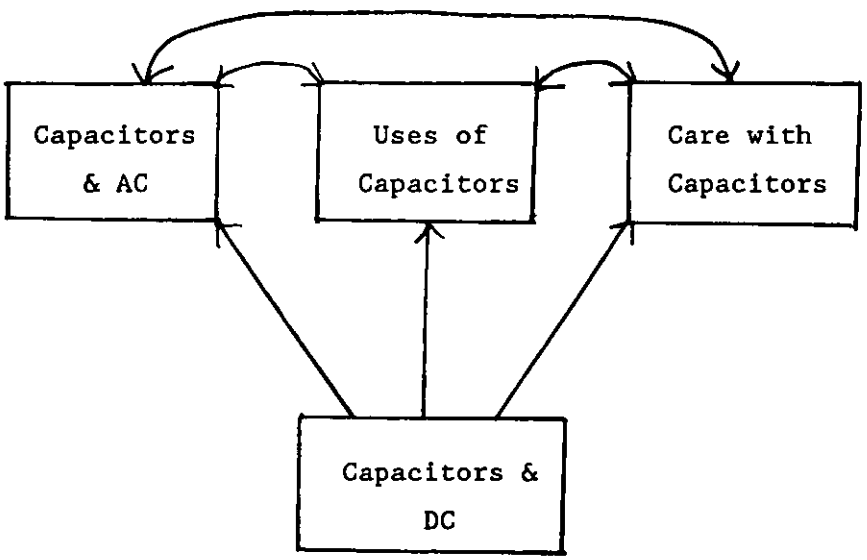
(ii) Module 1 Units



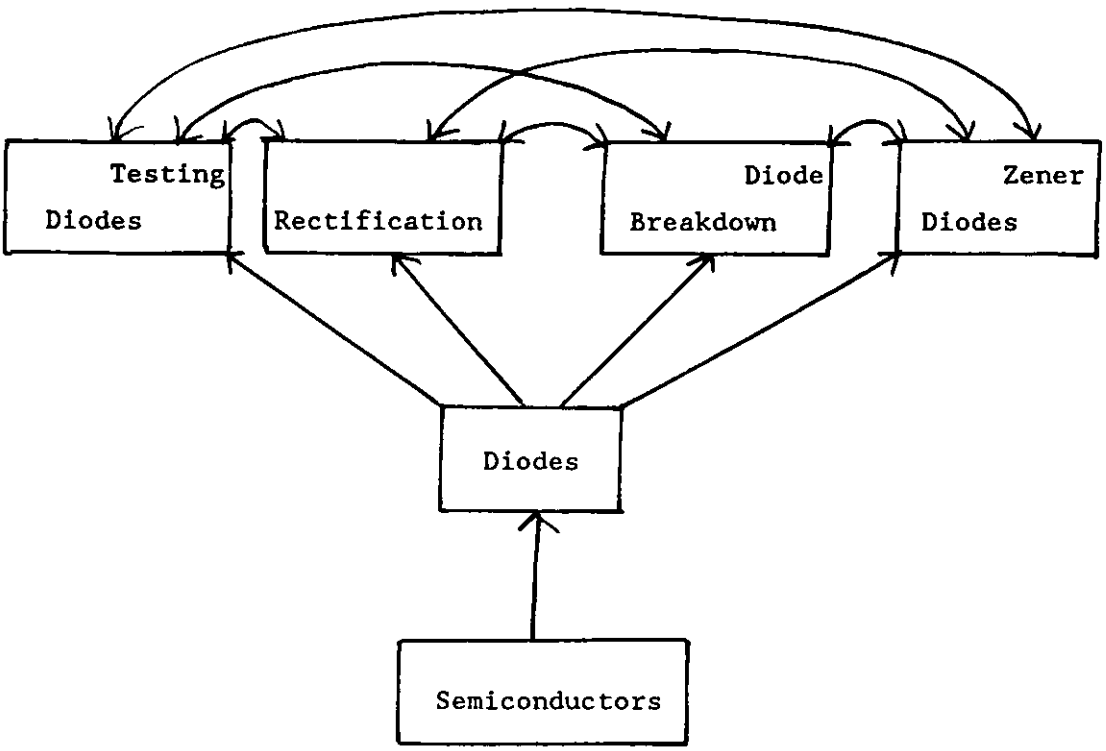
(iii) Module 2 Units



(iv) Module 3 Units



(v) Module 4 units



(vi) Module 5 Units

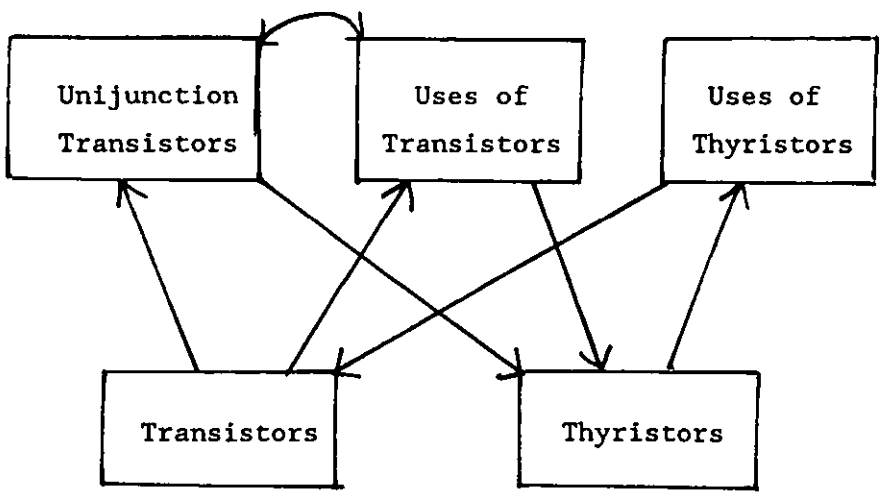
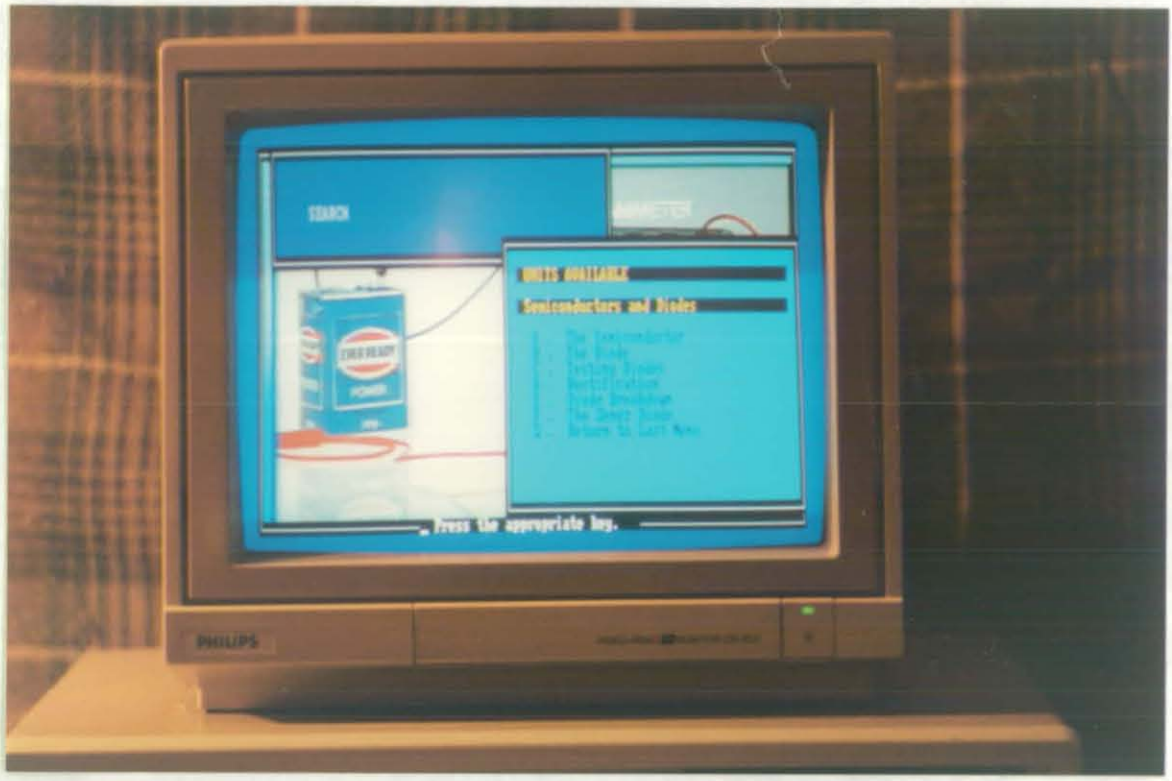
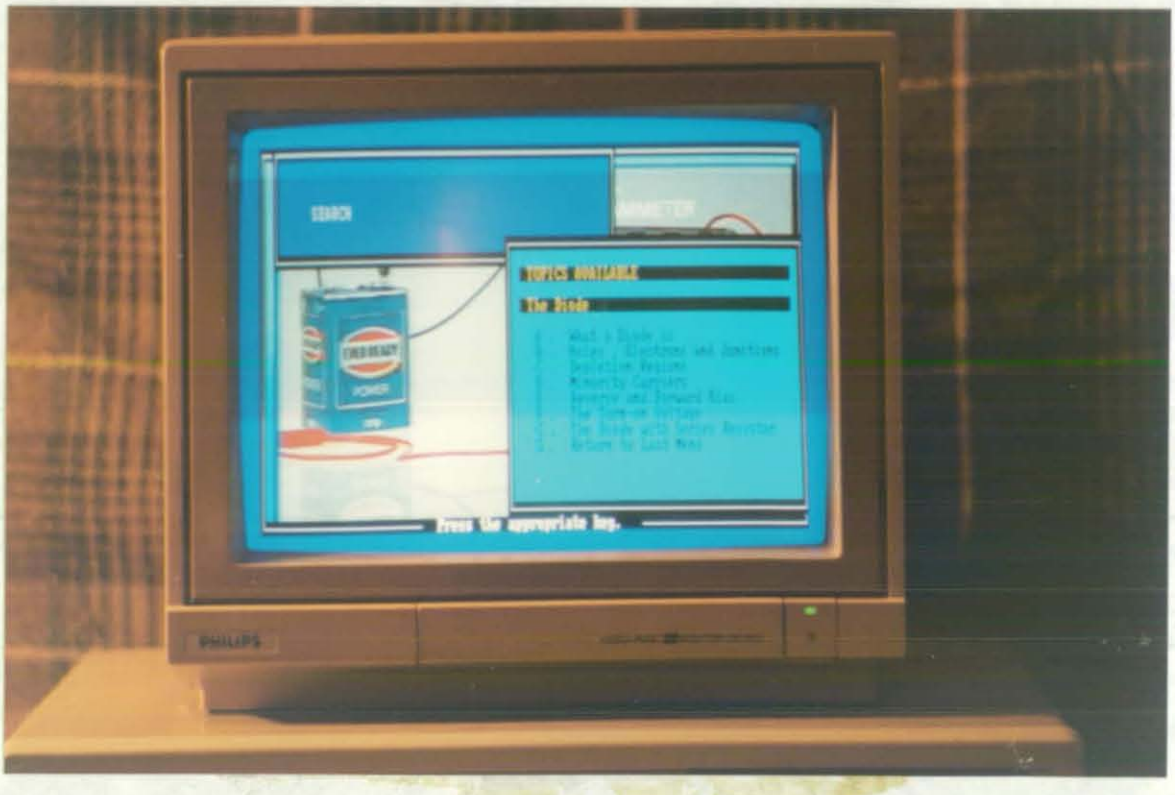


Figure 5.12

Screens for the search option





structured for linking of keyword lists for modules, units and some of the topics which cannot be adequately identified by single keywords is shown in Appendix 5.16.

Certain general groupings were also found to be necessary, shown in Appendix 5.17. Thus in the example above, the keyword list for 'transistor' identifies Unit 19 on 'The Transistor' but linking this list with general list 1 identifies Unit 23 on 'Uses of Transistors', and linking it with general list 7 identifies Topic 106 on 'Testing Transistors'.

The test for linked keyword lists needs to be carried out before that for single keywords, as the linked lists also occur as single keyword lists, and otherwise would always be detected at this stage. The testing sequence for keywords used in WITS, for which the PROLOG coding is shown in Appendix 5.18, is as follows:

1. The name of a module, unit or topic as used in the course is searched for in the input sequence.

2. The input sentence or list has all unwanted words removed.

3. It has the final 's' removed from all words which have one.

4. If one word only remains, this is tested against single keyword lists.

5. If more than one word remains, the list is tested for linked keywords.

6. If this fails, the list is tested for single keywords.

7. If this fails, a message is printed indicating that the input is not 'understood'.

It will be seen that this is a 'thesaurus' method of detecting the student's meaning from the input entered. In attempting to locate concepts, rather than individual English words, it has a similar aim to Schank's 'conceptual dependency' method of analysing natural language input, though it is not nearly so elaborate. Certain words convey fairly clearly a

particular concept the student is trying to convey. For example, if 'plot', 'graph', 'curve' or 'characteristic' occurs, it is reasonably certain that the student is looking for information on graphs or characteristic curves, and a linked keyword will specify the type of graph. Clearly the context of the specific knowledge domain of Electronics is important: 'plot' in the context of a spy game might convey something entirely different. The SSE disc contains sequences on diode, transistor and thyristor graphs.

5.6 Conclusion

5.6.1 WITS as a transferable shell

The WITS system was designed and built to test the ideas on the requirements of a course-oriented ITS described in the previous chapters. It can lay claim to being intelligent in the flexibility of the environments offered to the student and in the use of some AI techniques, notably a rule-based structure, use of the language PROLOG, and simple natural language input.

A feature of WITS is that it forms a transferable shell, and its teaching expertise can be applied to other knowledge bases on other videodiscs. The structure of the program was designed as a general teaching program, independent of any specific knowledge domain. Appendix 5.9 shows data relating to the Van Gogh disc produced by Philips, corresponding to Appendices 5.7 and 5.8 for Electronics. The system was found to work quite adequately on this very different disc. This will be discussed again in Section 8.2.2, and some screens showing WITS operating on the Van Gogh videodisc are shown in Figure 8.1.

5.6.2 WITS as a declarative front-end

Much of the information on the Electronics videodisc consists of 'declarative' factual material, which is presented to the student as first-time knowledge or revision. However, there are also many question sequences authored onto the disc for interaction. The questions built into WITS for

assessment and building a student model, described in the next chapter, also contributed to making the prototype a highly interactive and 'procedural' system.

The WITS system in conjunction with the Epic Solid State Electronics disc could with some modification be used as a declarative 'front-end' for an ITS such as SOPHIE, which was designed as a 'procedural', interactive system for students to practise Electronic trouble-shooting.

Other CAL programs relating to Electronics could be 'called up' at suitable points in the course. The WITS system occupies less than 1 Mb of computer storage, and the main program occupies about 340K of computer RAM. Computer memories of 1000K (as on the Opus used here) are now common. Even with a large knowledge database the system could easily be stored on a hard disc of 20 or 30 Mb along with many supporting programs. More information could be provided on other videodiscs, loaded as required. It is likely that this is the form whole course systems will eventually take, consisting of a front-end organising program which calls upon other programs to cover specific aspects.

The WITS prototype Electronics system could be described as a declarative front-end for procedural ITSs, called for by Richardson (see Section 2.5.3), or it could be described as a videodisc front-end, as called for by Laurillard (see Section 3.6).

5.6.3 WITS as a teacher-based system

The system as described in this chapter performs two of the main tasks of the teacher, in organising the student's learning and in presenting first-time, declarative knowledge. How the system covers another of the teacher's main tasks, that of assessing the student and interacting with the student to reinforce procedural aspects of learning, is covered in the next chapter.

Referring back to the analysis of the teacher's role in Section 4.2.1, most aspects of the teacher's role listed there are covered to some extent by WITS. Those aspects not mentioned in this chapter or the next one are

covered implicitly by the material on the videodisc. For example, teaching aims are highly subject-specific, and such aims for this course (apart from that of coaching students to take an exam) will have been formulated by the designers of the Electronics material on the disc. Also, there is much material on the disc which has been designed for interest value and relevance. There is little evidence of humour, or of attempts to put across a flavour or a personality, but this may be due to the nature of the subject, and in a medium using television techniques it is clear that such things could be attempted on a suitably designed, purpose-made videodisc.

The student model of WITS, the part of a CAL system which is often taken to identify it as an ITS, is described in the next chapter.

Chapter 6

The Student Model of WITS

6.1 Introduction

The inclusion of a student model has almost come to be the feature which characterises intelligent tutoring systems. "A characteristic showed by many ITSs is that they infer a model of the student's current understanding of the subject matter and use this individualised model to adapt the

instruction to the student's needs." (VanLehn,1988.) VanLehn goes on to refer to the construction of a student model as 'diagnosis'.

Since WITS was modelled on the teacher, the student model in WITS was largely based on the conventional process of student assessment. Teachers deal daily with models of their students and refer to the process as 'assessment'. Traditionally teachers set their students questions to answer in order to assess them. In fairly recent years objective testing has become popular, particularly multiple choice testing, partly because it lends itself to marking by computer.

Assessment can be divided into formal assessment, where written work is set, collected, marked and gone over, and informal assessment, where the teacher asks questions or sets small tasks. The response to informal assessment is immediate, and the result of the assessment is often only recorded in the teacher's memory. Formal assessment is more rigorous, but slower. The work may no longer be fresh in the student's mind when it is gone over.

It might be said that assessment is sometimes unsatisfactory in schools. A major reason for this is the volume of marking a teacher has to deal with; if a secondary teacher has six different classes in the week, which is quite common, with an average of thirty pupils per class, this will produce 180 pieces of work to mark per week. There is pressure to set work less frequently, or mark roughly. Another reason is the crowded nature of syllabuses, particularly in science. In order to cover everything, teachers often have to adopt a lecturing approach, and find that adequate assessment of every topic covered is not possible. Yet another reason is the pressure on teachers' time from other quarters, in the form of meetings, new courses, pastoral work, union activities and many others which have grown up lately.

A computer will spend as much time with a student as required. It will mark work in a few hundred nanoseconds, and it will not make any mistakes. It cannot cope with natural language or spoken input in answer to problems, but provided the form of the assessment can be made unambiguous, the computer can cope with the volume of a student's assessed work rather better than a human teacher.

In assessment by a whole course ITS it would be desirable to include the best features, but not the worst, of human assessment. A useful form of student assessment would be one which derived broad generalisations about the student from relatively few questions, as in informal assessment by a teacher, giving immediate feedback.

The description of the method of assessment outlined in this chapter, and used in WITS, is based on an internal paper circulated at Leicester Polytechnic (Callear, 1988).

6.2 A Probability Method of Assessment

6.2.1 Some criticisms of conventional multiple choice testing.

It has been mentioned that Laurillard (1985) encountered problems with ambiguous questions, which was also a problem in the preliminary project carried out (see Section 3.5.2). This discussion will be confined to five-answer multiple choice (MC) testing, which is unambiguous, commonly practised and widely accepted to be objective. Several problems may be identified.

a) Conventional MC testing is inefficient, answers being either right or wrong. A large number of questions is needed to obtain an assessment of any accuracy.

b) There is considerable doubt over what is being tested. It often seems that the question was set simply to test the vague proposition: "Can the student 'do' the subject at this level?"

c) A conventional MC test does not differentiate between different student abilities, when interest is growing in describing the nature of student abilities in a 'profile' rather than simply assigning an overall grade. (See Hodson and Brewster, 1986.)

6.2.2 A strategy for improving MC testing.

There is a need here to express partial correctness in the student's answers. Each answer needs to be coded with an 'answer parameter' expressing its degree of correctness. To a designer of expert systems the notion of partially correct information brings to mind the use of probabilities.

Inexact reasoning using probabilities often makes use of Bayes' Theorem, as for example in the expert system PROSPECTOR (Duda et al, 1976). Bayes' Theorem enables a probability for the truth of a hypothesis to be built up and successively refined from pieces of supporting evidence as they are received.

It is possible to say that one of the purposes of assessment is to determine the truth of a hypothesis: that the student, having been taught the course, has taken in what has been taught. The first piece of evidence towards how much the student has absorbed comes with the answer to the first question. Successive answers provide further evidence. Bayes' Theorem may be stated:

$$P(H:E) = \frac{P(E:H).\ P(H)}{P(E:H).\ P(H) + P(E:notH).(1-\ P(H))}$$

In this context the quantities may be defined as follows:

P(H:E) - the probability that the student has taken in the course, given a certain answer. (The probability of the hypothesis H given the evidence E.) This can be taken as the assessment of the student.

P(E:H) - probability of the student giving this answer (rather than some other) if the student has taken in the course. (The probability of this evidence E if the hypothesis H is true.) This is the crucial answer parameter attached to the answer the student has given.

P(H) - the probability that the student has understood the course, in the absence of the supporting evidence of this answer. (The probability of the hypothesis H in the absence of any evidence.) This will have an initial value, and thereafter will be each preceding P(H:E). The initial value to assign to P(H) is open to some debate, but in the prototype system it was found best, by trial and error, to set it at 0.5.

P(E:notH) - probability of this answer if the student has not understood the course. (The probability of the evidence E if the hypothesis H is not true.) For a five answer MC question, it is reasonable to take P(E:notH) as 0.2.

Mention may be made in passing that there are other ways of dealing with inexact reasoning in ES and AI. Appendix 6.1 (i) shows, in broad outline, an alternative way of dealing with student answers using certainty factors, based on Shortliffe's treatment in MYCIN (Shortliffe, 1976). This approach is considered less sound than one using probabilities, and was not used for WITS, so it will not be referred to again.

6.2.3 The meaning of the answer parameter, P(E:H).

P(E:H), the answer parameter, is the probability of this answer if the student has taken in the course. It is the fraction of students who have taken in the course who would give this answer.

It would be possible to measure P(E:H) for a particular question. It would first be necessary to decide what is meant by 'students who have taken in the course', and the criterion for this might be, in relation to an O-level question, that the student has passed O-level. The question could thus be given to a sample of students who have passed O-level to standardise the question.

P(E:H) resembles in some ways a 'degree of correctness' mark a teacher might give for e.g. an essay. However, it is really quite different. To

standardise a conventional question, it is likely that a sample of all levels of student who have done the course would be used. Here, to be consistent with Bayes' Theorem, a sample of students known to be at the required level would need to be used.

A further difference becomes apparent on looking at the type of question required by this model. If a MC question had several answers close in meaning, one best and others less good, one might tend to think highly of a student who picked the best answer, and assign this answer a high 'degree of correctness'. Using probabilities the best answer would be assigned a relatively low P(E:H), close to the probabilities for the other answers, as less students might be expected to get it right than with more clear cut answers.

6.2.4 Using the hypothesis probability for assessment.

It can be shown that in this application of Bayes' Theorem, answer parameters of greater than 0.2 will increase the value of the hypothesis probability P(H:E) and answer parameters of less than 0.2 will decrease it, as shown in Appendix 6.1 (ii). For an able student who chooses good answers, with answer parameters greater than 0.2, the measured probability of having taken in the course will increase to approach a value of 1, while for a poor student who chooses poor answers, with answer parameters less than 0.2, the value will fall to approach 0. The probability after a certain number of answers can be taken to assess the student, as shown in Figure 6.1 (i).

A quantity which is also useful in analysis of the student's performance is the natural logarithm of the odds, defined as:

$$\ln \frac{P(H:E)}{1-P(H:E)}$$

This increases with a straight line graph. The average increase of ln(odds) per question (the gradient of the graph) will be roughly constant for each student, and will be greater for more able students, as shown in Figure 6.1 (ii). This is a good measure of student ability.

6.3 Use of the Probability Method in WITS

6.3.1. Determination of answer parameters or P(E:H) values.

The answer parameters P(E:H) for each question are real probabilities which could be measured in a standardisation exercise by presenting the questions to a group of students of the required level. An alternative to this is to have the questions and answers designed by an educational expert, who also provides the information to determine the answer parameters.

With an expert method, the allocation of actual probability values cannot be left to the expert, who may have no knowledge of probabilities. In the Electronics prototype system, the process was as follows:

1. The expert designs a question with five partially correct answers.

2. The expert identifies the answers as either the best answer, a near answer, an answer that is factually correct but not a good answer, or an incorrect answer.

3. The probabilities for this distribution of answer types are filled in, having been previously determined. There are in fact 15 possible distributions of a best answer, plus near, factually correct and incorrect answers in different numbers.

Best and near answers were given answer parameters > 0.2, other answers < 0.2. An additional constraint on the parameters, brought out by initial trials, was that they should all be fairly close to the value of 0.2, as otherwise there were large swings above and below the initial probability, and a very large number of questions was needed to obtain a steady assessment.

This behaviour was brought out by initial mathematical trials with the method. The form these initial trials took was as follows. Appendix 6.2 (i) shows a program which reads in a set of answer parameters and from them gives sets of probability values and odds which may be plotted. Initial

sets of random answer parameters within a certain range, e.g. between 0.1 and 0.2, or 0.2 and 0.3, were generated by a statistical package, and the same package was used to plot the probabilities and odds after processing by the program. An example is shown in Appendix 6.2, (ii) to (iv).

6.3.2 Assessing different student variables.

One of the advantages of the probability method described here is that it is theoretically possible with such a model to assess several different student variables within one question. This is because, in applying Bayes' Theorem, the same piece of evidence can be used to measure the probability of several different hypotheses, each as though the others did not exist. Probabilities are often used successfully in this way, for example in medical expert systems, with a much greater number of variables.

There is some agreement that certain abilities, which are important to recognise in a student, are identified in Bloom's Taxonomy (Bloom, 1956), and based on this, the student abilities assessed in WITS were chosen as follows:

a) RECALL: The ability to remember facts.

b) UNDERSTANDING: The ability to relate facts to each other, and know the meaning of rules.

c) APPLICATION: The ability to apply rules and solve problems.

d) INSIGHT: The ability to extrapolate abilities in the subject to new, unseen situations in original ways.

It was necessary to allocate answer parameters, five for each question, in each of these student variables. The graphs of Figure 6.2 show the basis for doing this, and Figure 6.3 gives an example of a question used in the trials of the prototype system. The reasoning behind the construction of the graphs is now described.

The variables chosen are progressive. A question to test recall may test only recall, but a question to test understanding will necessarily, to some extent, test recall as well. Similarly, a question to test application will also test understanding and recall, and a question to test insight will also test application, understanding and recall.

To see how these variables vary with answer type, it may be assumed that as they are progressive, a sample of students with insight is generally more able than a sample possessing recall (to take the extremes of the scale). To standardise a question which assesses both recall and insight, a group of each type of students is needed. If the question is given to both groups, a high proportion of the students with insight can be expected to give the best answer, as they are more able. In the other group, a lower proportion will give the best answer, as they are generally less able. Thus a graph of insight against answer type will show a higher peak for the best answer than will a graph for recall, as shown in Figure 6.2.

It should be emphasised that these graphs, and the probabilities derived from them, are not constructed from empirical data but are in the realm of 'expert knowledge'. They were constructed using commonsense knowledge of the quantities involved, then modified by trial and error in the light of experiments. Numerous tests were carried out as described above to investigate the way the quantities involved behaved, before embarking on the student trials. From the graphs of Figure 6.2 answer parameters were calculated for each of the fifteen possible question types, i.e. combinations of types of answer.

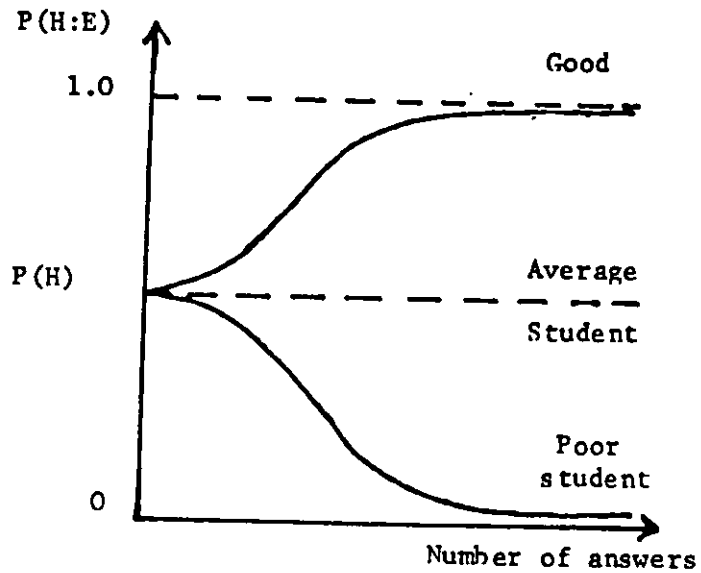
The task of of allocating answer parameters to questions was carried out as described in the last section. In addition the expert had to decide which was the highest of the student variables that each question tested. If a question did not test a variable, the answer parameter for that variable was 0.2.

A listing of the 40 questions used in WITS is given in Appendix 6.3, in the form of a written test given to one group in the evaluation trials (see Chapter 8), and the coding of one of these in the program data is given in Appendix 6.4. The data relating to the answers to these questions can be found in Appendix 6.5. The last five arguments of these facts give the key to the five possible answers A, B, C, D or E, in order, either best (b), near (n), correct (c) or incorrect (i).

Figure 6.1

Graphs for the Probability Assessment Method

(i) Variation of Hypothesis probability P(H:E) with number of answers



(ii) Variation of natural logarithm of odds on the hypothesis with number of answers.

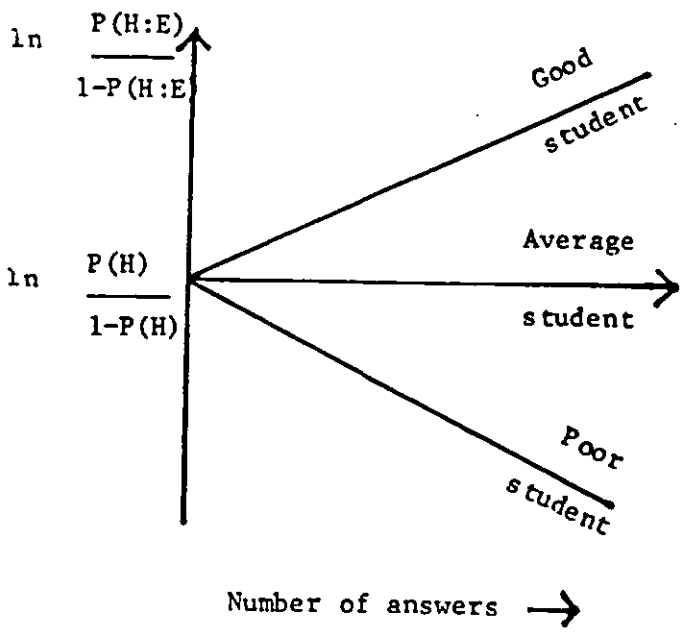


Figure 6.2

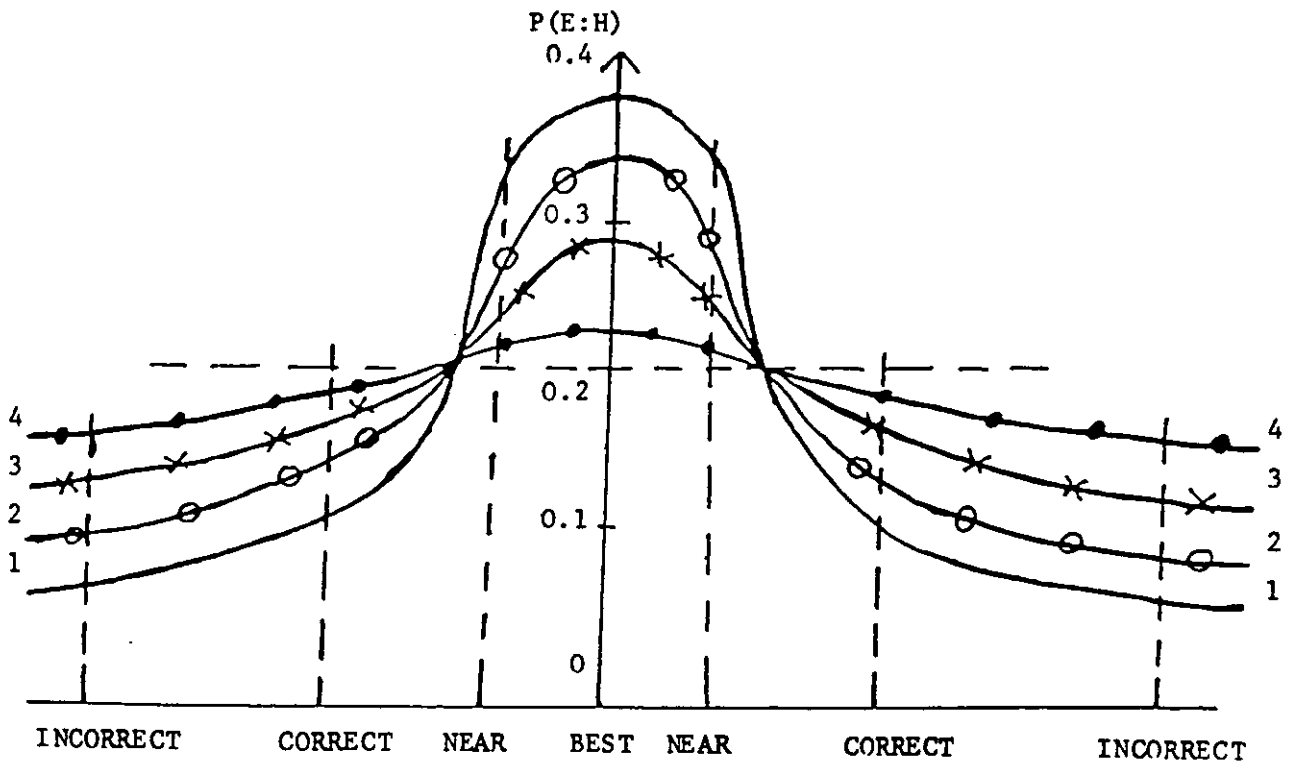
Answer parameters for different abilities

1. Recall

2. Understanding

3. Application

4. Insight



The graphs are used to determine the relative weighting of different types of answer in each student variable. Probabilities (totalling unity) are then calculated for the five types of answer present in different question types.

Figure 6.3

Example question used in the WITS trials

Question:

If you had to modify a cathode ray tube to give a brighter picture, for example to use in bright daylight, which of the following do you think would be best? (Without damaging the cathode ray tube!)

A. Reverse the voltage across the anode and the cathode (INCORRECT)

B. Increase the voltage across the anode and cathode slightly. (CORRECT)

C. Increase the voltage across the anode and cathode considerably. (CORRECT)

D. Increase the current through the cathode filament considerably. (CORRECT)

E. Increase both anode voltage and filament current slightly. (BEST)

Answer parameters allocated for the question type with answer distribution one best, one incorrect and three factually correct:

	RECALL	UNDERSTANDING	APPLICATION	INSIGHT
A	0.18	0.17	0.16	0.15
B	0.19	0.19	0.18	0.16
C	0.19	0.19	0.18	0.16
D	0.19	0.19	0.18	0.16
E	0.24	0.27	0.32	0.36

The presentation of such questions to the student and the calculation of probabilities from the answers is described in the next section.

6.3.3 Handling of Assessment by WITS

There are four stages in building up the student model, or assessing the student:

(i) Timing of questions.
(ii) Selection of questions.
(iii) Presentation of questions.
(iv) Analysis of answers.

(i) Timing of questions

After each showing of a topic, the program checks how many have been seen since the last question session. A decision is made whether to initiate a new question session or not on the basis of this number of topics and the student's ability. This is returned to later.

In INSTRUCT mode questions are simply presented without option. In CHOICE mode the student can refuse questions up to three times, and is then transfered to REVISE mode. In REVISE mode there is no compulsion to do questions, but the student is prompted after a period determined as in the other modes. The rules for the timing of questions are given in Appendix 6.6.

(ii) Selection of questions

The program keeps a count and a record of the topics covered since the last question session, and a search is first made for suitable questions on these recent topics. A list of such questions is made, and they are presented in random order to the student until exhausted.

When a question is no longer available on a recent topic, a list is made up of all suitable questions which the student has still not done, i.e. questions on topics covered in INSTRUCT mode, and covered or known in

CHOICE mode. This list is of all questions not yet done in REVISE mode. These questions are presented in random order to the student until either they are exhausted or the required number for the question session is reached.

If no question is found, because there are none suitable or they have all been done, the student, having been told there is to be a question session, is now told, with an apology, that there are none available. The rules for the selection of questions are given in Appendix 6.7.

(iii) Presentation of questions

An example of how a question looks to the student is shown in Figure 6.4. It will be noted that with each question there is an explanation of the answer. The sequence is as follows:

1. The question is presented on screen to the student.

2. The student presses A, B, C, D, or E to answer.

3. This first answer is recorded in the student model as the answer given for that question.

4. If the answer is a best or near answer for the question the student is congratulated and the explanation appears. After reading it the student can continue to another question.

5. If the first answer is a 'correct' or incorrect answer, the hint is shown and the student is given another try.

6. If this second answer is a best or near answer, the procedure of 4 is followed.

7. If the second answer is a correct or incorrect answer, the student is informed and the explanation appears. The student can then continue.

The questions, via the hint and explanation process, thus serve a teaching as well as a testing function. The student's second answer is only for his or her own self-testing use, and is not recorded. The rules for the presentation of questions are given in Appendix 6.8.

(iv) Analysis of answers

The student's first answer is recorded and used to update the probabilities that the student has an attainment in the four student variables of recall, understanding, application and insight. This is done after each question using Bayes Theorem, as described earlier in the chapter.

The PROLOG rule which updates the hypothesis probabilities for all the variables after the student has returned an answer is given in Appendix 6.9. This draws on facts which summarise the data for the questions, shown in Appendix 6.5.

The third argument here is the question identification number: for example, 192 indicates question 2 on Unit 19. The seventh argument is the highest student variable the question tests, the eighth is the best answer and the last five arguments are the answer codes for the five possible answers in the order A, B, C, D or E, i.e. the best (b), near (n), correct (c) or incorrect (i).

The sixth argument in these facts is the question type, from 1 to 15. The fifteen question types, representing the fifteen possible combinations of four types of answer are summarised in the facts of Appendix 6.10. These contain the probabilities to be attached to the best answer for those of the four student variables which the question tests, and also the probabilities to be attached to the near, correct and incorrect answers. These are derived from the graphs of Figure 6.2.

The rule which actually calculates the updated probability, using Bayes Theorem is the rule 'calc' in Appendix 6.11. The new probabilities for the student variables, having been calculated, are now recorded.

The complete process of assessment of the student by WITS, leading to updating of the student model, is shown in Figure 6.5.

6.3.4 Representation of the Student Model

The student model exists in two forms, a temporary or 'soft' version held in the computer's memory at run time, and a permanent or 'hard' version held in a student file saved onto hard or floppy disc. There are three types of data recorded in the student model:

(a) Data recording the part of the course covered, and the position reached.

(b) Data relating to student ability.

(c) Data relating to the student's use of the system.

The rule which initialises the student model during the first session is given in Appendix 6.14. This 'asserts' PROLOG facts such as the starting values of the student variables recall, understanding, application and insight, and the number of times commands such as 'search' and 'profile' have been used. These facts are subsequently updated as required during the course.

A type of data which does not appear in this initialising rule, because it is not updated but simply asserted once, is that recording the part of the course covered. This records such things as topics seen and questions done. This data appears in the student file, which is the true representation of the student model, containing everything that needs to be known about the student from one session to the next. This is in the form of a large, single frame, with standard types of data that can be slotted in.

A typical student file is shown in Appendix 6.15. The rules which handle the saving and reading back of the student file are given in Appendix 6.16.

Much of this information is available to the student in a different form in a student progress profile. The screens of a typical student profile are shown in Figure 6.6. The first screen gives outline data of the course covered, and the second one the probabilities of attainment in the student variables measured, with a histogram for comparison.

The third screen gives the student abilities in the four variables measured, as a percentage, on a scale on which an ideal student who gave the best answer each time would score 100% and a worst student who gave the worst possible answer each time would score 0%. These abilities are calculated using ln(odds) quantities, and the rules which do this are given in Appendix 6.17. To calculate the abilities in this way a record has to be kept of probabilities which would be attained by an ideal student and also by a worst student, as well as the real student. This explains the extra items in Appendix 6.17. The complex rules presenting the histograms on the screen are not included, as they run to several pages of code.

A further source of information for the student is a printed report. An example of such a report is given in Appendix 6.18. It contains the information of a profile with additional information on the way the student has used the system. Also in the report are comments on the student's performance. The convenient way in which such comments can be derived from different conditions in PROLOG is shown in Appendix 6.19. The lengthy rules which write the report are not included here.

6.4 Adaptation to the student in WITS

The general philosophy behind WITS was to make the system flexible in such a way that the student could adapt it to his or her own needs. This was achieved in several ways:

1. The student can choose a learning environment, and change from one to another easily.

2. The student can design his or her own route through the course in CHOICE mode.

3. By the use of modes and facilities the student can either be led through the course, or can choose what to see at any time at will.

4. The student can use the facilities provided in different proportions, spending a large part of the time using the search facility, keyword searching, doing questions or studying his or her profile as required.

Figure 6.4

Screens showing a typical question

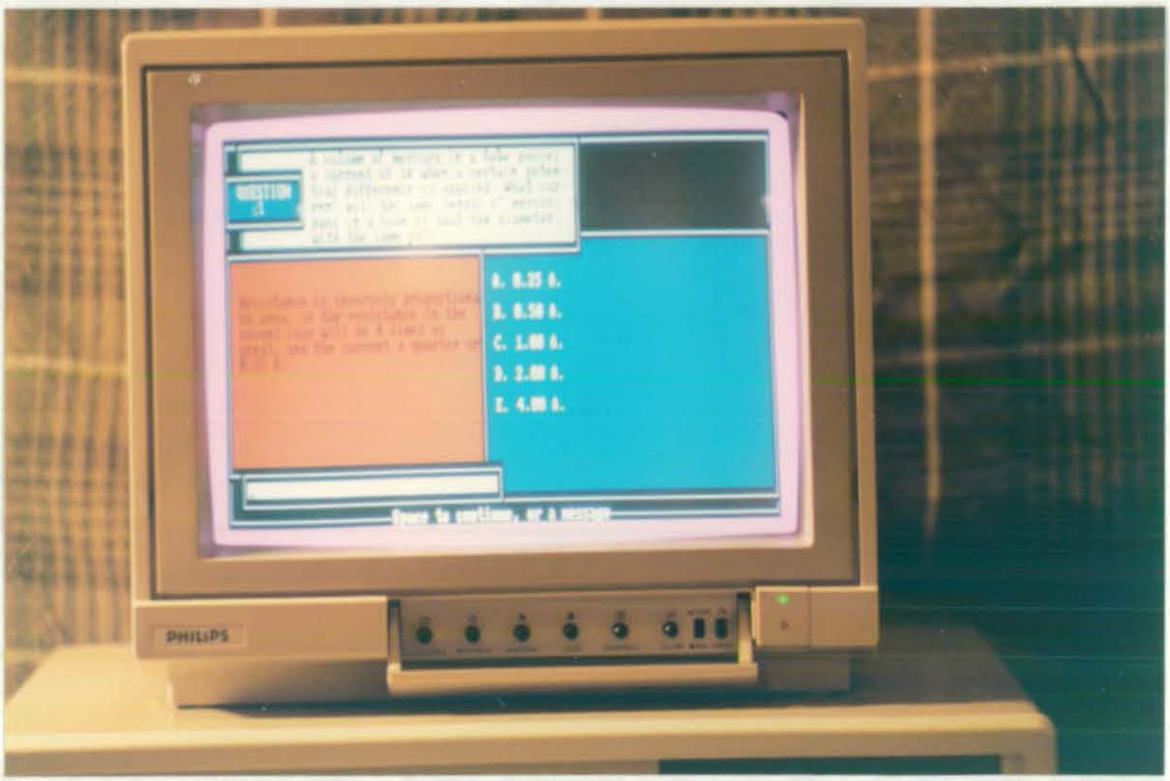


Figure 6.5

The process of assessment and updating of the student model in WITS

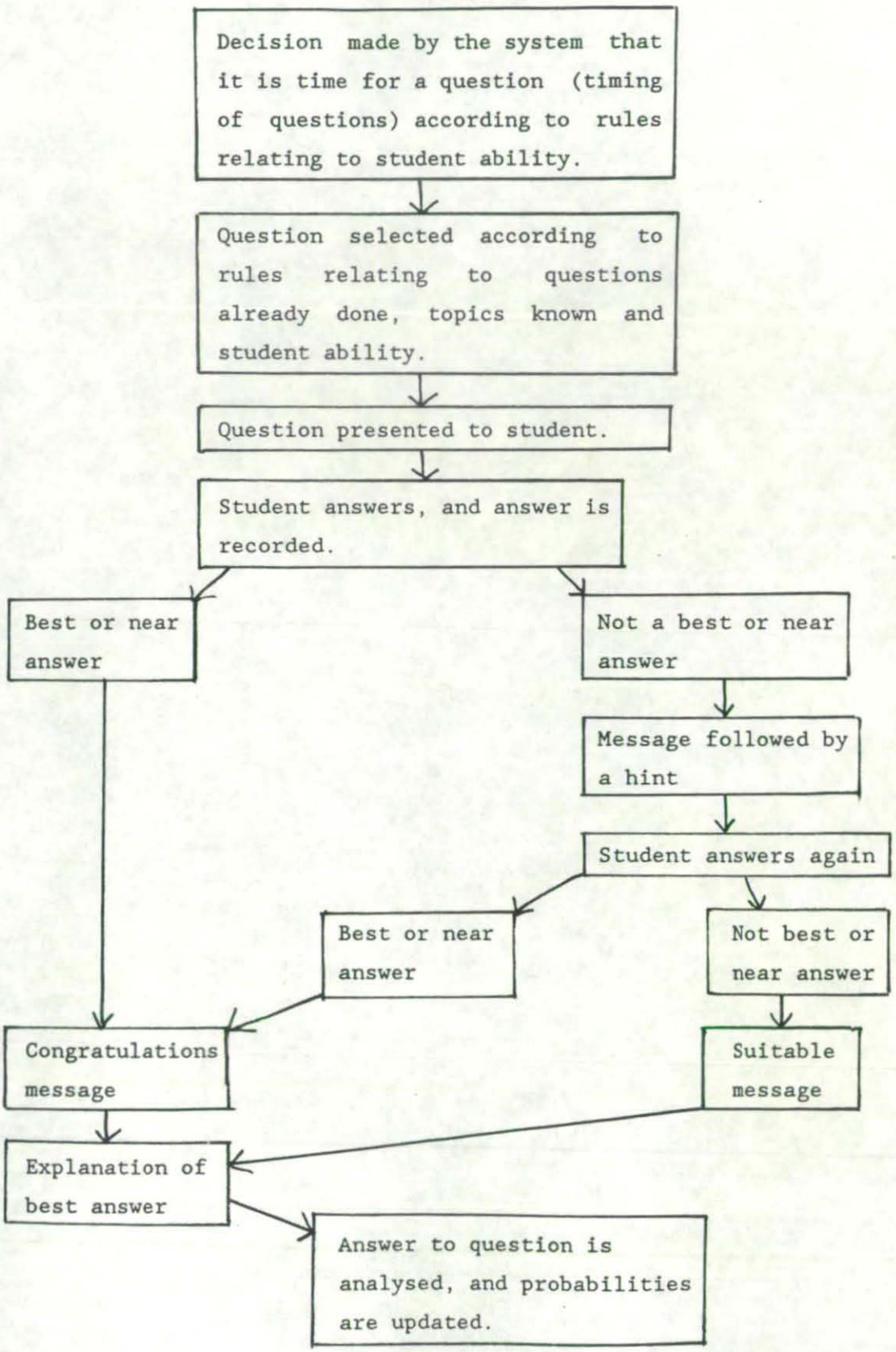


Figure 6.6

Screens showing a typical student profile

However, there are several ways in which the system adapts to the student in ways outside the student's direct control.

1. From the student's performance in questions he or she is placed in a category from A to E. This is made known to the student via the progress profile and could be a powerful spur to some students. The rules and facts relating to student category are shown in Appendix 6.12.

2. Depending on the student category, the number and timing of questions are varied. Able students are given more questions less frequently, less able ones less questions more frequently. This is because less able students are less likely to retain information for as long, and because it is usual to assume that they have less concentration and need a change more often. The rules which control this 'pacing' of the student are shown in Appendix 6.13.

3. In the question facts, shown in Appendix 6.5, the second argument represents the hardness of the question on a scale 1-6. For example, questions on some topics may be considered unsuitable for certain student categories, either because they are too easy or too hard. Certain types of question are hard, viz. the higher numbered question types in Appendix 6.10, and might be withheld from students of lower categories. This facility, although catered for in the data format, was not developed for use in the trials of WITS through time constraints.

4. In the module, unit and topic facts shown in Appendix 6.8, the second, third and fourth arguments respectively represent the hardness of the subject matter. This allows the material of the course to be modified for different abilities or categories of student. For example, a number of topics might be excluded for less able students. If the course were being used by a certain type of student, the system could be 'set' to exclude topics that were not on the syllabus. This facility, although catered for in the data format, was not developed for use in the trials of WITS through time constraints.

There are other ways in which the system might be made to adapt to the student. One student in the trials would have liked some longer videodisc sequences, and a system of grouping topics together into longer sequences for more able students would be useful.

6.5 Summary

The student model in WITS was of a type which a teacher might aim to build up during the teaching of a course. In order to build up this model, a method of assessment using probabilities was developed, which can be summarised as follows:

1. A student variable is expressed as a hypothesis about the student.

2. The probability that this hypothesis is true is measured using the student's answers to questions.

3. Answers are coded with answer parameters, which are their probabilities of being given if the hypothesis is true.

4. The model requires special questions and answers to be compiled and coded by an 'expert' according to strict criteria.

5. Bayes' Theorem is used to calculate the probability of the truth of the hypothesis about the student.

6. The value of probability after a certain number of questions can be used to assess student performance, and a measure of student ability is the natural logarithm of the odds on the truth of the hypothesis.

7. Different student variables can be measured simultaneously.

Within the limitations of this method, certain problems with conventional testing are reduced, and there are a number of advantages, as follows:

1. The method is more efficient, giving a detailed assessment of the student with less questions and answers.

2. It is possible to say with greater clarity what is being tested. The probability method represents defined variables by measurable quantities. This is represented by the comparison in Figure 6.7.

3. Different student abilities can be tested, combining the testing of several in the same question. The method lends itself to compiling profiles of students.

4. There are possibilities of extending the method to achieve more accurate assessment of more student variables, and of achieving greater objectivity by accurate standardisation of questions and answers.

One limitation of the probability method of assessment is that an assumption is implicitly made, namely that items of evidence (answers) are independent. If this is not the case, assessments by the method may be only approximate. (This was pointed out by Dr. Hinde at Loughborough University and Professor Teather at Leicester Polytechnic.) However, as most teachers acknowledge, assessment of students is an imprecise art by any method. A number of studies has shown that teachers' subjective assessments of students vary considerably (some are listed by Mehrens and Lehmann, 1969). The 'overlap' of questions is an acknowledged problem with most kinds of student testing.

Similarly, the independence of evidence is an acknowledged problem with expert systems, which has been written about at length (see Johnson, 1986, and Kyburg, 1987). Although it gives rise to concern, this does not prevent expert systems yielding useful results. Forsyth comments: "There is almost bound to be some degree of association in any real set of indicators. The practical problem is to minimise it." (Forsyth, 1984). Providing questions are designed to test different areas of knowledge, non-independence of evidence should not detract from the usefulness of this method too greatly.

Figure 6.7

Probability and Conventional Assessment Compared

	CONVENTIONAL TESTING	PROBABILITY METHOD
Answer parameter	Mark for an answer, right or wrong (cf.digital 0 or 1)	Probability of the answer being given, from 0 to 1 (analog)
Assessment parameter	Percentage of marks	Probability that an ability is present
Student ability parameter	Percentage of marks	Gradient of ln(odds) graph
What is measured	Unclear	A specified hypothesis (or several) about the student

Chapter 7

Evaluation of WITS

7.1 How the Evaluation Was Carried Out

7.1.1 Criteria for Evaluation

The WITS system was built as one solution to the problems of designing intelligence into a whole-course CAL system. In that a working system was built, which overcame the programming problems and functioned as planned, WITS was successful.

There has been interest recently in the question of how to evaluate ITSs. In a book that has appeared since the evaluation of WITS was planned and carried out, Littman and Soloway (1988) distinguish between formative and summative evaluation, commenting that as few ITSs can be called 'finished', evaluation is of necessity formative. They add that there is no standard set of evaluation methods for evaluating ITSs; "the field of ITSs is too young." Littman and Soloway distinguish between external and internal evaluation; the first is whether the ITS has some positive educational effects on the student, and the second is whether the ITS does certain educationally valid things it set out to do.

A system intended to teach students clearly needs to be evaluated with students. It was decided to examine the following criteria in evaluating WITS with students:

1. Did it teach a whole course effectively?
2. Did it assess students accurately?
3. How did the students and others react to WITS, particularly to its 'intelligent features'?

The way WITS performed in relation to these criteria will be examined after a discussion of the methods used and the problems involved. It will be seen that the third of these criteria is not capable of evaluation by quantitative measurements, being subjective and qualitative. It was worthy of inclusion as a means of encouraging open-ended findings, in the spirit of Action Research, described in the next section.

7.1.2 Action Research

In some ways the trials of WITS that were carried out resembled the software evaluations carried out regularly on new programs by software manufacturers and computer journals, but they could be more closely related to educational research.

Cohen and Manion (1980) differentiate between normative and interpretative approaches to educational research, as follows:

"The traditionalist (normative researcher) approaches social reality with preconceptions, manifest in his briefcase crammed with questionnaires, attitude-scales and structured interview schedules. The interpretive researcher, by contrast, will start with the social world as it is and, almost in the spirit of an eavesdropper, will tune in to it on its terms with unstructured interviews, participant observation, natural conversation and the like ... we see both the normative and interpretive approaches to the study of social reality as being equally valid".

It seems that data drawn from either objective or subjective data, or both, are acceptable in educational research. In evaluating WITS, it was possible to collect objective data in the form of keypress counts which recorded student use of the system, and also subjective data in the form of students' answers to questions. The probability method of assessment consisted of questions subjectively designed, but objectively administered and analysed. WITS thus combined both a normative and interpretive approach.

Among the methods frequently used by researchers in education, the approach to the WITS trials has most in common with action research. (For a description, see Stenhouse, 1980; Nixon, 1981; or Somekh,1986.) This is a method which now has a fairly long pedigree, having been initiated in the USA in the late 1940s.

"The conventional definition of action research identifies it as a small-scale intervention in the functioning of the real world and the close examination of the effects of such intervention. It is essentially a practical method and directly relevant to the situation giving rise to it in that the findings may be implemented in the near future to effect lasting improvement ... action research provides an orderly framework for problem-solving and for new developments, and to this extent is superior to the impressionistic approach that so often characterises developments in education; it is also empirical in that it relies on actual observations and behavioural data; and it is flexible and adaptive, allowing for changes during trial periods and sacrificing control for responsiveness and immediate experimentation and innovation." (Cohen and Manion, 1980.)

In evaluating this system the approach used can be seen as that of a piece of action research. Giving the system to students to use represented a small-scale intervention in their education; the effects were closely examined, using actual observations and behavioural data; the approach was flexible and adaptive, involving changes during the trials; and the aim was the better design of such systems to effect lasting improvement.

7.1.3 Evaluation of the first two criteria

An ideal way of evaluating WITS in respect of the first two criteria listed above would have been to take two groups of students, one taught normally in school as a control group, and the other learning from WITS as an experimental group. The two groups would have been carefully matched and would have contained a substantial number in each.

A central feature of WITS, and probably the one with most claim to 'intelligence', was the probability assessment method. To evaluate this, an ideal way would have been to give the two matched groups a probability method test, and also to give both groups a conventional test.

Perhaps the most satisfactory result to come from this sort of evaluation would have been a close correlation in all the tests done by the students. This would indicate that teaching by WITS was comparable to teaching by a human teacher, and also that assessment by the probability method gave similar results to conventional testing.

Comparing a computer-based course with a human teacher's course is not comparing like with like. Also, the probability model described here is so different from conventional testing, for example in attempting to measure different student abilities, that it is hard to compare the two.

In the case of significantly different results it would be difficult to interpret them. If test results were higher for the experimental group it could be argued that the control group had nevertheless gained educational advantages from the human teacher which had not been assessed. Conversely, if results were higher for the control group, it could be argued that considerations such as individual teaching and objectivity made teaching by a system such as WITS nevertheless worthwhile, at least for certain students in certain circumstances. If results from the two assessment methods were different, it would be hard to decide which had in fact given the more reliable results.

For these reasons it was considered best not to attempt a rigorous statistical evaluation, even had one been possible. Subjective methods were likely to yield as much, perhaps more, information in a smaller informal evaluation. In fact a statistical evaluation of the type outlined above would not have been possible, because of the problems involved, which are enumerated in the next section.

7.2 Constraints on the Evaluation

7.2.1 Equipment constraints

Although the hardware performed well, and software problems were few, only one videodisc player and one videodisc were available, and therefore only one set of equipment.

Students could only use this one at a time. Also because of the expense of the equipment and consequent security considerations, access to the equipment by the students needed to be restricted. It was fortunate that a small room was available in Leicester Polytechnic in which to site the equipment.

7.2.2 School constraints

As the WITS system was based on the expertise of the school teacher, school students were required to evaluate it. The security considerations mentioned above meant that the equipment had to be situated in Leicester Polytechnic, and it was fortunate that a school which was nearby was willing to cooperate.

Evaluation of a whole course system is not like evaluation of a computer program dealing with one educational topic, which might be carried out on a one-afternoon-per-student basis. Because a system like WITS aims to teach a substantial part of a course, it makes substantial demands on the students' time and energy, and takes weeks rather than hours. This increases the problems of evaluation considerably.

The problems with respect to students were:

(a) The students needed a substantial amount of time on the equipment.

(b) This had to be fitted into their private study periods, which did not coincide with Polytechnic teaching periods.

(c) They had to walk to the Polytechnic and wait to be let into the project room.

(d) They were left to work on the system alone, and if problems arose it was not always possible to obtain help immediately.

Although teachers in the school which supplied the students were interested in the project and were extremely helpful, they understandably had some

reservations. They were unwilling for students to be used as 'guinea pigs' in any way which might detract from their learning. This consideration alone meant that the notion of a group of students learning from WITS as their only source of instruction was not possible. Any use of WITS had to be in addition to, rather than instead of, their normal course.

7.2.3 Videodisc constraints

Although the WITS 'shell' was designed to teach a whole course independently of a teacher, it used a commercially available videodisc which, although the best available at the time, was not ideal in several ways.

(a) It was designed as a revision course rather than a first-time learning course.

(b) It was designed for City & Guilds Electronics, whereas the school students were doing 'A' Level courses.

(c) It was partially practical in approach, requiring equipment such as oscilloscopes and multimeters to be available by the side of the computer system. This was not possible, and the students had to refer to equipment when they returned to their school. Surprisingly, they did not appear to have particular difficulty with the practical aspects of the course.

(d) Although concerned largely with practical topics, the SSE disc did not make use of videodisc simulations involving measurements from the screen, usually considered a strong feature of interactive video in science and technology.

(e) There were many mistakes on the videodisc, some in the electronics (which the students pointed out). Where frames containing errors had been detected before mastering, they had been corrected with new frames but had been placed on the disc non-consecutively, seemingly at random. The reject frames had been left on the disc.

(f) The documentation explaining where things were on the disc was far from adequate.

(g) As mentioned in Chapter 5, the grouping of topics on the disc was unsatisfactory and had to be reorganised.

7.2.4 Other constraints

Some of the most important problems in evaluating WITS are difficult to classify. One has been explained above, namely that of interpreting any results obtained because of not comparing like with like.

The attempt made in the prototype WITS to measure four student variables simultaneously, recall, understanding, application and insight appears to have been without precedent.

These abilities in a student are not usually measured separately, but are looked upon as contributing to the overall 'blanket' grade given to the student. If such an attempt were made, the abilities would probably be measured by separate tests (Bloom, 1956, discusses how to do this) or by separate questions within one test. Another possibility would be to associate certain questions with certain abilities and use cluster analysis. No work has been found which attempts to separate these abilities in individual students or to apply simultaneous assessment techniques of this kind. To evaluate the assessment method, therefore, by comparing its assessment with one of a more conventional form to measure the same variables, would involve devising another new method of assessment, and there would be doubt as to which method was more reliable.

The end result of these problems was to limit the number of students and to limit the form of evaluation to a relatively subjective and qualitative form.

7.3 Description of the Trials

7.3.1 The trials carried out

Trials were carried out with the help of students and staff at a sixth form college near to Leicester Polytechnic. Several modifications were carried out to WITS during the trials. A 'bug' was discovered which caused the program to stop while counting the topics completed (using a recursive method). Another 'bug' caused errors in timing the sessions. These were both discovered in the first week and corrected. Corrections were also made to the data where relevant. Routines to handle chains of videodisc sequences and to print out reports were built in during the early part of the trials. The profile of progress was modified for greater clarity.

Initially the aim was to test WITS as an ITS, and to evaluate the probability assessment method as part of it. It became apparent during the trials that it was quite possible to separate the probability assessment method from the ITS, and to evaluate it separately. An arrangement was made to give an additional group of students the questions from WITS as a written test. A modified version of WITS was prepared to print out the questions and handle the results for this group.

In all, three groups of students took part in the trials, which are now described.

7.3.2 Group A (mature students)

This group consisted of three mature students, all of whom knew the subject matter of the course. They were, respectively, a teacher of Electronics, an Electronics engineer, and a computer engineer who was a former Electronics degree student. They spent a total of just under 10 hours on the WITS system. (See Table 7.1.)

These students were asked to try out the system and give their opinions, and were expected to perform highly when assessed by the system. They filled in a questionnaire, Q1, which collected their opinions of WITS and their self-assessments of the abilities measured in the assessment. (See Appendix 7.1.)

7.3.3 Group B (school students)

This group consisted of students from the school who were in their first year of an 'A' Level Technology course. They were just commencing the subject of Electronics, and worked through the WITS course in addition to their scheduled lessons. This involved coming to the Polytechnic for one of their private study periods each week, lasting 1 hour 10 minutes (less a few minutes spent in walking). This is shown in Table 7.2.

These students completed questionnaire Q1, and in addition their teacher completed a questionnaire Q2 with assessments of the measured student abilities in respect of each student. (See Appendix 7.2.)

7.3.4 Group C (written test students)

This group consisted of school students who were in their first year of an 'A' Level Physics course. They did not work through the WITS course, but did the questions from WITS as a written test.

The question paper is shown in Appendix 6.3, and a sample answer sheet is shown in Appendix 7.3. The answers from each student's answer sheet were fed through a special version of the WITS program which provided a report on each student similar to that produced on students who did the WITS course. An example is given in Appendix 7.4, which can be compared with Appendix 6.18. The calculation of probabilities and student abilities was as in the WITS course, except that all students had done all the questions, and in the same order, which was not the case for the course.

These students completed a questionnaire Q3 similar to questionnaire Q1, but without the questions on the WITS system (see Appendix 7.5.) Their teacher also completed questionnaire Q2.

7.4 Results: The Effectiveness of Teaching by WITS.

As explained above, it was not possible to set up a trial to test whether WITS was able to teach Electronics as a stand-alone whole course. However, there will remain a possibility of carrying out an evaluation of this kind

if a suitable videodisc becomes available, or if it becomes possible to master one, and if suitable students and circumstances can be found, perhaps in adult education.

Although WITS was not evaluated as a stand-alone course, it covered the whole of the students' introductory Electronics course (with some discrepancies) and in this sense was a whole course system.

The students were receiving normal lessons at the same time, and it was impossible to assess what they learned from WITS and what from their teacher, so the effectiveness of didactic, first-time teaching in WITS could not be evaluated in an objective way. However, it was possible to assess the effectiveness of the system from students' behaviour and subjective comments, and this is discussed in a later section.

In view of the difficulties in evaluating the effectiveness of teaching in WITS, the main effort in the trials carried out was towards evaluating the 'intelligent features' of WITS and the probability assessment method of the system.

7.5 Results: The Assessment of Students by WITS

7.5.1 WITS, teacher and self-assessments

WITS assessed students separately in the four areas of ability, recall, understanding, application and insight. Initially, a probability of having achieved the required standard in each of these areas was calculated, then in each area ln(odds) was calculated and taken as representing student ability in that area. Each ability was then converted to a percentage on a scale on which an ideal student, who gave the best answer to every question, would score 100%, and the worst possible student, who gave the worst possible answer to every question, would score 0%. The student thus received a percentage in each of the four areas.

WITS also gave the student an overall category determined from the average of these four percentages. The student assessment was available to the

student at all times during the course in the form of an on-screen profile of progress (see Figure 6.6) and as a printed report at the end of the course (see Appendix 6.18).

As already pointed out, such detail, theoretically possible within the probability method, is not usually attempted in student assessment. Other ways of assessing such variables are not readily available. Thus in an attempt to evaluate the accuracy of the WITS assessments, they were compared with estimated assessments of the same quantities by the students' teacher, and also with self-assessments by the students themselves.

It was considered rather too demanding to ask the students and the teacher to estimate such abilities as those being measured as a percentage, so they were asked to estimate them as either 'very good', 'good', 'average', 'below average' or 'poor' (see Appendices 7.1, 7.2 and 7.5.) Similarly, the teacher was asked to estimate general ability as a grade A to E, corresponding roughly to percentages used in WITS and shown in Appendix 7.2. This raw data is shown in Tables 7.3 to 7.5 for the three groups of students.

For comparison, the student self-assessments were converted to a five point scale on which 'poor' was 1 and 'very good' was 5, and the teacher's assessments were converted to a five point scale on which E was 1 and A was 5. The percentages calculated by WITS for the four student abilities were used as they stood.

7.5.2 Group A (mature students)

Table 7.6 compares WITS assessments of the four abilities for Group A with students' self-assessments. Students were not asked to estimate their own general ability. It is clear that there is close agreement between the WITS and self-assessments.

These quantities cannot be assumed to be normally distributed, so a non-parametric method of testing for correlation is required. The Spearman rank order correlation coefficient for all these assessments is 0.91, which is significant to the 1% level.

TABLE 7.1

Sessions and times spent by Group A (mature WITS students)

STUDENTS	NO. OF SESSIONS	TIME SPENT	
		hours	mins
A1	8	4	50
A2	2	2	37
A3	4	2	27
Total Time		9	51

TABLE 7.2

Sessions and times spent by Group B (school students)

STUDENTS	NO. OF SESSIONS	TIME SPENT	
		hours	mins
B1	5	2	18
B2	13	7	12
B3	7	5	49
B4	12	8	13
B5	8	4	24
B6	5	3	49
Total Time		31	45

TABLE 7.3

Probability assessment data on students in Group A

STUDENT	ASSESSMENT							
	RECALL		UNDERSTANDING		APPLICATION		INSIGHT	
	WITS	SELF	WITS	SELF	WITS	SELF	WITS	SELF
A1	69.3	VG	70.5	G	77.0	VG	87.5	VG
A2	66.2	G	67.3	G	63.1	A	57.6	A
A3	68.7	G	65.5	G	60.6	A	50.5	A

WITS assessments are percentages

VG	Very good
G	Good
A	Average
BA	Below average
P	Poor

TABLE 7.4

Probability assessment data on students in Group B

STUDENT	ASSESSMENT											
	RECALL			UNDERSTANDING			APPLICATION			INSIGHT		
	W	S	T	W	S	T	W	S	T	W	S	T
B1	36.4	A	G	37.7	A	G	23.6	BA	A	23.9	G	A
B2	50.5	G	G	50.1	A	A	42.5	A	A	14.4	A	A
B3	50.6	BA	A	51.2	P	P	44.7	A	A	40.0	G	A
B4	53.1	G	VG	51.5	A	VG	48.8	G	G	30.0	A	G
B5	50.9	G	A	52.0	A	A	44.7	A	BA	51.1	A	A
B6	51.2	A	G	50.8	VG	G	54.4	A	A	48.5	A	A

Column W : WITS assessment as a percentage
Column S : Student's self assessment
Column T : Teacher's assessment

VG Very Good (converted to 5)
G Good (4)
A Average (3)
BA Below Average (2)
P Poor (1)

TABLE 7.5

Probability assessment data on students in Group C.

STUDENT	ASSESSMENT											
	RECALL			UNDERSTANDING			APPLICATION			INSIGHT		
	W	S	T	W	S	T	W	S	T	W	S	T
C1	46.1	A	BA	44.7	BA	BA	43.3	BA	BA	46.6	A	BA
C2	42.2	G	A	41.6	BA	A	16.0	A	A	19.1	A	A
C3	54.2	G	A	55.0	BA	BA	44.7	A	A	63.3	A	BA
C4	77.9	A	A	76.3	VG	A	75.7	G	A	80.2	G	A
C5	66.7	A	A	68.9	A	BA	61.5	G	BA	65.2	G	BA
C6	54.1	A	G	55.0	G	A	58.2	A	A	73.4	G	A
C7	42.2	A	BA	38.6	P	BA	30.8	BA	BA	34.6	BA	BA
C8	49.0	BA	BA	49.0	A	BA	39.3	BA	BA	52.9	G	BA
C9	64.3	-	G	63.0	-	G	62.1	-	G	70.0	-	G
C10	60.5	-	A	59.8	-	BA	62.4	-	A	77.8	-	BA

Column W : WITS assessment as a percentage
Column S : Student's self assessment
Column T : Teacher's assessment

VG Very Good (converted to 5)
G Good (4)
A Average (3)
BA Below Average (2)
P Poor (1)

TABLE 7.6

WITS and student self-assessments in different abilities for Group A.

Student	Assessments (corrected to numerical 5 point scale)	
	WITS	Self-assessment
A1 (recall)	69.3	5
A2 (recall)	66.2	4
A3 (recall)	68.7	4
A1 (undstg)	70.5	4
A2 (undstg)	67.3	4
A3 (undstg)	65.5	4
A1 (appln)	77.0	5
A2 (appln)	63.1	3
A3 (appln)	60.6	3
A1 (insight)	87.5	5
A2 (insight)	57.6	3
A3 (insight)	50.5	3
Spearman rank order correlation (N=12)	0.91	

TABLE 7.7(a)

WITS, self-assessments and teacher assessments in different abilities for Group B

STUDENT	ASSESSMENT (corrected to 5-point numerical scale)											
	RECALL			UNDERSTANDING			APPLICATION			INSIGHT		
	W	S	T	W	S	T	W	S	T	W	S	T
B1	36.4	3	4	37.7	3	4	23.6	2	3	23.9	4	3
B2	50.5	4	4	50.1	3	3	42.5	3	3	14.4	3	3
B4	53.1	4	5	51.5	3	5	48.8	4	4	30.0	3	4
B5	50.9	4	3	52.0	3	3	44.7	3	2	51.1	3	3
B6	51.2	3	4	50.8	5	4	54.4	3	3	48.5	3	3
SROCC (N=5)	W-S: 0.37 W-T: 0.50 S-T: 0.23			0.25 0.00 0.43			0.70 0.30 0.60			0.0 0.25 0.37		
SROCC (N=20)	W-S: 0.35 W-T: 0.30 S-T: 0.49											

W WITS assessment
S Student's self-assessment
T Teacher's assessment
SROCC Spearman rank order correlation coefficient

TABLE 7.7 (b)

WITS and teacher overall assessments for Group B

Student	Assessments	
	WITS	Teacher
B1	2	2
B2	2	2
B4	3	4
B5	3	1
B6	3	4
Spearman Rank Order Correlation Coefficient (N=5)	0.43	

TABLE 7.8(a)

WITS, self-assessments and teacher assessments in different abilities for Group C

STUDENT	ASSESSMENT (corrected to 5-point numerical scale											
	RECALL			UNDERSTANDING			APPLICATION			INSIGHT		
	W	S	T	W	S	T	W	S	T	W	S	T
C1	46.1	3	2	44.7	2	2	43.3	2	2	46.6	3	2
C2	42.2	4	3	41.6	2	3	16.0	3	3	19.1	3	3
C3	54.2	4	3	55.0	2	2	44.7	3	3	63.3	3	2
C4	77.9	3	3	76.3	5	3	75.7	4	3	80.2	4	3
C5	66.7	3	3	68.9	3	2	61.5	4	2	65.2	4	2
C6	54.1	3	4	55.0	4	3	58.2	3	3	73.4	4	3
C7	42.2	3	2	38.6	1	2	30.8	2	2	34.6	2	2
C8	49.0	2	2	49.0	3	2	39.3	2	2	52.9	4	2
C9	64.3		4	63.0		4	62.1		4	70.0		4
C10	60.5		3	59.8		2	62.4		3	77.8		2
SROCC N=8 or 10	W-S(8): 0.04 W-T(10): 0.55 S-T(8): 0.55			0.83 0.37 0.60			0.73 0.51 0.60			0.64 0.37 0.46		
SROCC N=36 or 40	W-S(32) 0.62 W-T(40) 0.37 S-T(32) 0.53											

W WITS assessment
S Student's self-assessment
T Teacher's assessment
SROCC Spearman rank order correlation coefficient

TABLE 7.8(b)

WITS and teacher overall assessments for Group C

Student	Assessments	
	WITS	Teacher
C1	3	1
C2	1	2
C3	3	2
C4	5	3
C5	4	2
C6	4	4
C7	2	2
C8	3	1
C9	4	5
C10	4	2
Spearman Rank Order Correlation Coefficient (N=10)	0.62	

The students of Group A had access to their printed reports while filling in their questionnaires, and so may, subconsciously or otherwise, have agreed with the WITS assessments when making their self-assessments. This is likely because these students, who knew the subject, tended to score highly. Perhaps the high degree of agreement indicates a combination of close correlation and a high degree of approval of the WITS assessment.

7.5.3 School students (Group B)

Table 7.7(a) compares the WITS assessments of the four abilities for Group B with students' self-assessments and with teacher assessments, and Table 7.7(b) compares overall assessments by WITS with those by the teacher. In this group, student B3 was not included, as the 'bug' which caused the program to stop in the early stages (mentioned in Section 7.3.1) may also have caused some answers to be lost, and made this student's data suspect.

The first table shows that there were positive or zero correlations between all the assessments. There is no obvious pattern to the correlations for different abilities assessed, though the correlations are somewhat higher for recall and application. Perhaps the students and teachers understood these as more distinct abilities than the others.

Overall, using the data in this table, there was a somewhat higher correlation in this group between the teacher's assessments and the students' self-assessments than between WITS and either of the other assessments. The teacher and the students in this group seem to have agreed with each other on the students' separate abilities rather better than either agreed with WITS.

The second table shows a substantial positive correlation between the WITS overall assessments and the teacher's assessments. The implication here is that the reliability of the WITS assessments was comparable to the reliability of the other methods of assessment, both for the four separate abilities and overall.

7.5.4 Group C (written test students)

Table 8.8(a) compares the WITS assessments of the four abilities for Group C with students' self-assessments and with teacher assessments, and Table 7.8(b) compares overall assessments by WITS with those by the teacher.

The first table shows that there were positive correlations between all combinations of the three methods of assessment for all the four abilities assessed. In some cases the correlation was high.

Overall, using the data in this table, there was a substantial positive correlation in this group between the WITS assessments and the students' self-assessments. There was also a substantial though lower positive correlation between the teacher's assessments and the students' self-assessments. There was a positive though lower correlation between the WITS assessments and the teacher's assessments. In this group WITS and the students agreed with each other rather better then either agreed with the teacher. In fact it seems that in this group the assessments all agreed with each other rather well, probably as well as could be expected.

The second table shows that there was a substantial positive correlation between the WITS overall assessments and those of the teacher. Again, the implication here is that the reliability of the WITS assessments was comparable to the reliability of the other methods of assessment, both for the four separate abilities and overall.

7.5.5 Some inferences from the comparisons

Of the three methods of assessment compared here, only the WITS assessments involved objective measurement. The students' and teachers' assessments were subjective estimates of abilities which they were not normally accustomed to estimating, except in the case of teachers' estimates of students' general abilities. The teachers' assessments were collected on an informal basis, with no supervision. They had access to both the WITS assessments and the students' own, and may have been influenced subconsciously or otherwise by them. The students' self-assessments were collected by the teachers. The students in Group B had access to the WITS assessments through the transparency of the system, and this may have influenced their self-assessments.

Assessment in general is accepted to be a field where a high degree of accuracy is hardly possible. Teachers' subjective estimates of students are known to show considerable variation. (See Mehrens and Lehrmann, 1969.) Some research has indicated that teachers are substantially influenced by non-educational factors (see, for example, Keddy, 1971). There is little research on students' self-assessments. Such assessments are not usually given much credibility and are never used in schools for serious assessment, though there was no reason to doubt the sincerity of the assessments made here.

In view of this, any inferences from the findings in the WITS trials must be regarded as broad indications only. However, some tentative inferences can be drawn.

When attempting to assess different student abilities, in all groups of students there was a positive or zero correlation between the assessments by WITS and the self-assessments of the students, and also between the assessments by WITS and the assessments by teachers where these were collected. In some cases the correlations were significant. The correlations between the teachers' assessments and the students' self-assessments were not consistently higher. (The correlation was higher in Group B and lower in Group C.)

Correlations between the overall assessments of WITS and the teachers was positive and substantial in both groups B and C. This is perhaps the best indication that WITS had assessed the students accurately, as it involves a quantity which the teachers were used to estimating. The WITS assessments agreed with the student self-assessments better than the teachers' assessments did in Group C, but not as well in Group B. It is more likely that this is due to differences in assessment technique between the two different teachers than to a variation in WITS or in the students, the former being an objective assessment and the latter spread over an number of people.

Agreement between the WITS assessments and the students' self-assessments seemed to increase with the maturity of the students. It was rather better for the 'A' Level Physics students (Group C) than the 'A' Level Technology students (Group B), and very good for the students who knew the subject

(Group A). If the WITS assessments are taken to be accurate, this is consistent with the students' self-assessments improving as they mature and gain in self-knowledge.

It appears that, within the normal limitations of student assessment, there was nothing in the trial results to indicate that WITS had assessed the students inaccurately, or was greatly at odds with either the teachers' assessments or the students' self-assessments. Agreement between WITS and both teachers and students separately was comparable to the agreement between teachers and students.

7.5.6 The results treated as profiles

If the scores for different abilities are taken as individual profiles, as in the WITS system (see Figure 6.7), it will be seen that the abilities appear to be quite independent. While in many cases the 'progressive' abilities go steadily down (e.g. B2 and B4) or steadily up (e.g. C6 and C10), there are examples of students (e.g. B6 and C5) who 'peak in the middle'.

It is not easy to generalise about what the results showed about the different abilities themselves. On the whole, good students who did well overall did best in the 'higher' abilities of application and insight (e.g. C4, C5 and C9), and poorer students did worst (e.g. B1, C1 and C2), as might be expected. This was not always the case, however, as a comparison of B2, B5 and C3 shows; these students have similar scores in recall and understanding, but considerably different scores in application and insight. The mature students of Group A, who as expected were generally able, all did reasonably well in the higher abilities but did not necessarily achieve their highest scores in them. The best 'student' of all, appropriately A1, who was a practising teacher of Electronics, scored highly in everything and very highly in the higher abilities.

It is noticeable in Table 7.8 (a) for Group C that an unusually high proportion (7 out of 10) of the profile graphs 'sag in the middle', with the score for recall higher than that for understanding, and the score for insight higher than that for application. Whether this is due to a

peculiarity of the questions given, or to a peculiarity of the method, or to coincidence, or to some intrinsic quality of the abilities measured, is hard to say. It is not true of Table 7.7 (a) for Group B.

A possible reason for differences between the groups is that there was a major difference between Groups A, B and C. Group C did all 40 of the questions in the system, while the others did as many as they chose to in the time available (see Table 7.14). In the case of Group A this was an average of 23 (varying from 14 to 33), and in the case of Group B an average of 32. (All but one, who did 17, did 34 or 35.) It is contended that the probability method gives a reasonably accurate 'running score' in each ability, as it is intended to simulate a teacher's informal assessment of the student, such as might be based on verbal questions given at random. However, if students are given different questions, and different numbers of them, there will clearly be some dependence on the questions actually given, just as there would be with questioning by a human teacher.

Thus Groups A, B and C, in a statistical sense, were far from being comparable groups. The probability assessment method was being used here in two distinct ways:

(a) As a technique for assessment in an ITS, in which it was attempting to simulate informal questioning by a teacher (Groups A and B).

(b) As an examination technique for formal assessment (Group C).

The difference in use might account for the higher correlations obtained with Group C than Group B, but this needs to be set against the fact that the highest correlations of all were obtained with Group A. What can be said is that positive correlations were obtained in all Groups, as described above, and given that informal assessment can be expected to be less accurate than formal assessment, whether by a teacher or by an ITS, WITS seems to have performed reasonably well in both methods of use.

7.6 Results: How Students and Teachers Reacted to WITS.

7.6.1 Verbal reactions of teachers

Verbally, students and teachers were enthusiastic about WITS. The two heads of departments from the school who initially viewed a demonstration version of the program, without videodisc, were interested and accepted involvement immediately. The students from Group B who worked on WITS maintained throughout that it was useful to them, borne out by their attendance throughout a whole term. The teacher from the school who acted as one of the 'students' in Group A was particularly enthusiastic, inquiring repeatedly about costs and how the system could be procured for the school. This teacher wrote a note at the end of the project saying that both teacher and students had "really enjoyed" using the system.

It is interesting to compare this attitude of students, and particularly teachers, with the negative attitudes to the TICCIT whole course project reported in Section 3.3. It seems that teachers are now receptive to such a notion. It must be said, however, that WITS was used here as an auxiliary to normal teaching and not as a stand-alone system, and teachers may have viewed it differently had it been otherwise.

7.6.2 System interaction data

In an effort to obtain more objective data on the students' use of WITS, a number of system interaction parameters were programmed to measure such things as keypresses, uses of various features, and times in different modes.

Most of the interaction measurements are shown in Table 7.9. All keypresses by the student were counted except whilst viewing videodisc sequences, and student interaction rate was calculated for the table by dividing this keypress count by the time spent not viewing sequences. Interaction rates varied from 4.1 to 7.6 keypresses per minute. Two thirds of the students covered (or said they already knew) the whole course, and the least covered was just under half. 'Misunderstandings', or cases where student input produced no useful response, were generally low, except in one case, indicating relatively little experimentation.

TABLE 7.9

Students in Group A and B profiled by INTERACTION PARAMETERS

Student	Interaction Rate (per min)	%	Misunderstandings	Refuse Qs option	Information option	Mode changes	Topics via SEARCH	Topics sought by keyword
Group A								
A1	6.6	100	11	0	2	10	39	3
A2	4.1	71	2	0	0	3	10	10
A3	3.9	45	1	0	1	0	0	8
Group B								
B1	4.6	55	9	0	2	1	5	1
B2	7.2	100	99	17	3	19	165	2
B3	4.9	100	6	4	2	5	91	6
B4	6.2	100	12	0	3	6	187	3
B5	6.0	100	7	0	2	6	22	0
B6	7.6	100	30	0	2	4	64	3
Average	5.7	86	177	2.3	2	6	65	4

TABLE 7.9 (contd.)

Student	Time in modes (%)			Course Distribution (%)		
	INSTRUCT	CHOICE	REVISE	VIEWING	QUESTIONS	INTERACTING
Group A						
A1	74	12	14	53	10	37
A2	13	36	51	38	26	36
A3	0	100	0	50	9	41
Group B						
B1	72	28	0	31	9	60
B2	9	44	47	45	7	48
B3	21	19	60	47	6	47
B4	10	30	60	62	11	27
B5	71	2	27	47	13	40
B6	90	0	10	42	10	48
Average	40	30	30	46	11	43

TABLE 7.10

Use of the STUDENT ENVIRONMENTS (MODES) in WITS (Q1,No.9)

STUDENT	How did you react to the three MODES?			Time spent in the modes			No. of mode changes
	Useful	Did not use them	Could have been simpler system (negative attitude)	INSTRUCT	CHOICE	REVISE	
Group A							
A1	x			74	12	14	10
A2	x			13	36	51	3
A3		x		0	100	0	0
Group B							
B1			x	72	28	0	1
B2	x			9	44	47	19
B3	x			21	19	60	5
B4	x			10	30	60	6
B5	x			71	2	27	6
B6	x			90	0	10	4
Percentage	78	11	11				
Average				40	30	30	6

TABLE 7.11(a)

Use of the PROFILE OF PROGRESS facility Q1, Nos. 1 and 2

STUDENT	Did you find it useful?					Did it encourage you to improve?			No. of times profile used
	Very	Fairly	No opinion	Not very	No use	Yes	Maybe	No	
Group A									
A1		x				x			7
A2	x					x			5
A3	x					x			3
Group B									
B1		x				x			1
B2	x					x			17
B3		x				x			7
B4		x				x			9
B5		x				x			17
B6	x						x		16
Percentage	44	56	0	0	0	89	11	0	
Average									9

TABLE 7.11(b)

Use of the PROFILE OF PROGRESS facility Q1, No.3

STUDENT	Which assessment in the profile did you find most interesting?						No. of times used profile
	Recall	Und	App	Insight	Equally interesting	Of no interest	
Group A							
A1					x		7
A2		x					5
A3					x		3
Group B							
B1					x		1
B2					x		17
B3			x				7
B4					x		9
B5		x					17
B6					x		16
Percentage	0	22	11	0	67	0	

TABLE 7.12

Use of simple NATURAL LANGUAGE or KEYWORDS in WITS

STUDENT	Did you find it useful?			Times used keyword option	Topics seen this way
	Yes	Not aware of it	Always used SEARCH instead		
Group A					
A1			x	3	1
A2	x			10	3
A3	x			8	2
Group B					
B1	x			1	0
B2	x			2	0
B3	x			6	2
B4			x	3	0
B5			x	0	0
B6	x			3	0
Percentage	67	0	33		
Average				4	1

TABLE 7.13

Opinions of WITS assessment

STUDENT	Did you feel your final report was					Student Category
	Very accurate	Fairly accurate	Irrelevant	Not very accurate	Completely wrong	
Group A						
A1		x				A
A2		x				B
A3		x				B
Group B						
B1			x			D
B2		x				D
B3		x				C
B4	x					C
B5		x				C
B6			x			C
Group C						
C1		x				C
C2			x			E
C3		x				C
C4	x					A
C5	x					B
C6			x			B
C7		x				D
C8		x				C
C9	No student self-assessment provided					B
C10						B
Percentage	18	59	23	0	0	

TABLE 7.14

Students profiled by PROBABILITY ASSESSMENT

Student	Quests done	Best	Near	Correct	Incorrect	2nd tries
Group A						
A1	33	19	3	6	5	11
A2	22	10	6	1	5	6
A3	14	8	2	2	2	4
Group B						
B1	17	3	4	2	8	10
B2	34	14	2	9	9	18
B3	35	11	6	10	8	18
B4	35	14	4	8	9	17
B5	35	11	6	10	8	18
B6	34	12	5	7	10	17
Group C						
C1	40	13	6	7	14	
C2	40	9	9	8	14	
C3	40	16	5	10	9	
C4	40	23	8	5	4	
C5	40	13	4	2	6	
C6	40	16	5	10	9	
C7	40	10	6	11	13	
C8	40	14	5	10	11	
C9	40	20	5	7	8	
C10	40	16	7	10	7	
Average		13.3	5.2	7.1	8.4	
Percentage		39	15	21	25	

TABLE 7.14 (contd.)

Student	Recall	Undstanding	Application	Insight	Average	Category
Group A						
A1	69	71	77	88	76	A
A2	66	67	63	58	64	B
A3	69	66	61	51	62	B
Group B						
B1	36	38	24	24	31	D
B2	51	50	43	14	40	D
B3	51	51	45	40	47	C
B4	53	52	49	30	46	C
B5	51	52	45	51	50	C
B6	51	51	54	49	51	C
Group C						
C1	46	45	43	47	45	C
C2	42	42	16	19	30	E
C3	54	55	45	63	54	C
C4	78	76	76	80	78	A
C5	67	69	62	65	66	B
C6	54	55	58	73	60	B
C7	42	39	31	35	37	D
C8	49	49	39	53	48	C
C9	64	63	62	70	65	B
C10	61	60	62	78	65	B
Average	55.5	55.3	50.3	51.6	53.3	C

Questions were rarely refused, again with one exception. The 'help' information available was consulted by all students except one, sometimes up to three times.

Most students spent their time fairly evenly between viewing the course on videodisc and interacting otherwise with the system, with considerably less time spent answering questions. There were some questions, which were not used for assessment, on the videodisc, and these do not show on the 'time spent on questions'. The interaction time included looking through the search option lists, reading the system information, studying the profile, obtaining printed reports, and in some cases leaving the system running for some minutes at the end of the session before the session was terminated. The low proportions of time spent on questions were disappointing, as students were meant to do them slowly and carefully. However, they had been trained at school to do conventional MC questions quickly as though in an examination.

Time spent in the three environments or modes varied considerably. Most students spent most of their time in either INSTRUCT or CHOICE mode, experimenting for a while with the other, then spent some time at the end of the course in REVISE mode. Two students who did not finish the course, A3 and B1, spent no time in REVISE mode.

7.6.3 Reactions to intelligent features

All students who used the system were given a questionnaire to collect their self-assessments, and also to collect their reactions to the system, specifically to the 'intelligent' features. (They were not described thus to the students.) The questionnaire is in Appendix 7.1.

Students were asked how they reacted to the three environments or modes of WITS. (See Table 8.10.) Seven students (78 %) found them useful; one (A3) who did not spend long on the course, did not use them; and one (B1) had a negative attitude, choosing the option "there could have been a simpler system", but this student only changed mode once, and did not use REVISE mode. It is possible that this student completely failed to understand the use of the modes.

The profile of progress option was used frequently, some students using it 16 or 17 times, about half the number of questions. As questions were presented to most students two at a time, these students probably consulted their profile after every question session. (See Table 7.11 (a).) All students said the profile option was either 'very' or 'fairly' useful. All said it encouraged them to improve their performance, except one who said 'maybe'. Most students found all the abilities measured in the profile 'equally interesting', three being most interested in understanding (2) and application or problem solving (1). None found their profile 'of no interest'. (See Table 7.11 (b).)

A simple form of natural language was included in WITS, allowing the student to enter sentences or key words or phrases to search for a topic. Use of this facility was disappointing.

Two thirds of students said they found it useful, while one third said they always used the 'search' option instead, but the keypress counts show that it was used very little with any success. Table 8.12 suggests that the students tried to find a few topics using keywords, but actually saw very few by the method, and so seem to have given up. There are several possible reasons for its lack of success.

(a) The 'search' option had the advantage of showing the structure of the modules, units and topics, which was hidden using keywords.

(b) If they put in a module or unit name, to see a topic within the module or unit its whole title had to be typed in, which students may have found tedious.

(c) Students in INSTRUCT or CHOICE mode were not allowed to see topics out of order this way (though this applied to 'search' as well).

All the students who were assessed by WITS were given printed reports on their performance and asked what they thought of them. The majority (77 %) thought they were 'very' or 'fairly' accurate, including 18 % who thought they were very accurate. (See Table 8.13.) No students said their report was 'not very accurate' or 'completely wrong', but four (23 %) said that they thought the report was 'irrelevant'. This was in the middle of a five

point scale and may simply indicate that these students did not wish to express an opinion, or they may have felt the assessments in abilities like recall and understanding really were irrelevant, as their 'A' Level exam when it came would not assess such things.

7.6.4 Some general comments

A number of suggestions was made for the improvement of WITS. These included more sound effects, more sequences with sound, and several suggestions for additional topics that were not already on the videodisc. Two students wanted more questions, and one wanted a breakdown of the assessment by unit or module. Two students commented that too many keypresses were required between sequences, and one that 'space' should be used to go onto the next frame of a stills sequence, not F for forward. One student said some sequences were too short, and another that some could be strung together. These are sound suggestions and several could be included in a future version.

There was considerable variation in different students' use of WITS. Although the group of mature student (Group A) were rated much higher in the assessment, as was to be expected (see Table 7.14), rather surprisingly they were not noticeably more adventurous in their use of the system (see Table 7.9). To take extreme examples, compare students B1 and B2 in Table 8.9. Student B1 showed the least involvement, with low counts in the use of all facilities, and only changing mode once. Student B2 on the other hand showed great experimentation, refusing questions until ready (although nearly all were eventually completed), pressing wrong keys and questioning the system, using the more flexible modes more often, changing mode frequently, and using the search, profile and information options frequently.

The picture of student B2 is of a quick, self-confident, enthusiastic key-presser, who did rather badly in the assessment probably because he refused questions early on and then tried to do them too quickly at the end of the course. It would be possible to extend the 'report' rules of WITS to pick out and comment upon such students.

7.7 Conclusions

The evaluation of the system could be regarded as a piece of 'action research'. It formed a small-scale intervention in the students' education, with observations and modifications made as the evaluation progressed. A mixture of normative and interpretive methods was used.

Certain constraints, such as having only one set of equipment, the best videodisc available being lacking in some ways, and reservations on the part of the school, meant that WITS had to be evaluated with a small number of students on an informal basis. Three groups of students were used, a small group of mature students and a larger group of school students, both of which worked on the computer system, and another group who were assessed using the probability method independently of the computer system.

Did WITS teach a whole course effectively?

The WITS system functioned as planned and a number of students worked through a course on Solid State Electronics on the system for a total of some 40 hours. It was not possible to assess the effectiveness of WITS' teaching objectively, due to difficulties of separating the teaching students received from the system and their school. Subjective reactions from students and teachers indicated that it was well received, and was considered to be effective as a supplementary system.

Did WITS assess students accurately?

The probability method of assessment was evaluated by comparing its results with assessments by teachers and the students themselves. There were positive correlations between assessments by WITS and the other assessments, and there were no significant differences between the assessments by any of the three methods. The large majority of students thought their assessment was very or fairly accurate, and none thought it was inaccurate.

How did students react to the 'intelligent' features?

The students used the 'intelligent' features of WITS extensively. The large majority found the different environments or modes useful. All students found the profile of progress, a reflection of system transparency, either very or fairly useful. Nearly all students said the accessibility of the profile encouraged them to improve. The natural language or keyword entry facility was said to be useful, but in fact it was used little. This may have been due to the way it functioned in the system. Verbally, the students and teachers were enthusiastic about WITS.

Chapter 8

Conclusions

8.1 Introduction

The comment by Blaine and Smith (1977), reproduced in Section 1.4, is worth repeating here: "There is significant further progress to be made in AI by systematically dealing with an entire curriculum within the AI paradigm." In this chapter some findings relevant to AI, or more specifically ITS, will be outlined.

Several features were incorporated into WITS which are usually associated with Expert Systems, Artificial Intelligence generally or Intelligent Tutoring Systems. Some of the intelligent features could be evaluated, as explained in Chapter 8, by collecting students' opinions and keypress counts. The assessment method was evaluated by comparing its assessments with assessments from other sources. Some of the intelligent features, for example the use of a rule-based PROLOG program, could only be evaluated by whether they were successfully programmed and contributed to the functioning of the system as a whole.

The findings from the WITS system are first summarised; some spin-offs and further research suggestions are examined; and a retrospective appraisal of the course-oriented approach is given.

8.2 Findings from designing and building WITS

8.2.1 The rule-based PROLOG teaching module

The entire WITS program was written using PROLOG, and some examples of rules from the program are given as appendices to Chapters 5 and 6. PROLOG was found to be an excellent language in which to program a whole course system using videodisc material. The videodisc sequences are readily classified and accessed as groupings of PROLOG facts.

PROLOG is not easy to learn, and its use is not straightforward in a large program because different techniques are required which are not described in most textbooks. However, its suitability repays the trouble taken. Its predicates lend themselves to structured programming quite as well as the procedures of PASCAL. Its declarative composition readily lends itself to extending and modifying the program. Perhaps most important, the simple database structure gives a fluid and easily understood relationship between the knowledge base and the rules which handle it.

8.2.2 WITS as a transferable shell or front-end

The system was designed mainly as a prototype to teach Electronics, but it was programmed in such a way that there would be no system messages or features specific to the subject of Electronics. It was possible to

substitute for the Electronics database another one relating to the Vincent Van Gogh disc from Philips, converting the prototype into a system to teach a course on the subject of the painter. The new database is shown in Appendix 5.9, and some examples of frames from the Van Gogh course are given in Figure 8.1.

By and large, the WITS 'shell' was found to be transferable to another videodisc. The WITS system was demonstrated as a videodisc 'front-end' at the National Interactive Video Centre, to representatives of the centre and of the Council for Educational Technology, and it aroused considerable interest and approval.

This exercise brought out some problems of transferability. For example, the rectangular frame with a blue border that was retained throughout for the SSE disc was unsuitable for the Van Gogh disc, as the border obscured the outer part of the full-frame paintings. (See Figure 8.1.) Also, the highly structured system of modules, units and topics rested uneasily with an 'arts' subject such as Van Gogh, rather than science. Other problems would probably emerge with other discs.

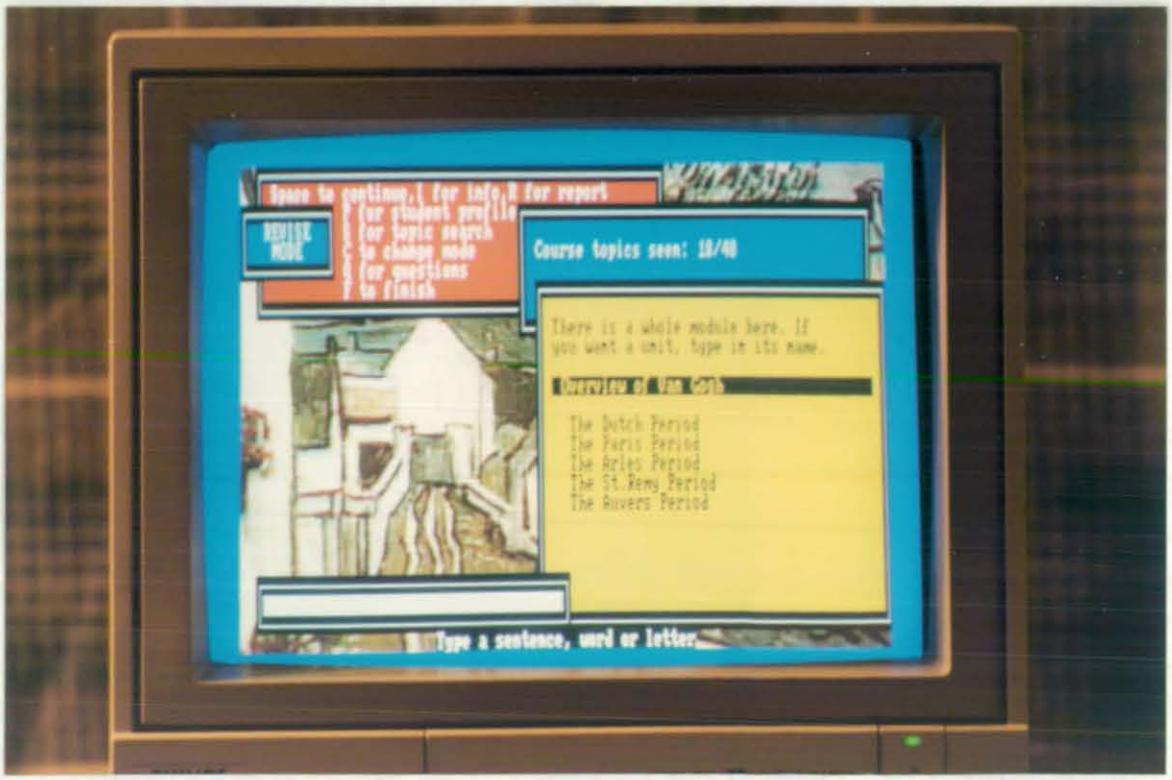
On the evidence of this project the model of an ITS with a rule-based PROLOG teaching module and a videodisc knowledge base, linked by a database of PROLOG facts, was found to be a good basis for a teaching system that is transferable from one knowledge base to another.

8.2.3 The flexible learning environments

When the project started in 1984 the term 'environment' was used to refer to a particular type of ITS. (See, for example, Miller, 1982.) While planning the form of this whole course ITS, it was felt, with regard to such findings as those of Ferguson (1984) and Laurillard (1985), that to hold the attention of a variety of students and accommodate their needs over the duration of a whole course, a choice of environments would be needed, that is, considerable flexibility. In so far as flexibility is a feature of intelligence, the environments or modes built into WITS were regarded as an intelligent feature.

Figure 8.1

Viewing screens of the Van Gogh videodisc course



It appears that there is now some consensus that a suitable environment is a characteristic feature of an ITS. In a recent book of authoritative papers (Polson and Richardson, 1988) an environment component is identified and described by Burns and Capps, and elaborated upon in a special paper by Burton, who says: "Learning is greatly enhanced by a proper facilitating environment ... " The importance attached to the choice of environments in WITS seems to have been vindicated by the importance that is now attached to the environment component of an ITS by practitioners in the field.

As described in Chapter 7, the modes or environments were found to be useful by the students who used WITS. The choice between three modes was probably enough. More would have been confusing, though it is possible that INSTRUCT and CHOICE mode could have been combined to leave two. There is room for some 'tinkering' between the facilities offered in the different modes.

A finding of the preliminary inquiry, borne out in practice, was that in a whole course ITS a choice of learning environments, and some means for the student to modify the learning environment, are highly desirable.

8.2.4 Natural language in WITS

The student is able, at any time, to type in whole sentences or keywords as well as commands, so that WITS can be said to have a form of natural language interface. Topics are presented to the student if appropriate in response to keywords detected in the input.

In programming terms this interface worked well. It was found possible to classify a knowledge base of 121 topics, 23 units, 5 modules and several additional topics, i.e. more than 150 objects, using keyword groups or paired keyword groups. Although of a simple form, the WITS natural language interface could readily by extended.

In terms of student use this aspect of the interface worked less well. Students preferred to find topics via the search option, which listed the topics available for them to choose from.

An elaborate natural language interface which could parse an input sentence, and perhaps using, for example, such devices as Schanks'ideas of conceptual dependency (Schanks, 1977), was found to be unnecessary in this sort of application. It was found that students do not want to be bothered to type in whole sentences, or even words. They want menus and single keypresses, perhaps because they are already conditioned by other programs they have used.

However, the usefulness of the keyword searching facility is likely to depend on the subject. Although it was not much used by the Electronics students, it was noticeable that in the Van Gogh disc application it was easier to find a particular artist's work by typing in the name than by looking through alphabetical groupings in the search option.

8.2.5 The student model and transparency

An elaborate student model was built up of the student in WITS based on the kind of student variables teachers and educators are involved with. Such a model could be regarded as a profile of the type some schools are now trying to produce on students. It was readily converted to an end-of-course report on the student. The probability assessment method was found to lend itself particularly to the compilation of such a student model, and to the compilation of student profiles and reports. It may have applications in these fields.

The student model was made available to the student in the form of an on-screen profile, at all times. This form of system transparency, letting the user know how the system is using the information gathered, is accepted as a desirable feature of expert systems. It was found to be appreciated by the students, who said in most cases that seeing their profile increased their involvement and effort with questions, and therefore stimulated their learning.

8.2.6 Probability assessment in WITS

A probability assessment method was devised specially for the WITS system, and it is claimed that it is an improvement on conventional assessment in several ways:

a) It is more efficient.

b) It makes clearer what is being assessed or measured.

c) It allows several variables to be measured separately in one test.

This probability assessment method arose from the whole course approach, where there was a need to assess students in detail with an economical number of questions. It was found to give assessments of students which were not significantly different from the teacher's assessments of the students, or from the students own self-assessments, though more detailed than would normally be given. There was a positive correlation between all these assessments. The probability assessment method is described in detail in Chapter 6, to which the reader is referred.

8.3 Suggestions for further research

WITS was designed as an ITS, with characteristics of a transferable expert system shell. As such it can also be described as a 'front-end' for other ITSs, and also for videodisc applications, in educational or in other fields. Used only in INSTRUCT mode, WITS resembles a conventional CAL program offering rigid, linear instruction. Used in REVISE mode, it could be used with a suitable videodisc for any application where structured search and keyword search methods would be useful - for example, in point of sale applications. The ideas in this project could be applied to other areas in this way.

The clearest possibility for further research is into the role a system such as WITS might play in existing educational institutions. This is commented upon further in Section 8.4.

The probability assessment method offers several possibilities for further research. It is described here as a hypothesis, needing further mathematical development and testing.

For purposes of evaluating the WITS prototype, answer parameters were determined by an 'expert judgement' method (see Section 7.6. In theory these parameters can be measured using samples of students already at the required standard. It would be of great interest to test the probability method rigorously with measured parameters.

Evaluation of the probability method of measuring several student abilities at once was carried out here by comparing the probability assessments with teachers' assessments and student self-assessments. This is a subjective evaluation, and a more objective one could be carried out by comparing the probability assessments of students with assessments obtained from separate tests to determine the abilities, as described by Bloom (1956).

To facilitate the estimation of answer parameters in the student abilities of recall, understanding, application and insight, a simple graphical algorithm was devised to describe the relationship of these abilities to each other (see Section 7.7). There is scope for research in devising tests for other student abilities which educators might want to measure, and determining algorithms for estimating answer parameters.

There is a possibility of using the probability assessment method for user-modelling in adaptable computer interfaces. One of the problems with such interfaces is that they need to build a model of the user very quickly, on a minimum of evidence, if they are to be useful, and if they are not to spend an unacceptable amount of time questioning and assessing the user. The efficiency of the probability model in quickly producing a usable model offers some promise here.

Probability assessment might have applications in the field of assessment generally, outside of computer assisted learning. It might, for example, enable conventional objective testing by examining bodies to be made more efficient; it might bring about greater clarity about what is being tested; and it might enable several different student variables to be assessed.

In particular, probability assessment lends itself to the compiling of student profiles and reports. Applications in this area might be a useful area of study. A research project suggested by the present one would be to design an ITS to write profiles and reports on students. WITS goes some way towards this, and such a system might make use of probability assessment. In a passage which seems to call for the coming of the micro,

Keating (1969) says: "Since there is invariably some degree of stereotyping in the sense of all forms of report falling into more or less broad categories, could one not find a way of increasing the phrases available for comment and at the same time lightening the actual work of writing them?" The simple reports produced in WITS showed that PROLOG is well suited to this exercise, and there is scope here for an intelligent profiling and report writing system working on direct assessments of the student, or authored responses from teachers.

Two other areas for research arising indirectly from this project are as follows. First, there is a need in interactive video for a method of searching a videodisc by image recognition. To be able to search for an image among the 50,000 on a videodisc by comparing it with a sample image would be useful, for example, in police work such as searching through filed fingerprints or photographs; perhaps more useful in an ITS would be the facility to search for an image described by a few mathematical parameters, compiled in response to an input keyword.

Second, there is a need for development of a method of natural language analysis which can be used to mark essay-type answers from students, even in a restricted comprehension-test form. Although a solution to the problem of universal natural language input remains elusive, it might be solved in this restricted domain. Mehrens and Lehmann (1969, Chapter 8) list advantages and disadvantages of essay-type tests, concluding that in spite of poor predictive validity, limited course content sampling, unreliability of marking and expense, they are still popular because they can indirectly measure attitudes, values and opinions; they represent good learning experiences; and they encourage structured, coherent expression.

8.4 Orientation of the research

An obvious, even necessary, extension to this research, as mentioned in Section 8.3, is to examine how a course-oriented ITS might be used in existing or future educational institutions.

Such a system when further developed might take some of the work-load of assessment and first-time teaching away from the teacher. There is concern at present that teachers have had too many demands heaped upon them, and computerised assistance to create more time could only be beneficial. It is

likely that a system such as that described here would be most useful in science or technical subjects, where a crowded syllabus of hierarchical topics needs to be covered; where video images of everyday applications are useful; and where examples and exercises need to be presented to the student.

It is not intended to suggest here that a 'whole course ITS' could cover the whole of a course to the extent of dispensing with the human teacher. There are areas where it is hard to see ITSs competing with human teachers in the near future, if ever, for example in the supervision of practical work, projects, report-writing, essays, and in providing tutorial help with individual problems. Where a course-oriented ITS would be useful is in freeing the teacher for more of these essentially human activities.

The approach to ITSs and to CAL generally at present seems to be to present the teacher with single, one-off programs which are useful in covering one or more aspects one or more topics, which may or may not be extended to other topics. An acceptance that computer systems could, with present technology, be used to take much of the burden of whole courses off the teacher's shoulders, and a determined effort to develop such systems, might produce a large increase in the use of computers in education, and considerable improvements in teachers' and students' achievements.

8.5 Summary of findings

The project investigated the problems of designing intelligence into a course-oriented ITS. It was essentially a programming project undertaken in a computing department, and achieved its main objective in that certain arguments were arrived at through investigation, and an ITS was designed and built to test them, which functioned as planned. The investigation, the experience of programming the ITS, and the results of evaluating the ITS produced the following findings.

8.5.1 Approaches to ITS

There seems to be no consensus about an achievable, short-term approach to ITS. Approaches today, such as the machine-learning approach of Self (1985) at Lancaster, or the cognitive theory approach of Anderson (1985) at

Carnegie-Mellon, would seem to be long term. A course-oriented approach provides an achievable short-term objective enabling usable ITSs to be built.

The problems with early attempts at whole course CAL projects, arising mainly from limited technology and negative teacher attitudes, appear to have diminished considerably. A whole course approach is virtually without precedent in the field of ITS, but can profitably draw on experience in the fields of CAL and IV.

It is possible today to develop initial systems to a stage at which they can be used by students. With modern computing techniques such as high level languages, large capacity micros and interactive video, there is no need for ITSs to be purely research prototypes.

Although on the whole expert systems do not transfer well to ITSs, it is possible with modern technology to design an ITS like an expert system shell, transferable from one area of subject knowledge to another.

If ITSs are to be used widely outside of 'toy' domains, they should accept that didactic, declarative instruction is required as well as interactive, procedural tutoring. The declarative aspects of an ITS are more transferable than the procedural aspects, and it is possible to design declarative front-ends for procedural ITSs.

An approach which can give direction to ITS research is a teacher-based approach. CAL systems tend to resemble tutors in that they operate on a one-to-one basis, but the techniques and methodology of teachers and the large body of educational research have much to offer, particularly in the areas of knowledge organisation, presentation and assessment. A teacher-based ITS can be looked upon as an expert system which simulates the teacher as expert.

8.5.2 The design of a course-oriented ITS

A course-oriented system can usefully be designed around the 'traditional' ITS structure of knowledge-base, teaching module and student model with the addition if relevant of environment and interface components (as described in Polson and Richardson, 1988).

PROLOG was found to be highly suitable for programming a project of this type, handling student interaction in real time with no perceptible delays. It lends itself to modular programming, program clarity and later extension and modification. Certain difficulties arising from the use of PROLOG in a large program deserve to be looked into.

The general approach of the educational research method known as action research can be useful in evaluating an ITS with students.

8.5.3 The knowledge base

Interactive video provides an efficient way of representing the large knowledge base required in a course-oriented ITS. It can include not only subject knowledge but a large amount of teaching expertise as well. It could provide a useful way of extending existing procedural ITSs to include declarative knowledge.

The organisation of knowledge for a whole course is most simply modelled on traditional, hierarchical structures favoured by textbooks, syllabuses and teachers.

8.5.4 The tutoring module

The teaching component of a course-oriented ITS needs to handle a larger amount of knowledge than topic-based ITSs. It needs to handle it in relatively large 'chunks', which can correspond to sequences on the videodisc. This is a coarse-grain approach. The fine-grain handling of knowledge can be carried out within a videodisc sequence.

Perhaps the main intelligence requirement in a whole course ITS is that it should be flexible, offering a variety of learning environments. This was achieved in the system built by offering three environment modes, with the facility to change between them at will.

Simple natural language, allowing keyword searching, was found not to be used extensively in the system that was built, the students preferring menus and commands. This may have been due to the way it was implemented, or to the subject matter.

8.5.5 The student model

The design of the student model in a whole course ITS can correspond to a teacher's assessment of a student. A whole course system can usefully base its student model on teachers' profiles of students, a field where educational research is continuing. Such a model should be available to the student, for the same reasons that 'transparency' is highly valued in expert systems. Research into games indicates that immediate feedback of this type promotes involvement, and this was borne out in the evaluation of the system built here.

To compile the student model, objective, unambiguous multiple choice questions of the type familiar to students may be used. It is possible to modify conventional testing using 'fuzzy reasoning' techniques of the type used in expert systems. In this project this led to a probability assessment method which, evaluation suggested, compared satisfactorily with conventional assessment.

With the probability assessment method used here, it is theoretically possible to assess the student more efficiently, that is, using less questions. It is also possible to be more clear about what is being measured, as the questions and answers are designed to test specific hypotheses about student ability, and the quantities involved are real probabilities, capable of being objectively measured. It is also theoretically possible with this method to assess several student variables or abilities within one test. Student abilities of recall, understanding, application and insight were assessed here, and results were found to show no significant difference between teachers' assessments and the students' self-assessments.

An assumption made in the probability method of assessment needs to be pointed out, namely that questions and student answers are assumed to be independent of each other. This is unlikely to be wholly the case, and it raises theoretical problems with the method. However, this is a common problem with most forms of assessment and many expert systems, as pointed out in the summary to Chapter 6, and it need not detract greatly from the usefulness of the method in practice.

Overall, the course-oriented approach of the project led to a number of findings and spin-offs, the most notable being the probability assessment method. It is an approach which could usefully be applied to other ITS projects.

References

Alderson, G. and Dewolf, M., 1984, **Guide to Effective Screen Design**, Computers in the Curriculum, Chelsea College, London.

Anderson, J. R., 1983, **The Architecture of Cognition**, Harvard University Press, Cambridge, MA.

Anderson, J. R., 1988, **The Expert Module**, in Polson, M. C. and Richardson, J. J. (Eds).

Anderson, J. R., Boyle, C. F. and Yost, G., 1985, **The Geometry Tutor**, Paper at 9th Int. Conf. on A.I., Carnegie-Mellon, USA.

Anderson, J. R., Boyle, C. F. and Reiser, B. J., 1985, **Intelligent Tutoring Systems**, Science, Vol. 228, April 26th.

Anderson, J. R., Corbett, A. T. and Reiser, B. J., 1986, **Essential Lisp**, Addison Wesley, Reading, MA.

Anderson, J. R., Farrell, R. G. and Reiser, B. J., 1985, **Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming**, Paper at 9th Int. Conf. on A.I., Carnegie-Mellon, USA.

Anderson, J. R. and Kline, P. J., 1979, **A Learning System and its Psychological Implications**, paper at 6th Annual Conference of the International Journal of Computer Aided Instruction.

Anderson, J. R. and Skwarecki, E., 1986, **The Automated Tutoring of Introductory Computer Programming**, Communications of the ACM, Vol.29, p.842-849.

Atkinson, R. C., 1976, **Adaptive Instructional Systems: some attempts to optimise the learning process**, in Klahr, D. (Ed).

Banet, B., 1979, **Computers and Early Training**, Calculators/Computers, Vol. 3, p.17.

Barr, A., Beard, M. and Atkinson, R.C., 1976, **The Computer as a tutorial laboratory: the Stamford BIP project**, Int. Journ. Man-Machine Studies, Vol. 8, p.567-96.

Barzilay, A. and Pople, H. E., 1984, **Spirit: An evolutionally designed intelligent tutoring system**, LRDC Tech. Report No. UPITT/LRDC/ONR/APS-15, University of Pittsburgh, PA.

Bayard-White, C., 1985, **An Introduction to Interactive Video**, National Interactive Video Centre, Council for Educational Technology, January.

BBC, 1988, **The Ecodisc**, sales literature available from the British Broadcasting Corporation.

Benyon, D., 1986, **Justification and Outline of a Knowledge-Based Interface System**, CARIS Working Paper No.5, School of Computing and Mathematical Sciences, Leicester Polytechnic.

Billingsley, P. A., 1982, Navigation through hierarchical menu structures. Does it help to have a map? Proceedings of the Human Factors Society, 26th Annual Meeting.

Blaine, L. and Smith, R. L., 1977, Intelligent CAI: The Role of the Curriculum in Suggesting Computational Models of Reasoning, Proc. of Annual Conf., Seattle Assoc. for Computing Machinery.

Bloom, B. S., 1956, Taxonomy of Educational Objectives, Longmans.

Bobrow, D. G. and Collins, A. (Eds), 1975, Representations and Understanding: Studies in Cognitive Science, Academic Press, NY.

Bobrow, D. G. et al, 1977, GUS, A Frame-Driven Dialog System, Artificial Intelligence, Vol. 8, p.155-173.

Boen, L. L., 1983, Educational Technology Research: Teaching with an Interactive Video-Computer System, Educ. Technology, March.

Bonar, J., Cunningham, R. and Schultz, J., 1986, An object-oriented architecture for intelligent tutoring systems, in Meyronits, N. (Ed), p.269-276.

Bramer, M. A., 1984, Expert Systems: the vision and the reality, in Bramer, M. A. (Ed).

Bramer, M. A. (Ed), 1984, Research and Development in Expert Systems, Cambridge.

Bratko, I., 1986, Prolog Progamming for A.I., Addison-Wesley.

Brighton Polytechnic, 1989, Telsoft System, Telsoft Interactive Video Education Centre, Brighton Polytechnic, Falmer, Brighton.

Brock, W. H., 1973, H. E. Armstrong and the Teaching of Science, 1880-1930, Cambridge University Press.

Brown, S. A., 1980, What do they know: a review of criterion-referenced assessment, Her Majesty's Stationery Office.

Brown, J. S. and Burton, R. R., 1975, Multiple Representations of Knowledge for Tutorial Reasoning in Bobrow, D. G. and Collins, A. (Eds).

Brown, J. S. and Burton, R. R., 1978, Diagnosis Models for Procedural Bugs in Basic Mathematical Skills, Cognitive Science, Vol.2, Bolt, Beranek & Newman.

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 Sleeman D. and Brown J. S. (Eds).

Brown, J. S. and Vanlehn, K., 1980, Repair Theory: A generative theory of bugs in procedural skills, Cognitive Science, Vol.4, p.379-426.

Burns, H. L. and Capps, C. G., 1988, Foundations of Intelligent Tutoring Systems: An Introduction, in Polson, M. C. and Richardson, J. J. (Eds).

Burton, R. R., 1982, Diagnosis bugs in a simple procedural skill in Sleeman D. and Brown, J. S. (Eds).

Burton, R. R., 1988, The Environment Module of Intelligent Tutoring Systems, in Polson M. C. and Richardson, J. J. (Eds).

Burton, R. R. and Brown, J. S., 1982, An investigation of computer coaching for informal learning activities, in Sleeman, D. and Brown, J. S. (Eds).

Callear, D. H., 1976, The role of the head of science in innovations in science teaching, unpublished M. Ed. Thesis, King's College, University of London.

Callear, D. H., 1984, Production of Computer-Controlled Videodiscs in Education, D.P.S.E. Dissertation, South Bank Polytechnic.

Callear, D. H., 1986, An Intelligent Teaching System to teach Physics: the arguments for and against, and a design under construction, paper presented to Computers in Physics Teaching Conference, Institute of Physics, at Essex University, January.

Callear, D. H., 1988, A model of student assessment based on probabilities, CARIS Working Paper No.26, School of Computing and Mathematical Sciences, Leicester Polytechnic, September.

Carbonell, J. R., 1970, AI in CAI: An Artificial-Intelligence Approach to Computer Aided Instruction, IEEE Trans. on MMS, December.

Clancey, W. J., 1979, Tutoring rules for guiding a case method dialogue, Int.J.Man-machine Studies, Vol.11, p.25-49.

Clark, D., 1986, Aspects of Menu Design, Journal of Computer Assisted Learning, Vol. 2, No. 3, Dec, p.172.

Clark, D. R., 1984, The Role of the Videodisc in Education and Training, Media in Education and Development, December.

Clemens, J. K., 1982, Video Discs: three choices, IEEE Spectrum, March.

Cline, H. F., (Ed), 1986, The Electronic Schoolhouse: The IBM Secondary School Computer Education Program, Lawrence Erlbaum Associates Publications.

Cline, H. F. and Schneiderman, M. B., 1986, The Electronic Schoolhouse: Summary and Recommendations, in Cline, H. F. (Ed), Ch.8.

Clocksin, W. F. and Mellish, C. S., 1981, Programming in Prolog, 1st. Edition, Springer-Verlag.

Cohen, L. and Manion, L., 1980, Research Methods in Education, Croom Helm, London.

Colbourne, M. J., 1984, Expert systems in Education , Proc. of Canadian Information Processing Society, May 9-11, p.186-190.

Computers in the Curriculum / United Kingdom Atomic Energy Authority, 1985, The Nuclear Reactor Simulation, Longmans Software.

Coombs, M. J. and Alty, J. L. (Eds), 1983, Computing Skills and the User Interface, Academic Press, London (Computers and People Series).

Cornes, P., 1986, CBT: Computer Based Training, Computer Bulletin, March.

Crocker, A. C., 1974, **Statistics for the Teacher**, National Foundation for Educational Research.

Crowder, N. A., 1959, **Automatic tutoring by means of intrinsic programming**, in Galanter, E. (Ed).

D'Agapeyeff, A., 1983, **Expert Systems, Fifth Generation and UK Suppliers**, Manchester NCC Pubs.

Davies, N. G. et al, 1984, **Tutor - a prototype ICAI system**, in Bramer, M. A. (Ed).

Daynes, R., Brown, R. D. and Newman, D. L., 1981, **Field Test Evaluation of Teaching with Videodiscs**, EITV, March.

DES, 1980 (1), **Microelectronics in Education: a Development Programme for Schools and Colleges**, Department of Education and Science, London.

DES, 1980 (2), **A Framework for the School Curriculum**, Department of Education and Science Welsh Office.

DES, 1982, **Science Education in Schools: a Consultative Document**, Department of Education and Science Welsh Office,

Doulton, A., 1984, **Interactive Video in Training**, Media in Education and Development, December.

Duda, R., Gaschnig, J., and Hart, P., 1980, **Model design in the prospector consultant system for mineral exploration**, in Mitchie, D. (Ed).

Duda, R. O., Hart, P. E., and Nilsson, N. J., 1976, **Subjective Bayesian methods for rule-based inference systems**, Proc. of the National Computer Conference, AFIPS, Vol.45, P.1075-1082.

Duke, J., 1983, **Interactive Video: Implications for Education and Training**, Council for Educational Technology Working Paper 22.

Edmonds, E. A., 1983, (1), **Adaptive Man-Computer Interfaces**, in Coombs, M. L. and Alty, J. L. (Eds).

Edmonds, E. A., 1983, (2), **Designing High Quality Software for Schools**, presented to an MEP Curriculum Software Conference, 3 May, Tring.

Feigenbaum, E., 1971, **On generality and problem-solving**, Machine Intelligence, Vol.6.

Ferguson, D. L., 1984, **Intelligent Computer-Based Tutoring Systems**, Frontiers in Education Conference Proceedings, IEEE, Session 18A1.

Fisher, R. B. and Howe, J. A. M., 1982, **The Potential of Expert System Based Training Aids**, Dept. of AI, University of Edinburgh, DAI Working Paper No.121, August.

Ford, L., 1984, **Intelligent Computer Aided Instruction**, in Yazdani, M. and Narayan, N. (Eds).

Forsyth, R. (Ed), 1984, **Expert Systems: Principles and Case Studies**, Chapman & Hall.

Fox, B., 1982, **A revolution for video discs**, New Scientist, 21st October.

Fox, B., 1983, Best of Both Media, Times Educational Supplement, 25th November.

Fox, B., 1984, Not so groovy, Times Educational Supplement, 6th July.

Freeman, D., 1986, Man or Mouse? Personal Computer World, Vol.9, No.3, P.152-5, March.

Galanter, E. (Ed), 1959, Automatic Teaching: The State of the Art, Wiley, NY.

Galton, M. and Eggleston, J., 1979, Some Characteristics of Effective Science Teaching, European Journal of Science Education, Vol. 1, p. 75-86.

Genesereth, M. R., 1982, The Role of Plans in Intelligent Teaching Systems, in Sleeman, D. and Brown, J. S. (Eds).

Glaser, R. (Ed), 1964, Teaching Machines and Programmed Learning II, National Educational Assoc., Washington.

Goldberg, A. and Ross, J., 1981, Is the Smalltalk-80 system for children? Byte, Vol.8, No.6, p.347-68.

Goldstein, I. and Carr, B., 1977, The Computer as Coach: An Athletic Paradigm for Intellectual Education, Proc. of the Annual Conf. for Computing Machinery, Oct. 16-19, p.227-233, Seattle.

Goldstein, I. P., 1982, The Genetic Graph: a representation for the evolution of procedural knowledge, in Sleeman, D. and Brown, J. S. (Eds).

Gomersall et al, 1985, Letters to Physics Bulletin, Physics Bulletin, Vol. 36, Dec., No. 12, p. 483; Vol. 36, July, No. 7, p. 279.

Hall, W. C., 1973, The Development of the Learning Model and its use in teaching and assessment in the Schools Council Integrated Science Project, unpublished Ph.D. thesis, Chelsea College.

Hartley, J. R. and Sleeman, D. H., 1973, Towards More Intelligent Teaching Systems, Int. Journ. Man-Machine Studies, Vol.5, p.215-236.

Hartnell, T., 1984, Exploring Artificial Intelligence on Your BBC Micro, Interface Publications.

Hartog, P. and Rhodes, E. C., 1936, The Marks of Examiners, Macmillan, London.

Hayes-Roth, B., 1985, A blackboard architecture for control, Journal of A.I., Vol. 26, p. 251-321.

Heaford, J. M. 1983, Myth of the Learning Machine, Sigma Technical Press.

Henderson, R. et al, 1983, Effects of Interactive Video/Computer Instruction on the Performance of Underachieving Students in Mathematics, Annual Mtg. of the American Educational Research Assoc., Apr. 11-14, Montreal.

Hills, P., 1984, The DTI Initiative in Interactive Video, European Training Agency Conference Proceedings, 12th April.

Hodson, D. and Brewster, J., 1985, **Towards Science Profiles**, School Science Review, Vol.67, No.239, December.

Hon, D., 1982, **Interactive Training in Cardiopulmonary Resuscitation**, Byte, June.

Humphries, K., 1988, **Computerising a School Record System**, Computer Education, No. 58, p. 2-3, February.

Huntley, J. S. and Alessi, S. M., 1985, **Evaluation of the DISCWRITER Authoring System**, Videodisc and Optical Disc, Vol. 5, No. 5, Sept-Oct, p. 356-372.

Ingle, R. and Jennings, A., 1981, **Science in Schools: Which Way Now?**, University of London Institute of Education Studies in Education Series No. 8.

Isbouts, J. P., 1983, **Making a Videodisc on Vincent Van Gogh**, EITV, September.

Joobbani, R. and Talukdar, S. N., 1985, **An Expert System for Understanding Expressions from Electric Circuit Analysis**, 9th. International Conference on A.I., Carnegie-Mellon.

Keating, L., 1969, **The School Report**, Kenneth Mason.

Keddie, N., 1971, **Classroom Knowledge**, in Young, M. F. D. (Ed).

Kehrberg, K. T., Pollack, R. A., 1982, **Videodiscs in the Classroom: An Interactive Economics Course**, Creative Computing, January.

Kewney, G., 1981, **Interaction in Perspective**, Business Video, October.

Kimball, R., 1982, **A self-improving tutor for symbolic integration**, in Sleeman, D. and Brown, J. S. (Eds).

Klahr, D., (Ed), 1976, **Cognition and Instruction**, Erlbaum Associates, Hillsdale, NJ.

Kohler, T. R., 1977, **Optical Videodisc Technology**, Proc. of the Annual Conference, Seattle Assoc. for Computing Machinery.

Laurillard, D. M., 1983, **Interactive Futures**, Times Educational Supplement, 21st October.

Laurillard, D. M., 1984 (1), **Interactive Video and the Control of Learning**, Educational Technology, Vol. 24, No. 6.

Laurillard, D. M., 1984 (2), **Imaginative Use of Videodiscs**, Times Higher Educational Supplement, 22nd June.

Laurillard, D., 1985, **The Teddy Bears' Disc**, Media in Education and Development, March.

Lawler, R. W. and Yazdani, M. (Eds), 1987, **Artificial Intelligence and Education, Volume 1: Learning Environments and Tutoring Systems**, Ablex Publishing Corp.

Lea, P., 1988, **Teaching with Interactive Videos**, Electrical Contractor, March.

Lecarne, O. and Lewis, R. (Eds), 1975, Computers in Education, North-Holland.

Leveridge, L. L. and Lyons, D. S., 1983, Which Disc Player for Education? A Comparative Evaluation, EITV, July.

Lewis, B. N. and Pask, G., 1964, The theory and practice of adaptive teaching machines, in Glaser, R. (Ed).

Lewis, R. (Ed), 1985, Journal of Computer Assisted Learning (JCAL), first edition March/April 1985, Blackwell Scientific Publications.

Linderholm, D., 1987, A Machine for all Times, Personal Computer World, January.

Littman, D. and Soloway, E., 1988, Evaluating ITSs: The Cognitive Science Perspective, in Polson, M. C. and Richardson, J. J. (Eds).

Luehrmann, A., 1977, Intelligent Videodisc Systems - implications for education, in Seidel, R. J. and Rubin, M. L. (Eds).

Malone, T. W., 1981, Towards a Theory of Intrinsically Motivating Instruction, Cognitive Science Vol. 4, p.333-369.

Mably, C., 1984, Interactive Videodiscs in Primary School, North East London Polytechnic.

Maddison, A., 1982, Microcomputers in the Classroom, Hodder and Stoughton.

McKinnell, I., 1987, Ventura Publisher, Personal Computer World, January, Vol. 10, No. 1, p. 178-182.

McLeish, J., 1968, The Lecture Method, Cambridge Institute of Education.

Mehrens, W. A. and Lehmann, I. J., 1969, Measurement and Evaluation in Education and Psychology, Holt, Rinehart and Winston, Inc.

Meyronits, N. (Ed), 1986, Proceedings of 1st Conference on Object Oriented Programming, Systems, Languages and Applications, NY: ACM.

Miller, J. R., 1988, The Role of Human-Computer Interaction in Intelligent Tutoring Systems, in Polson, M.C. and Richardson, J.J.

Miller, M. L., 1982, A structured planning and debugging environment for elementary programming, in Sleeman, D. and Brown, J. S.

Minsky, M. A., 1975, A framework for representing knowledge, in Winston, O. (Ed).

Mitchell, A., Perfect, H. and Boyd, J., 1985, Scope for Profiling: Computer Support for Teachers in Developing Profiled Reports, Programmed Learning and Educational Technology, Vol. 22, No. 4, November.

Mitchie, D. (Ed), 1980, Expert Systems in the Micro Age, Edinburgh.

Morrison, F., 1975, Planning a large-scale computer-assisted instruction installation - the TICCIT experience, in Lecarne, O. and Lewis, R. (Eds).

National Interactive Video Centre, 1985, An Introduction to Interactive Video, Council for Educational Technology, January.

Naylor, C., 1983, Build Your Own Expert System, Sigma Press.

Nixon, J., 1981, A Teacher's Guide to Action Research, Grant MacIntyre.

Nuffield Foundation, 1966-73, Nuffield O-level Science Projects, Physics, Chemistry and Biology, Longmans.

Nuffield Foundation, 1970-73, Nuffield Advanced Science Projects: Biological Science 1970, Physics 1971, Chemistry 1971, Physical Science 1973, Longmans.

O'Shea, T., 1982, A self-Improving Quadratic Tutor, in Sleeman, D. and Brown, J. S. (Eds).

O'Shea, T., 1985, The Learning Machine, BBC Television series.

O'Shea, T. and Self, J., 1983, learning and Teaching with Computers, Harvester Press.

Papert, S., 1980, Mindstorms: Children, Computers and Powerful Ideas, Harvester Press.

Papert, S., 1984, Giving Maths Power to the Children, Interview in Practical Robotics, November/December.

Parkes, A. P., 1987, Towards a Script-Based Representation Language for Educational Films, Programmed Learning and Educational Technology, Vol. 24, No. 3, p. 234-246, August.

Parsloe, E., 1983, Interactive Video, Epic Industrial Communications, Ltd., Sigma Press.

Pask, G., 1960, The Teaching Machine as a Control Mechanism, Transactions of the Society of Instrument Technology, June, London.

Phillips, C., 1987, Defeating the Object, Personal Computer World, Vol. 10, No. 1, p. 184-189, January.

Polson, M. C. and Richardson, J. J., 1988, Foundations of Intelligent Tutoring Systems, Lawrence Erlbaum Associates, Hillsdale, N.J.

Pountain, R., 1986, New tricks for old mice, Personal Computer World, Vol. 9, No. 110, p. 168-173, October.

Psotka, J., Massey, L. D., Mutter, S. A. (Eds), 1988, Intelligent Tutoring Systems: Lessons Learned, Lawrence Erlbaum Associates.

Quillian, M. R., 1966, Semantic Memory, unpublished Ph.D. dissertation, Carnegie Institute of Technology, Pittsburgh, PA.

Rich, E., 1979, Building and Exploiting User Models, International Journal of CAI, p. 720-722.

Richardson, J. J., 1988, Directions for Research and Applications, in Polson, M. C. and Richardson, J. J. (Eds).

Roach, K., 1984, Interactive Video: The Cardiff Experience, Media in Education and Development, December.

Roberts, F. C. and Park, O. C., 1983, Intelligent Computer Assisted Instruction: An Explanation and Overview, Educational Technology, December, p. 7.

Rockman, S., 1983, Interaction: How Much Do We Really Need? The AIT Video/Computer Project, EITV, January.

Romiszowski, A. J., 1986, Developing Auto-Instructional Materials, Kogan Page Ltd.

Ross, P., 1987, Intelligent Tutoring Systems, Journal of Computer Assisted Learning, Vol. 3, No. 4, December.

Ross, P., Jones, J. and Millington, M., 1986, User Modelling in Intelligent Teaching and Tutoring, paper at Trends in Computer Assisted Education Conference at University of Lancaster, 15th-17th April, Economic and Social Research Council, Blackwells.

Rowntree, D., 1977, Assessing Students: How shall we know them?, Harper and Row.

Rushby, N. J., 1979, An Introduction to Educational Computing, Croom Helm, London.

Sands, M. R. and Hull, R., 1985, Teaching Science: A teaching skills workbook, Focus on Education Series, p. 60, Macmillan Education.

Schank, R. and Abelson, R., 1977, Scripts, Plans, Goals and Understanding, Erlbaum Associates, Hillsdale, N.J.

Schools Council, 1970s, Published Courses: Schools Council Integrated Science Project, Science 5 to 13, Project Technology, Schools Council.

Schools Council, 1975, Working Paper No. 53, Her Majesty's Stationery Office.

Seidel, R. J. and Rubin, M. L. (Eds), 1977, Computers and Communications: Implications for Education, Academic Press.

Seidner, C. J., 1984, User Preference re: CBI Menus Choice Mechanisms, Digital Equipment Corporation Educational Services.

Self, J. A., 1974, Student Models in Computer-Aided Instruction, International Journal of Man-Machine Studies, Vol. 6, p. 261-276.

Self, J., 1985, A perspective on intelligent computer-assisted learning, Journal of Computer Assisted Learning, Vol. 1, p. 159-166.

Self, J., 1987, IKBS in Education, Education Review, Vol. 39, No. 2.

SERC/DoI, 1983, Intelligent Knowledge-Based Systems: a programme for action in the UK, Science and Engineering Research Council / Department of Industry, London.

Shayer, M., 1972, Some aspects of the strengths and limitations of Piaget's Developmental Psychology to the planning of secondary school science courses, unpublished M. Ed. thesis, University of Leicester.

Shayer, M. and Adey, P., 1981, Towards a Science of Science Teaching, Heinemann Educational.

Shortliffe, E., 1976, Computer Based Medical Consultations: MYCIN, American Elsevier, NY.

Simons, G. L., 1984, Introducing Artificial Intelligence, NCC Publications.

Skinner, B. F., 1954, The Science of Learning and the Art of Teaching, Harvard Educational Review, Vol. 24, p. 86-97.

Sleeman, D., 1982, Assessing aspects of competence in basic algebra, in Sleeman, D. and Brown, J. S. (Eds).

Sleeman, D., 1987, PIXIE, A Shell for Developing Intelligent Tutoring Systems, in Lawler, R. W. and Yazdani, M. (Eds).

Sleeman, D. and Brown, J. S., 1982, Intelligent Tutoring Systems, Academic Press. (Computers and People Series, Ed. Gaines, B. R.)

Sleeman, D. and Hendley, R. J., 1982, ACE: A system which analyses complex explanations, in Sleeman, D. and Brown, J. S. (Eds).

Sleeman, D. and Ward, R. D., 1988, Intelligent Tutoring Systems in Training and Education: Prospects and Problems, paper at British Computer Society Expert Systems Conference, Brighton.

Sloman, A., 1983, Intelligent Systems: a Brief Overview, for Science and Engineering Research Council study of Architectures of IKBS, Workshop 3, 16-17 March.

Snowberry, K., Parkinson, D. and Sisson, N., 1983, Computer Display Menus, Ergonomics, Vol. 26, No. 7, p. 699-712.

Soloway, E. and Vanlehn, K., 1985, A.I. and Education, Tutorial No. 10 at 9th A.I. Conference, Carnegie-Mellon, USA.

Soloway, E., et al, 1983, Meno-II: An AI-Based Programming Tutor, Journal of Computer Based Instruction, Vol. 10, Nos. 1 & 2, p. 20-34.

Somekh, B., 1986, Action Research and the Micro, in Classroom Action Research Network (CARN) Special Bulletin, Cambridge Institute of Education, Cambridge.

Stansfield, J., Carr, B., and Goldstein, I., 1976, Wumpus Advisor I: A first implementation of a program that tutors logical and probabilistic reasoning skills, MIT AI Lab. Memo No. 381.

Steinberg, E., 1977, Review of student control in computer-assisted instruction, Journal of Computer Based Instruction, Vol. 3, p. 84-90.

Stenhouse, L. (Ed), 1980, Curriculum Research and Development in Action, Heinemann Educational.

Stevens, A., Collins, A. and Goldin, S.E., 1982, Misconceptions in students' understanding, in Sleeman, D. and Brown, J. S. (Ed).

Stevens, A. and Collins, A., 1977, The Goal Structure of a Socratic Tutor, Proc. of the Annual Conference, Seattle Assoc. for Computer Machinery, Bolt, Beranek and Newman.

Stott, D. H., 1979, The Teaching Challenge of Mixed Ability, British Journal of Teacher Education, Vol. 5, No. 2, p. 62.

Suppes, P., 1967, Some theoretical models for mathematics learning, Journal of Research and Development in Education, Vol. 1, p. 5-22.

Sutton, C. (Ed), 1981, Communicating in the Classroom, Hodder and Stoughton.

Sutton, C. R. and Haysom, J. T. (Eds), 1974, The Art of the Science Teacher, McGraw Hill.

Taylor, T. D., 1987, A Description of IBM's Learning System/1, Optical Information Systems, p. 282-288, July/August.

Tsai, S. Y. and Pohl, N., 1978, Student Achievement in Computer Programming; Lecture Versus Computer Aided Instruction, University of Santa Clara.

Turing, A. M., 1950, Computing Machinery and Intelligence, Mind, Vol. LIX, No. 236.

Uhr, L., 1969, Teaching Machine programs that generate problems as a function of interaction with students, Proceedings of the 24th National Conference, p. 125-134.

VanLehn, K., 1983, Felicity conditions for human skill acquisition : Validating an A.I. based theory, Xerox Parc Technical Report CIS - 21, Palo Alto, CA.

VanLehn, K., 1988, Student Modelling, in Polson, M. C. and Richardson, J. J. (Eds).

Wenger, E., 1987, Artificial Intelligence and Tutoring Systems, Morgan Kaufmann Publications Inc., CA.

Whitten, P., 1982, Videodisc: Slow Motion Revolution? Corporate Video, May.

Williams, K., 1984, Interactive Videodisc at the Open University, Media in Education and Development, December.

Winston, P. H. (Ed), 1975, The Psychology of Computer Vision, McGraw-Hill, NY.

Winston, P. H., 1977, Artificial Intelligence, Addison-Wesley, NY.

Weisenbaum, J., 1966, ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine, Communications of the ACM, Vol. 9, No. 1, January.

Yazdani, M., 1984, Knowledge Engineering in PROLOG, in Forsyth, R. (Ed).

Yazdani, M., 1988, Intelligent Tutoring Systems: An Overview, in Lawler, R. W. and Yazdani, M. (Eds).

Yazdani, M. and Narayan, N., (Eds), 1984, Artificial Intelligence: Human Effects, Ellis Horwood.

Young, M.F.D. (Ed), 1971, Knowledge and Control, Collier-Macmillan.

Appendices

Appendix 1.1

ITS and Other Systems Referred to in the Thesis

Categories:		
	ITS	Intelligent Tutoring System
	AI	Artificial Intelligence System/Prototype
	ES	Expert System
	CAL	Computer Assisted Learning Project
	IV	Interactive Video System

In rough chronological order. Dates are those of first report, where known. Some projects are still continuing.

SAKI (1964)
Pask, Gordon.
USA
CAL tutor for typists and punchcard operators.
See Lewis and Pask, 1964.
Self-organising automatic keyboard instructor. Adaptive system which changes the difficulty of keyboard exercises as the student progresses. Deals with 'sensori motor skills' area.

ELIZA (1966)
Weizenbaum, Joseph
MIT, MA, USA.
AI program for general conversation.
See Weizenbaum, 1966.
Conducts a realistic dialogue by means of 'semantic trickery'. Throws back user's comments; reacts to trigger subjects; gives random, standard responses; recalls earlier topics.

ADAPTIVE TUTORS (1967)
Suppes, Patrick
Stanford University, CA, USA.
ITS generative tutors for mathematics.
See Suppes, 1967; Goldberg and Suppes, 1972.
Early AI work. Successive models have evolved to understand more successfully the validity of students' proofs. Examples generated at different levels of difficulty. See later EXCHECK, which evolved from this work.

GPS (1969)
Newell, Allen and Simon, Herbert.
Carnegie-Mellon University, CA, USA.
ES for heuristic search experiments.
See Ernst and Newell, 1969.
'General Problem Solver'. Used depth-first search of data-base of rules, etc, to solve theorems, then chess problems. Best-first was used later. Pioneer expert system.

SCHOLAR (1970)
Carbonell, Jaime
Bolt, Beranek and Newman, Camb. Mass., USA.
ITS dialogue tutor for geography.
See Carbonell, 1970.
Dialogue system using a semantic network to build up a student's knowledge by question and answer. Called 'mixed initiative' CAL, also uses the term information structure oriented, ISO.

DENDRAL (1971)
Feigenbaum, Edward and others.
Stanford University, CA., USA.
ES for mass spectrogram interpretation.
See Feigenbaum, 1971.
Chemistry ES which applied rule-based problem solving to real-world situations.

EXCHECK (1972)
Blaine, Lee; Smith, Robert L; supervisor Suppes, Patrick.
USA, Stanford University, CA.
ITS whole-course tutor for logic and set theory.
See Blaine and Smith, 1977; Suppes, 1981.
Students constructed maths proofs, and EXCHECK checked them so that all homework and exams were done interactively in real time. Associated book covers the instruction. From Suppes' early work.

SHRDLU (1972)
Winograd, Terry.
USA, Stanford University.
AI natural language robot program.
See Winograd, 1972.
Program manipulated simulated wooden blocks in response to natural language instructions, and explains what it is doing. An exercise in transparency.

KIMBALL'S TUTOR (1972)
Kimball, Ralph and Smallwood, Richard.
USA, Stanford University, CA.
ITS self-improving tutor for symbolic integration.
See Kimball, 1982.
The tutor conducted a question and answer dialogue, accepting 'symbolic heuristics' as the student suggested steps towards a solution to an archive-selected example. Allowed student to follow poor paths. No student model. Tutor acquired better strategies from the students.

LOGO (1973)
Papert, Seymour.
USA, MIT, MA.
CAL programming environment.
See Papert, 1980.
Language devised in 1967, developed by Papert as one of the first environments for learning. Claims to be a training in logic, extendable to other areas. Does not address the problem of teaching relevant practical information and skills.

BIP (1975)
Atkinson, R.C., Beard, M. and Barr, A.
USA, Stanford University, Calif.
ITS programming tutor for BASIC
See Barr et al, 1975 and Goldstein and Carr, 1977.
A student model is maintained of the student's familiarity with various skills, and the next task posed to the student is generated on the basis of which skills are currently known, according to a tutorial strategy.

SOPHIE I, II, III, etc. (1975)
Brown, J.S. and Burton, R.R.
USA, Bolt, Beranek and Newman, Camb. Mass.
ITS problem solving environment for electronics amplifier.
See Brown and Burton, 1975.
Reactive evironment extending SCHOLAR's mixed-initiative CAL. Uses natural

language parser system. Student finds a fault in a circuit by question and answer. No student model.

MYCIN (1976)
Shortliffe, E.H.
USA, Stanford University, CA.
ES for medical diagnosis.
See Shortliffe, 1976.
Rule-based, gives reasons for its choice. Highly influential ES, perhaps because the domain is most understandable. Many imitators and extensions. EMYCIN (empty MYCIN) was an early ES shell. GUIDON was based on EMYCIN.

SMALLTALK (1976)
Learning Research Group, Xerox.
USA, Xerox Parc Palo Alto Res. Centre, CA.
CAL programming environment.
See Goldberg et al, 1982.
LOGO-type environment for educational development, covering programming, graphics, music and other subjects. Includes TRIP to help with algebra problems, and others.

WUSOR-1, WUSOR-2 (1976)
Stansfield, J., Carr, B. and Goldstein, I.P.
USA, MIT, Mass.
ITS student modelling coach for exploration game.
See Stansfield et al, 1976 and Goldstein, 1982.
Built on the game 'Hunt the WUMPUS', devised by Yob, 1970. Employs genetic graphs with rules as nodes and relationships as links. Nodes are procedural skills, links are evolution of learning. Enables student misconceptions to be represented.

TICCIT (1976)
MITRE Corp, for National Science Foundation of America.
USA, Brigham Young University, WA.
CAL whole-course system for several subjects.
See Mitre Corp, 1976; Bunderson, 1974.
'Time-shared Interactive Computer Controlled Information Television'. Project to investigate the feasibility of CAL on a large scale at less cost than traditional teaching. Based on two minis with up to 128 terminals. Calculus and English initially. Material from six experts. Still used, but not widely.

PLATO (1976)
National Science Foundation of America.
USA, Illinois University, IL.
CAL supplementary system for several subjects.
See Alpert, 1975,; Murphy and Appel, 1977.
'Programmed Logic for Automated Teaching Operation.'Project to start up large scale CAL, using time-sharing on up to 1000 terminals, supplying 'rich learning environments', with courseware produced by teachers. Optional use for teachers and students. Still used, but not widely.

WEST (1976)
Burton, R.R. and Brown, J.S.
USA, Bolt, Beranek and Newman, Mass.
ITS problem solving coach for arithmetic game
See Burton and Brown, 1976.
The tutor or expert uses its knowledge of the domain together with a student model to decide what to say and when. Uses the PLATO game 'How the West was Won'.

WHY (1977)
Stevens, Albert and Collins, Allan.
USA, Bolt, Beranek and Newman, Camb. Mass.
ITS dialogue tutor for meteorology (rainfall).
See Stevens and Collins, 1977, Stevens et al, 1982
Script based, debugging, dialogue approach. Takes student goals as main structural component.

SITS (1977)
O'Shea, Tim
UK, Leeds University.
ITS self-improving tutor for quadratic equations.
See O'Shea, 1982.
'Self-Improving Teaching System', but also referred to as O'SHEA'S TUTOR. Teaches quadratic equations by inspection. Evaluated with 30 students. Uses production rules to change its strategy according to success of student responses.

ACE (1978)
Sleeman, D.H. and Hendley, R.J.
UK, Leeds University.
ITS problem solving environment for spectroscopy.
See Sleeman and Hendley, 1978
'Problem solving monitor.' Helps the student to interprete nuclear magnetic resonance spectrographs. No student model. Tells student if incorrect, explains why last try was wrong, asks for a natural language explanation and comments on it.

MACSYMA ADVISOR (1978)
Genesereth, Michael R.
USA, Stanford University, Calif.
ITS diagnostic tutor for mathematical problems.
See Genesereth, 1982.
Performs operations on maths expressions a student types in, in response to simple commands. Deduces the student's plan of action from the steps carried out, and hence diagnoses misconceptions or bugs. No student model.

PROSPECTOR (1978)
Duda, Richard; Gaschnig, John and Hart, Peter.
USA, SRI International.
ES for geological exploration.
See Duda et al, 1980.
Made news in 1982 when it was right about molybdenum deposits in Washington State, and human geologists were wrong. Uses rule models. Contributed to making ESs acceptable.

BUGGY (1978)
Brown, J.S. and Burton, R.R.
USA, Xerox Palo Alto Res. Centre, Calif.
ITS diagnostic system for arithmetic.
See Brown and Burton, 1978.
System for student modelling without any tutoring. Uses a procedural network. Informs teacher or student what he or she is doing wrong. Extended in DEBUGGY.

GUIDON (1979)
Clancey, William J.
USA, Stanford University.
ITS tutor based on ES for medical diagnosis.
See Clancey, 1982.

Derived from MYCIN. Student plays role of physician, questions on case studies compared with those of MYCIN. Uses tutoring rules to guide a discussion. Heavily criticised.

GRUNDY (1979)
Rich, Elaine
USA, Carnegie-Mellon University, PA.
HCI user modelling system.
See Rich, 1979; Ross, 1986.
Advises a user on the choice of a suitable library book. Has stereotype readers represented by hierarchy of frames to make up a user stereotype, and can learn to modify stereotypes.

LMS (1981)
Sleeman, D.H. and Smith, M.J.
UK, Leeds University.
ITS diagnostic system for algebraic equation solving.
See Sleeman and Smith, 1981, Sleeman, 1982.
Produces a student model without any tutoring. Evaluated as successful when the model gives same answers as student does. Uses production rules to generate hypotheses to form the student model.

SNIFFER (1981)
Shapiro, D.
USA, Yale University.
ITS programming system for PROLOG.
See Shapiro, 1981; Murray , 1985.
Examples offered, user supplies info, "bugs are corrected by synthesising correct clauses or by searching among perturbations of buggy clauses." Queries the user and thus resembles a tutor, but no student model.

TRIP (1981)
Gould, L and Finzer, W.
USA, Xerox Parc Palo Alto Res. Centre, CA.
CAL program for algebra.
See Gould and Finzer, 1981.
Written in SMALLTALK. Uses a WIMP system. Generates problems, teacher can adjust parameters for different students. No student model or subject understanding, but part of an 'environment' type of approach.

DEBUGGY (1982)
Burton, R.
USA, Xerox Palo Alto RES. Centre, Calif.
ITS diagnostic tutor for arithmetic (subtraction).
See Burton, 1982.
Builds on BUGGY's detection of bugs, and hypothesises to diagnose them. IDEBUGGY interacts with the student to establish and remove the bug (INTERACTIVE DEBUGGY). Generates test problems for the student. Tested with many students.

SPADE-0, 1 and 2 (1982)
Miller, M.L. and Goldstein, I.P.
USA, MIT, Calif.
ITS programming environment for turtle LOGO.
See Miller, 1982 and Ford, 1984.
Contains a design model. For the teaching and learning of structured planning and debugging of computer programs.

CPR PROJECT (1982)
Hon, David
USA, American Heart Association, Dallas, TX.
IV system for cardio-pulmonary resuscitation.
See Hon, 1982.
Uses a mannikin with sensors connected, and two TV screens, one to give instruction from a videodisc, the other interaction via a computer. Expensive, but claimed to be highly successful and cost-effective as well as saving lives.

SIERRA (1983)
VanLehn, Kurt.
USA, Xerox Parc Palo Alto Research Centre, CA.
ITS problem solving tutor for arithmetic.
See VanLehn, 1983.
Large AI based program in several phases, based on Brown's Repair Theory and VanLehn's Step Theory, involving 'felicity conditions'. Elaborate formulation of some conventional teaching practice. Took 100 hours of processing time per lesson in LISP.

MENO-II (1983)
Soloway, Elliot; Rubin, Eric; Woolf, Beverly; Bonar, Jeffrey
USA, Yale University.
ITS programming environment for PASCAL.
See Soloway et al, 1983.
Meno was a Plato discourse. Has a database of 18 common bugs as templates, and on finding a mismatch with the student's effort, gives remedial instruction using a network of tutoring rules.

ATDSE (1983)
Attisha, M. and Yazdani, M.
UK, Exeter University
ITS diagnostic tutor for arithmetic subtraction errors.
See Attisha and Yazdani, 1983.
'Automated Teacher for Diagnosing Subtraction Errors'. Knowledge based system with a control program that identifies numbered errors in a store. Generates random problems, but apparently no task difficulty model. Gives help. Appears to be for 8-bit, 32k, BBC type system, in Pascal. To be extended to multiplication.

LISP TUTOR (1984)
Elsom-Cook, Mark.
UK, Warwick University.
ITS programming tutor for LISP
See Elsom-Cook, 1984.
Stand-alone tutor. The paper describes the design considerations, concentrating mainly on the student model.

SPIRIT (1984)
Barzilay, A. and Pople, H.E.
USA, Pittsburgh University, PA.
ITS dialogue tutor for probability theory.
See Barzilay and Pople, 1984.
Acts as a tutor that mostly observes without interference, intervening when things go wrong, i.e. a monitor. Also holds a question and answer type dialogue. Constructs a student aptitude model. Has grown 'evolutionally'. Sponsored by US navy.

TEDDY BEAR DISC (1984)
Laurillard, Diana and others at BBC and Open University
UK, Open University, Milton Keynes
IV disc for materials technology.
See Laurillard, 1985.
Uses a dramatised court case situation to teach metallurgy and materials science. Simple video-question and answer technique, with considerable degree of student control. Uses existing BBC program material.

PEEKAPS I (1984)
Ferguson, David L.
USA, New York University, NY.
ITS front-end tutor for engineering problem solving.
See Ferguson, 1984.
Aims to be a generalised front-end for various environments or 'microworlds'. Based on Papert's theory of student-oriented learning. In two main parts, a knowledge structure and a task analyser.

TUTOR (1984)
Davies, N.G.; Dickens, S.L. and Ford, L.
UK, Cambridge, Logica UK Ltd.
ITS front-end tutor for Highway Code (traffic lights).
See Davies et al, 1985; Ford, 1986.
Aims to be a front-end for different knowledge bases. Based on GUIDON and SOPHIE. Student model records last three interactions. Has a natural language interface with 19 sentence types. Multi-tasking teaching strategy module. On VAX in PROLOG.

SS ELECTRONICS DISC (1985)
EPIC/EETPU (Director Eric Parsloe).
UK, EPIC Industrial Ltd, London.
IV on solid state electronics.
See Epic, 1985 (sales literature)
Designed for conventional branching use at Level 1, or under computer control at Level 3. Domain is highly structured, therefore useful in a prototype for a whole course.

TALUS (1985)
Murray, William R.
USA, Texas University, TX.
ITS programming system for LISP.
See Murray, 1985.
Acts as a 'domain expert'. A complete ITS tutor "would include a student model, a dialogue manager, courseware and additional expertise."

MIZAR (1985)
Blair, Howard and Trybulec, Andrzej
USA, Connecticut University, CT.
AI reasoning system for logic.
See Trybulec and Blair, 1985.
Takes a traditional approach to instruction in logic, based on theorems. Not a tutor, but students can submit their proofs to MIZAR for checking. On VAX under UNIX. Used in Maths teaching. Proofs highly readable.

ANDERSON'S LISP TUTOR (1985)
Anderson, John R; Reiser, Brian J; and Farrell, Robert G.
USA, Carnegie-Mellon University, PA.
ITS programming tutor for LISP.
See Reiser et al, 1985.
Based on Anderson's ACT (Adaptive Control of Thought) learning theory. A number of routine features (eg. immediate error feedback). Also claims to represent the goal structure of the problem solving, and prevent low-level error 'noise'. In use with students.

THE GEOMETRY TUTOR (1985)
Anderson, John R; Boyle, C. Franklin; and Yost, Gregg.
USA, Carnegie-mellon University, PA.
ITS diagnostic tutor for geometry.
See Anderson et al, 1985.
Three-part system: tutor, interface and 'ideal and buggy rules', IBR. The IBR contains correct and incorrect rules for matching. Uses an ideal rule model, but does not talk of a student model. Uses ACT (Advanced Computer Tutoring) production rule system.

CIRCUIT TUTOR (1985)
Joobbani, Rostam and Talukdar, Sarosh N.
USA, Carnegie-Mellon University, PA.
ITS diagnostic tutor based on ES for circuit analysis.
See Joobbani and Talukdar, 1985.
An ES that can examine the expressions students write, diagnose errors in them and suggest corrections. Uses divide-and-conquer, forward chaining and hypothesise and test. Uses expert opinion from lecturers at CMU.

UNIX USER MONITOR (1986)
Ross, peter and others
UK, Edinburgh University.
HCI diagnostic system for learning UNIX.
See Ross et al, 1986.
Analyses the way a user sets about learning UNIX, so as to stop him if he goes wrong. Monitor system. Uses a 'blackboard' interface.

Appendix 1.2

(i) Some arguments for Computer Assisted Learning (CAL) Systems

1. They can provide, in micro-based systems, INDIVIDUAL teaching where students have the machine which acts as their tutor all to themselves.

2. In a classroom, a student is one of many requiring the teacher's attention, but with a computer the student receives all its ATTENTION.

3. An electronic system can give INSTANT FEEDBACK. The student does not have to wait days or weeks for a human teacher to mark work.

4. A teaching system can be more ACCURATE and RELIABLE in its marking and record keeping than a human teacher.

5. Assessment and marketing in such a system is totally OBJECTIVE.

6. A teaching system gives STUDENT CONTROL in the learning process.

7. The student can choose how much TIME is spent on a topic repeating work as often as required.

8. The system can use frequent CONTINUOUS ASSESSMENT.

9. The student can receive PROGRESS REPORTS as required, and can thus adjust his or her own level of effort.

10. The system can, if using video techniques, use GOOD TEACHERS AND PRESENTERS at all times.

11. Electronic systems can use NEW METHODS of instruction which are simply not available otherwise, e.g. simulations.

Appendix 1.2

(ii) Arguments against Computer Assisted Learning (CAL) Systems (Based on O'Shea, 1983.)

1. They cannot converse in NATURAL LANGUAGE,except in a limited way. With most systems students have to respond to the system with one word, or menu type answers. Systems such as Carbonell's (1970) SCHOLAR and Brown and Burton's (1975) SOPHIE have attempted to solve this problem.

2. They lack FEASIBILITY, in the sense that they cannot deal with unanticipated, or unprogrammed, responses. Systems such as Weisenbaum's (1966)ELIZA and Bobrow's (1977) GUS have tackled this problem.

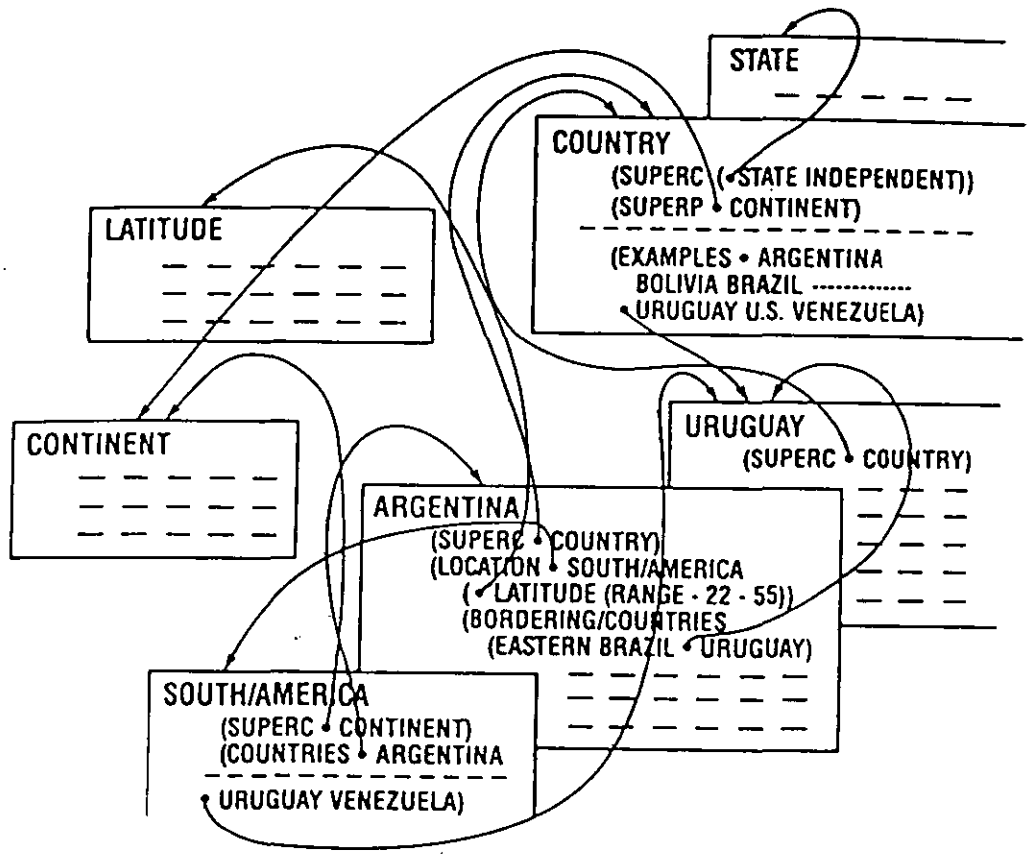
3. In general they cannot understand the NATURE OF MISTAKES, or analyse mistakes and act accordingly. However, some intelligent systems have attempted to identify and diagnose student mistakes, for example Brown and Burton's (1982) DEBUGGY system.

4. They cannot in general PROFIT FROM EXPERIENCE with students, or experiment with the teaching strategy. Again, some intelligent systems have attacked this aspect, notably O'Shea's (1977) self-improving quadratic tutor, SITS.

5. Some non-educational disadvantages are LIMITED INFORMATION HANDLING, SHORTAGE OR SOFTWARE AND EXPENSE OF HARDWARE. These limitations are disappearing as the technology advances.

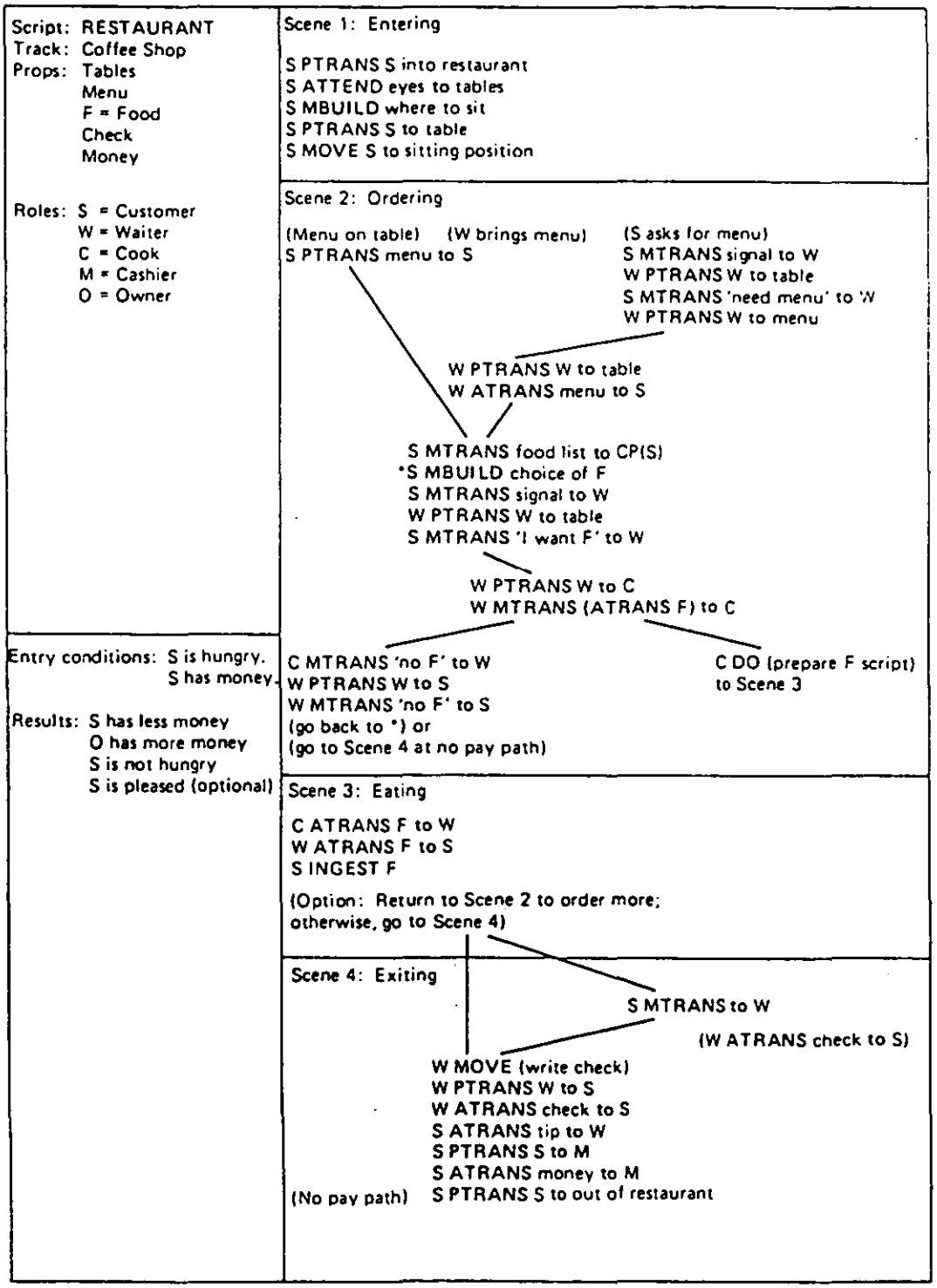
Appendix 2.1

Semantic network used by Carbonell (1970) in SCHOLAR



Appendix 2.2

Example of a 'restaurant' script (Schank, 1977)



Appendix 3.1

Acronyms in the field of computer assisted learning (CAL)

CAL	Computer Assisted Learning
CAI	Computer Aided Instruction
CBL	Computer Based Learning
CBI	Computer Based Instruction
CBT	Computer Based Training
CML	Computer Managed Learning
CMI	Computer Managed Instruction
CME	Computer Managed Education
CET	Computer Enhanced Training
ICAL	Intelligent Computer Assisted Learning
ICAI	Intelligent Computer Aided Instruction
ITS	Intelligent Tutoring Systems
IV	Interactive Video

Appendix 3.2

Some Comments on the Potential of Interactive Video

"The enormous potential of the combined media for educational applications is yet to be fully explored." (Laurillard, 1983.)

"The Department of Trade and Industry wishes to encourage the rapid development of interactive video expertise, application and experience in the UK. This will be based mainly, but not exclusively, on the new technology of videodiscs." (Hills, DTI, 1984.)

"This technology offers a tremendous range of educational applications to the imaginative teacher." (Laurillard, 1984.)

"Interactive video has the advantages of both these technologies [computing and television], offering in a single medium an unique blend of visual and textual information and computer assisted learning techniques. This convergence of technologies promises a new and flexible tool capable of providing the learner with a rich variety of facilities to support a wide range of teaching and training needs." (Bayard-White, CET, 1985.)

(For review articles on IV, see Fox, 1982 and 1984, Laurillard, 1984, Clark, 1984, and Doulton, 1988. For comprehensive surveys, see Duke, 1983, and Parsloe, 1983.)

Appendix 3.3

Description of the preliminary project

The system used

The HARDWARE of the system consisted of an Apple IIe computer, with two disc drives and a printer, controlling an industrial tape recorder through a switching unit or interface (see Figure 3.5). There were originally two colour monitors connected - due to peculiarities of the equipment one monitor would not show computer screens in colour, and the other would not show video sequences. The use of two screens was found to be distracting, and only one was used, sacrificing colour in the computer output.

An initial problem was that there was so much equipment on the desk top that the monitor was on a stand about 0.5m above the computer keyboard. This was found to be much too high for comfort, and the stand was lowered, highlighting the problems of ergonomic design of the workstation. The complexity of equipment can be daunting to a student, and as a general principle it might be best to keep visible equipment to a minimum.

The SOFTWARE of the system consisted of the IVL (Interactive Video Learning) system originated by Dalroth Computer Products Ltd., which cost #1,220 including VAT in 1984. It came on an Authoring Disc used for writing the programs and a Course Disc used for priming student discs, and included a BCD interface card for the Apple and a substantial manual.

Also bought with the system were a tutorial disc and workbook, and a development disc and workbook, at a cost of #295. The tutorial disc contains a computer program using the IVL system which duplicates the workbook - this is "designed to help you produce training courses or teaching programs more efficiently", and is really a textbook on course production as approached in the USA. The development disc contains an IVL program to help a teacher design a particular course, and again it duplicates the workbook. It helps the teacher to decide on a target audience, on objectives, on testing methods and so on, then gives a printout of them if required at the end. (See Appendix 3.4.)

The program produced

For the experimental program the subject "Radioactivity and Atomic Physics" was chosen, a substantial but fairly self-contained topic, occupying one to three chapters in the average Physics O-level textbook, and of interest to non-students and non-scientists. The target audience was regarded as an O-level student or an interested, intelligent adult learner. The arrangement of units within the subject is shown in Appendix 3.5.

The IVL system required the program to be written in segments, grouped together in blocks. Segments can be of two types, tests and presentations. Tests are of four types, true or false, missing words, multiple choice or free answer. Presentations are of five types, video scenes, computer text, graphic text, printout or student choice. All of these segment types were explored in the experimental program except graphics and printouts, which would have greatly increased the amount of time to be spent on the system.

It would be a useful facility for the student to be able to obtain printouts of some of the screens in the course, which was not possible. It would also be useful to him or her to obtain a printout of test results at the end, but although the teacher can obtain these results using an extra control program, it is not possible for the student to obtain a printout.

The blocks and segments used are indicated in the IVL printout in Appendix 3.6. The video scenes used are indicated in the printout of Appendix 3.7. The program used 86 segments out of a maximum of 90, 16 blocks out of a maximum of 20, and 49 video scenes out of a maximum of 50. The number of segments per block varied from 1 to 15, the maximum being 15. The program thus tested the system near to its limits.

Taking the main blocks of the program in order, the Radioactivity Unit showed a film made by the Educational Foundation for Visual Aids in 1963 called "The Discovery of Radioactivity", broken down into short sequences and tested after each sequence. The Atoms Unit did the same with another EFVA film made in 1958 called "Conquest of the Atom". The films had been transfered to videotape by a simple process involving a projector and a television camera, and were edited onto the videotape to be used with the program using two videotape recorders. The images were thus 'third hand', but on a small screen were found to be of acceptable quality.

The Experiments Unit was based on an ITV programme in the series Physics in Action. Three experiments were shown, one of which involved drawing a graph (on graph paper) to calculate half life from an experiment shown. This programme was recorded 'from the air' and was edited onto the program tape using two videotape recorders. The fourth section, also 'off air', showed some applications of radioactivity. The Summary Notes Unit gave a choice of four information sequences, on Elements and Compounds, on Atoms and Atomic Particles, on Alpha, Beta and Gamma Rays, and on Isotopes. These sequences showed stills of pictures and diagrams with voice over, followed by summary screens of computer text. The sequences were made specially using a television camera. The Questions Unit contained a short test of five multiple choice (MC) questions. This could be extended to contain more MC questions if required.

It was found possible to write even an extensive course of this kind relatively quickly using the IVL authoring system. The whole project took just a few weeks for a single worker (with the help of a presenter to link sequences), including the shooting and editing of the tape material.

Appendix 3.4

Example of IVL assistance in program development

Course: RADIOACTIVITY

Organisation: LEICESTER POLYTECHNC

Reference: 250

Characteristics of Target Audience

WELL MOTIVATED
INTERESTED ADULTS OR O-LEVEL STUDENTS
ANY AGE
SOME KNOWLEDGE OF SCIENCE
WILLING TO SPEND TIME
ANY JOB, PAST OR FUTURE
PREFERENCE FOR INDEPENDENT LEARNING
SCIENTIFIC OR ECOLOGICAL INTERESTS

Conditions for Entry

SHOULD HAVE GOOD ENGLISH
O-LEVEL LEARNING STANDARD
SOME (NOT MUCH) KNOWLEDGE OF SCIENCE
INTERESTED AND MOTIVATED
NO SPECIAL RESOURCES REQUIRED

Overall Course Aims

THE AIMS OF THE COURSE ARE:

1 GIVE A QUQLITATIVE KNOWLEDGE OF RADIOACTIVITY

2 ENCOURAGE SOME UNDERSTANDING AND APPRECIATION OF SCIENCE

3 PROVIDE SOME BACKGROUND FOR THE PROBLEMS AND BROADER IMPLICATIONS OF THE SUBJECT

4 PROVIDE A COURSE IN A RATHER OBSCURE SUBJECT THAT A STUDENT CAN FOLLOW SLOWLY AND CAREFULLY AT HIS OWN PACE

5 PROVIDE A SELF-STUDY COURSE OF A SIMPLE, QUALITATIVE TYPE FOR PEOPLE WHO REQUIRE A KNOWLEDGE OF RADIOACTIVITY, EITHER FOR INTEREST OR SELF-ADVANCEMENT

Specific Learning Objectives

AT THE END OF THE COURSE STUDENTS WILL BE ABLE TO:

1 RECALL A LIMITED BODY OF FACTS CONCERNING RADIOACTIVITY

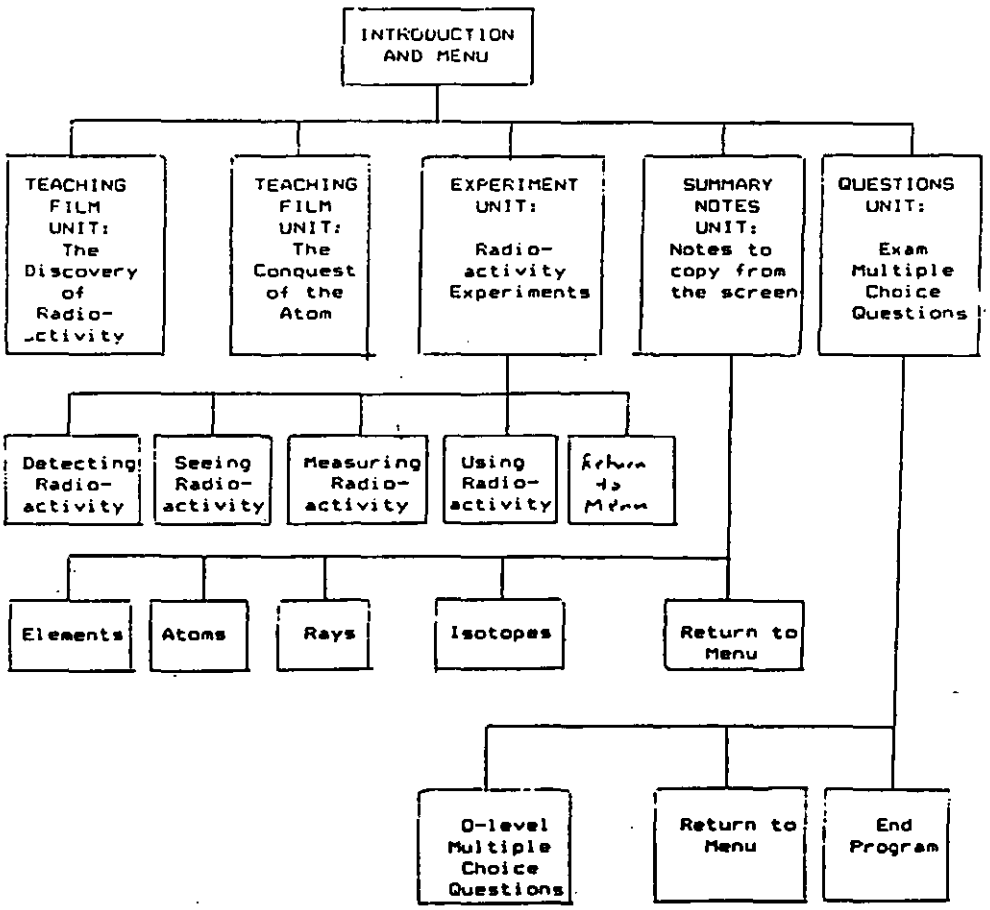
2 UNDERSTAND SOME OF THE CONCEPTS IT EMBRACES

3 DECIDE FOR THEMSELVES UPON ISSUES ARISING FROM THE SUBJECT MATTER, E.G. UPON THE DANGERS INVOLVED IN HANDLING RADIOACTIVE MATERIALS

4 ANSWER QUALITATIVE QUESTIONS ON RADIOACTIVITY, OF THE TYPE SET BY O-LEVEL EXAMINATION BOARDS

Appendix 3.5

Units in the Radioactivity and Atomic Physics preliminary IV/CAL project



Appendix 3.6

Blocks and segments in the preliminary IV/CAL project

Course name: RADIOACTIVITY

Certificate: Yes
Graphics on disk 2: No
Time Kit: No
Free Choice option: No
Opening Segment Number:1

Block	Block Name	First Seg	Next Logical Block
1	INTRODUCTION	1	2
2	CHOICE	3	14
3	RADIOACTIVITY	4	2
4	CRAD	21	2
5	ATOMS	36	2
6	BATOM	40	2
7	CATOM	47	2
8	EXPERIMENTS	64	2
9	DETECTING	66	8
10	SEEING	68	8
11	MEASURING	70	8
12	USING	73	8
13	SUMMARY	75	2
14	QUESTIONS	84	2
15	MC QS	86	14
16	BRAD	14	2

No	SegType	TestType	TestOf	Block	Sc	Branching 1	2	3	4	5	Title
1	Text			1		2					
2	Video			1	1	3					
3	Stud.Choice										
4	Free Answer	Pre/Post	3	3		3	5	5			
5	Video			3	2	6					
6	Video			3	45	7					
7	Free Answer	Ordinary		7		14	7	21			
8	Video			3	3	9					
9	Video			3	42	10					
10	Free Answer	Ordinary		3		12	11	11			
11	Video			3	15	15					
12	Video			3	12	13					
13	Free Answer	Post	3	3		3	3	3			
14	Video			16	4	15					
15	Video			16	43	16					
16	Free Answer	Ordinary		16		18	16	17			
17	Video			16	15	27					
18	Video			16	44	19					
19	Free Answer	Ordinary		16		12	19	20			
20	Video			16	15	32					

Appendix 3.7

Video scenes used in the preliminary IV/CAL project

INTERACTIVE VIDEO LEARNING
********** ***** ********

VIDEO LOGGER SCENE DETAILS

Scene	Start Frame	Start Time	End Frame	End Time	Length Frames	Length Time
1	425	0:00:17.00	1412	0:00:56.48	987	0:00:39.48
2	1638	0:01:05.52	2112	0:01:24.48	474	0:00:18.96
3	2429	0:01:37.16	2603	0:01:44.12	174	0:00:06.96
4	2807	0:01:52.28	2910	0:01:56.40	103	0:00:04.12
5	3177	0:02:07.08	3346	0:02:13.84	169	0:00:06.76
6	4457	0:02:58.28	7563	0:05:02.52	3106	0:02:04.24
7	7581	0:05:03.24	10842	0:07:13.68	3261	0:02:10.44
8	10865	0:07:14.60	14375	0:09:35.00	3510	0:02:20.40
9	14393	0:09:35.72	16670	0:11:06.80	2277	0:01:31.08
10	16700	0:11:08.00	18514	0:12:20.56	1814	0:01:12.56
11	18544	0:12:21.76	25613	0:17:04.52	7069	0:04:42.76
12	26176	0:17:27.04	26402	0:17:36.08	226	0:00:09.04
13	26588	0:17:43.52	26773	0:17:50.92	185	0:00:07.40
14	26935	0:17:57.40	27093	0:18:03.72	158	0:00:06.32
15	27205	0:18:08.20	27390	0:18:15.60	185	0:00:07.40
16	27556	0:18:22.24	28014	0:18:40.56	458	0:00:18.32
17	28264	0:18:50.56	28511	0:19:00.44	247	0:00:09.88
18	29898	0:19:55.92	35130	0:23:25.20	5232	0:03:29.28
19	35151	0:23:26.04	39757	0:26:30.28	4606	0:03:04.24
20	39792	0:26:31.68	44010	0:29:20.40	4218	0:02:48.72
21	44033	0:29:21.32	47697	0:31:47.88	3664	0:02:26.56
22	47748	0:31:49.92	53232	0:35:29.28	5484	0:03:39.36
23	53255	0:35:30.20	58539	0:39:01.56	5284	0:03:31.36
24	58561	0:39:02.44	58678	0:39:07.12	117	0:00:04.68
25	59332	0:39:33.28	59497	0:39:39.88	165	0:00:06.60
26	59655	0:39:46.20	59825	0:39:53.00	170	0:00:06.80
27	60011	0:40:00.44	60209	0:40:08.36	198	0:00:07.92
28	60412	0:40:16.48	60630	0:40:25.20	218	0:00:08.72
29	60819	0:40:32.76	61771	0:41:10.84	952	0:00:38.08
30	61940	0:41:17.60	62929	0:41:57.16	989	0:00:39.56
31	63100	0:42:04.00	63645	0:42:25.80	545	0:00:21.80
32	64139	0:42:45.56	64405	0:42:56.20	266	0:00:10.64
33	64446	0:42:57.84	70477	0:46:59.08	6031	0:04:01.24
34	70505	0:47:00.20	75008	0:50:00.32	4503	0:03:00.12
35	75043	0:50:01.72	84483	0:56:19.32	9440	0:06:17.60
36	84498	0:56:19.92	92406	1:01:36.24	7908	0:05:16.32
37	93489	1:02:19.56	94665	1:03:06.60	1176	0:00:47.04
38	94704	1:03:08.16	95323	1:03:32.92	619	0:00:24.76
39	95405	1:03:36.20	98882	1:05:55.28	3477	0:02:19.08
40	99190	1:06:07.60	100658	1:07:06.32	1468	0:00:58.72
41	93489	1:02:19.56	95323	1:03:32.92	1834	0:01:13.36
42	10885	0:07:15.40	25633	0:17:05.32	14748	0:09:49.92
43	10885	0:07:15.40	16690	0:11:07.60	5805	0:03:52.20
44	16720	0:11:08.80	25633	0:17:05.32	8913	0:05:56.52
45	4477	0:02:59.08	10862	0:07:14.48	6385	0:04:15.40
46	29908	0:19:56.32	39767	0:26:30.68	9859	0:06:34.36
47	39802	0:26:32.08	47707	0:31:48.28	7905	0:05:16.20
48	47758	0:31:50.32	56549	0:37:41.96	8791	0:05:51.64
49	39802	0:26:32.08	58549	0:39:01.96	18747	0:12:29.88

Appendix 5.1

Information text available to the student
within the WITS program, as coded

```
m(m,71,6, "          LEARNING MODES").
m(m,71,9, " There are three ways you can go").
m(m,71,10,"through this course, or three").
m(m,71,11,"'learning modes':").
m(m,71,13,"           INSTRUCT mode").
m(m,71,14,"           CHOICE mode").
m(m,71,15,"           REVISE mode").

m(m,72,6, "          INSTRUCT MODE").
m(m,72,9, " This is intended for students who").
m(m,72,10,"know practically nothing about the").
m(m,72,11,"course.").
m(m,72,13," All you do is keep pressing the").
m(m,72,14,"space bar, and you are taken ").
m(m,72,15,"through the course").

m(m,73,6, "In INSTRUCT mode:").
m(m,73,8, "1. The order of material is fixed.").
m(m,73,10,"2. You cannot miss out topics.").
m(m,73,12,"3. You cannot miss out questions.").
m(m,73,14,"4. You can only view again topics").
m(m,73,15,"   already covered.").
m(m,73,17,"5. Questions can only be on topics").
m(m,73,18,"   that have been covered.").

m(m,74,6, "            CHOICE MODE").
m(m,74,9, " This is intended for students who").
m(m,74,10,"have covered some of the course").
m(m,74,11,"previously.").
m(m,74,13," You can pick modules and units to").
m(m,74,14,"miss out. After this, you keep ").
m(m,74,15,"pressing the space bar to be taken").
m(m,74,16,"through the remainder of the ").
m(m,74,17,"course material.").

m(m,75,6, "In CHOICE mode:").
m(m,75,8, "1. You are given choices of units").
m(m,75,9, "   and modules wherever possible.").
m(m,75,10,"2. You can miss out topics.").
m(m,75,11,"3. You can miss out questions up").
m(m,75,12,"   to three times.").
m(m,75,13,"4. You can see again topics already").
m(m,75,14,"   seen or given as known.").
m(m,75,15,"5. You can be given, or can ask ").
m(m,75,16,"   for, questions on any topic seen").
m(m,75,17,"   or given as known.").

m(m,76,6, "           REVISE MODE").
m(m,76,9, " This is intended for students who").
m(m,76,10,"have been through the course or").
m(m,76,11,"already know the subject.").
m(m,76,14," You have to find your own topics").
m(m,76,15,"to go through, either by name or ").
m(m,76,16,"via the SEARCH option.").
```

```
m(m,77,6, "In REVISE mode:").
m(m,77,8, "1. You investigate by yourself.").
m(m,77,10,"2. No topics are offered to you.").
m(m,77,12,"3. You can miss out questions,").
m(m,77,13,"   though some will be offered.").
m(m,77,15,"4. You can see any topic you find.").
m(m,77,17,"5. You can request questions on any").
m(m,77,18,"   topic.").

m(m,81,9, " The questions in this course are").
m(m,81,10,"all multiple choice.").
m(m,81,13," In each case, you choose the best").
m(m,81,14,"answer from A, B, C, D or E.").

m(m,82,6, " The questions may be different ").
m(m,82,7, "from multiple choice questions you").
m(m,82,8, "have done before.").
m(m,82,10," They do not consist of one correct").
m(m,82,11,"and four incorrect answers.").
m(m,82,13," There is always a best answer, but").
m(m,82,14,"sometimes other answers are near to").
m(m,82,15,"it and also correct. Other answers").
m(m,82,16,"may be factually correct but not").
m(m,82,17,"good answers. Others may be quite").
m(m,82,18,"wrong.").

m(m,83,7, " The questions may sometimes seem").
m(m,83,8, "harder than the usual ones, because").
m(m,83,9, "there are several close answers.").
m(m,83,12," However, you get credit for every").
m(m,83,13,"answer, and if you do not pick the").
m(m,83,14,"best one, picking a near answer may").
m(m,83,15,"be nearly as good. You should aim").
m(m,83,16,"at doing the best you can each time").
m(m,83,17,"so as to get a good average.").

m(m,84,6, " Some things you may find useful").
m(m,84,7, "for certain questions, and while").
m(m,84,8, "going through the course generally,").
m(m,84,9,"are:").
m(m,84,11,"    a) Pen and rough paper.").
m(m,84,13,"    b) A calculator.").
m(m,84,15,"    c) Books to refer to.").
m(m,84,17," There's no hurry - the computer").
m(m,84,18,"will always wait for you!").

m(m,118,6, "       THIS TEACHING SYSTEM").
m(m,118,8, "WITS is a prototype teaching system").
m(m,118,9, "to teach solid state electronics by").
m(m,118,10,"computer.").
m(m,118,12,"The course material is on a video-").
m(m,118,13,"disc made by Epic, Ltd. and it is ").
m(m,118,14,"played on a Philips laservision").
m(m,118,15,"videodisc player. It is controlled").
m(m,118,16,"by a program written in Prolog run").
m(m,118,17,"on an Opus IBM PC-type computer, ").
m(m,118,18,"with 1 megabyte of RAM.").
```

```
m(m,119,8,"Here are some things you can do:").
m(m,119,10," * name a topic you want to study (but").
m(m,119,11,"   not when in 'instruct' mode)").
m(m,119,13," * GO ON with the current learning plan").
m(m,119,15," * CHANGE your mode of studying").
m(m,119,17," * ask for INFORMATION of various kinds").

m(m,121,8,"Some more things you can do:").
m(m,121,10," * ask for QUESTIONS").
m(m,121,12," * get help with PROBLEMS.").
m(m,121,14," * receive a REPORT or profile.").
m(m,121,16," * FINISH the session.").
m(m,121,18,"Press space for a summary.").

m(m,122,7,"Summary of commands - type a letter and").
m(m,122,8,"enter or return, as follows:").
m(m,122,10,"  space/g   go on or continue").
m(m,122,11,"    i       information").
m(m,122,12,"    p       profile of progress").
m(m,122,13,"    s       search for a topic").
m(m,122,14,"    m       miss out a topic").
m(m,122,15,"    c       change mode").
m(m,122,16,"    q       questions").
m(m,122,17,"    f       finish").

m(m,123,7,"If you're having problems getting").
m(m,123,8,"through, remember:").
m(m,123,9," * Unless using space to continue,").
m(m,123,10,"   end messages with ENTER or ").
m(m,123,11,"   carriage return.").
m(m,123,12," * Spell correctly.").
m(m,123,13," * One letter will do for certain").
m(m,123,14,"   messages,like H for help.").
m(m,123,15," * Check your message is correct before").
m(m,123,16,"   pressing ENTER.").
m(m,123,17," * If there are still problems, try").
m(m,123,18,"   putting it a different way.").
```

Appendix 5.2

Command keywords in the program code

```
/* KEYWORD TEST FACTS */

test(skip_wanted,[m,miss,skip,omit]).
test(continue,[" ",g,continue,go]).
test(want_to_finish,[f,stop,end,halt,quit,finish,bye]).
test(mode_to_change,[c,change,mode,changemode]).
test(profile_wanted,[p,profile,progress]).
test(search_wanted,[s,search,find]).
test(report_wanted,[r,report]).
test(info_needed,[i,info,information,data,record,course]).
test(question_wanted,[q,question,questions,test,tests]).
```

Appendix 5.3

Rules for screen layout in the three modes

```
/* New screen rules */

newscreen_inout(Topics,_,Maxtop,revise,_,_):-
        screen(23),str_int(S1,Topics),str_int(S2,Maxtop),
        concat("Course topics seen: ",S1,A),
        concat(A,"/",B),concat(B,S2,X),
        shiftwindow(3),clearwindow,
        field_str(0,1,8," REVISE "),
        field_str(1,1,8,"  MODE  "),shiftwindow(4),
        clearwindow,window_attr(23),field_str(1,1,30,X),
        shiftwindow(5),clearwindow,window_attr(104),!.

newscreen_inout(Topics,_,Maxtop,instruct,Module,Unit):-
        str_int(S1,Topics),str_int(S2,Maxtop),
        X=["Course topics seen: ",S1,"/",S2],
        frontcap(Module,Module2),frontcap(Unit,Unit2),
        append(["Module:"],Module2,M),append(["Unit:"],
        Unit2,U),screen(23),shiftwindow(3),clearwindow,
        window_attr(23),field_str(0,1,8,"INSTRUCT"),
        field_str(1,1,8,"  MODE  "),
        shiftwindow(4),clearwindow,window_attr(23),
        telelistover(5,M,allcaps),telelistover(6,U,allcaps),
        telelistover(7,X,allcaps),shiftwindow(5),clearwindow,
        window_attr(104),!.

newscreen_inout(_,Adtops,Maxtop,choice,Module,Unit):-
        str_int(S1,Adtops),str_int(S2,Maxtop),
        X=["Course topics seen or known: ",S1,"/",S2],
        frontcap(Module,Module2),frontcap(Unit,Unit2),
        append(["Module:"],Module2,M),append(["Unit:"],
        Unit2,U),screen(23),shiftwindow(3),clearwindow,
        window_attr(23),field_str(0,1,8," CHOICE "),
        field_str(1,1,8,"  MODE  "),
        shiftwindow(4),clearwindow,window_attr(23),
        telelistover(5,M,allcaps),telelistover(6,U,allcaps),
        telelistover(7,X,allcaps),shiftwindow(5),clearwindow,
        window_attr(104),!.

newscreen_inout(_,_,_,_,_,_):-
        errormess("Error in new screen.").

frontcap([],[]).
frontcap([F|Ft],[F2|Ft]):-frontchar(F,C,Ct),ch_cap(C,C2),
frontchar(F2,C2,Ct).

errormess(E):-shiftwindow(7),clearwindow,window_attr(199),
        write(" ",E),getspace,exit.
```

Appendix 5.4

Rules for collecting student input

```
/* BASIC INTERACTION (INPUT) CLAUSES */

/* Takes in list of wordlists (eg.topics) and a list of
escape options (eg. [["Return to course"],["Return to
menu"]]) and returns chosen wordlist. */

menulist(Opt,Esc,Choice):-
        make_up(Opt,NewOpt,'A',[],C1),
        count(Esc,N),M=91-N,char_int(M1,M),
        make_up(Esc,NewEsc,M1,[],C2),
        append(NewOpt,NewEsc,Joined),
        listlist(10,Joined,allcaps),
        append(C1,C2,C),
        !,getkeypress(C,Xin),ch_cap(Xin,X),
        findChoice(X,Joined,Choice).

make_up([],[],_,C,C).
make_up([Z|Zt],[P|Pt],Q,C,Cx):-
        Z=[L|Lt],frontchar(L,F,Fs),ch_cap(F,F2),
        frontchar(L2,F2,Fs),Cap=[L2|Lt],str_char(Q1,Q),
        Next=[". "|Cap],P=[Q1|Next],char_int(Q,N),
        M=N+1,char_int(Y,M),
        make_up(Zt,Pt,Y,[Q|C],Cx).

findChoice(X,[[L|Lt]|_],R):-
        str_char(L,X),Lt=[_|Pt],Pt=[Q|Qt],
        frontchar(Q,Z,Zt),char_int(Z,N),M=N+32,
        char_int(Z2,M),frontchar(Qx,Z2,Zt),R=[Qx|Qt].
findChoice(X,[_|Lt],R):-findChoice(X,Lt,R).

/* Lists a list of wordlists, each on a new line.
   Startline and type of sentence specified. */

listlist(_,[],_).
listlist(Line,[H|T],Type):-
        H=[Z1|Zt],frontchar(Z1,Zf1,Zs),
        ch_cap(Zf1,Zf2),frontchar(Z2,Zf2,Zs),
        Hx=[Z2|Zt],append([" "],Hx,List),
        telelistover(Line,List,Type),
        M=Line+1,listlist(M,T,Type).

/* Accepts Y,y,N,n only and returns Y or N */

getyesno(X):-screen(16),readchar(C),gety(C,Y),X=Y,
        modify(increment,record,interactions,0,0),!.
getyesno(X):-screen(13),
        modify(increment,record,interactions,0,0),
        getyesno(X),!.

gety('Y','Y'). gety('y','Y'). gety('N','N'). gety('n','N').
```

```
/* Accepts specified list of case independent keypresses,
   returns one */

getkeypress(L,X):-screen(2),readchar(C),
        ch_cap(C,Q),member(Q,L),X=Q,
        modify(increment,record,interactions,0,0),!.
getkeypress(L,X):-screen(13),
        modify(increment,record,interactions,0,0),
        getkeypress(L,X),!.

/* Accepts space only */

getspace:-screen(12),repeat,readchar(X),getspace1(X).
getspace1(X):-X<>' ',
        modify(increment,record,interactions,0,0),
        screen(13),!,fail.
getspace1(' '):-modify(increment,record,interactions,0,0).


/* Rule to get student input */

get_input(Wordlist):-screen(1),screen(15),
        input_to_screen(0,[],Charlist,_),
        alter(Charlist,Charlist2),
        chlist_string(Charlist2,String),
        string_wordlist(String,Wordlist),
        modify(increment,record,interactions,0,0).

/* Take in student message as string, printing to screen */

input_to_screen(N,Wkgcharl,Charlist,X):-
        readchar(X),M=N+1,M<35,
        not(spacecheck(N,Wkgcharl,_,X)),
        subinput(X,N,Wkgcharl,Retchars),
        input_to_screen(M,Retchars,Charlist,_),!.
input_to_screen(0,_,['g'],' ').
input_to_screen(_,C,C,'z').

spacecheck(0,[],['g'],' ').

subinput(X,_,C,C2):-char_int(X,M),M>31,M<127,
        append(C,[X],C2),chlist_string(C2,S),
        cursor(0,1),write(S),!.


subinput(X,_,C,C2):-char_int(X,M),M=8,rev(C,Z),
        Z=[_|Ys],rev(Ys,C2),chlist_string(C2,S),
        clearwindow,cursor(0,1),write(S).

alter([],[]).
alter([H|T],[X|Y]):-char_int(H,H1),in_word(H1,X1),
        char_int(X,X1),alter(T,Y),!.
alter([_|T],T).

string_wordlist(S,[H|T]):-
        fronttoken(S,H,S1),!,string_wordlist(S1,T),!.
string_wordlist(_,[]).
```

```
/* Data to change sentence to l.c. list */

in_word(C,C):- C>96, C<123.                 /* a,b ... z */
in_word(C,L):- C>64, C<91, L = C+32.        /* A,B ... Z */
in_word(C,C):- C>47, C<58.                  /* 1,2 ... 9 */
in_word(39,39).                                 /* ' */
in_word(45,45).                                 /* - */
in_word(32,32).                                 /*   */
```

Appendix 5.5

Subordinate screen layout rules

```
/* Screen message rules */

screen(1):-modeX(revise),shiftwindow(6),clearwindow,
        field_str(0,0,31,"Type a sentence, word or letter"),!.

screen(1):-not(modeX(revise)),shiftwindow(6),clearwindow,
        field_str(0,0,31,"Space to continue, or a message"),!.

screen(2):-recordX(in_chain,1),shiftwindow(6),clearwindow,
        field_str(0,0,31,"Make a choice when topic ends."),!.

screen(2):-shiftwindow(6),clearwindow,
        field_str(0,0,31,"  Press the appropriate key."),!.

screen(3):-tele(clear),shiftwindow(2),clearwindow,
        window_attr(71),shiftwindow(3),clearwindow,
        window_attr(23),field_str(1,2,6," WITS"),
        shiftwindow(4),clearwindow,window_attr(23),
        X="Solid State Electronics",field_str(1,2,30,X),
        shiftwindow(5),clearwindow,window_attr(104),!.

screen(4):-shiftwindow(8),clearwindow,window_attr(199),
        field_str(1,4,25,"Essential files missing."),
        field_str(3,6,20,"Please start again."),!.

screen(5):-makewindow(3,23,7,"",2,0,4,12),
        field_str(1,2,6," WITS"),
        makewindow(8,23,0,"",11,1,9,35),
        field_str(2,1,33,"A whole course intelligent"),
        field_str(4,1,33,"teaching system using videodisc."),
        field_str(6,1,33,"David Callear, 1988"),getspace,
        shiftwindow(8),clearwindow,
        field_str(2,1,25,"Please put the data disc"),
        field_str(4,1,25,"in floppy disc drive A."),!.

screen(7):-shiftwindow(1),shiftwindow(8),clearwindow,
        window_attr(23),field_str(3,1,15,"Please check"),
        field_str(4,1,25,"correct videodisc loaded."),!.

screen(8):-shiftwindow(8),clearwindow,window_attr(23),
        field_str(1,1,25,"Please put in your"),
        field_str(2,1,25,"student disc."),
        field_str(3,1,25,"If you are just starting,"),
        field_str(4,1,25,"put in a blank disc with"),
        field_str(5,1,25,"the operating system."),getspace,
        dev_window(Dev),shiftwindow(8),clearwindow,
        window_attr(Dev),!.

screen(9):-disc(send,"BORDER 2"),disc(send,"VIDEO ON"),
        tele(clear),changeX(backframe,title,F),
        concat("STILL ",F,D),disc(send,D),
        getspace,disc(send,"BORDER 1"),!.
```

```
screen(10):-shiftwindow(6),clearwindow,
        field_str(0,0,31,"P=PAUSE,S=START,R=REPEAT,E=END"),!.

screen(11):-shiftwindow(6),clearwindow,
        field_str(0,0,31,"F=FORWARD,B=BACK,R=REPEAT,E=END"),!.
        screen(12):-shiftwindow(6),clearwindow,
        field_str(0,0,31,"Press the space bar to continue"),!.

screen(13):-shiftwindow(7),clearwindow,window_attr(207),
        field_str(0,2,20,"  Please try again."),!.
screen(14):-shiftwindow(6),clearwindow,
        field_str(0,0,31," One frame - space to continue "),
        readchar(X),X=' ',!.
screen(14):-screen(13),screen(14),!.

screen(15):-shiftwindow(7),clearwindow,window_attr(112),!.

screen(16):-shiftwindow(6),clearwindow,
        field_str(0,0,31," Press Y for yes, or N for no. "),!.

screen(17):-tele(clear),shiftwindow(2),clearwindow,
        window_attr(23),cursor(2,2),write("  UPDATE"),
        shiftwindow(5),clearwindow,window_attr(48),!.

screen(18):-tele(clear),
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),
        shiftwindow(5),clearwindow,message(90),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),nl,nl,write("    Page 1"),
        shiftwindow(5),clearwindow,message(118),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),nl,nl,write("    Page 2"),
        shiftwindow(5),clearwindow,message(119),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),nl,nl,write("    Page 3"),
        shiftwindow(5),clearwindow,message(121),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),nl,nl,write("    Page 4"),
        shiftwindow(5),clearwindow,message(122),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" SYSTEM INFO"),nl,nl,write("    Page 5"),
        shiftwindow(5),clearwindow,message(123),getspace,!.

screen(19):-disc(send,"VIDEO OFF"),tele(clear),
        shiftwindow(7),clearwindow,write("   GOODBYE!"),!.

screen(20):-tele(clear),shiftwindow(2),clearwindow,
        window_attr(23),cursor(3,5),write("INFORMATION"),
        shiftwindow(5),clearwindow,window_attr(48),!.

screen(21):-tele(clear),
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" QUESTION INFORMATION"),nl,nl,
        write("    Page 1"),
        shiftwindow(5),clearwindow,message(81),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" QUESTION INFORMATION"),nl,nl,
        write("    Page 2"),
        shiftwindow(5),clearwindow,message(82),getspace,
```

```
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" QUESTION INFORMATION"),nl,nl,
        write("    Page 3"),
        shiftwindow(5),clearwindow,message(83),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" QUESTION INFORMATION"),nl,nl,
        write("    Page 4"),
        shiftwindow(5),clearwindow,message(84),getspace,!.

screen(22):-recordX(quests_count,N),str_int(S,N),
        shiftwindow(3),clearwindow,
        write(" QUESTION"),nl,cursor(1,4),
        write(S),shiftwindow(7),clearwindow,
        write("          QUESTION TIME"),!.

screen(23):-shiftwindow(2),clearwindow,window_attr(71),
        field_str(0,1,43,"Space to continue,
                I for info,R for report"),
        field_str(1,10,25,"P for student profile"),
        field_str(2,10,20,"S for topic search"),
        field_str(3,10,20,"C to change mode"),
        field_str(4,10,20,"Q for questions"),
        field_str(5,10,20,"F to finish"),!.

screen(25):-tele(clear),shiftwindow(2),clearwindow,
        window_attr(23),cursor(3,5),write("SEARCH"),
        shiftwindow(5),clearwindow,window_attr(48),!.

screen(26):-tele(clear),shiftwindow(2),clearwindow,
        window_attr(23),cursor(3,5),
        write("PROFILE OF STUDENT PROGRESS"),
        shiftwindow(5),clearwindow,window_attr(48),!.

screen(27):-tele(clear),shiftwindow(2),clearwindow,
        cursor(3,5),write("SYSTEM INFORMATION"),
        shiftwindow(5),clearwindow,!.

screen(28):-tele(clear),
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,
        write("    Page 1"),
        shiftwindow(5),clearwindow,message(71),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 2"),
        shiftwindow(5),clearwindow,message(72),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 3"),
        shiftwindow(5),clearwindow,message(73),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 4"),
        shiftwindow(5),clearwindow,message(74),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 5"),
        shiftwindow(5),clearwindow,message(75),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 6"),
        shiftwindow(5),clearwindow,message(76),getspace,
        shiftwindow(2),clearwindow,cursor(2,1),
        write(" MODE INFORMATION"),nl,nl,write("    Page 7"),
        shiftwindow(5),clearwindow,message(77),getspace,!.
```

```
screen(29):-dev_window(Dev),shiftwindow(8),clearwindow,window_attr(Dev),!.

screen(30):-shiftwindow(1),clearwindow,shiftwindow(7),
        disc(send,"BORDER 2"),disc(send,"INDEX ON"),
        clearwindow,
        write(" F=forw,B=back,P=play,G=goto,E=end"),!.

screen(31):-shiftwindow(6),clearwindow,
        field_str(0,0,31,"LAST ONE. B=BACK,R=REPEAT,E=END"),!.

screen(_).


/* Prints out a message with no recursion */

message(N):-N>=400,N<500,submessage(3,N),!.
message(N):-N>=500,rand(3,X),M=N+X-1,submessage(1,M),!.
message(N):-submessage(1,N).

submessage(1,N):-not(syst(development)),tele(open),
        submessage(2,N),!.

submessage(2,N):-mX(m,N,L,X),char_int(CR,13),Z=L-5,
        field_str(Z,1,40,X),write(CR),fail,!.
submessage(2,_):-flush(mic),writedevice(screen),
        closefile(mic),!.

submessage(3,N):-mX(m,N,L,X),Z=L-4,cursor(Z,1),
        write(X),fail,!.
submessage(3,_):-flush(mic),writedevice(screen),
        closefile(mic).
```

Appendix 5.6

Videodisc controlling rules

```
/* Basic rules to open and close channel */

tele(open):-openwrite(mic,"MIC"),writedevice(mic),!.
tele(close):-char_int(CR,13),write(CR),flush(mic),
        writedevice(screen),closefile(mic),!.

/* Clears graphics/teletext screen */

tele(clear):-shiftwindow(1),window_attr(34),clearwindow,!.

/* Operates the disc player */

disc(send,X):-syst(mic),tele(open),write(X),tele(close),!.

/* Prints string (Line) in boxed graphics text */

telelinebox(N,Telemessage):-shiftwindow(5),
        frontchar(N,C1,T2),frontchar(T2,C2,_),
        str_char(S1,C1),str_char(S2,C2),
        concat(S1,S2,S),str_int(S,X),M=X-5,
        field_attr(M,1,40,6),field_str(M,2,38,Telemessage).

telelineover(N,Telemessage):-
        M=N-5,field_str(M,1,40,Telemessage).

/* Writes list of strings, standard message text, in boxed
   or unboxed teletext or current graphic text, line
   specified as integer,type cap/cap_st/st/allcaps/none */

telelistover(Line,[L|Ls],Type):-checkchange(L,L3),
        capit(L3,Q,Type),teleLover(Ls,Type,"",R),
        fullst(Type,F),concat(Q,R,X),concat(X,F,Z),
        telelineover(Line,Z),!.

/* Writes list of strings, in boxed teletext or red on
   black graphic text, line specified as string, type
   cap/cap_st/st/allcaps/none */

telelistbox(Line,[L|Ls],Type):-checkchange(L,L3),
        capit(L3,Q,Type),teleLover(Ls,Type,"",R),
        fullst(Type,F),concat(Q,R,X),concat(X,F,Z),
        telelinebox(Line,Z),!.

/* Support for telelist predicates */

convert2dig(N,P,Q):-A=(N div 10)+48,B=(N mod 10)+48,char_int(P,A),
        char_int(Q,B).
teleLover([],_,Q,Q).
teleLover([L|Ls],Type,Q,R):-changeX(leave,"",L),
        concat(Q," ",X),concat(X,L,Y),
        teleLover(Ls,Type,Y,R),!.
teleLover([L|Ls],allcaps,Q,R):-checkchange(L,L2),
        capit(L2,L3,allcaps),concat(Q," ",L4),
        concat(L4,L3,X),teleLover(Ls,allcaps,X,R),!.
```

```
teleLover([L|Ls],Type,Q,R):-checkchange(L,L2),
        concat(Q," ",L3),concat(L3,L2,X),
        teleLover(Ls,Type,X,R).

checkchange(L,L2):-changeX(perspron,L,L2),!.
checkchange(L,L2):-changeX(discword,L,L2),!.
checkchange(L,L).

capit(L,Q,Type):-changeX(check_caps,"",Type),
        frontchar(L,C,Ls),ch_cap(C,C1),frontchar(Q,C1,Ls),!.
capit(L,L,_).

fullst(Type,"."):-changeX(check_stop,"",Type).
fullst(_,"").

/* Sequence display rules */

show_topic(T):-set_up(after_interacting),
        show(T),set_up(after_viewing),!.

/* Video sequence (non-recursive) */

show(T) if tX(T,vid,N,F1,F2,_) and chain(T,Presses,T1,T2),
        modify(add,record,in_chain,1,0),audio(N),
        concat("PLAY ",F1,A),concat(A,",",B),
        concat(B,F2,C),disc(send,C),disc(send,"VIDEO ON"),
        disc(send,"FADE 255,200/V"),tele(clear),
        gotoframe(1,' ',' ',Presses,T1,T2),
        modify(retractall,record,in_chain,0,0),!.

show(T) if tX(T,vid,N,F1,F2,_),
        audio(N),concat("PLAY ",F1,A),concat(A,",",B),
        concat(B,F2,C),disc(send,C),disc(send,"VIDEO ON"),
        disc(send,"FADE 255,200/V"),tele(clear),
        screen(10),shiftwindow(8),
        repeat,
        readchar(X),ch_cap(X,Z),char_int(Z,Y),
        choice(1,Y,0,0,F1,F2,[]),!.

/* Random stills (non-recursive) */

show(T) if tX(T,ran,_,_,_,L) and chain(T,Presses,T1,T2),
        modify(add,record,in_chain,0,0),count(L,N),L=[F|_],
        concat("STILL ",F,D),disc(send,D),
        disc(send,"VIDEO ON"),disc(send,"FADE 255,200/V"),
        tele(clear),modify(update,record,framecount,1,0),screen(11),
        repeat,
        readchar(Z),screen(11),ch_cap(Z,Y),char_int(Y,X),
        recordX(framecount,C),choice(2,X,N,C,"","",L),
        gotoframe(1,' ',' ',Presses,T1,T2),
        modify(retractall,record,in_chain,0,0),!.

show(T) if tX(T,ran,_,_,_,L),count(L,N),L=[F|_],
        concat("STILL ",F,D),disc(send,D),
        disc(send,"VIDEO ON"),disc(send,"FADE 255,200/V"),
        tele(clear),modify(update,record,framecount,1,0),
        screen(11),
        repeat,
        readchar(Z),screen(11),ch_cap(Z,Y),char_int(Y,X),
        recordX(framecount,C),choice(2,X,N,C,"","",L),!.
```

```
/* Consecutive stills (non-recursive) */

show(T) if tX(T,con,M,F,_,_) and chain(T,Presses,T1,T2),
        modify(add,record,in_chain,0,0),concat("STILL ",F,D),
        disc(send,D),disc(send,"VIDEO ON"),
        disc(send,"FADE 255,200/V"),tele(clear),
        modify(update,record,framecount,1,0),N=M-1,screen(11),
        repeat,
        readchar(Z),screen(11),ch_cap(Z,X),char_int(X,Y),
        recordX(framecount,C),choice(3,Y,N,C,F,"",[]),
        gotoframe(1,' ',' ',Presses,T1,T2),
        modify(retractall,record,in_chain,0,0),!.

show(T) if tX(T,con,M,F,_,_),concat("STILL ",F,D),
        disc(send,D),disc(send,"VIDEO ON"),
        disc(send,"FADE 255,200/V"),tele(clear),
        modify(update,record,framecount,1,0),N=M-1,screen(11),
        repeat,
        readchar(Z),screen(11),ch_cap(Z,X),char_int(X,Y),
        recordX(framecount,C),choice(3,Y,N,C,F,"",[]),!.

/* one (single still) type */

show(T) if chain(T,Presses,T1,T2) and tX(T,one,_,F,_,_),
        concat("STILL ",F,D),disc(send,D),
        disc(send,"VIDEO ON"),disc(send,"FADE 255,200/V"),
        tele(clear),gotoframe(1,' ',' ',Presses,T1,T2),!.

show(T) if tX(T,one,_,F,_,_),concat("STILL ",F,D),
        disc(send,D),disc(send,"VIDEO ON"),
        disc(send,"FADE 255,200/V"),tele(clear),screen(14).

show(T):-bound(T),shiftwindow(5),clearwindow,cursor(2,2),
        write("Topic sequence not found."),nl,getspace.

/* Chain selector rules */

gotoframe(1,' ',' ',Presses,T1,T2):-recordX(in_chain,_),
        getkeypress(Presses,Press),Presses=[Ans|_],
gotoframe(2,Press,Ans,[],T1,T2),!.
gotoframe(1,' ',' ',_,_,_).
gotoframe(2,Ans,Ans,[],T1,_):-show(T1),!.
gotoframe(2,_,_,[],_,T2):-show(T2).

chain(92,['3','1','2','4'],921,922).
chain(921,['1','2','3','4'],923,924).
chain(922,['1','2','3','4'],923,924).
chain(923,['3','2','1'],925,926).
chain(924,['3','2','1'],925,926).

/* Sequence controlling rules */

/* vid (video sequence) type */

choice(1,80,0,0,_,_,[]):-
        disc(send,"STILL"),!,fail.                    /* P */
choice(1,83,0,0,_,F2,[]):-concat("PLAY ,",F2,D),
        disc(send,D),!,fail.                          /* S */
```

```
choice(1,82,0,0,F1,F2,[]):-concat("PLAY ",F1,A),
        concat(A," ",B),concat(B,F2,D),
        disc(send,D),!,fail.                          /* R */
choice(1,X,0,0,_,_,[]):-X<>69,!,fail.        /* Mistake */
choice(1,X,0,0,_,_,[]):-X=69,
        disc(send,"STILL"),!.                         /* E */

/* ran (random stills) type */

/* Jump out of chain and end */

choice(2,69,_,_,"","",_):-recordX(in_chain,_),
        modify(retractall,record,in_chain,0,0),!.

/* End of sequence in chain */

choice(2,70,N,C,"","",L):-C=N-1,recordX(in_chain,_),M=C+1,
        getatomnumber(M,L,X),concat("STILL ",X,Y),
        disc(send,Y),!.

choice(2,70,N,C,"","",L):-C=N,
        screen(31),M=C+1,getatomnumber(M,L,X),
        concat("STILL ",X,Y),disc(send,Y),
        modify(increment,record,framecount,0,0),
        !,fail.                                  /* F endstop */
choice(2,70,N,C,"","",_):-C>=N,
        screen(31),!,fail.                       /* F endstop */

choice(2,70,N,C,"","",L):-C<=N,
        M=C+1,getatomnumber(M,L,X),
        concat("STILL ",X,Y),disc(send,Y),
        modify(increment,record,framecount,0,0),
        !,fail.                                          /* F */
choice(2,66,_,C,"","",L):-C>1,
        M=C-1,getatomnumber(M,L,X),
        concat("STILL ",X,Y),disc(send,Y),
        modify(update,record,framecount,M,0),
        !,fail.                                          /* B */
choice(2,82,_,_,"","",[L|_]):-
        concat("STILL ",L,X),disc(send,X),
        modify(update,record,framecount,1,0),
        !,fail.                                          /* R */
choice(2,66,_,C,"","",_):-C<=1,
        modify(update,record,framecount,1,0),!,fail.
choice(2,X,_,_,"","",_):-X<>69,!,fail.        /* Mistake */
choice(2,69,_,_,"","",_):-!.                             /* E */

/* con (consecutive stills) type */

/* Jump out of chain and end */

choice(3,69,_,_,_,"",[]):-recordX(in_chain,_),
        modify(retractall,record,in_chain,0,0),!.

/* End sequence in chain */

choice(3,70,N,C,_,"",[]):-C=N,
        recordX(in_chain,_),disc(send,"STEP"),!.
```

```
choice(3,70,N,C,_,"",[]):-C=N,
        screen(31),disc(send,"STEP"),
        modify(increment,record,framecount,0,0),
        !,fail.                                  /* F end stop */
choice(3,70,N,C,_,"",[]):-C>=N,
        screen(31),!,fail.                       /* F end stop */
choice(3,70,N,C,_,"",[]):-C<=N,
        disc(send,"STEP"),M=C+1,
        modify(update,record,framecount,M,0),
        !,fail.                                              /* F */
choice(3,66,_,C,_,"",[]):-C>1,
        disc(send,"STEP /R"),M=C-1,
        modify(update,record,framecount,M,0),
        !,fail.                                              /* B */
choice(3,82,_,_,F,"",[]):-concat("STILL ",F,X),
        disc(send,X),
        modify(update,record,framecount,0,0),
        !,fail.                                              /* R */
choice(3,66,_,C,_,"",[]):-C<=0,
        modify(update,record,framecount,0,0),
        !,fail.                                   /* B stop */
choice(3,X,_,_,_,"",[]):-X<>69,
        !,fail.                                  /* Mistake */
choice(3,69,_,_,_,"",[]).                                    /* E */


/* Audio track rules for video sequences */

audio(0):-disc(send,"AUDIO ON"),!.
audio(1):-disc(send,"AUDIO ON/A1"),disc(send,"AUDIO OFF/A2"),!.
audio(2):-disc(send,"AUDIO ON/A2"),disc(send,"AUDIO OFF/A1"),!.
```

Appendix 5.7

Videodisc data facts

(First 30 out of 121.)

```
/* Data for showing videodisc sequences */

t(1,vid,0,"00050","00327",[]).
t(2,vid,0,"00328","00624",[]).
t(3,vid,0,"00644","01230",[]).
t(4,vid,0,"01231","01581",[]).
t(5,vid,0,"01582","01833",[]).
t(6,vid,0,"01834","02215",[]).
t(7,vid,0,"02216","02875",[]).
t(8,vid,0,"02876","03404",[]).
t(9,vid,0,"03405","03642",[]).
t(10,vid,0,"03643","03953",[]).
t(11,vid,0,"03967","04450",[]).
t(12,con,13,"03954","",[]).
t(13,ran,0,"","",["04451","04452","04453","04454",
          "04455","04456","18345","04458","04459",
          "04460","04461","04462","18346","04464",
          "04465","04466","04467","04468"]).
t(14,vid,0,"04469","04828",[]).
t(15,vid,0,"04829","05646",[]).
t(16,con,5,"05647","",[]).
t(17,vid,0,"05835","06043",[]).
t(18,vid,0,"06044","06532",[]).
t(19,vid,0,"06533","07174",[]).
t(20,con,7,"07175","",[]).
t(21,vid,0,"07182","07403",[]).
t(22,vid,0,"05653","05834",[]).
t(23,vid,0,"07404","07645",[]).
t(24,vid,0,"07646","07961",[]).
t(25,vid,0,"07962","08180",[]).
t(26,vid,0,"08188","08415",[]).
t(27,con,5,"08417","08431",[]).
t(28,con,5,"08432","08436",[]).
t(29,one,0,"08439","",[]).
t(30,con,14,"18381","18394",[]).
```

Some of the facts for the course modules, units and topics

```
/* Module data: Number, level, keyname, choice of modules to
   go on to, choice of units within module to start with. */

modl(0,6,[solid,state,electronics],[1],[]).
modl(1,6,[direct,currents],[2],[1]).
modl(2,6,[alternating,currents],[3,4,5],[6]).
modl(3,6,[the,capacitor],[4,5],[9]).
modl(4,6,[semiconductors,"and",diodes],[3,5],[13]).
modl(5,6,[transistors,"and",thyristors],[3,4],[19,22]).

/* Unit data: Number, module, level, keyname, units
   to go on to. */

unit(0,0,6,[],[1]).
unit(1,1,6,[the,nature,of,electricity],[2]).
unit(2,1,6,[electric,circuits],[3,4,5]).
unit(3,1,6,[the,resistor],[4,5]).
unit(4,1,6,[internal,resistance],[3,5]).
unit(5,1,6,[uses,of,resistors],[3,4]).
unit(6,2,6,[what,ac,is],[7,8]).
unit(7,2,6,[the,cathode,ray,tube],[8]).
unit(8,2,6,[electrical,measurements],[7]).
unit(9,3,6,[capacitors,"and",dc],[10,11,12]).
unit(10,3,6,[capacitors,"and",ac],[11,12]).
unit(11,3,6,[uses,of,capacitors],[10,12]).
unit(12,3,6,[care,with,capacitors],[10,11]).
unit(13,4,6,[the,semiconductor],[14]).
unit(14,4,6,[the,diode],[15,16,17,18]).
unit(15,4,6,[testing,diodes],[16,17,18]).
unit(16,4,6,[rectification],[15,17,18]).
unit(17,4,6,[diode,breakdown],[15,16,18]).
unit(18,4,6,[the,zener,diode],[15,16,17]).
unit(19,5,6,[the,transistor],[20,21,22]).
unit(20,5,6,[the,unijunction,transistor],[21,22]).
unit(21,5,6,[uses,of,transistors],[20,22]).
unit(22,5,6,[the,thyristor],[23,19]).
unit(23,5,6,[uses,of,thyristors],[19]).

/* Topic data: Number, unit, module, level,
   keyname. */

top(0,0,0,6,[]).

top(1,1,1,6,[conductors]).
top(2,1,1,6,[insulators]).
top(3,1,1,6,[atoms,"and",electrons]).
top(4,1,1,6,[electrons,in,conductors]).
top(5,1,1,6,[electrons,moving,randomly]).
top(6,1,1,6,[electrons,under,a,voltage]).
top(7,1,1,6,[electrons,in,a,circuit]).
top(8,1,1,6,[the,waiting,room,analogy]).
top(9,1,1,6,[electric,currents]).
```

```
top(10,2,1,6,[an,electrical,convention]).
top(11,2,1,6,[the,plumbing,circuit,analogy]).
top(12,2,1,6,["ohm's",law]).

top(13,3,1,6,[what,resistors,are]).
top(14,3,1,6,[potentiometers]).
top(15,3,1,6,[double,current,potentiometers]).
top(16,3,1,6,[electrical,power]).

top(17,4,1,6,[what,internal,resistance,is]).
top(18,4,1,6,[car,head,lamps]).
top(19,4,1,6,[internal,resistance,circuits]).
top(20,4,1,6,[the,emf,of,a,cell]).

top(21,5,1,6,[linear,resistors]).
top(22,5,1,6,[thermistors]).
top(23,5,1,6,[the,use,of,thermistors]).
top(24,5,1,6,[the,voltage,dependent,resistor]).
top(25,5,1,6,[using,potentiometers]).
top(26,5,1,6,[high,stability,resistors]).

top(27,6,2,6,[sine,waves]).
top(28,6,2,6,[peak,values]).
top(29,6,2,6,[other,waveforms]).
```

Appendix 5.9

Facts for the course on Van Gogh

```
/* Module data: Number, level, keyname, choice of modules to go
   on to, choice of units within module to start with. */

modl(0,6,[vincent,van,gogh],[1,2,3],[]).
modl(1,6,[overview,of,van,gogh],[2,3],[1,2,3,4,5]).
modl(2,6,[van,goghs,paintings],[1,3],[6,7]).
modl(3,6,[influences,on,van,gogh],[1,2],[8,9,10,11]).

/* Unit data: Number, module, level, keyname, units
   to go on to. */

unit(0,0,6,[],[1,2,3,4,5]).

unit(1,1,6,[the,dutch,period],[2,3,4,5]).
unit(2,1,6,[the,paris,period],[1,3,4,5]).
unit(3,1,6,[the,arles,period],[1,2,4,5]).
unit(4,1,6,[the,stremy,period],[1,2,3,5]).
unit(5,1,6,[the,auvers,period],[1,2,3,4]).

unit(6,2,6,[paintings,by,period],[7]).
unit(7,2,6,[paintings,by,subject],[6]).

unit(8,3,6,[influences,by,name,bg],[9,10,11]).
unit(9,3,6,[influences,by,name,hm],[8,10,11]).
unit(10,3,6,[influences,by,name,pw],[8,9,11]).
unit(11,3,6,[influences,by,period],[8,9,10]).

/* Topic data: Number, unit, module, level,
   keyname. */

top(0,0,0,6,[]).

top(1,1,1,6,[the,period,reviewed]).
top(2,1,1,6,[van,goghs,letters]).
top(3,2,1,6,[the,period,reviewed]).
top(4,2,1,6,[van,goghs,letters]).
top(5,3,1,6,[the,period,reviewed]).
top(6,3,1,6,[van,goghs,letters]).
top(7,4,1,6,[the,period,reviewed]).
top(8,4,1,6,[van,goghs,letters]).
top(9,5,1,6,[the,period,reviewed]).
top(10,5,1,6,[van,goghs,letters]).

top(11,6,2,6,[the,dutch,period]).
top(12,6,2,6,[the,paris,period]).
top(13,6,2,6,[the,arles,period]).
top(14,6,2,6,[the,stremy,period]).
top(15,6,2,6,[the,auvers,period]).
top(16,7,2,6,[self,portraits]).
top(17,7,2,6,[portraits]).
top(18,7,2,6,[landscapes]).
top(19,7,2,6,[still,life,themes]).
top(20,7,2,6,[cities,"and",villages]).
```

```
top(21,8,3,6,[emile,bernard]).
top(22,8,3,6,[george,breitner]).
top(23,8,3,6,[paul,cezanne]).
top(24,8,3,6,[edgar,degas]).
top(25,8,3,6,[gerrit,dou]).
top(26,8,3,6,[paul,gauguin]).
top(27,8,3,6,[jean,guillaumin]).

top(28,9,3,6,[ando,hiroshige]).
top(29,9,3,6,[meindert,hobbema]).
top(30,9,3,6,[nicolaas,maes]).
top(31,9,3,6,[edouard,manet]).
top(32,9,3,6,[jacob,maris]).
top(33,9,3,6,[anton,mauve]).
top(34,9,3,6,[jean,francois,millet]).
top(35,9,3,6,[claude,monet]).

top(36,10,3,6,[camille,pissaro]).
top(37,10,3,6,[rembrandt,van,rijn]).
top(38,10,3,6,[pierre,auguste,renoir]).
top(39,10,3,6,[jacob,van,ruisdael]).
top(40,10,3,6,[george,seurat]).
top(41,10,3,6,[paul,signac]).
top(42,10,3,6,[henri,de,toulouselautrec]).
top(43,10,3,6,[jan,weissenbruch]).

top(44,11,3,6,[influences,in,holland]).
top(45,11,3,6,[influences,in,paris]).
top(46,11,3,6,[influences,in,arles]).
top(47,11,3,6,[influences,in,stremy]).
top(48,11,3,6,[influences,in,auvers]).

/* Following are dictionary topics not part of
   course */

top(100,0,0,6,[four,cut,sunflowers]).
top(101,0,0,6,[vincents,bedroom,in,arles]).
top(102,0,0,6,[an,introduction,to,van,gogh]).

/* Keyword data - topic lists */

kw(topic,16,[self,"self-portrait"]).
kw(topic,17,[portrait]).
kw(topic,18,[landscape,scene,tree,hill,windmill,gras,mountain]).
kw(topic,19,[still,life,"still-life"]).
kw(topic,20,[city,citie,town,village,house,building]).
kw(topic,21,[bernard]).
kw(topic,22,[breitner,britner,brietner]).
kw(topic,23,[cezanne,cezane,cezann]).
kw(topic,24,[dega,degas]).
kw(topic,25,[dou,doo]).
kw(topic,26,[gauguin,gaugin,guguin,guaguin,guagin]).
kw(topic,27,[guillaumin,gillaumin,guilaumin]).
kw(topic,28,[hiroshige]).
kw(topic,29,[hobbema,hobema]).
kw(topic,30,[mae,my,mice]).
kw(topic,31,[manet,manay]).
kw(topic,32,[mari]).
kw(topic,33,[mauve,move]).
kw(topic,34,[millet,milet]).
```

```
kw(topic,35,[monet,monay]).
kw(topic,36,[pissaro,pisaro,pizaro,pizzaro]).
kw(topic,37,[rembrandt,rembrant,rijn]).
kw(topic,38,[renoir]).
kw(topic,39,[ruisdael,ruisdyl,ruisdel]).
kw(topic,40,[seurat,surat]).
kw(topic,41,[signac,sinac,siniac]).
kw(topic,42,[toulouse,lautrec]).
kw(topic,43,[weissenbruch,weissenbruk,weisenbruch,weisenbruk]).

/* Dictionary topics */

kw(topic,100,[sunflower]).
kw(topic,101,[bedroom]).
kw(topic,102,[introduction,intro]).

/* Unit lists */

kw(unit,1,[dutch,holland]).
kw(unit,2,[pari,french,france]).
kw(unit,3,[arle]).
kw(unit,4,[remy,saint,"st.Remy","st-Remy","Saint-Remy"]).
kw(unit,5,[auver]).
kw(unit,8,[bg,"b-g"]).
kw(unit,9,[hm,"h-m"]).
kw(unit,10,[pw,"p-w"]).

/* Module lists */

kw(module,1,[vincent,gogh]).

/* Following are general keyword groupings (for
   Van Gogh disc) */

kw(general,1,[painting,picture,work]).
kw(general,2,[review,reviewed,account,summary,summarise,overview]).
kw(general,3,[period,phase,time]).
kw(general,4,[influence,contemporary,peer,contemporarie,friend]).
kw(general,5,[subject,topic]).
kw(general,6,[letter,writing]).

/* Keyname linking data (key module/unit/topic,
   number indicated, then two keyword lists
   identified which point to it: mod/unit/top/gen,
   first number, mod/unit/top/gen, second number */

kwlink(module,2,module,1,general,1).
kwlink(module,3,general,1,general,4).

kwlink(unit,6,general,1,general,3).
kwlink(unit,7,general,1,general,5).
kwlink(unit,11,general,3,general,4).

kwlink(topic,1,unit,1,general,2).
kwlink(topic,2,unit,1,general,6).
kwlink(topic,3,unit,2,general,2).
kwlink(topic,4,unit,2,general,6).
kwlink(topic,5,unit,3,general,2).
kwlink(topic,6,unit,3,general,6).
kwlink(topic,7,unit,4,general,2).
```

```
kwlink(topic,8,unit,4,general,6).
kwlink(topic,9,unit,5,general,2).
kwlink(topic,10,unit,5,general,6).
kwlink(topic,11,unit,1,general,1).
kwlink(topic,12,unit,2,general,1).
kwlink(topic,13,unit,3,general,1).
kwlink(topic,14,unit,4,general,1).
kwlink(topic,15,unit,5,general,1).
kwlink(topic,44,unit,1,general,4).
kwlink(topic,45,unit,2,general,4).
kwlink(topic,46,unit,3,general,4).
kwlink(topic,47,unit,4,general,4).
kwlink(topic,48,unit,5,general,4).

/* Data for showing videodisc sequences */

t(1,vid,1,"02770","09300",[]).
t(2,vid,2,"02770","09300",[]).
t(3,vid,1,"09435","15720",[]).
t(4,vid,2,"09435","15720",[]).
t(5,vid,1,"15800","30250",[]).
t(6,vid,2,"15800","30250",[]).
t(7,vid,1,"30390","36570",[]).
t(8,vid,2,"30390","36570",[]).
t(9,vid,1,"36630","44720",[]).
t(10,vid,2,"36630","44720",[]).
t(11,con,60,"09333","",[]).
t(12,con,29,"15762","",[]).
t(13,con,63,"30285","",[]).
t(14,con,33,"36580","",[]).
t(15,con,25,"44736","",[]).
t(16,con,7,"45103","",[]).
t(17,con,16,"45044","",[]).
t(18,con,17,"45061","",[]).
t(19,con,7,"45079","",[]).
t(20,con,15,"45087","",[]).
t(21,ran,0,"","",["44928","27000","15788","26800"]).
t(22,one,0,"07900","",[]).
t(23,ran,0,"","",["37250","44945"]).
t(24,one,0,"44907","",[]).
t(25,one,0,"44820","",[]).
t(26,ran,0,"","",["2800","26650","44930","28700","27200"]).
t(27,one,0,"44913","",[]).
t(28,ran,0,"","",["13700","13650"]).
t(29,one,0,"07400","",[]).
t(30,one,0,"44821","",[]).
t(31,one,0,"44901","",[]).
t(32,one,0,"07700","",[]).
t(33,one,0,"08100","",[]).
t(34,one,0,"04400","",[]).
t(35,one,0,"44904","",[]).
t(36,ran,0,"","",["10900","37100","45003"]).
t(37,ran,0,"","",["07100","44819"]).
t(38,ran,0,"","",["44924","09800"]).
t(39,one,0,"06800","",[]).
t(40,one,0,"44918","",[]).
t(41,one,0,"44921","",[]).
t(42,ran,0,"","",["11000","44936"]).
t(43,one,0,"07500","",[]).
t(44,ran,0,"","",[]).
```

```
t(45,ran,0,"","",[]).
t(46,ran,0,"","",[]).
t(47,ran,0,"","",[]).
t(48,ran,0,"","",[]).

t(100,one,0,"45082","",[]).
t(101,one,0,"21500","",[]).
t(102,vid,0,"00300","02047",[]).
```

Appendix 5.10

Main Rules of the Program

```
/* RULES TO DEAL WITH QUESTIONS */

/* Presents a Q until max no reached */

rule(1,_,_,_) if reached(question),
        record(student_cat,C),
        record(student_cat,C),maxtop(M),
        record(top_known_total,K),
        student_tops(C,M,K,_,Askno),
        record(quests_count,Q),Askno>Q,N=Q+1,
        assess1(select_quest,S,' '),
        modify(update,record,quests_count,N,0),
        set_up(quest),assess1(present_quest,S,A),
        modify(add,record,quest_done,S,0),
        assess1(record_quest,S,A),
        assess1(update_probs,S,' '),
        assess2(update_stud_cat,0,' '),!.

/* Catchall if no Qs found at all */

rule(2,_,_,_) if reached(question),
        record(quests_count,Q),Q=0,
        set_up(after_questions),
        newscreen and message(54),!.

/* Catchall if run out of Qs during Q session */

rule(3,_,_,_) if reached(question),
        record(student_cat,C),
        record(student_cat,C),maxtop(M),
        record(top_known_total,K),
        student_tops(C,M,K,_,Askno),
        record(quests_count,Q),Askno>Q,
        set_up(after_questions),
        newscreen and message(570),!.

/* Ends Q session when max no reached */

rule(4,_,_,_) if reached(question),
        record(student_cat,C),
        record(student_cat,C),maxtop(M),
        record(top_known_total,K),
        student_tops(C,M,K,_,Askno),
        record(quests_count,Q),Askno<=Q,
        set_up(after_questions),
        newscreen and message(570),!.

/* RULES TO DEAL WITH MISSING OUT TOPICS */

/* Skip irrelevant in REVISE */

rule(5,skip_wanted,_,_) if mode_is(revise),
        newscreen and message(1),!.
```

```
/* Skip not allowed in INSTRUCT */
rule(6,skip_wanted,Y,Z) if mode_is(instruct),
        newscreen and message(2),!.

/* Deal with skip in CHOICE */

rule(7,skip_wanted,_,_) if mode_is(choice)
        and not(reached(next_decided)),
        newscreen and message(132),!.

rule(8,skip_wanted,_,_) if mode_is(choice)
        and reached(next_decided),
        record(new_topic,T),count_tops(seen),
        modify(update,record,last_topic,T,0),
        modify(retractall,record,new_topic,0,0),
        modify(increment,record,skip_count,0,0),
        retractall(reached),increment_set(T),
        check_for_question,
        newscreen and message(3),!.

 /* COURSE CONTINUATION RULES ACTING ON 'SPACE' */

/* Continue meaningless in REVISE */

rule(9,continue,_,_) if mode_is(revise),
        newscreen and message(1),!.

/* Next rules are common to both INSTRUCT and
        CHOICE modes */

/* Course finished */

rule(10,continue,_,_) if not(mode_is(revise))
        and reached(end_of_course),
        newscreen and message(31),!.

/* Show topic in INSTRUCT or CHOICE */

rule(11,continue,_,_) if not(mode_is(revise))
        and reached(next_decided),
        record(new_topic,T),show_topic(T),
        retractall(reached),
        modify(retractall,record,new_topic,0,0),
        modify(update,record,last_topic,T,0),
        tidy_up(tops,T),increment_set(T),
        check_for_question,!.

/* RULES FOR INSTRUCT MODE */

/* New module */

rule(12,continue,_,_) if mode_is(instruct)
        and reached(new_mod),
        record(last_topic,T),top(T,_,M,_,_),
        Mx=M+1,modl(Mx,_,X,_,_),
        newscreen and message(32),
        telelistbox("12/CY/05",X,allcaps),
        modify(update,record,this_mod,Mx,0),
        modify(retractall,reached,new_mod,0,0),!.
```

```
/* New unit */

rule(13,continue,_,_) if mode_is(instruct)
        and reached(new_unit),
        record(last_topic,T),top(T,U,_,_,_),
        Ux=U+1,unit(Ux,_,_,X,_),
        newscreen,message(33),
        telelistbox("12/CY/05",X,allcaps),
        modify(update,record,this_unit,Ux,0),
        retractall(reached),!.

/* RULES FOR BOTH INSTRUCT AND CHOICE MODES */

/* Topic seen */

rule(14,continue,_,_) if not(mode_is(revise)),
        not(reached(new_unit)),
        not(reached(new_mod)),
        not(reached(new_mod_and_unit)),
        record(new_topic,T),record(top_seen,T),
        newscreen and message(34),
        top(T,_,_,_,X),
        telelistbox("13/CY/05",X,allcaps),
        getyesno(Yesno),show_or_not(Yesno,T),!.

/* Decide on next topic to show */

rule(15,continue,_,_) if not(mode_is(revise)),
        not(reached(new_unit)),
        not(reached(new_mod)),
        not(reached(new_mod_and_unit)),
        record(new_topic,T),
        top(T,_,_,_,X),newscreen and message(35),
        telelistbox("12/CY/05",X,allcaps),
        retractall(reached),save_file,
        assertz(reached(next_decided)),!.

/* RULES FOR CHOICE MODE */

/* Last module */

rule(16,continue,_,_) if mode_is(choice) and
        reached(new_mod) and reached(new_unit),
        getchoicelist(L,mods),count(L,N),N=1,
        L=[M],modl(M,_,X,_,_),
        modify(update,record,this_mod,M,0),
        modify(retractall,reached,new_mod,0,0),
        modify(retractall,reached,new_unit,0,0),
        assertz(reached(new_mod_and_unit)),
        newscreen,message(32),
        telelistbox("12/CY/05",X,allcaps),!.

/* Give choice of modules */

rule(17,continue,K,Y) if mode_is(choice) and
        reached(new_mod) and reached(new_unit),
        getchoicelist(L,mods),count(L,N),N>1,
        getlistnames(L,List,mods),
        newscreen,message(56),
```

```
        menulist(List,[],Z),modl(M,_,Z,_,_),
        modify(update,record,this_mod,M,0),
        modify(retractall,reached,new_mod,0,0),
        modify(retractall,reached,new_unit,0,0),
        assertz(reached(new_mod_and_unit)),
        newscreen,message(32),
        telelistbox("12/CY/05",Z,allcaps),!.

/* First unit in module if no choice and only one */

rule(18,continue,_,_) if mode_is(choice) and
        reached(new_mod_and_unit),
        getchoicelist(List,firstunit),count(List,N),
        N=1,List=[U],unit(U,_,_,Z,_),
        modify(update,record,this_unit,U,0),
        get_tops(U,T1,_,unit),
        modify(update,record,new_topic,T1,_),
        modify(retractall,reached,new_mod_and_unit,0,0),
        newscreen and message(33),
        telelistbox("12/CY/05",Z,allcaps),!.

/* Choice of first unit in module */

rule(19,continue,_,_) if mode_is(choice)
        and reached(new_mod_and_unit),
        getchoicelist(List,firstunit),
        getlistnames(List,L,units),
        newscreen,message(57),
        menulist(L,[],Z),unit(U,_,_,Z,_),
        modify(update,record,this_unit,U,0),
        get_tops(U,T1,_,unit),
        modify(update,record,new_topic,T1,_),
        modify(retractall,reached,new_mod_and_unit,0,0),
        newscreen and message(33),
        telelistbox("12/CY/05",Z,allcaps),!.

/* Choice of subsequent units in module */

rule(20,continue,_,_) if mode_is(choice)
        and reached(new_unit),
        getchoicelist(L,units),count(L,N),N>1,
        getlistnames(L,List,units),
        newscreen,message(57),
        menulist(List,[],Z),unit(U,_,_,Z,_),
        modify(update,record,this_unit,U,0),
        get_tops(U,T1,_,unit),
        modify(update,record,new_topic,T1,_),
        modify(retractall,reached,new_unit,0,0),
        newscreen and message(33),
        telelistbox("12/CY/05",Z,allcaps),!.

/* Last unit in module */

rule(21,continue,_,_) if mode_is(choice) and
        reached(new_unit),getchoicelist(L,units),
        count(L,N),N=1,L=[U],unit(U,_,_,X,_),
        modify(update,record,this_unit,U,0),
        get_tops(U,T1,_,unit),
        modify(update,record,new_topic,T1,_),
        modify(retractall,reached,new_unit,0,0),
```

```
        newscreen and message(33),
        telelistbox("12/CY/05",X,allcaps),!.

/* RULES FOR KEYWORD COMMANDS (MODE INDEPENDENT) */

/* Information */

rule(22,info_needed,_,_) if
        modify(increment,record,info_count,0,0),
        tele(clear),repeat,
        set_up(info),newscreen and message(510),!.

/* Profile */

rule(23,profile_wanted,_,_) if screen(24),
        modify(increment,record,profile_count,0,0),
        assess2(profile,1,' '),
        newscreen and message(510),!.

/* Change mode */

rule(25,mode_to_change,_,_) if newscreen,
        message(113),getkeypress(['C','R','I'],X),
        mode_is(M),retractall(mode),
        assertz(mode_is(M)),turn(Y,_,X,_,_,_,_,_,_,_),
        retractall(reached),changemode(Y),
        record(last_topic,T),increment_set(T),!.

/* Search */

rule(26,search_wanted,_,_) if
        modify(increment,record,search_count,0,0),
        repeat,set_up(search),
        newscreen and message(510),!.

/* Question */

rule(27,question_wanted,Y,Z) if newscreen,
        message(145),
        getkeypress(['A','B','C','Z'],X),
        chosen_quest(X),!.

/* Report */

rule(28,report_wanted,_,_) if screen(3),
        message(95),getspace,screen(3),
        message(96),assess2(report,1,' '),
        newscreen and message(510),!.

/* Finish */

rule(31,want_to_finish,_,_) if newscreen,
        telelineover(11,
        "Do you want to finish the session?"),
        getyesno(Z),nl,finish(Z),!.
```

```
/* RULES FOR KEYNAMES (MODE INDEPENDENT) */

/* Deal with keyname at right level in REVISE */

rule(32,keyname,K,T) if mode_is(revise)
        and level_checks(K,T),
        modify(increment,record,keyname_count,0,0),
        dealwith_and_show(K,T),!.

/* Topic already seen in INSTRUCT or CHOICE (show) */

rule(34,keyname,K,T) if topic_seen(K,T)
        and not(mode_is(revise)),
        modify(increment,record,keyname_count,0,0),
        dealwith_and_show(K,T),!.

/* Topic given as known in CHOICE (show) */

rule(35,keyname,K,T) if mode_is(choice)
        and topic_known(K,T),
        modify(increment,record,keyname_count,0,0),
        dealwith_and_show(K,T),!.

/* 'Wait' message if not seen or known in INSTRUCT
        or CHOICE */

rule(36,keyname,_,_) if not(mode_is(revise)),
        modify(increment,record,keyname_count,0,0),
        set_break,newscreen and  message(8),!.

/* MAIN CATCHALL RULE FOR INPUT NOT 'UNDERSTOOD' */

rule(37,X,K,T):-newscreen,message(560),
        modify(increment,record,misunderstandings,0,0).
```

Appendix 5.11

Rules for the initial phase of the program

```
/* Instructions for start_up procedure */

init(start_up):-set_up(prog),set_up(windows),
        screen(5),getspace,screen(7),
        consult("discdata.dba"),consult("progdata.dba"),
        consult("quesdat1.dba"),consult("quesdat2.dba"),
        consult("quesdat3.dba"),getspace,
        screen(8),set_up(windows_again),!.

/* Start_up fails due to missing file */

init(start_up):-errormess("Files missing.").

/* Start session other than first (student file present) */

init(initialise):-existfile("student.dba"),
        modify(retractall,record,interactions,0,0),
        consult("student.dba"),
        set_up(other_session_times),screen(3),
        recordX(X,1000),A=["hello,",X],
        telelistover(11,A,allcaps),
        telelineover(13,"Welcome back."),getspace,
        set_break,screen(3),message(510),
        recordX(last_topic,T),increment_set(T),
        !,fail.

/* Student file created but contains error */

init(initialise):-existfile("student.dba"),
        errormess("Error in student file.").

/* Start first session (file has not been created) */

init(initialise):-set_up(first_session_times),
        screen(3),message(101),get_input(D),D=[Firstname|_],
        screen(3),message(102),get_input(E),E=[Lastname|_],
        screen(3),telelineover(7,"This course is on "),
        modX(0,_,F,_,_),telelistbox("09/CY/05",F,allcaps),
        message(105),getkeypress(['W','L','N'],Sub),
        turn(Mode,Sub,_,How,Nm,_,_,_,_),
        assertz(record(Firstname,1000)),
        assertz(record(Lastname,2000)),
        assertz(record(How,3000)),
        assertz(mode_is(Mode)),init(student_model),
        screen(3),message(Nm),getspace,
        init(choice_or_not),recordX(last_topic,T),
        increment_set(T),save_file,screen(3),
        message(134),!,fail.
```

Appendix 5.12

Rules for the route planning facility for the student

```
/* Allow student to decide whether to miss mods and units
   in ADVICE mode */

init(choice_or_not):-not(modeX(choice)).
init(choice_or_not):-modeX(choice),newscreen,message(140),
        getspace,newscreen,message(141),getyesno(Yesno),
        changechoice(1,Yesno).

changechoice(1,'Y'):-set_up(start_choice).
changechoice(1,'N').

/* Rule to initiate route planning */

set_up(start_choice):-choosethings(mods_to_do,0,0,[]),
        choosethings(units_to_do,0,0,[]),
        newscreen,message(36),getspace.

choosethings(mods_to_do,0,0,[]):-
        repeat,choosethings(mods,0,0,[]),!.

/* Allows students to choose mods from list */

choosethings(mods,0,0,[]):-screen(3),message(128),getspace,
        findall(M,modX(_,_,M,_,_),Lst1),Lst1=[_|Lst],
        get_choice_list(Lst,[x],[],L),subtract(L,Lst,List),
        screen(3),message(129),listlist(10,List,allcaps),
        getyesno(Yesno),!,check(mods,Yesno,L,List),!.

/* Initialises unit selections */

choosethings(units_to_do,0,0,[]):-
        findall(K,recordX(mod_known,K),Lst),Lst=[],!.
choosethings(units_to_do,0,0,[]):-screen(3),
        findall(M,modX(M,_,_,_,_),Lst1),Lst1=[_|Lst2],
        findall(K,recordX(mod_known,K),Lst3),
        subtract(Lst3,Lst2,Lst),repeat,modX(N,_,X,_,_),
        not(recordX(mod_known,N)),member(N,Lst),
        choosethings(units,1,N,X),!.

/* Skip over unit revisions */

choosethings(units,1,N,_):-unitX(U,N,_,_,_),
        recordX(unit_known,U),!.
choosethings(units,1,N,_):-recordX(mod_known,N),!.

/* Allows student to choose units from
   successive module lists */

choosethings(units,1,N,X):-maxmod(M),
        repeat,choosethings(units,2,N,X),!,N=M.
```

```
choosethings(units,2,N,X):-findall(U,unitX(_,N,_,U,_),Lx),
        get_choice_list(Lx,[x],[],L),subtract(L,Lx,List),
        screen(3),telelistbox("06/CR/NB/CW",X,cap_st),
        message(135),listlist(10,List,allcaps),
        getyesno(Yesno),!,check(units,Yesno,L,List).

check(mods,'Y',[],_):-!.
check(mods,'N',X,_):-screen(3),message(130),getspace,!,fail.
check(mods,'Y',L,_):-assert_list(mods,L),screen(3),
        message(131),getspace,screen(3),message(142),
        count_tops(known),getspace,!.

check(units,'N',_,_):-screen(3),message(130),getspace,!,fail.
check(units,'Y',List,_):-assert_list(units,List).

/* These get revise lists from the student */

get_choice_list(_,Choice,Runlist,Listknown):-
        Choice=[end,your,selection],
        delete(Choice,Runlist,Listknown).

get_choice_list(Listofmods,X,Runlist,Listknown):-screen(3),
        telelineover(7,"Pick items to MISS OUT:"),
        menulist(Listofmods,[[end,your,selection]],Choice),
        delete(Choice,Listofmods,Newlistofmods),
        Newrunlist=[Choice|Runlist],
        get_choice_list(Newlistofmods,Choice,Newrunlist,
        Listknown).

/* These record the chosen mods and units */

assert_list(mods,[]).
assert_list(mods,[L|Ls]):-modX(M,_,L,_,_),
        modify(add,record,mod_known,M,0),
        assert_list(mods,Ls).
assert_list(units,[]).
assert_list(units,[L|Ls]):-unitX(U,_,_,L,_),
        modify(add,record,unit_known,U,0),
        assert_list(units,Ls).
```

Rules for navigating the transition nets of the course modules and units

```
/* Increments topic number and sets course parameters
   after a show */

/* Start up case */
increment_set(0):-not(mode_is(revise)),
        assertz(reached(new_mod)),
        assertz(reached(new_unit)),
        assertz(record(new_topic,1)),!.

/* End of course case */
increment_set(T):-bound(T),inc,!.

/* Increment in normal case */
increment_set(T):-Tn=T+1,top(Tn,_,_,X,_),level(L),
        L>=X,assertz(record(new_topic,Tn)),set(Tn),!.
increment_set(T):-Tn=T+1,increment_set(Tn).

/* Detects end of course */
inc:-mode_is(instruct),maxtop(M),
        record(top_seen_total,X),
        !,X>=M,assertz(reached(end_of_course)),!.
inc:-mode_is(choice),maxtop(M),record(seen_known,Z),
        !,Z>=M,assertz(reached(end_of_course)),!.

/* Sets flags for INSTRUCT */
set(T):-mode_is(instruct),record(last_topic,X),
        top(X,Ux,Mx,_,_),top(T,U,M,_,_),U<>Ux,
        assertz(reached(new_unit)),
        assertz(record(unit_done,Ux)),M<>Mx,
        assertz(reached(new_mod)),
        assertz(record(mod_done,Mx)),!.

/* Sets flags for  CHOICE */
set(T):-mode_is(choice),record(last_topic,X),
        top(X,Ux,Mx,_,_),top(T,U,_,_,_),
        U<>Ux,assertz(reached(new_unit)),
        assertz(record(unit_done,Ux)),
        getchoicelist(L,units),L=[],
        assertz(reached(new_mod)),
        assertz(record(mod_done,Mx)),!.
set(_).

/* Gets choice list of mods or units in CHOICE */

getchoicelist(L,mods):-
        getchoicelist2(L1,mods),L1=[],
        findall(M,modl(M,_,_,_,_),L2),
        delete(0,L2,L3),
        remove_some(L3,L3,L,mods),!.
```

```
getchoicelist(L,units):-
        getchoicelist2(L1,units),
        L1=[],record(this_mod,M),
        findall(U,unit(U,M,_,_,_),L2),
        delete(0,L2,L3),
        remove_some(L3,L3,L,units),!.

getchoicelist(L,ModUn):-getchoicelist2(L,ModUn),!.

/* Gets list of destination modules and units
   from last mod/unit, substitutes for those covered
   or known their own destination lists, removes
   those covered, known or at wrong level */

getchoicelist2(L,mods):-record(this_mod,M),
        modl(M,_,_,L1,_),!,subst_lists(L1,[],L2,mods),
        tailor(L2,[],L3),remove_some(L3,L3,L,mods),!.

getchoicelist2(L,firstunit):-record(this_mod,M),
        modl(M,_,_,_,L1),subst_lists(L1,[],L2,units),
        tailor(L2,[],L3),remove_some(L3,L3,L,units),!.

getchoicelist2(L,units):-record(this_unit,U),
        unit(U,_,_,_,L1),!,subst_lists(L1,[],L2,units),
        tailor(L2,[],L3),remove_some(L3,L3,L,units),!.

/* Substitutes dest. lists for those done */

subst_lists([],X,X,mods).
subst_lists([L|Lt],Q,R,mods) if record(mod_done,L),
        modl(L,_,_,X,_),!,append(X,Q,Qt),
        subst_lists(Lt,Qt,R,mods),!.
subst_lists([L|Lt],Q,R,mods) if record(mod_known,L),
        modl(L,_,_,X,_),!,append(X,Q,Qt),
        subst_lists(Lt,Qt,R,mods),!.
subst_lists([L|Lt],Q,R,mods):-subst_lists(Lt,[L|Q],R,mods),!.

subst_lists([],X,X,units).
subst_lists([L|Lt],Q,R,units) if record(unit_done,L),
        unit(L,_,_,_,X),!,append(X,Q,Qt),
        subst_lists(Lt,Qt,R,units),!.
subst_lists([L|Lt],Q,R,units) if mode_is(choice)
        and record(unit_known,L),unit(L,_,_,_,X),
        !,append(X,Q,Qt),subst_lists(Lt,Qt,R,units),!.

subst_lists([L|Lt],Q,R,units) if mode_is(choice)
        and unit(L,M,_,_,_) and
        record(mod_known,M),unit(L,_,_,_,X),
        !,append(X,Q,Qt),
        subst_lists(Lt,Qt,R,units),!.
subst_lists([L|Lt],Q,R,units):-
        subst_lists(Lt,[L|Q],R,units).

/* Tailors integer list to contain only one of each */

tailor([],X,X).
tailor([L|Lt],Q,R):-member(L,Lt),delete(L,Lt,Lx),
        tailor(Lx,[L|Q],R),!.
tailor([L|Lt],Q,R):-tailor(Lt,[L|Q],R).
```

```
/* Remove those done or at wrong level */

remove_some([],Q,Q,mods).
remove_some([],Q,Q,units).
remove_some([L|Lt],Q,R,mods) if record(mod_done,L),
        delete(L,Q,Qt),remove_some(Lt,Qt,R,mods),!.
remove_some([L|Lt],Q,R,mods) if level(X) and
        modl(L,Y,_,_,_) and Y>X,delete(L,Q,Qt),
        remove_some(Lt,Qt,R,mods),!.
remove_some([L|Lt],Q,R,mods) if mode_is(choice)
        and record(mod_known,L),
        delete(L,Q,Qt),remove_some(Lt,Qt,R,mods),!.
remove_some([_|Lt],Q,R,mods):-
        remove_some(Lt,Q,R,mods),!.

remove_some([L|Lt],Q,R,units) if record(unit_done,L),
        delete(L,Q,Qt),remove_some(Lt,Qt,R,units),!.
remove_some([L|Lt],Q,R,units) if level(X) and unit(L,_,Y,_,_)
        and Y>X,delete(L,Q,Qt),remove_some(Lt,Qt,R,units),!.
remove_some([L|Lt],Q,R,units) if mode_is(choice) and
        record(unit_known,L),delete(L,Q,Qt),
        remove_some(Lt,Qt,R,units),!.
remove_some([L|Lt],Q,R,units) if mode_is(choice) and
        unit(L,M,_,_,_) and record(mod_known,M),
        delete(L,Q,Qt),remove_some(Lt,Qt,R,units),!.
remove_some([_|Lt],Q,R,units):-remove_some(Lt,Q,R,units).

/* Changes list of integers to list of wordlists */

getlistnames([],[],_).
getlistnames([L|Lt],[Q|Qt],mods):-modl(L,_,Q,_,_),!,
        getlistnames(Lt,Qt,mods),!.
getlistnames([L|Lt],[Q|Qt],units):-unit(L,_,_,Q,_),!,
        getlistnames(Lt,Qt,units).
```

Appendix 5.14

Rules for the search option

```
set_up(search):-screen(25),
        telelinebox("06/NB/CB/02","MODULES AVAILABLE"),
        nl,findall(M,modX(_,_,M,_,_),ModlistX),
        modX(0,_,Q,_,_),delete(Q,ModlistX,ModlistY),
        rev(ModlistY,ModlistZ),
        removeL(ModlistZ,[],Modlist,module),
        telelistbox("08/CY",Q,allcaps),Z=[return,to,last,menu],
        menulist(Modlist,[Z],Module),repeat,seek(module,Module),
        !,Module=Z,!.

seek(module,Z):-Z=[return,to,last,menu].

seek(module,Module):-modX(M,_,Module,_,_),
        findall(X,unitX(_,M,_,X,_),Unitlist),
        Z=[return,to,last,menu],
        screen(25),telelinebox("06/NB/CB/04","UNITS AVAILABLE"),
        telelistbox("08/NB/CB/04",Module,allcaps),
        menulist(Unitlist,[Z],Unit),
        repeat,seek(unit,Unit),!,Unit=Z.

seek(unit,Z):-Z=[return,to,last,menu].
seek(unit,Unit):-unitX(U,_,_,Unit,_),
        findall(T,topX(_,U,_,_,T),Toplist),
        Z=[return,to,last,menu],screen(25),
        telelinebox("06/NB/CB/04","TOPICS AVAILABLE"),
        telelistbox("08/CY",Unit,allcaps),
        menulist(Toplist,[Z],Topic),
        repeat,seek(topic,Topic),!,Topic=Z.

seek(topic,Z):-Z=[return,to,last,menu].
seek(topic,Topic):-modeX(revise),topX(T,_,_,_,Topic),
        show_topic(T),tidy_up(tops,T),
        modify(increment,record,search_show,0,0),!.
seek(topic,Topic):-modeX(choice),topX(T,U,_,_,Topic),
        recordX(unit_known,U),show_topic(T),tidy_up(tops,T),
        modify(increment,record,search_show,0,0),!.
seek(topic,Topic):-modeX(choice),topX(T,_,M,_,Topic),
        recordX(mod_known,M),show_topic(T),tidy_up(tops,T),
        modify(increment,record,search_show,0,0),!.
seek(topic,Topic):-not(modeX(revise)),topX(T,_,_,_,Topic),
        recordX(top_seen,T),show_topic(T),tidy_up(tops,T),
        modify(increment,record,search_show,0,0),!.
seek(topic,_):-newscreen and message(8),getspace.

/* Removes mods/units at wrong level */

removeL([],L,L,module).
removeL([L|Lt],L2,L3,module):-modX(_,Q,L,_,_),level(X),X>=Q,
        removeL(Lt,[L|L2],L3,module).
removeL([_|Lt],L2,L3,module):-removeL(Lt,L2,L3,module).
```

Facts for the module, unit and topic identifying keywords

```
/* Identifying keyword lists. Must not have final 's', and topics
   found by one keyword must be positioned above generic keywords
   in the database, eg. 'zener' above 'diode'. */

/* Topic lists */

kw(topic,1,[conductor,conducting,conduct]).
kw(topic,2,[insulator,insulate,insulating,insulater,inserlate]).
kw(topic,5,[random,randomly,randem,randemly]).
kw(topic,6,[volt,voltage]).
kw(topic,8,[waiting,dentist,room]).
kw(topic,10,[convention]).
kw(topic,11,[plumbing,water,pipe,plumber]).
kw(topic,12,[ohm,"Ohm's","Ohms","Oms","Om's","Ohm","ohm's"]).
kw(topic,15,[double]).
kw(topic,14,[potentiometer,potentiometor]).
kw(topic,16,[power,watt,energy]).
kw(topic,18,[car,head,light]).
kw(topic,20,[emf,cell]).
kw(topic,21,[linear]).
kw(topic,22,[thermistor,thermister]).
kw(topic,24,[divider,divided,dividing]).
kw(topic,26,[stability,high]).
kw(topic,27,[sine,sin,wave,waveform]).
kw(topic,28,[peak,rms,mean,root]).
kw(topic,32,[unit,psu,supply,pack]).
kw(topic,33,[signal,generator]).
kw(topic,34,[digital,multimeter,dmm]).
kw(topic,41,[flask]).
kw(topic,42,[charging,charge,charged]).
kw(topic,44,[discharging,discharge,discharged]).
kw(topic,45,[bottle]).
kw(topic,46,[torch]).
kw(topic,48,[twins]).
kw(topic,49,[reactance,reactence,reactive]).
kw(topic,51,[cycle,frequency,frequencie,freqency]).
kw(topic,52,[series,parallel]).
kw(topic,53,[time,constant]).
kw(topic,54,[flash,gun]).
kw(topic,55,[shock]).
kw(topic,56,[coupling,couple,bypass,bypassing]).
kw(topic,59,[burst,bursting]).
kw(topic,64,[si,silicon,chip]).
kw(topic,65,[valence,valency]).
kw(topic,66,[covalent,covalency]).
kw(topic,67,[dope,doping,doped]).
kw(topic,68,[conductivity]).
kw(topic,69,["n-type",ntype]).
kw(topic,71,[hole,junction]).
kw(topic,72,[depletion,region]).
kw(topic,73,[carrier,minority]).
kw(topic,74,[reverse,forward,bias]).
kw(topic,75,[turn]).
```

```
kw(topic,84,[half,halfwave,"half-wave"]).
kw(topic,85,[full,fullwave,"full-wave"]).
kw(topic,87,[bias,reverse]).
kw(topic,89,[avalanche,snowball]).
kw(topic,90,[ball,billiard,snooker]).
kw(topic,94,[max,min,maximum,minimum]).
kw(topic,99,[amplify,amplification,amplifier]).
kw(topic,102,[multivibrator]).
kw(topic,105,[relaxation,relax]).
kw(topic,107,[switch]).
kw(topic,108,[inverter,invertor]).
kw(topic,109,[drive,base]).
kw(topic,110,[astable]).
kw(topic,114,[hooligan]).
kw(topic,117,[phase]).
kw(topic,119,[triac,diac]).
kw(topic,121,[dimmer,lamp]).

/* Following are additional topics not part of course */

kw(topic,130,[electronic,electron]).
kw(topic,131,[atom,atomic,nucleu]).

/* Unit lists */

kw(unit,2,[circuit,wiring,circit]).
kw(unit,4,[internal]).
kw(unit,3,[resistor,resistance,rheostat,resist]).
kw(unit,7,[cathode,ray,tube,crt,cro,oscilloscope]).
kw(unit,13,[semiconductor,semiconduct,semiconducter]).
kw(unit,16,[rectify,rectifying,rectified]).
kw(unit,17,[breakdown]).
kw(unit,18,[zener,zenor,zenar]).
kw(unit,14,[diode]).
kw(unit,20,[unijunction,unijuncsion,unijuntion]).
kw(unit,19,[transistor,transister,transistar,tranny]).
kw(unit,22,[thyristor,thyrister,thyristar]).

/* Module lists */

kw(module,2,[alternating,ac]).
kw(module,3,[capacitor,capacitance,capacity,condenser,condensor]).
kw(module,1,[current,dc,direct,electricity,electric,electrical]).
```

Appendix 5.16

Facts to link two lists of keywords

```
/* Keyname linking data (key module/unit/topic, number indicated,
   then two keyword lists identified which point to it:
   mod/unit/top/gen, first number, mod/unit/top/gen, second number */


kwlink(module,4,unit,13,unit,14).
kwlink(module,5,unit,19,unit,22).

kwlink(unit,1,module,1,general,2).
kwlink(unit,5,unit,3,general,1).
kwlink(unit,6,module,2,general,2).
kwlink(unit,8,module,1,general,4).
kwlink(unit,9,module,3,module,1).
kwlink(unit,10,module,3,module,2).
kwlink(unit,11,module,3,general,1).
kwlink(unit,12,module,3,general,1).
kwlink(unit,15,unit,14,general,7).
kwlink(unit,21,unit,19,general,1).
kwlink(unit,23,unit,22,general,1).

kwlink(topic,3,topic,130,topic,131).
kwlink(topic,4,topic,1,topic,30).
kwlink(topic,7,unit,2,topic,130).
kwlink(topic,12,topic,6,module,1).
kwlink(topic,13,unit,3,general,2).
kwlink(topic,17,unit,4,general,2).
kwlink(topic,19,unit,2,unit,4).
kwlink(topic,23,topic,17,general,1).
kwlink(topic,25,topic,14,general,1).
kwlink(topic,29,topic,27,general,3).
kwlink(topic,30,unit,7,general,2).
kwlink(topic,31,unit,7,general,1).
kwlink(topic,35,unit,1,general,4).
kwlink(topic,36,unit,2,general,4).
kwlink(topic,38,unit,1,unit,2).
kwlink(topic,39,module,3,general,2).
kwlink(topic,40,unit,1,module,3).
kwlink(topic,47,module,2,module,3).
kwlink(topic,50,topic,49,module,3).
kwlink(topic,61,topic,6,module,3).
kwlink(topic,62,module,3,general,7).
kwlink(topic,70,unit,14,general,2).
kwlink(topic,76,unit,3,unit,14).
kwlink(topic,77,unit,14,general,7).
kwlink(topic,78,unit,14,general,6).
kwlink(topic,79,unit,14,general,1).
kwlink(topic,80,unit,16,general,2).
kwlink(topic,81,topic,84,unit,16).
```

```
kwlink(topic,83,topic,85,unit,16).
kwlink(topic,88,unit,17,unit,18).
kwlink(topic,91,unit,18,general,2).
kwlink(topic,93,unit,18,unit,2).
kwlink(topic,96,unit,18,general,1).
kwlink(topic,97,unit,18,general,1).
kwlink(topic,98,unit,19,general,2).
kwlink(topic,100,topic,99,module,2).
kwlink(topic,101,unit,2,unit,19).
kwlink(topic,103,unit,20,general,2).
kwlink(topic,106,unit,19,general,1).
kwlink(topic,112,unit,22,general,2).
kwlink(topic,115,unit,22,general,6).
kwlink(topic,118,unit,22,general,1).
kwlink(topic,120,module,1,unit,22).
```

Appendix 5.17

Facts for additional general keyword groupings required

```
kw(general,1,[use,using,practical,experiment]).
kw(general,2,[explain,what,how,why,work,show,tell,nature]).
kw(general,3,[other,another,some,next]).
kw(general,4,[measure,measuring,find,determine,measurement]).
kw(general,5,[revise,revision,recap,revising,revised]).
kw(general,6,[plot,graph,characteristic,curve]).
kw(general,7,[test,tested,testing,fault]).
kw(general,8,[care,danger,explode,careful]).
```

Appendix 5.18

Rules controlling analysis of keyword input

```
/* Test for keyword */

analyse(Input,Keyword,0,x) if test(Keyword,Testlist),
        member(X,Input),member(X,Testlist),!.

/* Test for keyname */

analyse(Input,keyname,Number,Type) if
        keyname_found(Input,Number,Type),!.
analyse(_,nothing,0,x).

/* These three test for a keyname sublist of input sentence */

keyname_found(S,K,topic) if topX(K,_,_,_,X),sublist(X,S),!.
keyname_found(S,K,unit) if unitX(K,_,_,X,_),sublist(X,S),!.
keyname_found(S,K,module) if modX(K,_,X,_,_),sublist(X,S),!.

/* Removes dead wood from input sentence and tests for keyname */

keyname_found(S1,K,Type) if
        take_out(1,S1,S2),take_out(1,S2,S3),take_out(1,S3,S4),
        take_out(2,S4,S5),take_out(2,S5,S6),take_out(2,S6,S7),
        singular(S7,S8),count(S8,N),getkeyname(N,S8,K,Type).

/* Takes final 's' off any words with one to make them singular */

singular([],[]).
singular([L|Ls],[P|Ps]):-lop_S(L,P),singular(Ls,Ps).

lop_S(S,P):-string_chlist(S,C),rev(C,C_rev),C_rev=[C1|Cs],
        ch_cap(C1,C2),C2='S',rev(Cs,Ct),chlist_string(Ct,P).
lop_S(S,S).

/* Take out common 'leave' and 'ignore' words, up to three of
   each, all occurrences */

take_out(1,L,R):-member(X,L),changeX(leave,"",X),delete(X,L,R),!.
take_out(2,L,R):-member(X,L),changeX(ignore,"",X),delete(X,L,R),!.
take_out(_,L,L).

/* If one word remains,try it against single keywords,
   all singular */

getkeyname(1,[X],No,topic) if kwX(topic,No,List),member(X,List),!.
getkeyname(1,[X],No,unit) if kwX(unit,No,List),member(X,List),!.
getkeyname(1,[X],No,module) if kwX(module,No,List),member(X,List),!.

/* Next try pairs of linked keywords in the sentence */

getkeyname(_,S,N,topic):-kwlinkX(topic,N,K1,N1,K2,N2),
        getlist(K1,N1,L1),getlist(K2,N2,L2),
        member(X,S),member(X,L1),member(Y,S),member(Y,L2),!.
```

```
getkeyname(_,S,N,unit):-kwlinkX(unit,N,K1,N1,K2,N2),
        getlist(K1,N1,L1),getlist(K2,N2,L2),
        member(X,S),member(X,L1),member(Y,S),member(Y,L2),!.
getkeyname(_,S,N,module):-kwlinkX(module,N,K1,N1,K2,N2),
        getlist(K1,N1,L1),getlist(K2,N2,L2),
        member(X,S),member(X,L1),member(Y,S),member(Y,L2),!.

/* If pairs of keywords fail, try single keywords in the sentence */

getkeyname(_,S,N,topic):-kwX(topic,N,L),member(X,S),member(X,L),!.
getkeyname(_,S,N,unit):-kwX(unit,N,L),member(X,S),member(X,L),!.
getkeyname(_,S,N,module):-kwX(module,N,L),member(X,S),member(X,L).

/* Look-up table for the paired lists of keywords */

getlist(module,N,L):-kwX(module,N,L).
getlist(unit,N,L):-kwX(unit,N,L).
getlist(topic,N,L):-kwX(topic,N,L).
getlist(general,N,L):-kwX(general,N,L).
```

Appendix 6.1

(i) An approach to student answers based on uncertainty factors, as in MYCIN (Shortliffe, 1976)

All answers in a five-choice multiple choice question are regarded as significant in providing information about the student, and may be coded with something akin to a "degree of correctness". The coding used by Shortliffe involved a "measure of belief" between 0 and 1, and a corresponding "measure of disbelief" between 0 and -1. Shortliffe allowed for conflicting evidence, which could indicate at the same time both greater belief and greater disbelief, but here, where the belief is in the student's understanding as indicated by the answer to a question, it seems reasonable to assume that questions and answers can be framed to remove this ambiguity.

The measure of belief or disbelief (MB and MD), based on a new piece of evidence, is adjusted by an attenuation factor between 0 and 1 indicating the reliability of the new evidence. Here this might represent the general hardness of the question. To take an extreme example, the fact that an O-level student scores low on a degree level question does not tell us that he or she is unable - the evidence is inappropriate unless it is adjusted to allow for the level.

After each student answer, the overall measure of belief (MB) is updated according to:

$$MB[h:e1,e2] = MB[h:e1] + MB[h:e2] \times (1-MB[h:e1])$$

h is the hypothesis that the student understands the material taught, based on a new piece of evidence e2 and all previous evidence e1. The overall measure of disbelief (MD, negative) is updated in a similar way. A certainty factor CF is arrived at by simply combining the overall MB and overall MD as follows:

$$CF[h:e] = MB[h:e] - MD[h:e].$$

Some mathematicians are unhappy about the validity of the certainty factor approach. Most are rather happier about using probabilities.

(ii) Effect of answer parameter P(E:H) on hypothesis probability P(H:E).

When P(E:H) = 0, P(H!E) = 0. When P(E:H) is large, P(H:E) approaches 1.

A value of P(E.H) = 0.2 in the version of Bayes Theorem given here leaves

$$P(H:E)new = \frac{P(E:H).P(H:E)old}{P(E:H).P(H:E)old + 0.2(1 - P(H:E)old)}$$

$$= \frac{0.2.\ P(H:E)\ old}{0.2.P(H:E)old + 0.2\ (1-P(H:E)old)}$$

$$P(H:E)\ new = P(H:E)old.$$

Values of P(E:H) greater than 0.2 increase P(H:E), and vice versa.

Appendix 6.2

Computer simulations of student performance using the probability assessment method

(i) Program in BBC Basic to perform simulations of a student's performance, using the probability assessment method.

```
 10REM STUDENT SIMULATION
 20REM Takes in answer parameters as P(E:H) up to 50
 30 REM or until 0 entered
 40REM Returns values of P(H:E), Odds(H:E), and ln(Odds(H:E))
 50DIM PEH(50)
 60DIM PHE(51)
 70MODE 3
 80PROCheader:PRINT"Enter value of P(H): "
 90INPUT P1:PRINT:PHE(1)=P1
100REM Read in values
110PRINT"Type in P(E:H) values: "
120PRINT
130M=0
140T=0
150REPEAT
160M=M+1
170INPUT INPVAL
180PEH(M)=INPVAL
190T=T+INPVAL
200UNTIL M=50 OR INPVAL=0
210CLS:PROCheader:PRINT"P(H) is ";PHE(1)
220IF M<>1 THEN PRINT"Average P(E:H) is ";T/(M-1)
230PRINT"P(E:notH) is taken as 0.2"
240REM Calculate new P(H:E) using last P(H:E)
250REM as new P(H) and print out values
260PRINT
270PRINT"    Answer","    P(E:H)","    P(H:E)"," Odds(H:E)"
280N=2:IF M<>1 THEN REPEAT
290REM Calculate P(H:E)
300R=PHE(N-1)
310X=PEH(N-1)*R
320Y=0.2*(1-R)
330Z=X/(X+Y)
340PHE(N)=Z
350REM Calculate odds
360Q=Z/(1-Z)
370Z2=(INT(100*Z))/100
380Z3=(INT(Q*100))/100
390REM Calculate log(odds)
400Z4=(INT(100*LN(Q)))/100
410PRINT N-1,PEH(N-1),Z2,Z3,Z4
420N=N+1
430UNTIL Q>500 OR N=M+1
440END
450DEFPROCheader
460PRINT"Probability Assessment Simulation"
470PRINT"================================":PRINT
480ENDPROC
```

(ii) Sets of random answer parameters for simulations of a below-average student, an average student, and an above-average student, generated by MICROTAB for the BBC.

```
.IRANDOM 10 FROM 15 TO 21 IN C1

.MULTIPLY C1 BY 0.01

.NAME C1 'EHVALS'

.PRINT C1 C2

ROW   EHVALS        C2
  1     0.18
  2     0.19
  3     0.20
  4     0.18
  5     0.20
  6     0.18
  7     0.20
  8     0.18
  9     0.19
 10     0.18

.IRANDOM 17 TOO FROM 17 TO 23 IN C1

.MULTIPLY C1 BY 0.01

.PRINT C1 C2

ROW    EHVALS        C2
  1     0.22
  2     0.23
  3     0.20
  4     0.23
  5     0.18
  6     0.19
  7     0.18
  8     0.23
  9     0.17
 10     0.20

.IRANDOM 10 FROM 19 TO 25 IN C1

.MULTIPLY C1 BY 0.01

.PRINT C1 C2

ROW    EHVALS        C2
  1     0.22
  2     0.24
  3     0.25
  4     0.25
  5     0.20
  6     0.23
  7     0.21
  8     0.23
  9     0.21
 10     0.21
```

(iii) The answer parameters generated in (ii) processed by the Basic program in (i).

P(H) is 0.5
Average P(E:H) is 0.188
P(E:notH) is taken as 0.2

Answer	P(E:H)	P(H:E)	Odds(H:E)	LnO(H:E)
1	0.18	0.47	0.9	-0.11
2	0.19	0.46	0.85	-0.16
3	0.2	0.46	0.85	-0.16
4	0.18	0.43	0.76	-0.27
5	0.2	0.43	0.76	-0.27
6	0.18	0.4	0.69	-0.37
7	0.2	0.4	0.69	-0.37
8	0.18	0.38	0.62	-0.48
9	0.19	0.37	0.59	-0.53
10	0.18	0.34	0.53	-0.63

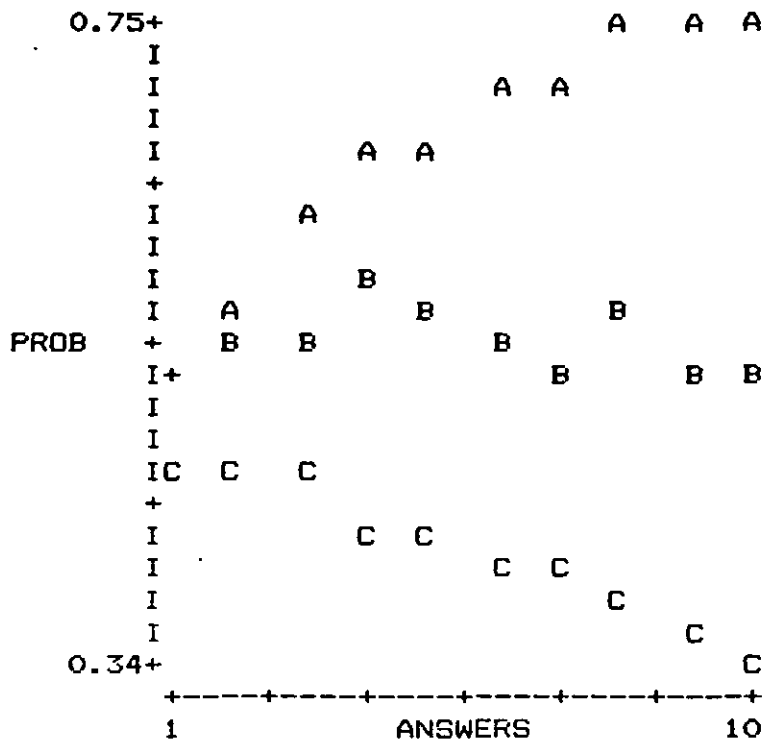
P(H) is 0.5
Average P(E:H) is 0.203
P(E:notH) is taken as 0.2

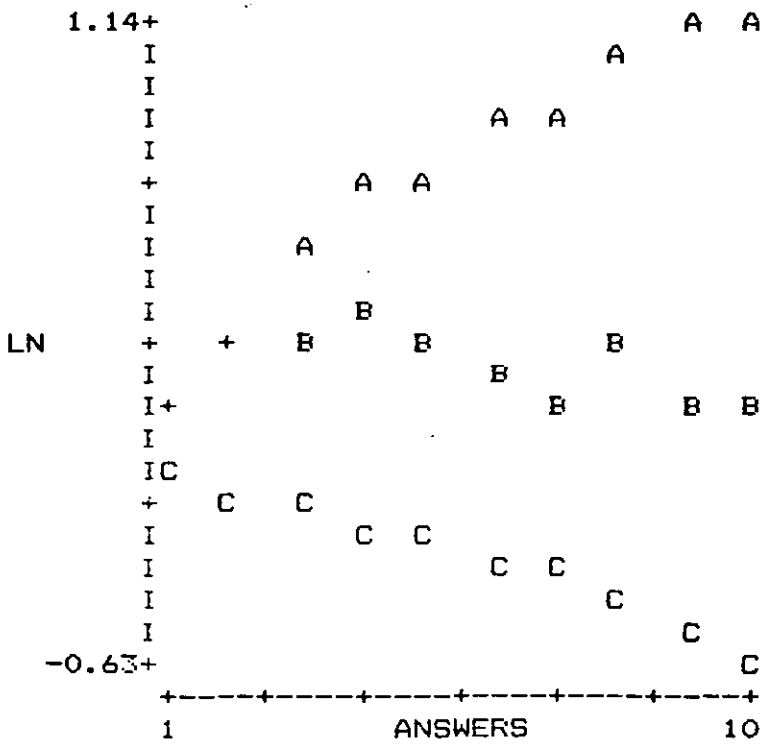
Answer	P(E:H)	P(H:E)	Odds(H:E)	LnO(H:E)
1	0.22	0.52	1.1	9E-2
2	0.23	0.55	1.26	0.23
3	0.2	0.55	1.26	0.23
4	0.23	0.59	1.45	0.37
5	0.18	0.56	1.3	0.26
6	0.19	0.55	1.24	0.21
7	0.18	0.52	1.11	0.11
8	0.23	0.56	1.28	0.25
9	0.17	0.52	1.09	9E-2
10	0.2	0.52	1.09	9E-2

P(H) is 0.5
Average P(E:H) is 0.225
P(E:notH) is taken as 0.2

Answer	P(E:H)	P(H:E)	Odds(H:E)	LnO(H:E)
1	0.22	0.52	1.1	9E-2
2	0.24	0.56	1.32	0.27
3	0.25	0.62	1.65	0.5
4	0.25	0.67	2.06	0.72
5	0.2	0.67	2.06	0.72
6	0.23	0.7	2.37	0.86
7	0.21	0.71	2.49	0.91
8	0.23	0.74	2.86	1.05
9	0.21	0.75	3	1.1
10	0.21	0.75	3.15	1.14

(iv) Graphs of the student performances simulated in (iii), as produced by MICROTAB.





Appendix 6.3

Listing of the Electronics questions used in the trials of WITS (in the form of a written test as given to Group C; see Chapter 7)

SOLID STATE ELECTRONICS TEST

These questions are all multiple choice.
In each case, you choose the best answer from A, B, C, D or E.

The questions may be different from multiple choice questions you have done before.
They do not consist of one correct and four incorrect answers.
There is always a best answer, but sometimes other answers are near to it and also correct. Other answers may be factually correct but not good answers. Others may be quite wrong.

The questions may sometimes seem harder than the usual ones, because there are several close answers. However, you get credit for every answer, and if you do not pick the best one, picking a near answer may be nearly as good. You should aim at doing the best you can each time so as to get a good average.

Some things you may find useful for certain questions are:

a) Rough paper.
b) A calculator.

Question 1

Which of the following depends on transistors for its operation?

A. A radio.
B. A computer.
C. A transformer.
D. A TV set.
E. A An electric bell.

Question 2

Transistors have replaced the early thermionic valves in electronic circuits. Which of the following advantages of transistors do you consider most important?

A. They are smaller.
B. They are cheaper.
C. They do not need heaters.
D. They work at low voltages.
E. They are more robust and reliable.

Question 3

From your knowledge of electricity so far, which of the following statements is most correct?

A. Some light bulbs contain nitrogen, BECAUSE the gas cools the filament.
B. Light bulbs have tungsten filaments BECAUSE tungsten has a high melting point.
C. Glass is used for light bulbs BECAUSE it is a good insulator.
D. Two resistors always have more resistance together than separately.
E. You should not short a battery BECAUSE you might get hurt.

Question 4

When materials are given a static charge of electricity by rubbing them, a positively charged material has:

A. More than its normal number of electrons.
B. A deficiency of electrons.
C. Extra protons.
D. Positive ions.
E. Positive holes.

Question 5

Which of the following statements are you most sure is correct?

A. Good conductors of electricity have atoms with easily removed electrons.
B. The number of electrons and nuclear charges in an atom are always equal.
C. The outer electrons in metallic conductors like copper are moving randomly for most of the time.
D. Insulators are substances whose electrons cannot leave the atoms.
E. Electricity only flows through solids with 'free' electrons.

Question 6

A thin copper wire and a thick copper wire are connected across a battery in series. Which of the following are you most sure will be the same in both wires?

A. The current.
B. The electricity passing a cross-section of the wire in one second.
C. The number of electrons per cc.
D. The electron velocity.
E. The electrons passing a cross-section of the wire in one second.

Question 7

A column of mercury in a tube passes a current of 1A when a certain potential difference is applied. What current will the same length of mercury pass in a tube of half the diameter, with the same pd?

A. 0.25 A.
B. 0.50 A.
C. 1.00 A.
D. 2.00 A.
E. 4.00 A.

Question 8

Which of the following explains resistance best?

A. The resistance of a uniform wire is proportional to its length.
B. Resistance is inversely proportional to area of cross-section.
C. The unit of resistivity is the ohm meter.
D. Voltage is resistance times current.
E. Some materials have zero resistance at Absolute zero.

Question 9

A resistor of R ohms is connected in a circuit between points X and Y. Suppose we need to measure the current I in R without breaking the circuit. We can:

A. Connect an ammeter between X and Y. Then I = ammeter reading.
B. Connect an ammeter between X and R. Then I = ammeter reading.
C. Connect an ammeter between X and Y. Then I = R x ammeter reading.
D. Connect a voltmeter between X and Y. Then I = R x voltmeter reading.
E. Connect a voltmeter between X and Y. Then I = voltmeter reading / R.

Question 10

A 24 ohm resistor uses 12W. A 16 ohm resistor in parallel with it, rated at 50 W, will use:

A. More watts.
B. Less watts.
C. 8 W.
D. 18 W.
E. 50 W.

Question 11

A rheostat set to a large resistance and a lamp, shining brightly, are connected in parallel to an accumulator. When the resistance of the rheostat is halved, the bulb will:

A. Shine much more brightly, and probably burn out.
B. Shine a little more brightly.
C. Change very little in brightness.
D. Shine very slightly less brightly.
E. Shine much less brightly.

Question 12

Three 3 ohm resistors are arranged in a triangle and a 12 V battery of internal resistance 2 ohms is connected to two corners of the triangle. The current from the battery is:

A. 6.0 A.
B. 3.0 A.
C. 3.4 A.
D. 1.1 A.
E. 4.0 A.

Question 13

How can a potentiometer be used to measure current?

A. A potentiometer cannot be used to measure current.
B. By passing the current through the potentiometer.
C. By balancing it against a known current in a resistance.
D. By passing it through a galvanometer connected to the potr.
E. By passing it through a known resistance and balancing the pd produced.

Question 14

Which of the following will NOT function if AC is used instead of DC?

A. An electric lamp.
B. An electric bell.
C. The heating element of an electric fire.
D. A voltameter used for electro-plating by electrolysis.
E. A meter for measuring current or voltage.

Question 15

Which of the following statements is most useful in measuring alternating currents and voltages?

A. Hot-wire meters measure rms values.
B. Moving coil meters can be adapted to measure AC.
C. Oscilloscope traces allow peak values to be found directly.
D. The meter wire potentiometer can be used to measure AC.
E. Voltameters (not voltmeters) cannot be used to measure AC.

Question 16

The electrons which form the beam in a cathode ray tube are produced by:

A. Making the screen positive.
B. Heating the filament.
C. Making the anode positive.
D. Making the 'grid' positive.
E. Making the focussing electrodes negative.

Question 17

20 V DC across the plates of an oscilloscope gives a deflection of 15 mm. An AC voltage produces a straight line 60 mm long. What is the root mean square value of the AC signal?

A. 16 V.
B. 28 V.
C. 32 V.
D. 40 V.
E. 56 V.

Question 18

If you had to modify a cathode ray tube to give a brighter picture, for example to use in bright daylight, which of the following do you think would be best? (Without damaging the cathode ray tube!)

A. Reverse the voltage across the anode and the cathode.
B. Increase the voltage across the anode and cathode slightly.
C. Increase the voltage across the anode and cathode considerably.
D. Increase the current through the cathode filament considerably.
E. Increase both anode voltage and filament current slightly.

Question 19

A meter has 5 ohms resistance, and reads full-scale when 100 microvolts is applied. How can it be adapted to read 1 milliamp full-scale?

A. Put a resistor in series with it.
B. Put 10 ohms in series with it.
C. Put a resistor in parallel with it.
D. Put 0.1 ohms in parallel with it.
E. Put 10 ohms in parallel with it.

Question 20

A sensitive meter used to detect a current has a resistance of 5 ohms and takes 0.015 A at full-scale. It is used in a circuit with a voltage of about 3 V. How can a resistor be used to protect the meter?

A. 190 ohms in parallel.
B. 190 ohms in series.
C. 200 ohms in parallel.
D. 200 ohms in series.
E. 250 ohms in series.

Question 21

Which factor is most important in the design of an efficient capacitor?

A. A large overlapping area for the 'plates'.
B. The conductivity of the substance between the plates.
C. The conductivity of the plates.
D. The permittivity of the substance between the plates.
E. The separation of the plates.

Question 22

A charged capacitor consisting of two metal plates with a dielectric or insulator between them:

A. Has positive charge on both plates.
B. Has positive charge on one plate.
C. Has a positive charge on one plate and a negative charge on the other.
D. Holds the charge in the dielectric.
E. Is not really charged at all, because the charges are equal and cancel each other out.

Question 23

AC is passed through a capacitor and a resistor in series. The ends of the capacitor are connected to the X plates of an oscilloscope, the ends of the resistor to the Y plates. What trace will be seen?

A. A circle.
B. An ellipse.
C. A figure of eight.
D. A straight line.
E. A sine curve.

Question 24

Three capacitors of 2 microfarads each can be joined in five ways: all 3 in series, all 3 in parallel, 1 in series with 2 in parallel with each other, 2 in series with each other in parallel with the third, and 2 in

parallel with each other in series with the third. Which capacitance (in microfarads) is NOT close to that of any combination?

A. 0.67.
B. 1.33.
C. 3.0.
D. 5.0.
E. 6.0.

Question 25

In which of the following do electrons carry the current:

A. A battery.
B. p-type semi-conductor material.
C. n-type semi-conductor material.
D. An electrolytic cell.
E. A cathode ray oscilloscope.

Question 26

When a current flows through a conductor in a magnetic field, a voltage (the Hall voltage) is produced across the sides of the conductor. What do you think this voltage will be proportional to?

A. The mean charge of the current carriers.
B. The resistivity of the material.
C. The concentration of the carriers.
D. The thickness of the conductor.
E. The sum of the rates of flow of all the carriers.

Question 27

A diode is connected in series with a battery, a rheostat and a milliammeter, and in parallel with a voltmeter. As the current is increased, the voltage hardly changes until a certain current is reached, then it increases suddenly. Which of the following summarises this diode behaviour?

A. It allows no current one way.
B. When reverse biased, it allows no current at first,then breaks down.
C. It allows current freely one way.
D. When forward biased,it first allows current freely, then breaks down.
E. The voltage across the diode is proportional to its resistance.

Question 28

A bridge rectifier is used to convert a moving coil dc meter to an AC meter. The current which passes through the meter is:

A. A steady direct current.
B. Intermittent bursts of current one way.
C. Current both ways but more one way.
D. A half-wave rectified current.
E. A full-wave rectified current.

Question 29

The zener effect in a zener diode is caused by:

A. Free electrons.
B. Electrons pulled out of atoms by an applied voltage.
C. Electrons removed from covalent bonds linking atoms by a voltage.
D. Electrons knocked out of atoms by other electrons.
E. All these factors.

Question 30

A zener diode has a breakdown voltage of 7.5 V and a maximum power rating of 1 W. This means:

A. It will be damaged by a voltage greater than 7.5 V.
B. It will start to conduct at a voltage greater than 7.5 V.
C. It will be damaged by currents greater than 0.1 A.
D. It will be damaged by currents greater than 0.133 A.
E. It will be damaged by currents greater than 0.15 A.

Question 31

Transistors (bipolar) are devices which:

A. Have three terminals.
B. Have a base, emitter and collector.
C. Can magnify small changes of current.
D. Are made from a sandwich of semi-conductors.
E. Have a source, drain and gate.

Question 32

One of the 'characteristic' curves for a transistor is approximately a straight line through the origin. What are plotted on the X and Y axes?

	X	Y
A.	V(C)	I(C)
B.	I(C)	I(B)
C.	V(B)	I(B)
D.	V(B)	I(B)

E. No such characteristic curve.

V=voltage, I=current, B=base, C=collector

Question 33

Why is it necessary to bias a transistor used in an amplifier to have a particular operating point on the mutual characteristic curve?

A. Otherwise, there will be distortion.
B. Otherwise, the current may 'go off' the curve.
C. There is one point where the curve is nearly linear.
D. The current must not go negative.
E. Otherwise, part of the current may be 'chopped off'.

Question 34

How does an astable multivibrator circuit operate?

A. It has no stable state.
B. It switches back and forth between two states of its own accord.
C. It remains in one state until switched to the other by a signal.
D. It always returns to one of its states after a certain time.
E. It has two transistors which are always in opposite states.

Question 35

Which of the following best explains a unijunction transistor?

A. It has two base terminals.
B. It has three terminals.
C. It operates as a 'switch' triggered by a certain voltage.
D. 'Triggering' causes the input impedance to become very small.
E. It has three terminals, emitter, base 1 and base 2.

Question 36

Which of the following can a transistor in a circuit be used for, better than any other device?

A. Make radios.
B. Draw characteristic curves.
C. Magnify small current changes.
D. Switch currents on and off.
E. Rectify currents.

Question 37

A transistor is connected with an oscillator across its E and B terminals, and a battery and speaker across E and C. With other simple components added, this could be used for:

A. Making a siren.
B. Testing hearing ranges.
C. Making an electric organ.
D. Making a burglar alarm.
E. Testing the transistor.

Question 38

When soldering a transistor into a circuit, its lead is held with a pair of pliers. This is because:

A. The lead has to be held somehow.
B. It earths any static electricity.
C. It prevents the transistor being heated.
D. It stops heat damaging the transistor.
E. It keeps electricity from the soldering iron away from the transistor by earthing it.

Question 39

The most useful distinguishing feature of a thyristor is:

A. It has three terminals, anode, cathode and gate.
B. It is a type of diode.
C. It is also known as the silicon-controlled rectifier or scr.
D. The current switched on by a gate current cannot be switched off by the opposite process.
E. It has four layers of semiconductor material.

Question 40

A burglar alarm can be constructed using a switch which sends a current to the gate of a thyristor, whose anode-cathode current then operates a bell. What makes the thyristor particularly suitable for such an alarm?

A. It is more reliable than using a relay or unijunction transistor.
B. Any number of window and door switches can be used.
C. Once on, the thyristor current cannot be stopped by switching the gate current off.
D. The thyristor takes a smaller current than a relay.
E. This is an all-electronic device, with no moving parts but the bell.

Appendix 6.4

Examples of the coding of questions in WITS

```
m(q,11,5,"Which of the following depends on").
m(q,11,6,"transistors for its operation?").

m(ch,11,6,"A. A radio.").
m(ch,11,8,"B. A computer.").
m(ch,11,10,"C. A transformer.").
m(ch,11,12,"D. A TV set.").
m(ch,11,14,"E. A An electric bell.").

m(h,11,6,"Transistors are used to form").
m(h,11,7,"amplifiers and 'gates'. Which").
m(h,11,8,"contain these?").

m(ex,11,6,"Radios and TV sets depend on").
m(ex,11,7,"transistors, but a computer ").
m(ex,11,8,"contains thousands, so this is ").
m(ex,11,9,"the best answer.").

m(q,12,3,"Transistors have replaced the early").
m(q,12,4,"thermionic valves in electronic").
m(q,12,5,"circuits. Which of the following").
m(q,12,6,"advantages of transistors do you").
m(q,12,7,"consider most important?").

m(ch,12,6,"A. They are smaller.").
m(ch,12,8,"B. They are cheaper.").
m(ch,12,10,"C. They do not need heaters.").
m(ch,12,12,"D. They work at low voltages.").
m(ch,12,14,"E. They are more robust and reliable.").

m(h,12,7,"All are important! Take your choice.").

m(ex,12,7,"All are important, but most would").
m(ex,12,8,"probably attribute the success of").
m(ex,12,9,"the 'solid state' transistor to its").
m(ex,12,10,"robustness and reliability.").
```

```
m(q,13,4,"From your knowledge of electricity so").
m(q,13,5,"far, which of the following statements").
m(q,13,6,"is most correct?").

m(ch,13,6,"A. Some light bulbs contain nitrogen,").
m(ch,13,7,"   BECAUSE the gas cools the filament.").
m(ch,13,8,"B. Light bulbs have tungsten filaments").
m(ch,13,9,"   BECAUSE tungsten has a high melting").
m(ch,13,10," point.").
m(ch,13,11,"C. Glass is used for light bulbs").
m(ch,13,12,"   BECAUSE it is a good insulator.").
m(ch,13,13,"D. Two resistors always have more res-").
m(ch,13,14,"   istance together than separately.").
m(ch,13,15,"E. You should not short a battery ").
m(ch,13,16,"   BECAUSE you might get hurt.").

m(h,13,7,"Which has the best reason to").
m(h,13,8,"explain the first part?").

m(ex,13,5,"B has the best reason for the").
m(ex,13,6,"statement. C and E are factually").
m(ex,13,7,"correct but have poor reasons.").
m(ex,13,8,"Glass is used for light bulbs").
m(ex,13,9,"mainly because it is transparent.").
m(ex,13,10,"Batteries are damaged by shorting,").
m(ex,13,11,"though a spark could burn you. A").
m(ex,13,12,"and D have incorrect reasons.").
```

Appendix 6.5

Facts containing answer data for the questions used in WITS

```
/* ( Qspecification [nsc,vsc,gsc], Level [all 6 in prototype],
     Qnumber, EarliestSuitableTopic, TopicFoundBy, Qtype,
     HighestApplicVariable, BestAns, AnsTypeA, ATB, ATC, ATD, ATE)

     e.g. quest(nsc,6,31,1,1,6,app,'D',i,i,n,b,i). */

     quest(nsc,6, 11,  1,  1,10,und,'B',n,b,i,n,i).
     quest(nsc,6, 12,  1,  1,15,ins,'E',n,n,n,n,b).
     quest(nsc,6, 13,  2,  2, 3,und,'B',i,b,c,i,c).
     quest(nsc,6, 14,  3,  3, 7,und,'B',i,b,i,c,n).
     quest(nsc,6, 15,  5,  5,14,und,'C',n,n,b,n,c).
     quest(nsc,6, 21,  7,  7,10,ins,'A',b,n,i,i,n).
     quest(nsc,6, 22, 12, 12, 1,ins,'A',b,i,i,i,i).
     quest(nsc,6, 31, 13, 13,12,und,'D',n,n,c,b,c).
     quest(nsc,6, 32, 13, 13, 3,app,'E',i,c,i,c,b).
     quest(nsc,6, 33, 16, 16, 2,app,'D',c,i,i,b,i).
     quest(nsc,6, 41, 20, 20, 6,app,'D',i,i,n,b,i).
     quest(nsc,6, 42, 20, 20, 1,app,'B',i,b,i,i,i).
     quest(nsc,6, 51, 25, 25, 6,ins,'E',i,i,n,i,b).
     quest(nsc,6, 61, 27, 27, 2,und,'D',i,i,i,b,c).
     quest(nsc,6, 62, 28, 28,14,und,'C',c,n,b,c,c).
     quest(nsc,6, 71, 30, 30, 2,und,'B',i,c,i,i,b).
     quest(nsc,6, 72, 31, 31, 1,app,'B',i,b,i,i,i).
     quest(nsc,6, 73, 31, 31, 4,ins,'E',i,c,c,c,b).
     quest(nsc,6, 81, 35, 35, 3,ins,'D',i,i,c,b,c).
     quest(nsc,6, 82, 35, 35, 7,ins,'D',i,c,i,b,n).
     quest(nsc,6, 91, 43, 43,10,und,'A',b,i,i,n,n).
     quest(nsc,6, 92, 45, 45, 8,und,'C',i,c,b,c,n).
     quest(nsc,6,101, 50, 50, 7,app,'B',n,b,i,c,i).
     quest(nsc,6,102, 52, 52, 1,app,'D',i,i,i,b,i).
     quest(nsc,6,131, 69, 69, 3,und,'E',i,c,c,i,b).
     quest(nsc,6,141, 73, 73, 5,ins,'E',c,c,c,c,b).
     quest(nsc,6,151, 78, 78, 3,app,'D',i,c,i,b,c).
     quest(nsc,6,161, 83, 83, 6,und,'E',i,n,i,i,b).
     quest(nsc,6,181, 92, 92, 2,und,'C',i,i,b,c,i).
     quest(nsc,6,182, 94, 94, 7,app,'D',i,n,i,b,c).
     quest(nsc,6,191, 98, 98, 4,rec,'B',c,b,c,c,i).
     quest(nsc,6,192,100,100, 4,und,'B',c,b,c,c,i).
     quest(nsc,6,193,101,101, 9,app,'C',n,c,b,c,c).
     quest(nsc,6,194,102,102,10,und,'B',n,b,i,i,n).
     quest(nsc,6,201,104,104, 9,und,'C',c,c,b,n,c).
     quest(nsc,6,211,106,106, 9,und,'C',n,c,b,c,c).
     quest(nsc,6,212,106,106,12,ins,'B',n,b,c,c,n).
     quest(nsc,6,213,106,106, 8,ins,'D',c,c,n,b,i).
     quest(nsc,6,221,114,114,12,ins,'D',n,c,c,b,n).
     quest(nsc,6,231,118,118, 7,ins,'C',i,c,b,i,n).
```

Appendix 6.6

Rules for the timing of questions

```
/* Decides if question due and acts accordingly */

check_for_question:-modeX(instruct),recordX(student_cat,C),
      recordX(tops_from_q,T),maxtop(M),
      recordX(top_known_total,K),
      student_tops(C,M,K,N,_),T>=N,
      modify(add,reached,question,0,0),
      screen(3),message(540),!.
check_for_question:-not(modeX(instruct)),recordX(student_cat,C),
      recordX(tops_from_q,T),maxtop(M),
      recordX(top_known_total,K),student_tops(C,M,K,N,_),
      T>=N,screen(3),message(500),
      getyesno(X),quest_choice(1,X),!.
check_for_question:-newscreen,message(550).

/* Question accepted immediately in CHOICE or REVISE */

quest_choice(1,'Y'):-modify(add,reached,question,0,0),
      screen(3),message(37),!.

/* Question refused in REVISE */

quest_choice(1,'N'):-modeX(revise),recordX(refusals,R),
      R<5,modify(increment,record,refusals,0,0),
      modify(update,record,tops_from_q,0,0),
      newscreen and message(510),!.
quest_choice(1,'N'):-modeX(revise),recordX(refusals,R),R>=5,
      modify(increment,record,refusals,0,0),
      newscreen and message(520),
      getyesno(X),quest_choice(2,X),!.
quest_choice(2,'Y'):-modeX(revise),
      modify(add,reached,question,0,0),
      screen(3),message(37),!.
quest_choice(2,'N'):-modeX(revise),
      modify(update,record,tops_from_q,0,0),
      modify(increment,record,refusals,0,0),
      newscreen and message(510),!.

/* Question refused in CHOICE */

quest_choice(1,'N'):-modeX(choice),
      recordX(current_refusals,R),R<3,
      modify(increment,record,refusals,0,0),
      modify(update,record,tops_from_q,0,0),
      modify(increment,record,current_refusals,0,0),
      newscreen and message(510),!.
quest_choice(1,'N'):-modeX(choice),recordX(current_refusals,R),
      R>=3,modify(increment,record,current_refusals,0,0),
      modify(increment,record,refusals,0,0),
      newscreen and message(52),
      getyesno(Y),quest_choice(2,Y),!.
```

```
quest_choice(2,'Y'):-modeX(choice),
      modify(add,reached,question,0,0),
      modify(update,record,current_refusals,0,0),
      screen(3),message(37),!.
quest_choice(2,'N'):-modeX(choice),
      modify(increment,record,refusals,0,0),
      modify(increment,record,current_refusals,0,0),
      changemode(revise),recordX(last_topic,T),
      increment_set(T),save_file.
```

Rules for the selection of questions

```
/* Selects a question on the topics recently covered.
   Fails if none available. */

assess1(select_quest,Question,' '):-
        findall(T,recordX(topic_since_q,T),L),
        get_quest_list(L,[],List),
        findall(Z,recordX(quest_done,Z),Newlist),
        subtract(Newlist,List,Nextlist),count(Nextlist,N),N>0,
        rand(N,M),getatomnumber(M,Nextlist,Question),!.

/* Selects any other suitable question, randomising */

assess1(select_quest,Question,' '):-
        findall(Q,quest(_,_,Q,_,_,_,_,_,_,_,_,_,_,_),L1),
        findall(Z,recordX(quest_done,Z),L2),subtract(L1,L2,L3),
        remove_unsuitable(L3,[],L),count(L,N),N>0,
        rand(N,M),getatomnumber(M,L,Question),!.

/* Selects any suitable question as found */

assess1(select_quest,Question,' '):-
        quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),
        not(recordX(quest_done,Q)),
        suitable_topic(T),Question=Q.

/* Gets list of all questions on recent topics. */

get_quest_list([],L,L).
get_quest_list([L|Lt],Runlist,List):-
        findall(X,quest(_,_,X,L,_,_,_,_,_,_,_,_,_),Qlist),
        append(Runlist,Qlist,Nextlist),
        get_quest_list(Lt,Nextlist,List).

/* Removes unsuitable questions from a possible list */

remove_unsuitable(L,_,L) if modeX(choice),!.

remove_unsuitable([],L,L):-!.

remove_unsuitable([L|Lt],Worklist,Newlist) if modeX(choice),
        quest(_,_,L,T,_,_,_,_,_,_,_,_,_),recordX(top_known,T),
        Nework=[L|Worklist],
        remove_unsuitable(Lt,Nework,Newlist),!.

remove_unsuitable([L|Lt],Worklist,Newlist):-
        quest(_,_,L,T,_,_,_,_,_,_,_,_,_),recordX(top_seen,T),
        Nework=[L|Worklist],
        remove_unsuitable(Lt,Nework,Newlist),!.

remove_unsuitable([_|Lt],Worklist,Newlist):-
        remove_unsuitable(Lt,Worklist,Newlist).
```

```
/* Subtracts lists, 1 from 2 to leave 3 - LOCAL */

subtract([],L,L).
subtract([L|Ls],Lst,List):-member(L,Lst),
       delete(L,Lst,X),subtract(Ls,X,List).
subtract([L|Ls],Lst,List):-not(member(L,Lst)),
       subtract(Ls,Lst,List).

/* Questions opted for */

chosen_quest('Z'):-newscreen and message(510),!.

chosen_quest('A'):-assess1(select_quest,Q,' '),
       ch_quest('B',give_quest,Q,""),
       newscreen,message(510),!.

chosen_quest(Option):-Option<>'A',newscreen,message(143),
        get_input(S),analyse(S,K,N,Type),
        ch_quest(Option,K,N,Type),!.

chosen_quest(_):-newscreen and message(54).

ch_quest('B',keyname,T,topic):-
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),
       suitable_topic(T),not(recordX(quest_done,Q)),
       ch_quest('B',give_quest,Q,""),!.

ch_quest('B',keyname,U,unit):-findall(X,topX(X,U,_,_,_),L),
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),member(T,L),
       suitable_topic(T),not(recordX(quest_done,Q)),
       ch_quest('B',give_quest,Q,""),!.

ch_quest('B',keyname,M,module):-findall(X,topX(X,_,M,_,_),L),
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),member(T,L),
       suitable_topic(T),not(recordX(quest_done,Q)),
       ch_quest('B',give_quest,Q,""),!.

ch_quest('C',keyname,T,topic):-
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),
       recordX(quest_done,Q),ch_quest('C',give_quest,Q,""),!.


ch_quest('C',keyname,U,unit):-findall(X,topX(X,U,_,_,_),L),
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),member(T,L),
       recordX(quest_done,Q),ch_quest('C',give_quest,Q,""),!.

ch_quest('C',keyname,M,module):-findall(X,topX(X,_,M,_,_),L),
       quest(_,_,Q,T,_,_,_,_,_,_,_,_,_),member(T,L),
       recordX(quest_done,Q),ch_quest('C',give_quest,Q,""),!.

ch_quest('B',give_quest,Q,""):-set_up(quest),bound(Q),
       assess1(present_quest,Q,A),bound(A),
       modify(increment,record,total_quests,0,0),
       assess1(record_quest,Q,A),assess1(update_probs,Q,' '),
       assess2(update_stud_cat,0,' '),
       modify(add,record,quest_done,Q,0),getspace,
       set_up(after_questions),newscreen and message(510),!.

ch_quest('C',give_quest,Q,""):-set_up(quest),bound(Q),
```

```
        assess2(present_quest,Q,A),bound(A),getspace,
        set_up(after_questions),newscreen,message(510),!.

ch_quest(_,keyname,_,_):-newscreen and message(54),!.
ch_quest(_,_,_,_):-newscreen and message(144).

/* Determines whether a question is suitable */

suitable_topic(T) if modeX(revise),topX(T,_,_,_,_),!.
suitable_topic(T) if modeX(choice) and recordX(top_known,T),!.
suitable_topic(T) if recordX(top_seen,T).
```

Appendix 6.8

Rules for the presentation of questions

```
/* Presents a question N collecting response X.
   Levels and question types not yet catered for. */

assessl(present_quest,N,X):-disc(send,"VIDEO OFF"),
       present(ch,N),present(q,N),screen(22),
       getkeypress(['A','B','C','D','E'],X),
       quest(_,_,N,_,_,_,_,B,_,_,_,_,_,_),
       modify(increment,record,total_answers,0,0),
       studentanswer(1,N,X,B).

/* Clauses for student's first answer */

studentanswer(1,N,X,X):-present(mess,60),getspace,
       present(ex,N),!.
studentanswer(1,N,X,B):-find_probs(N,_,_,X,_,_,_,_,_,n),
       present(mess,61),str_char(S,B),
       concat("The best answer was ",S,Z),
       cursor(9,1),write(Z),getspace,present(ex,N),!.
studentanswer(1,N,X,B):-find_probs(N,_,_,X,_,_,_,_,_,c),
       present(mess,65),getspace,present(h,N),
       modify(increment,record,total_answers,0,0),
       getkeypress(['A','B','C','D','E'],Y),
       studentanswer(2,N,Y,B),!.
studentanswer(1,N,X,B):-find_probs(N,_,_,X,_,_,_,_,_,i),
       present(mess,62),getspace,present(h,N),
       modify(increment,record,total_answers,0,0),
       getkeypress(['A','B','C','D','E'],Y),
       studentanswer(2,N,Y,B),!.

/* Rules for student's second answer after a hint */

studentanswer(2,N,X,X):-present(mess,60),
       getspace,present(ex,N),!.
studentanswer(2,N,_,B):-str_char(S,B),present(mess,63),
       cursor(8,1),write("The best answer was ",S,"."),
       getspace,present(ex,N).

/* Clauses for placing questions on the screen.
   q = main question, ch = choice part, h = hint,
   ex = explanation. */

present(q,N):-shiftwindow(2),clearwindow,
       window_attr(112),subpresent(q,N),!.
present(ch,N):-shiftwindow(5),clearwindow,
       window_attr(23),subpresent(ch,N),!.
present(h,N):-shiftwindow(9),clearwindow,
       window_attr(48),subpresent(h,N),!.
present(ex,N):-shiftwindow(9),clearwindow,
       window_attr(64),subpresent(ex,N),!.
```

```
present(mess,N):-shiftwindow(9),clearwindow,
       window_attr(104),subpresent(mess,N).

/* Secondary presentation clauses */

subpresent(q,N):-
        mX(q,N,M,Line),X=M-3,field_str(X,10,37,Line),fail,!.
subpresent(ch,N):-
        mX(ch,N,M,Line),X=M-4,field_str(X,2,38,Line),fail,!.
subpresent(h,N):-
        mX(h,N,M,Line),X=M-4,field_str(X,1,35,Line),fail,!.
subpresent(ex,N):-
        mX(ex,N,M,Line),X=M-4,field_str(X,1,35,Line),fail,!.
subpresent(mess,N):-
        mX(m,N,M,Line),X=M-5,field_str(X,1,35,Line),fail,!.
subpresent(A,B).
```

Appendix 6.9

Rules for updating hypothesis probabilities

```
/* Goes through recorded answers updating probabilities */

assess(update_all_probs,0,' '):-pu_value(P),
       update(recall,P),update(understanding,P),
       update(application,P),update(insight,P),
       update(ideal_recall,P),update(ideal_understanding,P),
       update(ideal_application,P),update(ideal_insight,P),
       update(worst_recall,P),update(worst_understanding,P),
       update(worst_application,P),update(worst_insight,P),
       findprobs(2,set_to_zero,0,0,0,0,0,0,0,0),
       modify(update,record,total_quests,0,0),
       update_answers(set_to_zero),
       modify(retractall,record,change_to_cat,0,0),
       modify(update,record,student_cat,3,0),
       assess1(update_all_probs,0,'Z').
assess(update_all_probs,0,' '):-!.

assess(update_all_probs,0,'Z'):-recordX(quest_done,Quest),
       assess1(update_all_probs,Quest,' '),!.
assess(update_all_probs,0,'Z'):-
       assess2(update_stud_cat,0,' '),!.

assess(update_all_probs,Quest,' '):-
       write(" Updating Question ",Quest),nl,
       assess1(update_probs,Quest,' '),!,fail.
assess(update_all_probs,Quest,' '):-!.

/* Updates the student probabilities with the latest answer */

assess(update_probs,Quest,' '):-retrieve_ans(Quest,Ans),
       quest(_,_,Quest,_,_,Q_Type,HiStudVar,Bestans,_,_,_,_,_),

       /* Update absolute student parameters */

       find_probs(Quest,Q_Type,HiStudVar,Ans,Ra,Ua,Aa,Ia,AnsType),
       findprobs(2,HiStudVar,0,0,0,0,0,0,0,0),
       modify(increment,record,total_quests,0,0),
       update_answers(AnsType),studentX(recall,R),
       studentX(understanding,U),
       studentX(application,A),studentX(insight,I),calc(R,Ra,Rn),
       calc(U,Ua,Un),calc(A,Aa,An),calc(I,Ia,In),update(recall,Rn),
       update(understanding,Un),update(application,An),
       update(insight,In),

       /* Update ideal student parameters */

       find_probs(Quest,Q_Type,HiStudVar,Bestans,Rb,Ub,Ab,Ib,_),
       studentX(ideal_recall,Ri),studentX(ideal_understanding,Ui),
       studentX(ideal_application,Ai),studentX(ideal_insight,Ii),
       calc(Ri,Rb,Rni),calc(Ui,Ub,Uni),
       calc(Ai,Ab,Ani),calc(Ii,Ib,Ini),
       update(ideal_recall,Rni),update(ideal_understanding,Uni),
```

```
     update(ideal_application,Ani),update(ideal_insight,Ini),

     /* Update worst student parameters */

     find_probs(0,Q_type,HiStudVar,'W',Rw,Uw,Aw,Iw,w),
     studentX(worst_recall,Rww),studentX(worst_understanding,Uww),
     studentX(worst_application,Aww),studentX(worst_insight,Iww),
     calc(Rww,Rw,Rnw),calc(Uww,Uw,Unw),
     calc(Aww,Aw,Anw),calc(Iww,Iw,Inw),
     update(worst_recall,Rnw),update(worst_understanding,Unw),
     update(worst_application,Anw),update(worst_insight,Inw).

/* Finds probs for a question, given a certain answer.
   Args: Quest,Q_type,HighestStudVar,Ans,4 probabilities,
   AnswerType. */

find_probs(Q,T,HSV,'A',Pb1,Pb2,Pb3,Pb4,AT):-
     quest(_,_,Q,_,_,_,_,_,AT,_,_,_,_),
     quest_type(T,AT,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4),!.

find_probs(Q,T,HSV,'B',Pb1,Pb2,Pb3,Pb4,AT):-
     quest(_,_,Q,_,_,_,_,_,_,AT,_,_,_),
     quest_type(T,AT,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4),!.

find_probs(Q,T,HSV,'C',Pb1,Pb2,Pb3,Pb4,AT):-
     quest(_,_,Q,_,_,_,_,_,_,_,AT,_,_),
     quest_type(T,AT,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4),!.

find_probs(Q,T,HSV,'D',Pb1,Pb2,Pb3,Pb4,AT):-
     quest(_,_,Q,_,_,_,_,_,_,_,_,AT,_),
     quest_type(T,AT,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4),!.

find_probs(Q,T,HSV,'E',Pb1,Pb2,Pb3,Pb4,AT):-
     quest(_,_,Q,_,_,_,_,_,_,_,_,_,AT),
     quest_type(T,AT,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4),!.

find_probs(0,Q_type,HSV,'W',Pb1,Pb2,Pb3,Pb4,w):-
     quest_type(Q_type,_,P1,P2,P3,P4),
     findprobs(1,HSV,P1,P2,P3,P4,Pb1,Pb2,Pb3,Pb4).

/* Finds probabilities, using as args HighestStudentVariable
   and 8 probs. The 4 probs for that studvar go in, applicable
   ones come out, with 0.2 for the others. */

findprobs(1,rec,P1,A,B,C,P1,0.2,0.2,0.2).
findprobs(1,und,P1,P2,A,B,P1,P2,0.2,0.2).
findprobs(1,app,P1,P2,P3,A,P1,P2,P3,0.2).
findprobs(1,ins,P1,P2,P3,P4,P1,P2,P3,P4).

findprobs(2,rec,0,0,0,0,0,0,0,0):-
     modify(increment,record,recall_count,0,0),!.
findprobs(2,und,0,0,0,0,0,0,0,0):-
     modify(increment,record,recall_count,0,0),
     modify(increment,record,understanding_count,0,0),!.
findprobs(2,app,0,0,0,0,0,0,0,0):-
```

```
     modify(increment,record,recall_count,0,0),
     modify(increment,record,understanding_count,0,0),
     modify(increment,record,application_count,0,0),!.
findprobs(2,ins,0,0,0,0,0,0,0,0):-
     modify(increment,record,recall_count,0,0),
     modify(increment,record,understanding_count,0,0),
     modify(increment,record,application_count,0,0),
     modify(increment,record,insight_count,0,0),!.
findprobs(2,set_to_zero,0,0,0,0,0,0,0,0):-
     modify(update,record,recall_c,0),
     modify(increment,record,insight_count,0,0),!.
findprobs(2,set_to_zero,0,0,0,0,0,0,0,0):-
     modify(update,record,recall_count,0,0),
     modify(update,record,understanding_count,0,0),
     modify(update,record,application_count,0,0),
     modify(update,record,insight_count,0,0).

/* Records answers given for each question */

record_quest(Q,'A'):-modify(add,record,gaveafor,Q,0),!.
record_quest(Q,'B'):-modify(add,record,gavebfor,Q,0),!.
record_quest(Q,'C'):-modify(add,record,gavecfor,Q,0),!.
record_quest(Q,'D'):-modify(add,record,gavedfor,Q,0),!.
record_quest(Q,'E'):-modify(add,record,gaveefor,Q,0).

/* Gets answer given for a recorded question */

retrieve_ans(Q,'A'):-recordX(gaveafor,Q),!.
retrieve_ans(Q,'B'):-recordX(gavebfor,Q),!.
retrieve_ans(Q,'C'):-recordX(gavecfor,Q),!.
retrieve_ans(Q,'D'):-recordX(gavedfor,Q),!.
retrieve_ans(Q,'E'):-recordX(gaveefor,Q).

/* Updates Best, Near, Correct and Incorrect Ans Totals */

update_answers(b):-modify(increment,record,best_ans,0,0),!.
update_answers(n):-modify(increment,record,near_ans,0,0),!.
update_answers(c):-modify(increment,record,correct_ans,0,0),!.
update_answers(i):-modify(increment,record,incorrect_ans,0,0),!.
update_answers(set_to_zero):-modify(update,record,best_ans,0,0),
     modify(update,record,near_ans,0,0),
     modify(update,record,correct_ans,0,0),
     modify(update,record,incorrect_ans,0,0).
```

Appendix 6.10

Facts for the fifteen possible combinations of answer types

```
quest_type(1,i,0.19,0.18,0.17,0.15). /* X 4 */
quest_type(1,b,0.25,0.29,0.33,0.38).

quest_type(2,i,0.18,0.18,0.16,0.15). /* X 3 */
quest_type(2,c,0.20,0.19,0.18,0.17).
quest_type(2,b,0.25,0.28,0.33,0.38).

quest_type(3,i,0.18,0.17,0.16,0.15). /* X 2 */
quest_type(3,c,0.20,0.19,0.18,0.17). /* X 2 */
quest_type(3,b,0.25,0.28,0.32,0.37).

quest_type(4,i,0.18,0.17,0.16,0.15).
quest_type(4,c,0.19,0.19,0.18,0.16). /* X 3 */
quest_type(4,b,0.24,0.27,0.32,0.36).

quest_type(5,c,0.19,0.18,0.17,0.16). /* X 4 */
quest_type(5,b,0.24,0.27,0.31,0.36).

quest_type(6,i,0.18,0.16,0.15,0.13). /* X 3 */
quest_type(6,n,0.23,0.25,0.27,0.29).
quest_type(6,b,0.24,0.26,0.29,0.32).

quest_type(7,i,0.17,0.16,0.14,0.13). /* X 2 */
quest_type(7,c,0.19,0.18,0.16,0.14).
quest_type(7,n,0.22,0.24,0.27,0.29).
quest_type(7,b,0.24,0.26,0.29,0.32).

quest_type(8,i,0.17,0.16,0.14,0.13).
quest_type(8,c,0.19,0.17,0.16,0.14). /* X 2 */
quest_type(8,n,0.22,0.24,0.27,0.28).
quest_type(8,b,0.23,0.25,0.28,0.31).

quest_type(9,c,0.19,0.17,0.15,0.14). /* X 3 */
quest_type(9,n,0.22,0.23,0.26,0.28).
quest_type(9,b,0.23,0.25,0.28,0.31).

quest_type(10,i,0.17,0.15,0.13,0.11).        /* X 2 */
quest_type(10,n,0.22,0.23,0.24,0.25).        /* X 2 */
quest_type(10,b,0.23,0.24,0.26,0.28).

quest_type(11,i,0.17,0.15,0.13,0.11).
quest_type(11,c,0.18,0.16,0.14,0.12).
quest_type(11,n,0.21,0.22,0.24,0.25).        /* X 2 */
quest_type(11,b,0.23,0.24,0.25,0.27).

quest_type(12,c,0.18,0.16,0.14,0.12).        /* X 2 */
quest_type(12,n,0.21,0.22,0.24,0.24).        /* X 2 */
quest_type(12,b,0.22,0.24,0.25,0.27).
```

```
quest_type(13,i,0.16,0.14,0.12,0.10).
quest_type(13,n,0.21,0.21,0.22,0.22).        /* X 3 */
quest_type(13,b,0.22,0.23,0.23,0.24).

quest_type(14,c,0.17,0.15,0.13,0.11).
quest_type(14,n,0.20,0.21,0.22,0.22).        /* X 3 */
quest_type(14,b,0.22,0.22,0.23,0.24).

quest_type(15,n,0.20,0.20,0.20,0.20).        /* X 4 */
quest_type(15,b,0.21,0.21,0.21,0.22).
```

Appendix 6.11

Rule to calculate the updated probability

```
/* Calculates new value of a student probability parameter from
   the old value and the question parameters,  using Bayes'
   Theorem */

calc(Oldprob,Q_prob,Newprob):-
     Newprob = (Q_prob * Oldprob) /
                  ((Q_prob * Oldprob) + (0.2 * (1  - Oldprob))).
```

Appendix 6.12

Rules for the student categories

```
/* Rule for updating student category */

assess(update_stud_cat,0,' '):-studentX(understanding,U),
     studentX(application,A),studentX(insight,I),
     studentX(recall,R),studentX(ideal_recall,Ri),
     studentX(ideal_understanding,Ui),
     studentX(ideal_application,Ai),
     studentX(ideal_insight,Ii),
     studentX(worst_recall,Rw),
     studentX(worst_understanding,Uw),
     studentX(worst_application,Aw),
     studentX(worst_insight,Iw),
     calc_vals(R,Ri,Rw,_,Rab),calc_vals(U,Ui,Uw,_,Uab),
     calc_vals(A,Ai,Aw,_,Aab),calc_vals(I,Ii,Iw,_,Iab),
     X=(Rab+Uab+Aab+Iab)/4,student_category(_,C,Lower,Upper),
     X>=Lower,X<Upper,modify(update,record,student_cat,C,0),
     modify(add,record,change_to_cat,C,0),!.


/* Facts to determine student category from percentage bands */

student_category('A',1,70,101).
student_category('B',2,55, 70).
student_category('C',3,45, 55).
student_category('D',4,30, 45).
student_category('E',5, 0, 30).
```

Appendix 6.13

Rules for pacing the student

```
/* These rules determine number of topics to be left between
   questions, and nunber of questions to give (last 2 args) */


/* Less than 1/4 course known (30 Qs in 120 tops) */

student_tops(5,Max,Known,12,3) if Known<Max div 4,!.
student_tops(Cat,Max,Known,8,2) if Known<Max div 4,Cat>2,Cat<5,!.
student_tops(Cat,Max,Known,4,1) if Known<Max div 4,Cat<3,!.

/* 1/4 to 1/2 of course known (30 Qs in 90 tops) */

student_tops(5,Max,Known,12,4) if Known<Max div 2,
        Known>=Max div 4,!.
student_tops(Cat,Max,Known,9,3) if Known<Max div 2,
        Known>=Max div 4,Cat>2,Cat<5,!.
student_tops(Cat,Max,Known,6,3) if Known<Max div 2,
        Known>=Max div 4,Cat<3,!.

/* More than 1/2 course known (30 Qs in 60 tops) */

student_tops(5,Max,Known,8,4) if Known>=Max div 2,!.
student_tops(Cat,Max,Known,6,3) if Known>=Max div 2,Cat>2,Cat<5,!.
student_tops(Cat,Max,Known,4,1) if Known>=Max div 2,Cat<3.
```

Appendix 6.14

Rule to initialise the student model

```
/* Following data is asserted then updated as required. Other
   data such as questions done, answers given, modules, units
   and topics done, etc, is asserted afresh as built up. */

init(student_model):-pu_value(P),

        /* Student variable starting values */

        assertz(student(recall,P)),assertz(student(understanding,P)),
        assertz(student(application,P)),assertz(student(insight,P)),

        /* Ideal student starting values */

        assertz(student(ideal_recall,P)),
        assertz(student(ideal_understanding,P)),
        assertz(student(ideal_application,P)),
        assertz(student(ideal_insight,P)),

        /* Worst student starting values */

        assertz(student(worst_recall,P)),
        assertz(student(worst_understanding,P)),
        assertz(student(worst_application,P)),
        assertz(student(worst_insight,P)),

        /* Counts for Qs testing different variables */

        assertz(record(recall_count,0)),
        assertz(record(understanding_count,0)),
        assertz(record(application_count,0)),
        assertz(record(insight_count,0)),

        /* Running data for course */

        assertz(record(student_cat,3)),
        assertz(record(this_mod,0)),
        assertz(record(this_unit,0)),
        assertz(record(last_topic,0)),
```

```
/* Course data preserved when changing mode */

assertz(record(instruct_this_mod,0)),
assertz(record(instruct_this_unit,0)),
assertz(record(instruct_last_topic,0)),
assertz(record(choice_this_mod,0)),
assertz(record(choice_this_unit,0)),
assertz(record(choice_last_topic,0)),

/* Data for Qs */

assertz(record(refusals,0)),
assertz(record(current_refusals,0)),
assertz(record(tops_from_q,0)),
assertz(record(quests_count,0)),

assertz(record(total_quests,0)),
assertz(record(total_answers,0)),
assertz(record(top_seen_total,0)),
assertz(record(top_known_total,0)),
assertz(record(seen_known,0)),

assertz(record(best_ans,0)),
assertz(record(near_ans,0)),
assertz(record(correct_ans,0)),
assertz(record(incorrect_ans,0)),

/* Data for student handling of course */

assertz(record(mode_changes,0)),
assertz(record(info_count,0)),
assertz(record(misunderstandings,0)),
assertz(record(skip_count,0)),
assertz(record(profile_count,0)),
assertz(record(keyname_count,0)),
assertz(record(search_count,0)),
assertz(record(search_show,0)),
assertz(record(key_show,0)).
```

Appendix 6.15

Typical student file saved by the program

```
mode_is("choice")
record("instruct_time_hrs",0)
record("instruct_time_mins",0)
record("instruct_time_secs",0)
record("revise_time_hrs",0)
record("revise_time_mins",0)
record("revise_time_secs",0)
record("john",1000)
record("smith",2000)
record("a little",3000)
record("instruct_this_mod",0)
record("instruct_this_unit",0)
record("instruct_last_topic",0)
record("choice_this_mod",0)
record("choice_this_unit",0)
record("choice_last_topic",0)
record("refusals",0)
record("current_refusals",0)
record("mode_changes",0)
record("skip_count",0)
record("search_show",0)
record("top_known_total",0)
record("top_seen",1)
record("top_seen",2)
record("top_seen",3)
record("search_count",1)
record("top_seen",4)
record("top_seen",5)
record("top_seen",6)
record("top_seen",7)
record("top_seen",8)
record("quest_done",15)
record("gaveafor",15)
record("change_to_cat",1)
record("top_seen",9)
record("unit_done",1)
record("top_seen",10)
record("top_seen",11)
record("top_seen",12)
record("unit_done",2)
record("quest_done",22)
record("gavebfor",22)
record("change_to_cat",5)
record("quest_done",11)
record("gavebfor",11)
record("change_to_cat",5)
record("quest_done",12)
record("gaveefor",12)
record("change_to_cat",4)
```

```
record("top_seen",13)
record("top_seen",14)
record("top_seen",15)
record("top_seen",16)
record("unit_done",3)
record("misunderstandings",1)
record("top_seen",17)
record("top_seen",18)
record("top_seen",19)
record("top_seen",20)
record("unit_done",4)
record("quest_done",31)
record("gavebfor",31)
record("near_ans",2)
record("change_to_cat",4)
record("quest_done",32)
record("gaveefor",32)
record("change_to_cat",3)
record("top_seen",21)
record("top_seen",22)
record("top_seen",23)
record("top_seen",24)
record("top_seen",25)
record("top_seen",26)
record("unit_done",5)
record("mod_done",1)
record("top_seen",27)
record("top_seen",28)
record("quest_done",51)
record("gaveefor",51)
record("change_to_cat",2)
record("top_seen",29)
record("unit_done",6)
record("top_seen",30)
record("top_seen",31)
record("top_seen",32)
record("quest_done",73)
record("gaveefor",73)
record("change_to_cat",1)
record("top_seen",33)
record("top_seen",34)
record("unit_done",7)
record("top_seen",35)
record("top_seen",36)
record("quest_done",81)
record("gavebfor",81)
record("application_count",6)
record("insight_count",5)
record("incorrect_ans",2)
record("change_to_cat",2)
record("top_seen",37)
record("top_seen",38)
record("unit_done",8)
record("mod_done",2)
record("this_mod",5)
record("top_seen",98)
record("top_seen",99)
record("quest_done",191)
record("gavebfor",191)
record("change_to_cat",2)
```

```
record("top_seen",100)
record("top_seen",101)
record("info_count",1)
record("top_seen",1)
record("keyname_count",8)
record("top_seen",1)
record("key_show",2)
record("quest_done",192)
record("gavebfor",192)
record("change_to_cat",2)
record("top_seen",102)
record("unit_done",19)
record("top_seen",103)
record("top_seen",104)
record("top_seen",105)
record("unit_done",20)
record("quest_done",201)
record("gaveafor",201)
record("change_to_cat",2)
record("top_seen",106)
record("top_seen",107)
record("top_seen",108)
record("top_seen",109)
record("quest_done",211)
record("gavedfor",211)
record("correct_ans",2)
record("change_to_cat",2)
record("profile_count",2)
record("top_seen",110)
record("top_seen",111)
record("unit_done",21)
record("this_unit",22)
record("framecount",2)
record("top_seen",112)
record("top_seen",113)
record("total_answers",18)
record("quest_done",13)
record("gavebfor",13)
record("recall_count",14)
record("understanding_count",13)
record("total_quests",14)
record("best_ans",8)
record("student_cat",2)
record("change_to_cat",2)
record("quests_count",0)
record("question_time_hrs",0)
record("question_time_mins",13)
record("question_time_secs",36)
record("viewing_time_hrs",1)
record("viewing_time_mins",13)
record("viewing_time_secs",26)
record("last_topic",114)
record("top_seen_total",55)
record("seen_known",55)
record("top_seen",114)
record("topic_since_q",114)
record("tops_from_q",1)
record("new_topic",115)
record("l_session_hrs",2)
record("l_session_mins",24)
```

```
record("1_session_secs",52)
record("2_session_hrs",0)
record("2_session_mins",0)
record("2_session_secs",0)
record("new_topic",115)
record("session",3)
record("new_topic",115)
record("interactions",276)
record("interaction_time_hrs",0)
record("interaction_time_mins",59)
record("interaction_time_secs",11)
record("3_session_hrs",0)
record("3_session_mins",1)
record("3_session_secs",31)
record("choice_time_hrs",2)
record("choice_time_mins",26)
record("choice_time_secs",23)
record("totaltime_hrs",2)
record("totaltime_mins",26)
record("totaltime_secs",45)
record("lastcheck_hrs",0)
record("lastcheck_mins",1)
record("lastcheck_secs",31)
student("recall",0.26046572032)
student("understanding",0.29329966375)
student("application",0.22749221312)
student("insight",0.1652754591)
student("ideal_recall",0.52082100403)
student("ideal_understanding",0.73132582758)
student("ideal_application",0.5334288871)
student("ideal_insight",0.55302867768)
student("worst_recall",0.028775692031)
student("worst_understanding",0.011409105259)
student("worst_application",0.03499742666)
student("worst_insight",0.029567854435)
```

Appendix 6.16

Rules for saving and reading back the student file

```
/* Saves student data */

save_file:-openwrite(student,"student.dba"),
        writedevice(student),write_to_file,
        writedevice(screen),closefile(student).

/* Writes terms to a file */

write_to_file:-mode_is(X),Term=mode_is(X),write(Term),nl,
        reached(X),Term2=reached(X),write(Term2),nl,fail.
write_to_file:-record(X,Y),Term=record(X,Y),
        write(Term),nl,fail.
write_to_file:-student(X,Y),Term=student(X,Y),
        write(Term),nl,fail.
write_to_file.
```

Appendix 6.17

Rules to calculate student abilities from the odds on the hypothesis

```
/* Calculates attainment and relative ability values from
   student's probability, ideal probability and worst
   probability */


calc_vals(Prob,Prob,Wprob,Att,100):-Att=Prob/(1-Prob),!.

calc_vals(Prob,Iprob,Prob,Att,0):-Att=Prob/(1-Prob),!.

calc_vals(Prob,Iprob,Wprob,Att,Rab):-
     Att = Prob / (1-Prob),
     Iatt = Iprob / (1-Iprob),
     Watt = Wprob / (1-Wprob),
     Rab = 100 * (ln(Att)-ln(Watt)) / (ln(Iatt)-ln(Watt)).
```

Appendix 6.18

Example of a Typical Student Report

STUDENT REPORT

NAME OF STUDENT:

You originally knew the course material not at all.

Your total time spent on the course has been 7 hrs, 33 mins and 13 secs in 8 sessions, and you have had 731 interactions with the system, an interaction rate of 1 interactions per minute.This does not include key presses while viewing topics.

You have seen 121 topics, and you said that you already knew 0 topics. Your total now either seen or known is 121, or 100 per cent of the course of 121 topics.

Your percentages of time spent in the three modes have been as follows:

INSTRUCT:	51.2 per cent.
CHOICE :	0 per cent.
REVISE :	48.7 per cent.

Your percentages of time spent on viewing topics, answering questions and interacting in other ways have been as follows:

VIEWING:	64.0 per cent.
QUESTIONS:	7.28 per cent.
INTERACTING:	28.6 per cent.

You have done 26 questions, and have given 41 answers altogether, that is, 15 answers required another try after a hint. Your first answers to questions have been as follows:

BEST ANSWERS:	6.
NEAR ANSWERS:	4.
CORRECT ANSWERS:	8.
INCORRECT ANSWERS:	8.

You refused questions 0 times.

Based on these answers, your performance has been not too good.

Your answers to questions give probabilities in certain areas as follows:

That you have good RECALL:	0.1238
That you have good UNDERSTANDING:	0.0878
That you are good at solving PROBLEMS:	0.0342
That you have INSIGHT in the subject:	0.0975

The probability is set at 0.1 to start, and should increase towards nearly 1 (or decrease) at a speed depending on your abilit each area. Note that the probability in a particular area may be low simply because only a few of the questions you have done tested that area, and there is little data to go on.

Your ability in each area, calculated from these probabilities, is as follows:

Ability in RECALL:	41.6 per cent	(26 questions)
Ability in UNDERSTANDING:	38.8 per cent	(26 questions)
Ability in APPLICATION:	24.7 per cent	(15 questions)
Ability in INSIGHT:	41.5 per cent	(8 questions)

These percentages are calculated on a scale on which a student who gave the best answer every time would get 100, and a student who gave the worst possible answer every time would get 0. On the basis of your performance in these different areas, we can make a few comments as follows.

On the basis of these abilities, your average performance has been average.

You scored best in insight, which is the highest of the skills measured, requiring application of your knowledge to new situations. This is a good sign for your future development in the subject.

Recall and insight were your two best areas. This is unusual, as recall is the most basic skill and insight the highest. It indicates an erratic approach - perhaps, for example, you only thought carefully with the harder questions.

You have been placed in category D, on the basis of your average ability. (The average of the four percentages above.) This category covers the range 30 to 45 per cent.

Additional course details are as follows:

You have looked at your profile 5 times.
You have skipped topics 0 times.
There have been 21 misunderstandings, when the computer was unable to understand you.
You have used the information option 2 times.
You changed mode 5 times.
You have used the search option 11 times, and have seen 73 topics while using it.
You have asked for topics by name 6 times, and have seen 0 topics in this way.

Appendix 6.19

Some rules relating to the student report

```
assessment(answers,B,N,C,I,X) if B>N,B>C,I>N,I>C,X="rather
        inconsistent, high in best answers and also incorrect
        ones.",!.
assessment(answers,B,N,C,I,"excellent.") if X=B+N,X>3*(C+I),B>N,!.
assessment(answers,B,N,C,I,"very good.") if X=B+N,X>2*(C+I),B>N,!.
assessment(answers,B,N,C,I,"good.") if X=B+N,X>2*(C+I),!.
assessment(answers,B,N,C,I,"quite good.") if X=B+N,X>C+I,!.
assessment(answers,B,N,C,I,"poor.") if X=C+I,X>2*(B+N),!.
assessment(answers,B,N,C,I,"not too good.") if X=C+I,X>1.5*(B+N),!.
assessment(answers,_,_,_,_,"about average."):-!.

assessment(average,R,U,A,I,"excellent") if
        Z=(R+U+A+I) div 4 and Z>70,!.
assessment(average,R,U,A,I,"very good") if
        Z=(R+U+A+I) div 4 and Z>55,!.
assessment(average,R,U,A,I,"good") if
        Z=(R+U+A+I) div 4 and Z>45,!.
assessment(average,R,U,A,I,"average") if
        Z=(R+U+A+I) div 4 and Z>30,!.
assessment(average,R,U,A,I,"rather poor"):-!.

assessment(ability,R,U,A,I,highestvar) if I>=R and I>=U and I>=A,
        write("You scored best in insight, which is the highest
        of the"),nl,write("skills measured, requiring application
        of your knowledge to"),nl,write("new situations. This is
        a good sign for your future development"),nl,write("in
        the subject."),nl,nl,!.

assessment(ability,R,U,A,I,highestvar) if A>=R and A>=I and A>=U,
        write("You scored best in application or problem-solving,
        one of"),nl,write("the higher skills measured. This may
        indicate an aptitude for"),nl,write("the practical,
        engineering side of the subject."),nl,nl,!.

assessment(ability,R,U,A,I,highestvar) if U>=R and U>=A and U>=I,
        write("You scored best in understanding, indicating that
        you are"),nl,write("starting to appreciate relationships
        and to see the structure"),nl,write("of the subject."),
        nl,nl,!.
```

```
assessment(ability,R,U,A,I,highestvar) if R>=U and R>=A and R>=I,
        write("You scored best in recall, which requires only the
        remembering"),nl,write("of facts. You will need to work
        on developing the higher skills"),nl,write("of
        understanding, problem-solving and insight."),nl,nl,!.

assessment(ability,R,U,A,I,general) if R>=U and R>=A and I>=U
        and I>=A,write("Recall and insight were your two best
        areas. This is unusual,"),nl,write("as recall is the most
        basic skill and insight the highest. It"),nl,
        write("indicates an erratic approach - perhaps, for
        example, you only"),nl,write("thought carefully with
        the harder questions."),nl,nl,!.

assessment(ability,R,U,A,I,general) if
        Z=(R+U+A+I) div 4 and Z>=50 and X=A+I and X>=R+U,
        write("You have done best with the higher areas of skill.
        This"),nl,write("indicates some maturity in the subject,
        and "),nl,write("promises well for your future
        development."),nl,nl,!.

assessment(ability,R,U,A,I,general) if
        Z=(R+U+A+I) div 4 and Z>=50 and X=R+U and X>=A+I,
        write("You have done best in the more basic areas of
        skill. In"),nl,write("spite of your good overall
        performance, you will need"),nl,write("to concentrate
        on the more advanced skills, namely"),nl,write("problem
        solving and the application of your knowledge"),nl,
        write("to new situations."),nl,nl,!.

assessment(ability,R,U,A,I,general) if
        Z=(R+U+A+I) div 4 and Z<50 and X=A+I and X>=R+U,
        write("You have done best with the higher areas of skill.
        This"),nl,write("indicates some degree of maturity in the
        subject, and "),nl,write("is rather at odds with your
        low overall performance."),nl,nl,!.

assessment(ability,R,U,A,I,general) if
        Z=(R+U+A+I) div 4 and Z<50 and X=R+U and X>=A+I,
        write("You have done best in the more basic areas of
        skill. This"),nl,write("indicates that you are not yet
        mature in the more advanced"),nl,write("skills, and will
        need to concentrate on them more."),nl,nl.
```

Appendix 7.1

Questionnaire Q1 to record self-assessments and opinions of WITS students

QUESTIONNAIRE

For students who have worked through the WITS Solid State Electronics program.

Name:

Please ring one response in each case.

1. Did you find the 'profile' facility which shows your progress at any point useful?

   VERY USEFUL | FAIRLY USEFUL | NO OPINION | NOT VERY USEFUL | OF NO USE

2. Did you find that consulting your 'profile' encouraged you to try to improve your results on the questions?

   YES | MAY HAVE HAD AN EFFECT | NO

3. Which of the assessments in the profile did you find most interesting, and usually looked for first?

   RECALL | UNDERSTANDING | APPLICATION | INSIGHT

   ALL EQUALLY INTERESTING | OF NO INTEREST

4. When answering questions or doing exams, do you feel that your own RECALL (that is, your memory for facts) is:

   VERY GOOD | GOOD | AVERAGE | BELOW AVERAGE | POOR

5. When answering questions or doing exams, do you feel that your basic UNDERSTANDING of Electronics (or ability to relate facts to each other and spot rules) is:

   VERY GOOD | GOOD | AVERAGE | BELOW AVERAGE | POOR

6. When answering questions or doing exams, do you feel that your ability to APPLY your knowledge (that is, ability to solve problems) is:

   VERY GOOD | GOOD | AVERAGE | BELOW AVERAGE | POOR

7. When answering questions or doing exams, do you feel that your intuition or INSIGHT (ability to deal with new knowledge and situations) is:

   VERY GOOD | GOOD | AVERAGE | BELOW AVERAGE | POOR

8. When you saw your final report from the course, did you feel it was:

VERY ACCURATE	FAIRLY ACCURATE	IRRELEVANT	NOT VERY ACCURATE	COMPLETELY WRONG

9. How did you react to the three modes of WITS, INSTRUCT, CHOICE and REVISE?

THEY WERE USEFUL	DID NOT USE THEM, JUST STAYED IN ONE MODE	THERE COULD HAVE BEEN A SIMPLER SYSTEM

10. Did you find it useful to type in keywords or sentences to ask for topics?

YES	WAS NOT AWARE YOU COULD DO THIS	ALWAYS USED THE 'SEARCH' FACILITY TO FIND THINGS INSTEAD

11. Were there things about the program you found irritating and would like to see removed? (Apart from 'bugs' in the program!) Please list any:

12. Were there things you would like to have seen included or added to make the program better? Please list any:

Appendix 7.2

Questionnaire Q2 to record teacher's assessments of WITS students

QUESTIONNAIRE

For the teacher of students who have worked through the questions of the WITS Solid State Electronics program.

Name of student:

Please ring one response in each case.

1. From your knowledge of this student, do you feel that his or her RECALL (that is, memory for facts) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

2. From your knowledge of the student, do you feel that his or her basic UNDERSTANDING of Electronics (or ability to relate facts to each other and spot rules) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

3. Do you feel that the student's ability to APPLY his or her knowledge (that is, ability to solve problems) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

4. Do you feel that the student's intuition or INSIGHT in the subject (ability to deal with new knowledge and situations) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

5. How would you assess this student in the subject, on a scale A to E ? Assume that the grades in the scale correspond to an examination performance as follows:

A	70 to 100%
B	55 to 69%
C	45 to 54%
D	30 to 44%
E	0 to 29%

(Please ring one letter.)

Appendix 7.3

Sample answer sheet for the written test

SOLID STATE ELECTRONICS TEST

Cross out the answer you consider best.

Name:

1	A	B	C	D	E
2	A	B	C	D	E
3	A	B	C	D	E
4	A	B	C	D	E
5	A	B	C	D	E
6	A	B	C	D	E
7	A	B	C	D	E
8	A	B	C	D	E
9	A	B	C	D	E
10	A	B	C	D	E
11	A	B	C	D	E
12	A	B	C	D	E
13	A	B	C	D	E
14	A	B	C	D	E
15	A	B	C	D	E
16	A	B	C	D	E
17	A	B	C	D	E
18	A	B	C	D	E
19	A	B	C	D	E
20	A	B	C	D	E

21	A	B	C	D	E
22	A	B	C	D	E
23	A	B	C	D	E
24	A	B	C	D	E
25	A	B	C	D	E
26	A	B	C	D	E
27	A	B	C	D	E
28	A	B	C	D	E
29	A	B	C	D	E
30	A	B	C	D	E
31	A	B	C	D	E
32	A	B	C	D	E
33	A	B	C	D	E
34	A	B	C	D	E
35	A	B	C	D	E
36	A	B	C	D	E
37	A	B	C	D	E
38	A	B	C	D	E
39	A	B	C	D	E
40	A	B	C	D	E

Appendix 7.4

Modified report for students in Group C

(based on their written test)

STUDENT REPORT

NAME OF STUDENT:

You have done 36 questions, and have given answers as follows:

BEST ANSWERS:	13.
NEAR ANSWERS:	4.
CORRECT ANSWERS:	6.
INCORRECT ANSWERS:	13.

Based on these answers, your performance has been rather inconsistent.

Your answers to questions give probabilities in certain areas as follows:

That you have good RECALL:	0.1754
That you have good UNDERSTANDING:	0.1584
That you are good at solving PROBLEMS:	0.0899
That you have INSIGHT in the subject:	0.2039

The probability is set at 0.1 to start, and should increase towards nearly 1 (or decrease) at a speed depending on your ability in each area. Note that the probability in a particular area may be low simply because only a few of the questions you have done tested that area, and there is little data to go on.

Your ability in each area, calculated from these probabilities, is as follows:

Ability in RECALL:	46.2 per cent	(36 questions)
Ability in UNDERSTANDING:	45.3 per cent	(35 questions)
Ability in APPLICATION:	40.4 per cent	(24 questions)
Ability in INSIGHT:	53.1 per cent	(14 questions)

These percentages are calculated on a scale on which a student who gave the best answer every time would get 100, and a student who gave the worst possible answer every time would get 0. On the basis of your performance in these different areas, we can make a few comments as follows.

On the basis of these abilities, your overall performance has been average.

You scored best in insight, which is the highest of the skills measured, requiring application of your knowledge to new situations. This is a good sign for your future development in the subject.

Recall and insight were your two best areas. This is unusual, as recall is the most basic skill and insight the highest. It indicates an erratic approach - perhaps, for example, you only thought carefully with the harder questions.

You have been placed in category C, on the basis of your average ability. This category covers the range 45 to 55 per cent.

Appendix 7.5

Questionnaire Q3 for Group C

(a modified version of Q1)

QUESTIONNAIRE

For students who have worked through the WITS Solid State Electronics program questions.

Name:

Please ring one response in each case.

1. When answering questions or doing exams, do you feel that your own RECALL (that is, your memory for facts) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

2. When answering questions or doing exams, do you feel that your basic UNDERSTANDING of Electronics (or ability to relate facts to each other and spot rules) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

3. When answering questions or doing exams, do you feel that your ability to APPLY your knowledge (that is, ability to solve problems) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

4. When answering questions or doing exams, do you feel that your intuition or INSIGHT (ability to deal with new knowledge and situations) is:

   VERY GOOD GOOD AVERAGE BELOW AVERAGE POOR

5. When you read the report on your performance in the questions, did you feel it was:

   VERY ACCURATE FAIRLY ACCURATE IRRELEVANT NOT VERY ACCURATE COMPLETELY WRONG