Developing a design specification for a computer-based embedded trainer for a navigation aid

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Master's Thesis. Submitted in partial fulfilment of the requirements for the award of Master of Philosophy at Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/27240

Publisher: © Susan Humphries

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

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

Attribution-NonCommercial-NoDerivs 2.5

You are free:

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

Under the following conditions:

**BY:** 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 licence).

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
Developing a Design Specification for a Computer Based Embedded Trainer for a Navigation Aid

by Susan Humphries

A Master’s Thesis

Submitted in partial fulfillment of the requirements for the award of

Master of Philosophy of the Loughborough University of Technology

4th August 1991

© by Susan Humphries
Developing a Design Specification for a Computer Based Embedded Trainer for a Navigation Aid

ABSTRACT

This thesis is submitted for the degree of Master of Philosophy at Loughborough University.

The thesis describes the design of a computer-based training system embedded in a navigation aid. Significantly, the range of techniques considered included so-called "intelligent" methods which allow more flexibility than conventional computer-based training techniques. Although the actual design of the training system was a desired outcome, the process was carried out with two more general objectives. One objective was to explore the advantages and disadvantages of embedded training systems. The other was to address the problem of how computer-based training systems may be designed in a methodical fashion.

The thesis discusses the advantages and disadvantages of embedded training. A conclusion is that unsupervised embedded training systems should be used with great care and probably not with beginners. However, they may be very suitable for continuation training. Their most useful application may be where people must be ready at all times to cope with emergencies, which rarely arise and therefore present few opportunities for practice.

The proposed design method is essentially pragmatic but takes account of relevant aspects of computer science and psychology. Two main features of the method are a clear development from the training objectives to the training material, and the use of Hierarchical Task Analysis. It is applied both to the navigation aid trainer, and to embedded training in general, demonstrating a principled method suitable for the design of many embedded training systems, and probably many other types of training.
The thesis also claims that the structure of the navigation trainer could be used, or adapted for use, in a wide variety of other computer-based training systems. Moreover, the elements of the trainer which are task-specific are contained within well-defined parts of the system. For a sufficiently similar training task, this means that not only could the same method be used to design the system, but some components of the navigation trainer could be re-used for training another task if the task-specific components were replaced. The re-usability of both the design process and the design structure are discussed. The navigation trainer is compared with a training system for information retrieval which was designed using an adaptation of the same method, although it is a quite different task from navigation.
4.2.4 Sequence of Events ................................................................. 89
4.3 Conclusions ............................................................................. 91
5. HIERARCHICAL TASK ANALYSIS OF NAVIGATION ................ 93
5.1 Conduct of the Task Analysis .................................................. 93
5.2 Scope ....................................................................................... 93
5.3 Assumptions .......................................................................... 94
5.4 Results .................................................................................... 95
6. DETAILED DESIGN OF THE EMBEDDED TRAINER ........... 108
6.1 Model Builder ......................................................................... 108
6.2 Plan Builder ............................................................................ 115
6.3 Executor .................................................................................. 123
6.4 Report Generator ..................................................................... 129
6.5 Tutor Interface ......................................................................... 130
6.6 Scenario Generator ................................................................. 133
6.7 Classifier ................................................................................ 134
6.8 Aid ......................................................................................... 135
7. THE ESA-QUEST TUTOR: A DEMONSTRATION OF THE ... 137
7.1 Design Approach .................................................................... 137
7.1.1 Objectives .......................................................................... 137
7.1.2 Task Description ................................................................. 137
7.1.3 Candidate Description ......................................................... 137
7.1.4 Training Material ................................................................. 138
7.2 Overview of the Demonstrator ............................................ 138
7.2.1 Methods of using the Quest Tutorial ................................. 138
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2.2 Lessons and Exercises</td>
<td>140</td>
</tr>
<tr>
<td>7.2.3 Hypertext</td>
<td>140</td>
</tr>
<tr>
<td>7.3 Structure of the System</td>
<td>141</td>
</tr>
<tr>
<td>7.3.1 Interface</td>
<td>141</td>
</tr>
<tr>
<td>7.3.2 Sequencer</td>
<td>141</td>
</tr>
<tr>
<td>7.3.3 Tutor</td>
<td>142</td>
</tr>
<tr>
<td>7.3.4 Student Model</td>
<td>142</td>
</tr>
<tr>
<td>7.4 Sequence of Events</td>
<td>143</td>
</tr>
<tr>
<td>7.5 Implementation</td>
<td>144</td>
</tr>
<tr>
<td>7.5.1 Problems Encountered</td>
<td>144</td>
</tr>
<tr>
<td>7.5.2 Testing</td>
<td>145</td>
</tr>
<tr>
<td>7.5.3 Demonstration at IRS Dialtech</td>
<td>145</td>
</tr>
<tr>
<td>7.6 The Full System</td>
<td>146</td>
</tr>
<tr>
<td>7.6.1 Demonstration at Frascati</td>
<td>146</td>
</tr>
<tr>
<td>7.6.2 General Feasibility</td>
<td>146</td>
</tr>
<tr>
<td>7.6.3 Improving the Lessons</td>
<td>147</td>
</tr>
<tr>
<td>7.6.4 Adding Exercises</td>
<td>148</td>
</tr>
<tr>
<td>7.6.5 Extended Exercises</td>
<td>148</td>
</tr>
<tr>
<td>7.6.6 Student's Own Problem</td>
<td>148</td>
</tr>
<tr>
<td>7.6.7 Link to ESA-Quest</td>
<td>149</td>
</tr>
<tr>
<td>7.6.8 Adding Test Material</td>
<td>149</td>
</tr>
<tr>
<td>7.6.9 Improving the Student Model</td>
<td>149</td>
</tr>
<tr>
<td>7.6.10 Lesson Editor</td>
<td>149</td>
</tr>
<tr>
<td>7.7 Conclusions</td>
<td>149</td>
</tr>
<tr>
<td>7.7.1 Comparison of the Structure</td>
<td>150</td>
</tr>
<tr>
<td>7.7.2 Lessons for the Navigation Trainer</td>
<td>151</td>
</tr>
<tr>
<td>8. DISCUSSION AND CONCLUSIONS</td>
<td>152</td>
</tr>
<tr>
<td>8.1 Unresolved Issues</td>
<td>152</td>
</tr>
<tr>
<td>8.1.1 Interpreting and Answering Students' Questions</td>
<td>153</td>
</tr>
<tr>
<td>8.1.2 Building the Student Model</td>
<td>154</td>
</tr>
</tbody>
</table>
8.1.3 Building the Tutorial Plan ............................................................. 154
8.1.4 Other Issues..................................................................................... 154

8.2 Training Supervision................................................................................... 155
8.3 Continuation Training................................................................................. 156
8.4 Advantages and Disadvantages of Embedded Training ...................... 157
  8.4.1 Cost and Fidelity............................................................................. 157
  8.4.2 Availability of Training Facilities during Operation ................... 158
  8.4.3 Continuation Training .................................................................... 158
  8.4.4 One-to-One Training ...................................................................... 159
  8.4.5 Updating of Training Material ...................................................... 159
  8.4.6 Conclusions about Embedded Training ........................................ 160

8.5 Generality of the Design ............................................................................ 160
  8.5.1 The Design Process ........................................................................ 160
  8.5.2 The Structure of the Trainer ............................................................ 161
8.6 Final Remarks ............................................................................................. 162

9. REFERENCES ............................................................................................ 164

FIGURES
  2.1 Classes of Feedback ........................................................................... 18
  2.2 The Flow of Instruction ........................................................................ 49
  3.1 Training Design Stages ........................................................................... 52
  4.1 Structure of the Navigation Trainer, with Scenario Generator ....... 87
  4.2 Structure of the Navigation Trainer, with real sensor data............. 89
  4.3 Sequence of Events once Training Session has started................. 90
  5.1 Main HTA for Navigation .......................................................................... 97
  5.2 Manual version of task 0.221 Get data for next step ................. 97
  5.3 Manual version of task 0.231 Receive input ..................................... 98
  5.4 Manual version of task 0.23223 Allow for tide ............................. 98
  5.5 Automatic version of task 0.221 Get data for next step ............... 98
  5.6 Automatic version of task 0.23223 Allow for tide ....................... 99
  7.1 Structure of the Quest Tutor .............................................................. 141
7.2  Sequence of Events once Training Session has started.............144

TABLES
5.1  Main tasks for navigation ..................................................100
5.2  Supplementary tasks for manual navigation ..........................103
5.3  Supplementary tasks for automatic navigation .......................106
1. INTRODUCTION

This thesis is submitted for the degree of Master of Philosophy at Loughborough University.

The thesis describes the development of a design specification for a computer-based embedded trainer for a navigation aid. An *embedded trainer* is one which is "embedded" in the operational equipment which the student is to learn to use. The training material, and the mechanisms to deliver it to the student, are regarded as being embedded in the operational system - which is invariably a computer-based, or computer-controlled, system. The training package enables the student to be trained on the operational equipment.

Frequently an embedded training package is simply a means of simulating incoming data so that the operational system can be used like a simulator, to provide controlled examples for students to practise on. More sophisticated systems may use any of the techniques of computer-based training. The navigation trainer described here is a complex system, intended to train students without supervision over several training sessions.

The practical constraints imposed by the operational equipment and its environment, and the task to be taught obviously have a large impact on the design of the training system. In this instance, the implications of the literature in both artificial intelligence and the psychology of learning and teaching have been explicitly considered as well. The hope is that this results in both a design and a design method which will be more successful than if only the practical constraints had been considered.

1.1 Historical Background

The work described in this thesis began at the Behavioural Sciences Division of the Admiralty Research Establishment (ARE) at Teddington. They were researching the feasibility of embedded training. ARE were particularly interested in the use of artificial intelligence to produce an "intelligent" trainer. In this context "intelligent" can be interpreted as "using the techniques derived from artificial intelligence research". They hoped that such techniques would provide highly flexible training systems and reduce the need for instructors to be present during
training. This was to contribute to the Navy's objective of increasing the amount of training which is done at sea.

ARE therefore commissioned a study to investigate the potential and limitations of embedded training, to discuss its feasibility, and to consider to what extent such systems could be used without an instructor being present (or even available). An element of the study was to build a small demonstration system to show what an embedded training system might look like.

A suitable computer-based system was needed to form the basis for the demonstrator. After some consideration, a navigation aid was chosen. This fulfilled another of ARE's research interests, and had the advantage that it was being designed at the time, so that the design of the main system and of the embedded training package could go on in parallel. This gives more prominence to the training material than usual. It had the disadvantage that the navigation aid was itself a research project, and therefore subject to major change.

Unfortunately, shortly after the start of the project a period of upheaval began at Teddington, as the establishment was shortly to be closed. Project management changed; the new manager was interested primarily in the navigation aid rather than embedded training. As funds were limited, it was decided that the navigation aid should be built as a demonstrator, but that the embedded training package should be merely a design exercise. As the project progressed, the situation at Teddington became worse, as staff left and support of various kinds was withdrawn. Some small experiments planned as part of the project could not be organised because of understandable lack of interest from ARE staff, who knew that neither the establishment nor the research work would continue.

However, a large part of the embedded training study was carried out before much of this took place. The final outcomes of the project were a report on the feasibility and limitations of embedded training, a demonstrator for a navigation aid, and a design for a training package to be embedded in it.

1.2 Brief Description of the Navigation Trainer

Navigation is a very practical skill. The navigation trainer was designed to provide students with plenty of practice. In normal use the navigation aid receives data from
the environment which the navigator observes and is required to react to. This data is not sufficiently controllable for most training situations, and so the embedded trainer has its own simulator which generates simulated environmental data for use during training. With the simulator running, the training system can control the events which the student must react to, and it knows exactly what those events are.

In order to give structure to the training sequence, the system builds a training plan. As it executes the plan, teaching the student, it observes the student's responses and draws conclusions about how much he knows and which tasks he can do. This information is used to build up a model of the student's abilities, and it may also be used to amend the training plan if that proves to have been based on false assumptions. The system can generate reports of the student's progress, on request.

The training system is based on a model of the navigation task constructed using Hierarchical Task Analysis. The analysis was done partly by being taught navigation at the Royal Navy's Petty Officer of the Watch navigation course, and partly by speaking to experienced navigators who are themselves instructors. The hierarchical task model underlies the student model, and is used to build the training plan. The hierarchical structure simplifies the planning process and presents a variety of options for the way in which the task may be taught.

The trainer described here may be more sophisticated than is required for another training situation. The navigation trainer must operate without supervision and therefore needs to be able to cope with a range of situations in a sensible fashion. Indeed, this report has been written without regard to resource constraints: a practical implementation would certainly need to be less ambitious than the system outlined here.

1.3 Nature of the Thesis

This thesis has been developed from the embedded training study described above. The study included the design of an embedded training system, although unfortunately that system was never built. One purpose of the thesis is to show how an embedded training system can be designed in a rational way, using an approach which is essentially pragmatic but which takes account of relevant aspects of computer science and psychology. The approach is applied both to the navigation
aid trainer, and to embedded training in general, demonstrating a principled method suitable for the design of many embedded training systems, and probably to many other types of training.

The thesis goes further than that, however. It claims that the structure of the navigation trainer could be used, or adapted for use, in a wide variety of other computer-based training systems. Moreover, the elements of the trainer which are task-specific are contained within well-defined parts of the system. For a sufficiently similar training task, this means that not only could the same method be used to design the system, but some components of the navigation trainer could be re-used for training another task if the task-specific components were replaced. The task model is necessarily specific to navigation, as is the student model which is based on it, but the existence of these elements, and the method of developing them, would apply to any subject amenable to Hierarchical Task Analysis. The parts of the system which build and execute the training plan, and the interface with the student, are more general.

In fact, this claim has been tested to a limited extent. A simple computer-based training system for information retrieval was built as a demonstrator for the European Space Agency using the method and structure described in this thesis. Obviously it was not possible to re-use any elements from the navigation trainer as that system was never built, but the structures of the two systems were very similar. Information retrieval is a very different task from navigation, and the demonstrator was not an embedded training system, but nonetheless the structure worked well with some alteration. The demonstrator was well received by ESA and is likely to be developed into a full system. The project is described in chapter 7.

For "intelligent" embedded training, the original subject of the Teddington study, the conclusions are that it should be used with great care and probably not with beginners. It is especially suitable for continuation training. Its most useful application may be where people must be ready at all times to cope with emergencies, which rarely arise and therefore present few opportunities for practice.

1.4 Organisation of the Thesis

The thesis considers embedded training systems in general, and describes the design
of an embedded training system for the DCRS navigation aid which was built at Teddington. It begins with a survey of relevant literature on psychology and computing in chapter 2. Some of the factors discussed in the literature are integrated into a model of instruction.

Chapter 3 describes the design method which will be taken, based on a systems approach. The implications of each step of the method are discussed both for embedded training in general and for the navigation trainer in particular. Chapter 4 describes the structure of the navigation trainer and shows how the structure fits in with the model of instruction developed in chapter 2. In chapter 5 the task description of navigation is given. This was prepared using Hierarchical Task Analysis, a method described in chapter 2, and one found to be very suitable. Chapter 6 describes the components of the navigation trainer in detail, including how the task description is used.

The information retrieval trainer built for ESA is described in chapter 7 as an example of the use of the design method and structure in a different context.

Finally chapter 8 describes the work which would have been done to complete the project if circumstances had allowed, and discusses the conclusions which have come out of the study. It includes the advantages and disadvantages of embedded training and shows when it should be considered as a training method.
2. LITERATURE SURVEY

In considering requirements for and constraints on a computer-based training system of any kind it would seem appropriate to survey literature in the two broad fields of psychology and computing. In psychology there is a wealth of literature on learning and teaching. Psychologists have also addressed the design and use of man-machine systems. Computer based training systems are inevitably constrained by what it is possible to achieve with current computing technology. Advances in artificial intelligence over the last few years have had an impact on research into computer-based training systems which has not yet been realised in commercially-available training systems. However, a number of training systems using techniques from artificial intelligence have been implemented by researchers and there are lessons to be learned from them.

2.1 Psychology

The main areas of psychology which are relevant to this study are the psychology of learning and teaching, especially in the context of industrial training, and the study of man-machine systems.

2.1.1 Learning

Various definitions of learning have been given. Hinrichs' (1976) seems to be typical: he defines learning as "a process by which an individual's pattern of behaviour is changed by experience". In general, an individual adapts its behaviour in order to improve its standing in the environment, both in the short term and in the long term by learning how to adapt better. The experience by which the individual changes its pattern of behaviour is knowledge of its environment and, crucially, knowledge of how its actions affect the environment. This is feedback or knowledge of results. It is an idea that has developed out of the long-standing study of conditioning and reinforcement.

2.1.1.1 Reinforcement

The contribution of behaviourism to the psychology of learning is important, and it is described in (Walker, 1975). Behaviourism started in 1913 when J.B. Watson suggested that psychologists should ignore inner feelings and thoughts. Not long afterwards, Pavlov began his famous experiments which led to the discovery of
classical conditioning, one of behaviourism's most important principles. Having discovered that dogs could be conditioned to salivate at the sound of a bell, Pavlov concluded that such conditioning is the basis of all habits, simple or complex. Also at this time Thorndike, a contemporary of Watson, stated his Law of Effect, which says that learning can take place without reasoning, by the favouring of accidental actions which turn out to have a beneficial effect. Over a series of trials an action can become so favoured ("strengthened") that it is carried out immediately it is required - it has been learnt.

By the 1930s, Skinner was working on operant conditioning. He demonstrated that he could teach a rat to carry out a complex series of actions by giving it food rewards for successively closer approximations to the required behaviour. Skinner called this process shaping, and the rewards positive reinforcers. This is rather different from classical conditioning. Loosely, classical conditioning involves the learning of an automatic response or reaction to a particular stimulus or situation, whereas in operant conditioning the subject learns to perform a deliberate action in order to achieve a goal (food, avoidance of punishment, etc). Stimulus-response (S-R) relationships are useful because the change in response to a stimulus can be accurately measured in a laboratory. Although it is difficult to generalise the idea to include any reasonably intelligent behaviour, S-R descriptions of behaviour are often useful.

Skinner made a detailed study of positive and negative reinforcement, and of punishment. Positive reinforcers are anything which we would seek, including such things as social approval and intellectual satisfaction as well as material benefits. As well as promoting learning, positive reinforcers may serve as incentives to use more effort in a behaviour already learnt. Negative reinforcers and punishment are both aversive stimuli which we would avoid, but they are distinct in their application: negative reinforcers encourage a particular behaviour as a means of escape from the reinforcer; punishments discourage a behaviour. In normal learning situations, positive reinforcers are to be strongly preferred, as they do not have the disturbing emotional side-effects which are associated with aversive stimuli.

Reinforcement has been given under a variety of schedules. It is not necessary for every response to be rewarded with a reinforcer. Rewards may be given after a fixed
number of responses (fixed ratio schedules), after a fixed interval of time (fixed interval schedules), or after a variable number of responses or interval of time (variable ratio and variable interval schedules). In experiments the usual training procedure is to start with continuous reinforcement (a fixed ratio schedule) while the behaviour is developed. After that an intermittent schedule can be introduced, although if too high a ratio is used immediately, the subject may stop responding before the required number of responses has been reached. In that case extinction of the response is said to have occurred. By gradually extending the ratio or the interval, the number of responses for each reinforcer can be increased considerably.

If (positive) reinforcement is completely withdrawn, extinction will inevitably occur, since the motivation for performing the response is the reinforcement. However, the schedule of reinforcement affects the rate of extinction: the greater the ratio or interval, the slower extinction occurs (Walker, 1975). That is to say, the less a behaviour will be rewarded, the more persistently it will be performed. One explanation of this, suggested by Amsel (1972), is that the removal of reinforcers causes frustration, which normally suppresses the response. Animals who have experienced intermittent reinforcement will have learned to tolerate such frustration, and will therefore go on responding for long periods during extinction.

However, as Walker points out, there is a cognitive difficulty in discovering that reinforcement has stopped after an intermittent schedule. If the ratio or interval is high, and especially if it is variable, it will take the subject some time to discover that reinforcement has stopped altogether.

In talking about reinforcers, behaviourists tend to ignore the information content of the feedback from an action. In most of their experiments the feedback from the experimenter serves only to favour the desired response above others. However, in most real situations the information content of the feedback is also significant: many actions are impossible to perform without some knowledge of intermediate results.

2.1.1.2 Feedback

As Annett explains, the term feedback derives from systems engineering, where a servo-mechanism adapts to its environment by means of a feedback loop which controls the operation of the mechanism according to the new state of the
environment, including the results of its previous actions. This analogy is helpful in understanding the role of feedback in learning. A servo-mechanism has a control function which determines how the feedback will affect the operation of the mechanism. Learning can be viewed as the adaptation of a control function so that the individual responds more appropriately to his environment. He cannot make this adaptation unless he knows the results of his actions.

An advantage of this view over older behaviourist views is that it closes the loop between the antecedents and the consequences of an action. Classical conditioning investigates the way that a response may be determined by an antecedent, and operant conditioning shows how an action may be performed in order to achieve its consequence. A feedback loop links these two, with the control function being the link.

Different classes of feedback have been identified. Blum and Naylor (1969) use a binary scheme as in figure 2.1.

In this scheme the broadest division of feedback is into intrinsic and extrinsic, of

```
feedback
  /\    /
intrinsic extrinsic
  /\     /
primary secondary
  /\      /
specific general augmented summary
```

Figure 2.1: Classes of Feedback

which extrinsic can be divided into primary and secondary, and so forth. The types are:

i. intrinsic: intrinsic to the organism, such as muscle tension;
ii. extrinsic: originating outside the organism;

iii. primary: intrinsic to the task;

iv. secondary: extrinsic to the task;

v. specific: specific to the action the individual performed;

vi. general: related to a group of actions or to an action as an example of its class;

vii. augmented: (of secondary feedback) immediate and detailed;

viii. summary: (of secondary feedback) delayed and general.

This nomenclature is by no means universally used, although the distinctions it makes are recognised. In particular, the training literature (including Annett) frequently regards both “intrinsic” and “primary” feedback as provided by the task, and uses the term intrinsic for both. Extrinsic feedback therefore becomes that provided for training purposes (which will not be available when the trainee graduates from the training course). This can be a more useful distinction in many cases.

Where feedback which is extrinsic to the task is used for training purposes, behaviourist theories might suggest that extinction will occur when the feedback is removed. However, this does not always happen. This is consistent with the idea that in many cases of human training feedback is primarily important for the information it provides, and that motivation is provided by other things. On the other hand, such research suggests that it would be wise to withdraw extrinsic feedback carefully, as it may have some motivating effect. (See section 2.1.2.5).

2.1.1.3 Individual Differences

In much of the research work on learning individual differences have been regarded as an effect to be overcome by careful experiment design. However, some researchers have studied individual differences in their own right. Campione et al (1984) review some of this work. They are particularly interested in the relationship between intelligence and ability to learn. Although some early experiments, especially those conducted by Woodrow (eg Woodrow, 1946) appeared to show no link between intelligence and the “amount of improvement shown by practice”, as
Woodrow carefully defined learning, Campione et al argue that this is because of the tasks chosen for these experiments. For example, Woodrow asked subjects to sort geometrical forms into boxes, and found no difference in performance between normal and retarded children. Later work suggests that no difference would be expected; Woodrow's task did not allow subjects to benefit from the use of learning strategies.

Campione et al describe several experiments in which normal and retarded children are given tasks where they can employ processing strategies (eg the use of contextual information, rehearsal, etc) and show that those children who employ these strategies perform better. This finding was confirmed by explicitly teaching the use of processing strategies to those children who did not spontaneously use them, upon which their performance improved to the level of the others. Similarly, when children were prevented from using their processing strategies (eg by giving interfering tasks to prevent rehearsal) their performance dropped markedly. By contrast, in tasks where learners cannot employ processing strategies (such as Woodrow's) no difference in performance was found between learners.

Of interest is the fact that older and more mature children employed processing strategies spontaneously, without having been taught them - such things are not normally taught in schools. However, younger and retarded children could use the strategies if they were taught to. Moreover, more mature learners are able to learn processing strategies from less detailed instruction than slower learners require. This lead Campione et al to conclude that the ability to profit from incomplete instruction is a major source of individual variation in intelligence.

Finally, Campione et al considered how well learners transfer skills from the learnt task to a similar one (see also section 2.1.2.2 below). They conclude that the ability to transfer from one task to another, and indeed the recognition that transfer is required, also distinguish learners of different abilities.

In conclusion, Campione et al, having studied the work of various researchers, including their own, suggest that quick learners spontaneously employ learning strategies, are able to profit from incomplete instruction, and can transfer what they have learnt from one task to another where appropriate. These conclusions are based almost exclusively on work with children, but it is reasonable to hypothesise that
these factors may account for some of the differences in learning ability between adults.

2.1.1.4 Skill Acquisition

Other researchers have considered the stages which an individual goes through while learning to perform a skill. For example, Fitts (1964) describes three stages of skill learning:

i. the cognitive stage, in which an understanding of the skill is learnt to the point where it can be performed at least to a crude approximation;

ii. the associative stage, in which procedures are practised, and errors in the understanding of the skill are gradually detected and eliminated;

iii. the autonomous stage, in which performance gradually improves, perhaps indefinitely.

Anderson (1981) develops these ideas under different names:

i. the declarative stage, corresponding to Fitts' cognitive stage, in which knowledge is encoded as a set of facts about the skill, which are used by general interpretive procedures;

ii. knowledge compilation, corresponding to Fitts' associative stage, a transition stage in which knowledge is converted from declarative to procedural form;

iii. the procedural stage, corresponding to Fitts' autonomous stage, in which the procedural form of the knowledge is tuned so that it will apply more appropriately and faster, and places less load on the information processing system.

Anderson incorporated these ideas into a computer simulation of some aspects of skill learning. His program (described in section 2.2.2.2 below) had considerable success in predicting students' behaviour in the course of learning to write mathematical proofs.

Shepherd et al (1977) investigated this process in students learning a complex diagnostic task. In an experiment in which students learned to diagnose faults in a process plant, they found that students who had been taught diagnostic rules were better at diagnosing unfamiliar faults than those who were given a theoretical description of the plant and its functions. They suppose that the theoretical
knowledge must be used by the student to derive diagnostic rules before it is truly useful for diagnosis. Once the diagnostic rules have been derived, the theory has served its purpose and can be safely forgotten. This process can be likened to compilation in computer programs, where the readily understandable, accessible high-level code is compiled into machine code which is difficult to understand or alter but which can be executed.

This view of learning may suggest that theoretical knowledge has no place in teaching. This idea should not be taken too seriously. To begin with, there may be no suitable rules which can be easily expressed and taught. Alternatively, the known rules may not cover every eventuality. Even if there is a complete set of rules, the theory, if it is readily understandable, may serve as a useful mnemonic while the rules are learned. Also, where more than one related topic is to be learned, the theory may apply to several of them while the rules may be specific to only one. If this is the case further considerations apply. It may be sensible to interleave two related topics so that the student can use the same theory for each before it is “compiled” and forgotten. Another consequence of this view of learning is that if theory will be useful later on it may be necessary to prevent it being forgotten once “compiled” knowledge is available.

A development of the idea of compilation of knowledge is that of goal-response distance (Shepherd and Kontogiannis, 1987). The goal-response distance is an indication of the difficulty a learner will have when constructing a response to a particular goal he has been set. A novice learner, asked to respond to a high level goal but having only low level responses at his disposal, faces a large goal-response distance which he will find very difficult to surmount. Later, the same learner may have compiled a repertoire of higher level responses in other learning situations. The goal-response distance will be smaller, and so he will find the task easier. Goal-response distance cannot be measured precisely, but there are certain instructional strategies which will reduce it or make it easier for the trainee to cope.

2.1.2 Teaching

Teaching is the process of encouraging learning. Sometimes teaching is achieved by a teacher standing up in front of a classroom and actively giving instruction to pupils; sometimes pupils are put in a situation where they can be expected to learn
and the teacher takes a back seat temporarily. The techniques available are legion. Whatever method is used, teaching consists of intentional attempts to foster learning, and to encourage particular things to be learnt.

The emphasis in teaching research is on finding out what arrangement of circumstances will be most likely to cause the student to learn the material provided. Some consideration has been given to incidental learning, where the student learns something other than what the teacher had in mind, but in general it is the problem of persuading the student to learn something specific which has received the attention.

Here it is appropriate to distinguish training from teaching in general. Hinrichs (1976) defines training as "any organisationally initiated procedures which are intended to foster learning among organisational members", where what is learnt "is intended to contribute to overall organisational objectives". The dividing line is far from clear, but generally when individuals are trained to perform a task then performance of the task itself is the ultimate goal, whereas in education general principles, learning strategies and more global learning may be more important. It is rarely possible to say that one learning situation is entirely one thing or the other. In the following discussion, the term training is generally used, but most of the material applies to more educational teaching situations as well.

Some teaching research has derived from the various models of learning, including the few which were described above. Much of it has developed fairly independently. Hinrichs (1976) says that "today there is no such thing as a complete, universally accepted, identifiable and verifiable psychology of training". The picture has not changed much since that statement was made. Generally teaching literature contains sets of principles, such as those given by Blum and Naylor (1968). These conclude that practice should be given in several sessions spaced over a period of time, that knowledge of results is probably best given immediately, that students' motivation must be kept up, and so forth.

2.1.2.1 Feedback

It is perhaps in the study of feedback that learning research has most bearing on teaching research. The view of feedback given above removes many of the logical
difficulties of thinking about knowledge of results, but it does not immediately suggest how feedback can best be manipulated by an instructor in order to foster learning. This issue has not been resolved in any conclusive way. Blum and Naylor (1968) survey some of the many experiments which have been conducted to try to gain some insight. The only clear conclusions seem to be:

- That feedback is useful to the extent that students make use of it. In more intellectual kinds of learning students may tend to ignore feedback, and the instructor may want to consider various tactics for persuading them to pay attention to it, or to pay attention to particular aspects of it (eg speed or accuracy);

- Feedback should be as immediate as possible. It should certainly be given before the student has the opportunity to forget the circumstances in which he acted. In applying this principle, however, an instructor needs to be clear exactly what learning he is trying to foster. If he would like the student to develop learning strategies he may want to consider a whole sequence of the student’s actions as a unit and hence delay some kinds of intrinsic feedback (such as “That action will lead to disaster”) until the whole unit is complete.

2.1.2.2 Transfer of Training

The other area where learning theory impinges is in the study of transfer. Transfer takes place when a student adapts what he has learnt about one task to a similar one he has not been trained to do. In operant conditioning terms, transfer is response generalisation. If transfer is complete, the student is able to perform the new task without additional training. Transfer is particularly important when comparing the training task to the real task, which is almost certainly different. It also takes place between training tasks. If a student has learnt how to make a left-handed widget, we expect him to need much less training, possibly none, to learn how to make a right-handed one. Negative transfer also occurs where two tasks are so similar that the student is confused and performs one worse as a result of learning the other. Blum and Naylor describe some experiments on transfer. They conclude that the results are conflicting, but in general the more similar the stimulus, or circumstances in which two tasks are required, the more likely is negative transfer. The more
similar the response, or tasks themselves, the more likely is positive transfer. This conclusion is expressed in the language of the older S-R school of learning (see section 2.1.1.1), which holds that learning involves the creation of stimulus-response connections, but it seems to make sense from a feedback point of view as well. Blum and Naylor also suggest that the likelihood of positive transfer decreases as retention increases: the less you remember something the less likely it is to help you learn something similar. Negative transfer is more complicated. It seems that it is more likely to occur if the interval between learning the two tasks is either very short, or moderately long. A fairly short, or very long interval is not so likely to cause negative transfer. This conclusion is confusing, and is not helpful in avoiding negative transfer in any particular instance.

It should be noted, however, that Campione et al (1984) regard the ability to transfer what has been learnt, and to recognise that transfer is appropriate, as factors that distinguish quick and slow learners (see section 2.1.1.3 above).

2.1.2.3 Programmed Learning

One of the outcomes of laboratory work on learning and reinforcement was the idea of programmed learning. The process has been widely discussed (e.g. Annet 1969, Walker, 1975). Initially in the form of programmed texts and later implemented as computer programs, programmed learning is a method of presenting material to a learner with minimum intervention from a teacher. Programmed texts are books in which the reader is frequently required to answer questions or fill in words. Teaching machines and computer programs present material and require the learner to answer questions in various ways.

Pressey type. Here the student is presented with a series of multiple choice questions, perhaps supported by a lecture, film, text or other material. A device automatically scores his answers right or wrong. For example, the student may give his answer by pushing a stylus into one of four holes in a board. The stylus can only be pushed home in the right answer hole, as the others are partly blocked.

Skinner type. Also called linear, short step or constructed response. Here material is broken down into a large number of very small steps, each to be presented to the student in a pre-determined order. Commonly a step is a sentence or two,
from which a key word has been omitted. The student fills in the missing word and is then shown the correct answer. By careful selection and ordering of steps, the student is led to the correct answer and rarely makes mistakes.

Crowder type. Also called *branching* or *intrinsic*. Here again there is an orderly presentation of material which has been divided into steps. The steps are typically larger than in the Skinner type, and a multiple choice question is given. If the student answers wrongly, the program branches to remedial material which explains why the answer was wrong. This type of programmed learning is barely practical without a computer.

Variations on these types are also encountered. However, the following characteristics are usually found:

- The material presented to the learner is carefully arranged along lines suggested by some systematic analysis of the subject matter or skill to be taught. However, the material is revised if required in the light of experience, so that it suits the student population using it.
- Each individual proceeds at his own pace. Training material is designed for, and used by, individuals.
- There is minimum intervention by the teacher.
- Each individual makes active responses. The response leads to greater involvement and allows immediate feedback to be given.
- Knowledge of results is given immediately.

Programmed learning is important because it is an application of laboratory findings to everyday situations: it has been used successfully in schools and colleges. Nevertheless, a considerable amount of research has been done to find out why it works, particularly whether knowledge of results promotes learning by reinforcement or by its information content. Skinner, and Pressey to a lesser extent, stress the reinforcement aspect, and so provide for students to make few mistakes in order to maximise the reinforcement effect. The information content in their systems is very low, simply a yes/no response (usually "yes"). Crowder, by contrast, is more concerned with the information content of the response, which in his system is much higher, including remedial material if appropriate. By allowing his students to make mistakes, he provides the system (usually a computer program) with information by
which it can tailor subsequent tuition to the student. As all these systems have been successful, reinforcement and information content are probably both important.

As Amsel (1960) points out, Skinner's analogy with operant conditioning is only superficial; where in operant conditioning each reinforced response is a closer approximation to the desired behaviour, in programmed learning the subject matter progresses and the required response is different and not a better approximation of the same behaviour.

Whatever the conclusion, there is no doubt that programmed learning has been very successful in some cases, especially when material is to be memorised. The method has been used by the Royal Navy, for example in teaching the Rule of the Road. This is a protocol governing the correct behaviour for a vessel encountering another at sea, and the visual and audible signals to be given by vessels in various circumstances. A programmed text has been produced in the form of a book entitled “The Rule of the Road at Sea”. It is a variant of the Skinner type. Each page of the text gives one or two sentences of information, accompanied by an illustration. At the bottom of each page is either a question to be answered by a word or short phrase, or a sentence with a blank to be filled in. Having given his response, the student turns the page to reveal the correct answer. Each step is short, and the answer can be found in the text or illustration above. Periodically there are summary pages containing some or all of the questions from the previous section. There is no remedial information; if the student answers wrongly he is instructed to study a page or section again.

2.1.2.4 Training Style

There has recently been interest in training style and training strategy. Stuart and Holmes (1982) regard training as “managing others’ learning activities”. By analogy with the study of leadership styles, they conclude that the best training style depends on the student’s learning maturity. This factor is defined in terms of the learner’s ability to set high but attainable goals, his willingness and ability to take responsibility for his learning, and his previous learning experience. For one student, it may vary from one learning event to another. Stuart and Holmes describe the way that students of different learning maturity are likely to react to the same approach from the trainer. The mature student may regard a participative, collaborative trainer
as empathic and stimulating, while the moderately mature student sees the same trainer as idealistic and unable to take responsibility, and the immature learner thinks he is ingenuine and not fulfilling his allotted role. Stuart and Holmes describe a variety of possible training styles and the reaction they may evoke in different students. They suggest that the trainer should vary his style throughout the learning process, as each student becomes more mature at learning the particular subject.

Whether or not learners can be reliably and accurately categorised, it is clear that learners do differ. These differences can be exploited, rather than minimised, by adjusting the teaching method. In the absence of a detailed theory of individual differences, a trial and error approach may still produce results.

2.1.2.5 Training Strategy

Hinrichs (1976) describes some of the techniques which are open to instructors. He divides the techniques into three groups which he calls content, process and mixed techniques. Content techniques are designed to impart substantive knowledge or information on a cognitive level. They include lectures, the use of audiovisual devices, and auto-instruction (he specifically mentions the use of programmed learning or computer-assisted learning). Process techniques are mainly intended to change attitudes, develop awareness of self and others, and improve interpersonal skills. They include role playing and sensitivity training. Under "mixed techniques" he includes conference discussions, case studies, simulations and games, and on-the-job training. On-the-job training is an important approach in industrial settings. However, Hinrichs lists the following disadvantages: it may be inefficient, the trainee may not be highly involved in the training process, the quality of instruction may be poor, and training frequently takes second place to getting the job done.

Singleton (1974) takes a systems view of training which is highly goal-orientated. He distinguishes between programmed operators, who simply have to follow detailed instructions, and concept operators, who must understand what they are doing in order to work out procedures for themselves. Training programmed operators is a straightforward process of demonstrating or describing the procedure required, requiring the trainee to practise the procedure, and giving feedback on his performance. Training concept operators, on the other hand, requires the teaching
and practice of a set of skills which the operators will subsequently combine to perform the task. They must also be trained to work out the procedures required. This is similar to the compilation process which Shepherd discusses (see section 2.1.1.4).

Singleton also distinguishes on-line and off-line training (i.e., on- and off-the-job). He finds the following disadvantages in on-line training: it may be expensive if materials are wasted; it may be dangerous; training and operating requirements are confused; and the relative frequency with which tasks are practised cannot be controlled. Neither he nor Hinrichs mentions that the conditions of practice (complexity, speed, occurrence of faults) are also not usually open to control in on-line training.

In discussing the provision of practice, Singleton describes the whole and part-task methods which are commonly used. Whole task training requires the trainee to perform the entire task immediately (although perhaps in a simplified version). There are no problems later on in training with linking together the different parts of the task into a whole, but the trainee may have difficulty in mastering the task in this way. Part task methods require the trainee to learn task elements separately. Very specific feedback can be given for each part of the task. Having mastered each part, he is taught to combine the parts, either all at once or progressively. Part task training frequently requires special training equipment.

Ohlsson (1985) also has much to say about training strategy. He criticises the psychological literature on the matter as being too global to be adequate in guiding the choice of strategy in a specific instance. He looks at classroom teachers for inspiration and observes that they employ plans to govern whole series of teaching actions. The plans are not hard and fast, but can be adapted, even abandoned, as required. Individual stages may be desired outcomes of teaching, or may be precursors to other stages, as when, in teaching subtraction, a teacher directs children’s attention to the fact that borrowing problems exist before teaching them how to solve them. Such a plan gives cohesion to a learning sequence. It saves effort on the part of the teacher, as it can be re-used. It presupposes both a particular view of the subject matter, and an hypothesis (probably unstated) about learning. Ohlsson also lists a whole set of teaching tactics related to each teaching objective.
For example, when a student has given a solution to an exercise, the teacher's tactics may be to give feedback on the solution and identify any errors. Alternatively, he may prompt the student to check his own solution, to describe the solution method, or to justify his own method.

In effect, this is a strategy for carefully removing feedback, as section 2.1.1.2 recommends.

When it comes to presenting new material to trainees, active teaching is sometimes assumed to be the only way. Teachers in classrooms use this technique most of the time, and even if their methods are not absolutely the best, they have been proved by time to produce results. Ohlsson draws attention to a strategy teachers occasionally employ, but one which is not often used in computer-based tutors. Having initially presented the material to the students by demonstrating the task to them, probably with some explanations and background information, they give the students a series of specially constructed exercises which lead them through the task with less and less support each time. For example, in the classroom the teacher first presents a technique to the students, explaining its uses and giving a demonstration. Then the students practise the mechanics of the technique with progressively more difficult exercises and less help from the teacher and from each other. They may begin with the teacher performing the technique while the class prompts, followed by individuals doing the task with correction, and finally the students working alone. Once students have mastered the mechanics of a skill they are given exercises which are as realistic as possible. This provides context: it gives them practice in deciding when and how to apply the new skill, and shows how it relates to the overall task.

2.1.3 Systems Approach to Training Design

Singleton (1974) and others have described the systems design approach. This approach to design begins by recognising that a complex machine cannot be designed without considering its purpose and how it fits into a larger system of machines and people. Each component is therefore designed in the context of the larger system. In particular, the systems design approach concentrates on objectives (of the component within the system), functions (as separate from the means of achieving them), and on an orderly design structure based closely on the objectives. The approach can be applied to training design, especially where it involves
machines.

2.1.3.1 Systems Model of Industrial Training

According to Hinrichs (see section 2.1.2), training consists of procedures to direct and encourage learning. In the case of industrial training there is a fairly rigidly defined organisation in the background. This makes the training objectives more concrete: organisations intend training procedures to encourage their members to learn skills which will improve their usefulness to the organisation. The organisation must not be ignored in designing training; it is at the behest of the organisation that training takes place. As John Hinrichs (1976) explains very clearly, training in an organisation can be modelled as an open system. Hinrichs models the training system on three levels, namely those of the trainee, the training department and the organisational management. The management makes business plans which, when compared with available manpower, may imply a training need. The training department prepares a training scheme to meet the need and administers it to the trainees. In order for the training system to survive, the management, the training department and the trainees must all support it: all have requirements which must be fulfilled before they are motivated to do so. The trainee, for example, may have any number of motivations for joining the scheme, including instructions from his superior, or the belief that it will further his career. His success in learning the material will depend on his motivation, personality and various environmental factors such as the time he is able to devote to the scheme, and the facilities provided. It will also depend on how well the training scheme has been designed by the training department. Once the scheme has begun the trainee will not want to continue unless he feels he is achieving his personal objectives. He must also feel he is becoming competent at the skills he is expected to learn: he needs to know he is achieving the learning objectives of the scheme.

Those trainees who continue with the scheme will learn something from it, and their behaviour will consequently change. The training department, observing this, has the opportunity to adjust the training program as it applies to each trainee, according to what that trainee appears to have learned. This is the process of individualisation of instruction. It allows the training program to concentrate on each trainee's weak areas, and neither to move on from a topic before he has mastered it nor dwell too
long on something he has already grasped. This not only saves time, but improves the trainee's motivation by increasing his sense of competence and achievement. It also allows for the material to be presented in a form that is most suited to the particular learning style preferred by the trainee.

2.1.3.2 Systems Approach to Training Design

The systems approach is intended for the design of man-machine systems in which there are identifiable communication and control channels. Inputs and outputs to each process can be described and monitored. This is not the case with training systems, in which the trainer and the trainees can all use their initiative to change the expected sequence of events. It is not unheard of for a trainer to learn from a trainee, for example. In general, interactions cannot be rigidly defined, and so at a detailed level the systems approach does not apply to training. However, at a high level, when types of interaction and general needs are considered, it is useful to apply the systems approach to training design. It emphasises objectives, and provides a structured method of design closely linked to those objectives. By separating functions from the means of achieving them, the choice between different training methods and media can be put off until other factors have been fully considered.

The systems approach to training design, as described by Singleton, recognises that a training programme is a procedure taking place within a system (the organisation) for the benefit of the system. This is a valuable admission, as it is inevitably true. Fortunately, there is little conflict between the organisation’s objectives and the individual’s as far as training is concerned: in general, both want the individual to receive training. Ignoring the individual’s requirements during training design can therefore result in their being partially fulfilled incidentally. Nevertheless, there must be a risk in the systems approach to training design, that emphasis on the system’s interests will lead to the trainee’s interests being neglected.

According to the systems approach, design starts with careful consideration of the objectives. These are both general (increase ability, reduce learning time or costs, or decrease the capabilities required) and specific (produce a certain number of operators who can perform task A). There are also requirements (not always stated) for the scheme to meet constraints of time, budget and standards, and to produce
trained operators who fit in with the way the organisation works.

The specific objective is to train people to perform a certain task. To this end, a description of the task is produced. It can take various forms, but it should describe all the procedures and decisions involved, and the information needed to make them. Singleton distinguishes between a task description, which is system-orientated, and a job description, which describes the same things from a human operator’s orientation.

A selection specification is derived from the job description. This is a description of the characteristics trainees should have to undergo the training. The selection specification is used to establish two things: firstly that trainees are capable of learning the required skills, and secondly that they all have certain skills at the beginning of the programme. The skills available in the target population must be considered to do this. The selection specification must be converted into selection tests which are actually used to select trainees from the available people. Development of selection specifications and selection tests are not well understood, and there is no guarantee that the tests will find the best people for the programme.

The training requirement is the difference between the job description and the selection specification. It is used to design the training programme.

Singleton gives the following six-stage procedure for designing training:

i. Obtain training requirement from discrepancy between selection specification and job specification;

ii. Consider kind of operator required; whether programmed or concept or intermediate (see previous section);

iii. Determine on-line/off-line separation;

iv. Consider methods of providing practice;

v. Devise methods for progressive combination of skills;

vi. Consider methods of implementing and evaluating the scheme.

However, apart from the first item, this list seems simply to record a few of the decisions that need to be taken during training design. Singleton does not justify the ordering, nor consider any other issues, such as choice of training medium, and he gives no advice for implementation and evaluation.
Evaluation is an important issue in the systems approach to training. The scheme has been designed according to stated objectives, and evaluation attempts to find out whether the objectives have in fact been achieved. It also leads to the scheme being improved. It is always difficult to measure the success of a training scheme. It may produce operators trained to the specified standards more cheaply than a previous training scheme, but their morale may be low, or their attitude unacceptable to other workers. A new scheme may be more costly but produce operators who make fewer mistakes. In other words, all the objectives of the scheme must be met, not just those which specify the standard trainees are to achieve. Even when it can be shown that a scheme is better than a previous one, it can almost certainly be improved still further.

2.1.3.3 Hierarchical Task Analysis

The task description can be provided by using hierarchical task analysis (HTA). HTA has proved to be a flexible and effective method for describing a wide variety of tasks. Annett and Duncan (1967) introduce the technique, but Duncan (1974) gives a more matured and detailed account. He describes some of the methods of task analysis which were being used at the time. Most of these assumed that there is one particular level at which tasks should be analysed, or specify a small, fixed number of appropriate levels. For example, one writer (Smith, 1964), states that tasks should be analysed in terms of jobs, duties, tasks and elements. It seems that debate at this time revolved around how many levels should be used, and what the levels should be. Duncan, on the contrary, argues that a task should be analysed progressively, at as many levels of detail as are required. He assumes that tasks can be analysed hierarchically, each operation being capable of further redescription until the required level of detail is reached. He gives a rule (a stopping rule) for identifying the required level of detail in terms of the probability $p$ and the cost $c$ of the operation being performed inadequately: redescription should stop when the product $p \times c$ is acceptably small. The probability $p$ is assessed by considering what trainees can already do by the time they come to learn the operation in question. The cost $c$ is defined by the system itself and includes waste, hazard, demands for support from outside, etc. The probability and cost do not need to be measured: a subjective judgement is sufficient to decide whether either one of them is so high
that analysis should continue, and hence training be more thorough. In particular, if either the probability of inadequate performance or the consequent cost is zero, analysis should stop.

The operations produced by Duncan's analysis are linked together by plans. A plan for achieving an operation \( A \) describes the way in which the operations in the redescription of \( A \) are combined to achieve \( A \). Duncan sees operations as being relatively easy to train, and plans as being relatively hard. A simple plan, consisting of carrying out a sequence of operations in a given order, will present no difficulty. However, the more complex plans required for fault diagnosis, for example, include decisions which cannot be fully specified and therefore training for such plans may be difficult both to specify and to evaluate.

Shepherd and Kontogiannis (1987) describe adaptations to the method which apply to HTA used specifically for training users of computer-based systems. The outcome of the HTA is a description of the task broken down hierarchically to the level of the operations through which the operator interfaces with the system, for example, when pumps are switched on or valves operated. Shepherd attempts to categorise the interfacing operations, the plans for combining the operations at each level, and the intermediate goals (or higher order operations), to enable the trainer to choose the training conditions and provide the trainee with the extrinsic information he needs at each stage to help him learn.

2.1.4 Man-machine Systems

Embedded training is a technique for training people to operate machines - in particular, computers. Section 2.2 below discusses computers and how they are used in training. The present section approaches the subject from the other side: it considers how people use machines.

The systems approach (eg Singleton, 1974) stresses the need to consider the machine and the user as a single system with integrated goals. Designing the machine entails designing the task the user is to perform. A major part of the design process is to decide which parts of the task will be performed by the machine and which by the human user. Singleton gives a list, modified from Fitts, showing the relative strengths of humans and machines, which is intended to be a guide in
deciding which parts of the task to allocate to each.

An important paper by Bainbridge (1982) points out some difficulties with the problem of assigning tasks in the man-machine system. Fitts' list described the human user as unreliable, imprecise and slow, and designers have tended to follow his lead in assigning as many tasks as possible to the machine. One ironical result of this attitude is that the human is left doing only those tasks which the designer cannot think how to automate. Such a task is monitoring the machine to make sure it is working correctly. (Fitt's list gives error correction as one of the human's strengths.) Consequently the human, who cannot be trusted to do the job himself, is required to monitor the machine to check that the machine does it correctly. Of course, since the human no longer does the job himself, he has no opportunity to preserve the skills he will need when he has to handle the machine's malfunction. As Bainbridge says, "By taking away the easy parts of his task, automation can make the difficult parts of the human operator's task more difficult." It is for the more fully automated systems, in which human intervention is rarely required, that most effort must be invested in training. Bainbridge also suggests ways in which interface design can give the human more control and a more coherent job. He concludes that the allocation of tasks between the human and the operator is much harder than some designers had assumed, and that training must be given more emphasis.

Singleton had actually mentioned this problem in his book. According to his view, the user is central to the task, and he delegates tasks to the machine. However, having stated this belief, he is not consistent in applying it. For example, he comments (p 58) "if [Task Description] is possible [ie complete and comprehensive], then it probably should not be a personnel problem since the man could be replaced by a mechanism." The systems approach does not make the man central to the task; that is the designer's responsibility. It is easy to see how the designer might get involved in the intricacies of the systems approach and forget such an additional requirement, as Singleton seems to do occasionally.

2.2 Computing

This section starts with a discussion of the uses to which computers are put in
teaching. The impact of artificial intelligence techniques on training is then covered. Finally, the advantages and limitations of embedded training are considered.

2.2.1 Uses for Computers in Teaching

Computers have been used in training and education almost as long as they have existed. The range of uses nowadays is vast. A survey among major corporations (Kearsley et al., 1981) divided the uses into simulation, exercises, drills, tutorials, and job aids and data management. The use of computers as training job aids (for the production of teaching materials, for example) is widespread, as is their use for management of training data - sometimes called Computer-Managed Training or CMT. These applications have much in common with other, non-teaching, uses of computers and will not be discussed further.

In the actual teaching process, computers have until recently had two roles, with little middle ground. The first of these is their use as a tool. Straightforward simulation falls into this category: in a conventional simulator all the teaching is done by an instructor, with the simulator providing the setting. Often the simulator provides more feedback than would normally be available, and records the student's performance at some level. Similarly, management games and economic models allow students to experiment with complex systems. Exercises present students with test exercises and frequently mark their answers (Hills et al., 1986; Kearsley et al., 1981). Other tools are database and text browsers which give students easy access to information. Computers can provide a situation for discovery learning to take place, as with Microworlds (eg Lawler, 1984). They can be used for demonstrations by taking advantage of their potential for simulation and graphics. Games of various kinds have training uses, and programming itself is often seen as a way to train analytic skills.

In all these systems, any explicit teaching is done by a human instructor, using the computer system as an aid. There are also systems which present teaching material to the students. This may be text or pictorial (graphics, video or stills) or a mixture of both. Traditionally programmed learning methods (see section 2.1.2.3 above) are used: the teaching material is interspersed with questions for the student to answer. Feedback can be given at whatever stage the designer chooses, including immediately after the student has answered. Producers of this kind of material have
claimed that the interaction with the computer improves motivation: certainly some people enjoy using computers, and there is an argument (see section 2.1.2.3 above) that the need to make active responses increases involvement. In many such systems a section can be repeated or remedial material included if a student makes a wrong answer. The student is not restricted to the speed of the class.

On the other hand the program must be designed so that the student’s answers can be anticipated: if he tries to do something unexpected the system will have no response available. This restricts the structure and content of the system and effectively prevents it being combined with any of the “tools” methods mentioned above as these would introduce too much freedom into the student’s behaviour.

2.2.2 Artificial Intelligence

Webber and Nilsson (1981) regard artificial intelligence (AI) as “a branch of computer science whose objective is to endow machines with reasoning and perceptual abilities.” Many of the efforts towards that end have consisted in making more explicit the knowledge contained in a program in order that it may be reasoned about. New techniques for representing knowledge have also enabled computer programming to handle areas previously unapproachable by computational methods.

AI covers many other areas, and uses many techniques not described here. Nilsson (1980) provides a thorough introduction to the subject.

2.2.2.1 Explanation

Probably the most talked-about AI program of all time is MYCIN (Shortliffe, 1974). By describing an expert diagnostician’s knowledge as a series of rules (“if X is the case, then do Y”) MYCIN was able to provide therapy advice for certain kinds of infectious diseases. Not only did MYCIN model a type of process not previously accessible to computers: because its knowledge was in a comprehensible format, it was able to provide an explanation of how it had arrived at its conclusions.

However, although the rules MYCIN used were comprehensible, its control structure was not. The diagnostic approach taken by MYCIN was not that used by experts, and it was not accessible, so MYCIN’s explanations were only intelligible on a small scale. The rules interacted in ways which were not immediately apparent, and
this made them difficult to alter. Later developments addressed these problems. Clancey (1983) describes some of MYCIN’s problems, and how they were overcome. He found that it was necessary to introduce more structure into the knowledge. “Meta-rules” were added, explicitly describing how the diagnostic rules were to be applied. The original uniform set of rules was reorganised to fit in the new structure. The explanatory power of the system, now called NEOMYCIN, was greatly improved. In fact, it was considered that the new system could be used for teaching medical students. Although MYCIN could now explain its own reasoning fairly well, it could not give the underlying justifications for any of its rules. It often happens that experts do not refer to underlying justifications, although they are useful memory aids for students. A justification was added to each rule where appropriate. With these improvements, NEOMYCIN was incorporated in GUIDON2, a program described in (Clancey & Letsinger, 1981), intended to demonstrate the expert’s reasoning process to students. The new program has been tried out with students and they seem to be able to understand it. Work on the MYCIN and GUIDON systems continues (Richer & Clancey, 1987).

Other attempts to make knowledge representation more explicit and accessible were conducted at MIT by William Swartout. He took the Digitalis Therapy Advisor and examined its shortcomings (Swartout, 1983). He found two major problems. Firstly, when asked to explain what it had done, the system frequently described steps which were essentially for its own house-keeping and not part of the medical reasoning process. The resulting explanation could be very confusing to anyone other than the programmer. A more serious problem was the absence of justification, causal and strategic knowledge. Swartout’s approach was different from Clancey’s. Noting that the necessary knowledge must all have been available when the program was written, he aimed to produce a system which could write the program itself. He used explicit statements of facts in the domain and of the principles used in working in the domain. He built a system which could use that knowledge to generate a new version of the Advisor, called XPLAIN, which was not only an efficient system in its own right, able to explain its own functioning, but also contained a record of how it had been generated. The new system could give causal explanations by referring to the models from which it was created. To prevent explanations including details not relevant to the user, he marked each domain principle with a
viewpoint describing who would be interested in hearing about it. This information was compiled into the final program so that facts irrelevant to a particular class of user could be filtered out of the explanation.

It is intriguing to note the parallel between Swartout's system and the views of Anderson and Shepherd (see section 2.1.1.4) about how cognitive skills are acquired. Swartout's system starts with declarative knowledge about its domain (facts and principles), together with general procedures (the system which generates, or compiles, the knowledge into the working system). The working system itself is procedural and efficient and not open to explanation; explanations are derived from the declarative knowledge from which the working system was generated.

2.2.2.2 User Modelling

One of the difficulties in providing a satisfactory explanation is that what is satisfactory varies from user to user and from time to time. Researchers have addressed this problem by building within the system some kind of model of the user. An attempt to communicate with the user is adapted to the user by reference to this model. The model may contain information about whether the user is an expert or a novice, whether (as in XPLAIN) he is a programmer or a primary user, it may try to infer what he knows, or any other fact about him, but basically its purpose is to make the interaction more specific to the user, and hence more helpful. There are different approaches to using such a model. Some systems (eg Innocent, 1982) view the adaptation process as being part of the ordinary interface with the system, so that each exchange with the user is adapted by reference to the user model. Greenberg and Witten (1985) point out some of the potential difficulties with that approach. The main one is that the user is also trying to build his own model of the system: this mutual modelling process may lead to instability. Another significant problem is dealing with inaccuracies in the user model: an error in the model could potentially make the system unusable.

A second approach to the use of user models, one sometimes used in systems intended to help novices, is that they should be used solely as the basis of advice and explanation to the user. Here the adaptive process is viewed as looking over the user's shoulder, waiting for the moment to intervene, or for the user's question. Such monitoring systems are treated very much as separate subsystems, which may
eventually be removed. The user interacts with the main system without intervention.

Sleeman (1985) describes five kinds of user model. These are:

* **scalar** A single number is used to represent a vital characteristic of the user, be it speed, experience, or whatever;

* **ad hoc** The information the system records is tied in very closely to what it can do, with no explicit psychological model of the user. These models can be identified because they have no meaning outside the context of the system;

* **profile** The user is modelled in terms of a series of attributes, each with a value;

* **overlay** The user is regarded as knowing a subset of what is to be known;

* **process** The model of the user describes what he can do, and can be executed to predict his behaviour. This kind of model may also contain wrong processes, leading to the prediction that the user will make a mistake, which can be used to diagnose his errors.

John Self (1983) describes an innovative approach to student modelling using machine learning techniques; let the machine and the student learn alongside each other. He applies this approach to a guided discovery learning system for climatic concepts. The system is assumed to be using the same learning method as the student, and so, learning from the same examples the student sees, its concepts are taken to be a model of the student's concepts. Based on its supposedly superior knowledge of concept learning strategies, the system can offer the student advice about the kinds of example to seek, and the appropriateness of examples he has sought, to help him to improve his concept learning strategies.

Anderson (1981) describes a system called ACT, which is a computer implementation of a theory of human learning. His theory says (see section 2.1.1.4) that knowledge is first encoded as declarative statements. In problem solving these declarative statements are acted on by general procedures. This process is quite slow and places a great burden on working memory, but it allows a considerable amount of error trapping to be done. Later, by two processes which he calls *composition* and *proceduralisation*, the declarative knowledge is compiled into a procedural form. Composition is a process of gradually combining (general) steps which are used together into one step. Proceduralisation is a process of making a general procedure
specific by replacing variables with particular values, thus reducing the load on working memory. The effect of these two processes is to produce a gradual speed up of the task.

In ACT, procedures are modelled by productions. A production is a condition-action pair (If these conditions hold, do these actions) which represents a step in the problem solving process. Productions are arranged hierarchically, in that the problem is decomposed step-wise until manageable components are arrived at. Productions are only created and never destroyed. This will often result in more than one procedure being applicable to any given situation. To resolve this conflict, ACT chooses between the productions based on their specificity (it prefers a more specific production) and on their strength. A production gains strength every time it is employed successfully. New productions are relatively weak. Eventually unsuccessful productions will become so weak that they never apply.

The process having been speeded up by compilation, a further stage of continuous improvement occurs which Anderson calls tuning. The ACT system models three tuning processes: generalisation, discrimination, and strengthening. Strengthening is the process just described. Generalisation is a process of combining together two procedures together by replacing specific values with variables, so that one procedure will now cover both previous cases as well as others of the same kind. Discrimination is a process of adding extra conditions to a procedure so it is not applied to so many cases. In some situations discrimination may also be able to create a new procedure which produces a different result in the cases so excluded.

The combination of these processes allows ACT to model some significant features of human learning (see Anderson, 1981 for details). He has modelled learning of language skills, geometry and a classification task, and compared ACT’s performance with that of human subjects. The results, while not conclusive, suggest his theory may be useful.

2.2.2.3 Planning

Another area of AI which is relevant here is planning. AI is an attractive approach to many planning problems, because they can be intractable using conventional algorithms, and yet not require genuine creativity for their solution. Planning is one
of the areas in which AI has been successfully used to solve real, non-laboratory problems.

In the laboratory, AI planning systems have usually been associated with robots, either real or simulated. Nilsson (1980) describes in some detail the classic robot planners STRIPS, DCOMP and NOAH, and variations on them. Early planners were given rules which showed how various goals could be achieved in terms of subgoals. Some of the subgoals would be actions which the system could directly carry out. Given a goal, the planner produced a simple linear plan by breaking each goal down into subgoals successively until a sequence of subgoals was obtained which could be carried out, i.e., as each subgoal appeared in the plan, the preconditions for achieving it had been met and it could immediately be executed.

This formulation is unable to cope with interacting subgoals, where the action of achieving one subgoal affects the preconditions for another subgoal. An improvement was to produce a non-linear plan with the subgoals only partially ordered at first. This gives an extra flexibility which allows some interacting subgoals to be achieved by careful ordering. The addition of extra actions by the planner gave further flexibility. Hierarchical planners, such as NOAH, were then introduced to handle more complicated planning problems. Instead of successively breaking a problem down in subgoals and then making sure that all the preconditions are satisfied, a hierarchical planner sorts out the preconditions at each level before breaking the goals down again. At each level of planning it may be necessary to make adjustments to the preconditions as more detail becomes apparent. By this process the planning problem at each stage is kept to a manageable size.

Early planners assumed that the world always behaves as expected: the plan itself is the end result. A more realistic approach considers that planning is not complete until the goal has actually been achieved. Before carrying out each action, a planner can check that the condition of the world is as it expected. A precondition may not exist, or the goal of the action may have already been achieved. Unexpected conditions require some amendment, either by patching the existing plan, or by creating a new plan. Some plans are drawn up with alternatives in them, to assist this process. Hierarchical planning is a particularly appropriate method if the world
is very unpredictable, as detailed planning of later stages can be postponed.

Peachey and McCalla (1986) consider the use of planning in intelligent tutoring systems: see section 2.2.3.3 below.

2.2.3 Artificial Intelligence in Teaching

Techniques from many branches of AI, such as natural language processing, planning, diagnosis, etc, may be used to develop teaching systems in various ways. To begin with, any system which contains information in the explicit, approachable form aimed at by AI is more likely to be useful for teaching. For example, by storing its information as easily understood rules, an expert system can have an explanation facility which enables it to answer some of the user’s questions - and this may help him learn. The same can be done with games and models, and many other systems. Students can start to use such a system with less preparation, and will need less supervision while they are working. These are the tool uses of computers in teaching. In this section we shall discuss the use of AI in programs which take over some of the instructor’s actual teaching. These systems are sometimes referred to as intelligent tutoring systems (ITS) or intelligent tutors.

2.2.3.1 Models in Intelligent Tutoring Systems

The “intelligence” of intelligent tutors derives from the explicit models which they contain. The aspects most frequently modelled are the subject matter, the trainee, and the instructor. Where subject matter models are concerned, the issues are precisely those discussed in section 2.2.2.1. The models of each individual trainee which an intelligent tutor builds are again not dissimilar from the user models of other AI programs. It is the use of these models in some teaching strategy, which can be regarded as either an implicit or an explicit model of an instructor, which is central to intelligent tutors.

2.2.3.2 Teaching Strategies

The instructor model in an intelligent tutor need not be a particularly accurate model of a human instructor. Indeed, it need not be deliberately modelled on the way humans teach. Nonetheless, it takes over some of the functions of a human instructor and is, to that extent, a model of one. The teaching strategy which the intelligent tutor is programmed to take will depend on the designer’s idea of a good
teaching strategy. That strategy may involve guiding the student closely through a series of exercises, as in Anderson’s LISP tutor (Anderson and Reiser, 1985). The LISP tutor is built on the principal that feedback should be immediate. A student is given exercises in the form of problems for which he must write LISP code. When he makes a mistake, or makes an input which is not on a path to the correct solution, the LISP tutor intervenes. If the student cannot correct his input, the tutor gives hints and will, if necessary, complete a piece of code so that the student can move on to the next section. Anderson is one of the few writers on intelligent tutors to report the results of an evaluation. He does not report in detail, but he claims that students learned the same material in 11.4 hours with a human tutor (one-to-one), 15 hours with the computer tutor, and 26.5 hours left to solve the problems on their own. However, it should be noted that this heavy handed approach to guidance may not be suitable in other settings. Students do not have the opportunity to practise (and are apparently not taught) problem-solving strategies. It may also be expected that students will not develop their learning strategies by using Anderson’s system, as the learning strategy is imposed by the system. On a practical level, such a tutor cannot be built for a particular task unless, at each step of the task, it is possible to say whether an action is on the solution path or not.

Not all intelligent tutors incorporate such an authoritarian strategy. Elsom-Cook (1987) describes a continuum between Anderson’s LISP tutor and free discovery learning environments which give the student no guidance whatsoever. Somewhere in the middle of the continuum are systems which allow the student some freedom to explore possibilities but try to prevent him developing suboptimal strategies. Such a system is West (Burton, 1982) which makes a deliberate decision about whether or not to intervene at each point where it knows a better move than the one the student made.

2.2.3.3 Planning in Intelligent Tutoring Systems

Although Ohlsson (1985, see section 2.1.2.5) suggests that an intelligent tutoring system will benefit from a plan, which will give cohesion to the teaching sequence, there is not much sign of his suggestion being followed up. However, Peachey and McCalla (1986) have considered the use of such a plan. Their proposed system includes two components, the Planner and the Executor, which build and use the
plan. The Planner constructs a non-linear plan, based on a student model, containing all possible paths through the teaching material. The Executor carries out the plan, taking alternative paths as circumstances dictate. It can recall the Planner if the plan should prove to be inadequate.

The plan consists of a sequence of steps, each intended to achieve an instructional goal. The steps are associated with teaching actions which are intended to achieve the instructional goals. The planner is non-linear, producing a partially ordered plan. The Executor is responsible for choosing the final ordering of steps. Peachey and McCalla disregard interactions between subgoals, but they mention a plan editor which could be added to remove bugs introduced by interacting subgoals, should there be any. Their planner is not hierarchical, and the whole plan is produced at the beginning, with no postponement.

Executing the plan involves carrying out the specified teaching actions. After teaching, it compares the actual student model with the expected student model to see whether the instructional goal has been achieved; if not, an alternative path must be taken, or a new plan constructed. Because there is no direct connection between teaching actions and instructional goals, and because the student model can only be an indirect view of the state of the student's mind, the chances of a mismatch between the expected outcome of a teaching action and the actual outcome is high; it is essential that the planning system as a whole is flexible enough to cope with unexpected outcomes.

2.2.3.4 Teaching Tactics

As well as these strategic decisions, which are usually system design decisions, an intelligent tutor also needs a set of tactics to employ. These are partly a design issue; indeed the decision may be taken entirely at design-time and only one tactic supplied for each teaching problem. It is also possible to include different teaching approaches which the instructional model can choose between in order to pick one which best matches the student's learning style, or in order to avoid repetition. This has been a neglected area, although the remedial system of Sleeman's PIXIE shell (Sleeman, 1987) provides several different types of remediation, including displaying the error the system has found, and showing a correct solution to the problem. Sleeman states that he intends to develop the remedial system so that it
tailors its explanations to suit the individual student's learning style (see section 2.1.1.3) and to evaluate the effect of different remediation tactics.

2.2.4 Embedded Training

An important decision in training design is whether to teach the task on the actual equipment or not. Training on the equipment itself reduces the problem of transfer of training (see section 2.1.2.2) but there may be difficulties. Training on the real equipment may not reveal the structure of the task to the trainees, and they may master the task better if they learn to do a simpler task first. The equipment may be dangerous, costly to run, or in use all the time. Embedded training can be used in some cases to get round difficulties of this kind by changing or adding to the data which is fed into or produced by the system. For example, a process control system may use artificial data to give the trainees practice at emergency procedures without the cost of actually shutting down the plant.

The term *embedded training* is generally applied to computer-based systems. This is not a necessary restriction, as the use of substitute materials in manufacturing equipment could be termed embedded training.

Various embedded training systems exist. The term usually refers to onboard simulators for ships and planes, which simulate sensor data. Brady and his team (in Kocher et al 1984, Lambert et al 1985 and Brady et al 1985) describe such systems for pilot training. By using helmet mounted displays for pilots it is possible to simulate other aircraft in a variety of situations. Such trainers have a high fidelity while providing more controllable training conditions at less cost than real exercises. In many respects embedded training systems of this kind are similar to ordinary simulators, so that although they may incorporate automatic analysis of the trainee's performance, any tuition must be given by a human instructor.

Another large group of embedded training systems exists, but they are not usually referred to by that name. They are systems to teach either computer programming or use of an operating system. Most of them are experimental systems built for research into intelligent tutoring systems. The LISP project mentioned in section 2.2.3.2 is such a system. The BIP project (Barr et al, 1976) teaching Basic, Wizard (Finin, 1983) for UNIX, SPADE-0 (Miller, 1983) for Logo, and Proust (Johnston &
Soloway, 1983) for Pascal are examples of systems using a wide variety of approaches to teaching programming and operating system skills. Each of them is an embedded training system because it is embedded in the computer which the student is learning to use. However, programming involves so few of the problems encountered in other embedded systems where communication between the system and the environment is significant, that these systems contribute little to the understanding of embedded training, except as examples of intelligent teaching systems.

2.3 Summary: a Model of Training

The preceding sections describe various issues which are important in the design and provision of training. Many of these issues can be drawn together into a model of the training process (see figure 2.2 below). The model is first described in terms of a human instructor; after that the requirements for a computer-based system to carry out the same processes are discussed. The model assumes students are being taught individually; if training is done in groups, the instructor may have to compromise between the needs of individual students.

2.3.1 The Model with a Human Instructor

At the start of a training session, the instructor begins by choosing the element of the subject matter to train next. He is likely to follow a plan which may be more or less explicit (Ohlsson, 1985). If he knows the student's history of learning this subject, he will choose an element, or adapt his plan, to suit the student. This may include altering the amount of material regarded as an element; material may be broken down into smaller elements for a slower student (section 2.1.1.3).

Having chosen an element to train, the instructor then decides how to adapt the training conditions to the individual student (section 2.1.2.5). In particular he can adjust the material given as pre-instruction or introduction, the conditions of practice, and the aids available to the student during practice including extrinsic feedback (sections 2.1.1.2 and 2.1.2.1).

Training can now begin. First the student is presented with the introductory, or pre-instructional material. He is then given some examples to practise on. The conditions of practice will be those chosen by the instructor. For example, the
complexity of the example, and the speed and accuracy required, can often be controlled. The conditions may vary from one example to the next. Additional aids, not available on the real task, may be provided to help the student. Chief of these is usually support and hints from the instructor. Finally, the student may be given knowledge of results extrinsic to the task to help him improve his performance. A
student needs to know whether his response was correct; if that information is not
provided by the task itself, he needs to be told. If the information is provided by the
task - ie the feedback is intrinsic to the task - he may need to have his attention
drawn to it, or it may be necessary to show him how to obtain the information.

The instructor will monitor the student’s performance on the practice examples, with
a view to refining his own opinion of what the student knows and how well he is
getting on.

The first cycle of instruction is now complete; the instructor must decide whether to
stop - either temporarily or finally - or continue with another cycle. If he continues
he may have more information about the student on which to base his choice of the
next element to teach and the training conditions to apply.

As training progresses, the training programme can be adapted more precisely to the
needs of the individual student. For example, his ability to transfer learning
between similar tasks can be judged, and used to decide the amount of detail given
to him when learning later task elements.

Before a student can be said to have mastered a task, he must be able to perform it
with the benefit of only that feedback which is intrinsic to the task; all extrinsic
feedback must be withdrawn. This should be done carefully in order to avoid losing
the motivating effect which the feedback may have (see section 2.1.1.2).

2.3.2 The Model with a Computer-Based Training Package

The same model can be used with a computer-based training package, although it
requires a certain level of sophistication and some particular features described here.
The model would not apply to a simple computer-based training package, such as a
programmed learning package, which would not have all the necessary features. For
a fully-developed training system capable of operating without the intervention of a
human instructor for considerable periods of time, some of the techniques of
artificial intelligence are recommended (see section 2.2.2).

In order to be able to select the next element for instruction, the training system
needs to have a set of options to choose from. If it is to choose the next element to
suit the student, it needs information about the student as an individual. The most
flexible way to achieve this is to have explicit models of the subject matter (section 2.2.3.1) and of the student (section 2.2.2.2). Even a crude user model can increase the flexibility of the system. An explicit model of teaching tactics will allow the system to use different tactics with different students - particularly valuable when adjusting training conditions (section 2.2.3.3). Rules for using these models to choose the next element to teach and to adjust the training conditions are also required.

Computer-based training systems sometimes have facilities for automatic planning (section 2.2.2.3) and generation of explanation (section 2.2.2.1). The next element to train must be selected according to some method, and a plan should give structure to the training process. Planning must be at least partly automatic, however, or the system will be required to follow a rigid plan regardless of the individual student's requirements. Rather, the system must be able to re-work all or part of the plan if it proves unsuitable. Although automatic generation of explanation has received much attention from researchers, this process still presents significant problems.

2.3.3 Training Design

The model just described relates to the sequence of events during training. However, before training can take place, the training programme must be designed. Because of the complexities involved, and because of the desire to monitor training effectiveness and amend training programmes when necessary, a systematic approach to training design is recommended. One such is the systems design approach, which is described in section 2.1.3. This approach concentrates on identifying the objectives of the training scheme, and on following an orderly design structure based closely on those objectives.

A design approach based on systems design is the subject of the following chapter.
3. THE DESIGN METHOD

3.1 Overall Approach

Following Singleton and others, we outline an approach to training design which starts from the premise that the human is being trained to fulfil a specified function within a system. The advantages of applying the systems design approach to training...
is that the design is considered in the context of the organisation, not in isolation. It also gives structure to the design process. Figure 3.1 shows the elements of the method.

All the elements in this plan are mentioned by Singleton, although stages 2 to 6 of his six-stage procedure (see section 2.1.3.2) all fall into "design and create training material." In other words, greater emphasis is placed here on the earlier stages of the process. Within each stage, we will not necessarily keep to Singleton's suggestions, but will draw material from any suitable source.

The method is strongly driven by the objectives, which are defined by the organisation which requires the training scheme. The objectives specify and scope the task to be taught. The task is now analysed and described using some suitable convention - here Hierarchical Task Analysis is recommended. The objectives are also used to set down acceptance criteria for the scheme. The criteria are used as the basis of acceptance tests; these are defined from the acceptance criteria (and hence the objectives) and the task analysis only, without reference to the design of the training scheme itself. The tests will be the only formal basis on which the training scheme is assessed. This is consistent with accepted procedure in software production.

In the meantime, the candidate population is identified and described. By referring to the task description, selection criteria and hence selection tests can be specified. These will be used to select trainees for the scheme from among the candidate population.

The training scheme itself is designed around the task description. A rough estimate of the training requirement (the difference between what trainees know or can do and what they are required to know or do) is obtained by comparing the task description and the selection criteria. Training material is designed and created to train each aspect of the training requirement.

The training scheme can now be conducted, using trainees selected from the candidate population. The results of the scheme will form the input the acceptance tests, which will determine the success or otherwise of the scheme. If necessary the scheme can be amended in the light of the results of the acceptance tests.
Each of these stages is discussed in more detail below. In the following sections, the implications of each stage of the approach for a general embedded training system are considered. The practical experience of applying the approach to the design of the embedded navigation trainer is then described. As the navigation trainer has not been built, some of the later stages of the method, particularly the conduct and assessment of the scheme, have not been carried out, and only some general remarks can be made here.

3.2 Objectives

The training design process begins from stated objectives. In general the objectives of a training scheme will be to train students to perform the task in question, while not costing too much, fitting in with the way things are currently done, being acceptable to instructors and students, not requiring too much in the way of external resources, and so forth. As Singleton points out (see section 2.1.3.2) there are likely to be unstated objectives, such as producing operators who fit in with the company’s way of working, or who are highly motivated. Identification and satisfaction of these rather vague objectives will increase the scheme’s chance of success.

System objectives are often divided into functional and non-functional objectives. Functional objectives describe things that the system must be able to do. For example, the requirement to interpret and score a trainee’s response to an exercise, and provide him with knowledge of his result, is a functional requirement. It is usually relatively easy to express functional requirements in a way which is both specific and testable. All other requirements are non-functional. A non-functional requirement might be to produce the training scheme within a certain cost or timescale. Non-functional requirements are sometimes described as functional constraints, although this is not always a helpful approach as it can lead to important requirements being added as an after-thought rather than regarded as design priorities. Non-functional requirements often relate to the whole system. They can be difficult to express in a way which is specific or testable. They are also typically more difficult to discover than functional objectives.

Identification and satisfaction of non-functional objectives will increase the scheme’s chance of success. It is important to spend time trying to establish all the
objectives for the training scheme, as the scheme will eventually be judged against all the objectives even if some of them were unknown to the designer. (See section 3.6 below on Acceptance Criteria and Tests.)

3.2.1 Embedded Training

The objectives for an embedded training system can be expressed in a wide range of ways. The task itself is constrained to be machine-based (we shall assume the machine is a computer), but this constrains the nature of the task very little. It can be a programmed or a concept operation, in Singleton's terms (see section 2.1.2.5), requiring on- or off-line training. There may or may not be emergency procedures, great pressure on the operator, periods of inactivity, expensive materials, and so forth. Operation of the machine may be the whole task or an aid to doing something else. The only thing that can be said with any certainty is that operating the machine is a significant part of the task.

3.2.2 Navigation Trainer

Fortunately in the case of the DCRS navigation aid we were in the ideal situation of designing the two parts of the system together: or rather, the main system was designed first, but with the training system in mind, and as development of the main system still continued as the training system was being designed, it would have been possible to make changes to the main system if necessary. The training system was therefore not as tightly constrained by the main system as it would have been if it were already complete and in use before the training system was designed.

However, some aspects of the main system were immutable, and the overall specification for the main system must also form a requirement for the training system as well. Consequently, the following sections describe the navigation system and the ways it impinges on the training system.

It should be noted that this document refers to the design of the navigation aid and not to its implementation, which is currently much more restricted than the system described here.
3.2.3 Navigation

Before going on to discuss the navigation aid, it will be helpful to describe the navigation task in some detail. For the moment I will describe the task as carried out using paper charts, pencil and parallel ruler in the time-honoured fashion.

The basic purpose of navigation is well known: it is to enable the vessel to get from A to B without encountering predictable hazards. There may be other constraints, such as speed, distance travelled, fuel consumption, secrecy, and so forth. The main part of the navigator's task while at sea is to make sure that he has the best possible estimate of where the vessel is. There are various ways to achieve this: the most reliable, simple and frequently used is known as a visual fix. Conspicuous visible objects, usually three, are sighted and their bearing from the vessel measured. The position of the vessel is found on the chart by drawing the bearing line from each object: the vessel is where the three lines cross. There are variations on the method for use when fewer than three objects can be seen. In close inshore waters it is not unusual for the navigator to take a fix every five minutes or more often. He must record the three bearings, measure each one off and transfer it, using a parallel ruler, to the location of the object, find the intersection, calculate the tidal drift being felt, and hence predict the vessel's likely location some way ahead, all before the next fix is due. While doing this he must not neglect to inform the captain of any events which are due soon - turns, for example - and warn him of the conditions he is going to encounter. In between taking visual fixes it is also necessary to keep track of the location, heading and speed of other vessels in the neighbourhood to avoid collisions. It is easy to see that the navigator is under considerable pressure in inshore waters.

The navigator also has access to many other sources of information, including radar, satellite transmissions, navigational sonar, etc. A visual fix will usually give a more accurate estimate of the vessel's location than any other method. Consequently, although the navigator frequently combines several sources of information to get the best fix, visual data is most important and, where time pressure forces him to choose just one type of data, he will be inclined to pick visual data. Throughout this report only the use of visual fix information is considered. This is a major part of the task of navigation, it is a complicated task in its own right, and it is often performed on
its own.

When he is unable to obtain fix information for any reason, the navigator has a
different set of problems. Because each sensor reading and each calculation
introduces an error, the position of “ownship” will never be known with complete
accuracy. If it is some time since the last fix was taken, the range of possible
positions may be quite large. One of his options is to calculate a pool of errors. This
shows the range of possible positions of ownship, based on the last fix position and
estimates of the errors inherent in each sensor reading. For example, the error in the
log reading means that the speed of the vessel is not known accurately, and so the
actual position of the vessel could be slightly ahead of or behind the calculated
position. The variabilities of equipment errors are known, although the estimates
may be out of date if the equipment has not been calibrated recently. Effects of the
environment are more difficult to assess. All the errors are combined to produce a
roughly elliptical shape on a piece of tracing paper, which is placed over the
vessel’s calculated position and shows the area in which the vessel should lie.

To make things easier for himself, the navigator makes a thorough plan of the
journey beforehand. The plan consists of straight-line segments called legs. Where
two legs join he plans a wheelover - a turn. The rate of turn will affect the turning
circle, and hence the point at which the wheelover must begin. From that point he
takes the bearing of a nearby visible object and marks it on the chart. This gives him
an easy test for making the wheelover: it is due when the object is at the marked
bearing. At the planning stage he also marks on the chart the speed and bearing for
each leg, taking into account the tidal flow he can expect to encounter. (He finds
this out by lengthy calculations involving interpolating between the tide values at
various points and times.) He decides how frequently he will want to make visual
fixes and marks the points at which they will occur. He can help himself by
highlighting hazards, including shallow water, marking the bearings of dangerous or
conspicuous objects for later use. Particular circumstances may require him to
prepare for other events as well: in a submarine he may need to plan when to make
a complete sweep of the horizon using the periscope.

The plan will be made well in advance, probably weeks ahead of the exercise itself.
Even so, it only helps to a limited extent because the conditions he encounters will
inevitably differ from what he predicts. The actual tidal flow may be far from the value shown by his tables - even in a different direction - and there may be hazards not shown on the chart, such as shipping. He may need to amend his plan or abandon it totally. Nevertheless the navigator must respond to whatever occurs. He must make the best use of the information provided to him, perform his calculations, and keep the captain informed at all times.

3.2.4 The Navigation Aid: DCRS

The DCRS navigation aid is intended to relieve the navigator of some of the more repetitive and mechanical of his tasks so that he can concentrate on the higher level, strategic picture. It is based around a digitised chart, displayed on a screen, which replaces the navigator’s traditional paper chart. The Functional Specification for the navigation aid (Walters, 1987) describes the two phases of operation. In the planning phase the user draws a navigation plan onto a selected digitised chart. Facilities are provided for him to change the plan, and add all the necessary details. When instructed, the aid checks for consistency, constraint violation, and so forth. It also produces a list of planned events. In the execution phase the aid is provided with sensor data (real or simulated) which it uses to plot the positions of ownship and other nearby vessels. The actual position of ownship at all times is compared with the navigation plan and the user is warned of forthcoming planned events. In both phases the aid carries out various calculations for the user, automatically where possible but otherwise at the user’s request.

The important benefits of the navigation aid to the user are firstly the digitised chart and plan editing facilities, which reduce the time needed to plan an exercise; secondly the database of useful information, such as tides, vessel data, and so forth, which is easier to consult than the conventional reference books; and thirdly the facilities for carrying out time-consuming calculations. A little more information on each of these will give a better idea of the uses of the navigation aid and hence of the training requirement.

3.2.4.1 Digitised Chart and Plan Editing

One or more digitised charts is stored in the navigation aid and called up by the
navigator according to his current mission. He can view the chart on the screen at different scales and choose whether or not to display information such as tidal flow and depth. To construct his plan, the user draws the legs onto the screen by moving the cursor. When any two consecutive legs are drawn, the system calculates the turning circle between the two. A default rate of turn is used, but the user is free to choose his own. The user sets the interval for visual fixes and the system calculates the points at which fixes will be taken and marks the chart accordingly. Other events are added as required. Tidal values for the time of day and month are calculated from stored tables. Shallow water and other hazards can be highlighted.

3.2.4.2 Database Information

Database information includes tide tables, information on ownship and other vessels, and details of visible objects such as buoys and lighthouses. Generally speaking the database provides information for the navigator to use either directly or as input to calculations. An important feature of the database is that the navigator is able to update it in a temporary fashion. For example, having been in an area for several days he may have more reliable data on current tides.

3.2.4.3 Calculations

Generally speaking most of the planning calculations can be performed during execution as well, although not vice versa. Exceptions are the checking calculations which are carried out only during planning. They take place at the navigator’s request; they ensure that the plan contains all the details it needs, and that the values are consistent. Consistency is checked both within the plan - eg the heading given for a leg must be the vector sum of the required heading and the tidal drift expected - and between the plan and known constraints - eg the vessel has a maximum rate of turn which must not be exceeded, and the depth of water must be sufficient at all times. A plan must be checked for completeness and consistency before it is executed.

The other planning calculations can all be carried out during execution as well. These are calculations of expected tidal drift from the database tide tables, heading to be steered to make good a given track given the current tidal flow, the calculation
of wheelovers, time to an event (including arrival at the destination), etc. Time to an event is more tricky during execution because the ownership might not be on the plan. In some cases the ownership will be so far off the plan that it is not possible to calculate the time to an event in any valid way. It is important for the system to know its own limitations and not try to make calculations with insufficient data unless explicitly instructed by the navigator. If possible, the system gives warnings before each event is due.

Other calculations are only available during execution. A fix actually involves a set of calculations, only a few of which concern the calculation of the fix itself. It acquires the bearings either directly from the source, if it is available in a suitable format, or typed in by the navigator. The system calculates fixes in the same way as the navigator would. It calculates the tidal drift experienced and marks the expected position of the vessel a small way ahead. It calculates the course the navigator will need to steer to make good his planned track given the calculated tide.

An important calculation during execution is the pool of errors. This can be done by the navigation aid much more quickly than the navigator could do it, and with less need to resort to short-cuts. It can be carried out more frequently, so that a good version is always available.

3.2.5 Other Objectives

The navigation trainer will have other objectives, such as not costing too much, producing trained staff who fit in with the way things are currently done, being acceptable to instructors and students, not requiring too much in the way of external resources, and so forth. At the moment it is not possible to specify what these objectives are, particularly since the trainer is not being designed within the organisation for which it is intended. It is only possible to say that there will almost certainly be some sort of objective of each of these types.

3.3 Task Description

Having clarified the objectives, the next stage in the design process is to obtain a thorough description of the task to be trained. Singleton says (see section 2.1.3.2) that the designer "should describe all procedures and decisions and the information
needed to make them."

There are various methods for doing this: hierarchical task analysis (HTA) has proved to be effective and flexible in other cases (see section 2.1.3.3) above. Whatever method is used, the resulting description must contain performance criteria, specifying the standard to which the task is to be trained. These standards will vary according to the circumstances of the task. If it is high pressure, requiring great speed and accuracy, then the performance criteria will be correspondingly high. If the task is not so demanding and students' skills can be perfected on the job, quite a low performance may be enough to get them started. Different standards will apply to different parts of the task. The performance criteria specify the standard students must achieve to graduate from this training scheme: higher standards may be required later on in the training process.

The task description is vital to the whole design process: it is used to define the selection and acceptance criteria, and to specify the training requirement which is used to design the training material itself.

3.3.1 Hierarchical Task Analysis

The method of hierarchical task analysis (HTA) was developed by Annett and Duncan (eg Duncan, 1974). It is outlined in section 2.1.3.3. We shall adopt the stopping rule put forward in (Shepherd and Kontogiannis, 1987). Their rule specifies that redescription of tasks shall stop when it reaches the operations through which the user interfaces with the system.

There are few suggestions for obtaining the information necessary to produce the task analysis. Duncan suggests approaching a skilled operator, although he points out that there is no guarantee that skilled operators use optimal strategies. In (Duncan, 1974), he observed a skilled operator at work, and then asked him how he was carrying out those parts of the task for which there was no observable action. Duncan then checked the strategy the operator reported by withholding information from him until he asked for it. Since more than one hierarchical arrangement can satisfactorily describe the same task, there is no one solution to the problem. A criterion must be that the hierarchy can account for all the operations, and combinations of operations, which are involved in performing the task. The plans
given should be suitable for training and preferably efficient. To achieve this, it is acceptable to use any source of information which is available.

An important preliminary to the task analysis is to decide what aspects of the task are being addressed by the current training scheme. For a well-defined task it may be obvious where the boundaries of the task lie. In general, though, it will be helpful to explicitly state what is to be included and what not. At first the task boundary can only be stated approximately. As the analysis progresses, the details of the boundary will become better defined.

3.3.1.1 Performance Criteria

Part of the description of an operation includes the performance criteria to which it must be performed. Some operations contain their own criteria implicitly: the operation of turning off a switch is performed adequately when the switch has been turned off. The operator can check that the action has been performed by observing that the switch is in the off position, or whatever other sign is provided. Other operations are not so clear: if a specified quantity of material is to be added to a process, the description must show the accuracy to which the quantity must be measured. Without this, it is not possible to tell whether the trainee has been successful: He cannot be given feedback on his performance, and there is no way to tell whether the operation has been successfully trained.

3.3.1.2 Training Criteria

Another set of criteria is involved in training, however. These are not part of the task itself, but of the training process. These training criteria indicate the performance which is required before the trainee is declared trained at a given operation. Training is, after all, not an isolated process, but is a strategy for getting people competent to perform a task in the real world. At some point a decision must be made that a trainee is competent and can be allowed to graduate from the training course.

The training criteria are closely related to the cost of inadequate performance. If there is no cost attached to performing a given operation inadequately, then a high error rate can be tolerated and the training criterion will be trivial. If the cost is very
high, it will be necessary to train to a very high standard before letting trainees loose on the real task. In that case the training criterion may be “attain the performance criteria in each of 20 trials”, or some such stringent requirement.

3.3.2 Embedded Training

The task description contains those parts of the task which it is intended to teach using the embedded trainer. The following kinds of material might be trained:

A How to use the main system - what it does and how to control it;
B How to perform the main task using the system;
C How to do without the main system.

It will not always be appropriate to train all three. If the main system is extremely easy to use, it may not be necessary to teach students the mechanics of operating it (A). If the main task is operating the system, then B may be trivial, consisting simply of instructions to report the state of the system from time to time. C will not always be necessary: often when there is an equipment failure, work stops until the system is fixed. It may also be extremely difficult, particularly if the tasks with and without the aid are very different. However, one of the prime advantages of having an embedded training system is that it can continue to give an experienced operator practice in rarely used emergency procedures, including what to do when the main system breaks down.

Training the operator to use the main system involves teaching him how to operate the aid, how to make it perform the functions he requires, what its limitations are, and, in some cases, the methods it uses. This is quite straightforward, at least in principle: it may be difficult to present the material to the student for a variety of reasons, but at least the information is all available.

Training the operator to perform his main task is more difficult for the automated system. The problem stems from the difficulty in describing the material in such a way that the computer can make use of it. If a strategic decision must be made for which there are no hard and fast rules, the system cannot make that decision for itself. Therefore it can only advise the student, either in general terms, or for fixed examples specified in advance. The system does not know enough to make the decision for itself, so it cannot give specific advice and it cannot assess the student’s
decision. The system cannot teach that decision to the student, and it cannot directly observe his competence and update its model of him. It can, however, present him with exercises in which he is required to make that decision. It can also employ other techniques for getting information on the student (such as asking him whether he feels competent to make the decision). In each case the difficulties of trying to train such nebulous points must be considered in deciding what to include in the task description.

There may also be difficulties in using the embedded trainer to teach emergency procedures to be carried out when the main system fails. Frequently the manual system is quite different from the automated, and may not be directly compatible with it. Certainly the main system is not available to provide information on the trainee’s behaviour. Some other method of observing the trainee will be necessary.

The task description need only contain those parts of the task which will be taught using the embedded trainer. However, unless the separation is clear-cut, it may be helpful to analyse the whole task before deciding which parts of it will be included in the embedded trainer and which will be taught by another method. Upon analysis, it may become clear that some parts of the task are too vaguely specified to be taught by an automated trainer, or that certain operations are too closely connected to be taught separately.

Performance and training criteria for embedded training systems are no different on the whole from those for other training systems. However, in some embedded training systems there will be the freedom to expose trainees to some parts of the real task before they have mastered it all, as part of the training process. This can only happen if the training criteria for some parts of the task refer to circumstances in the environment. The navigation aid is an example of such a system; it is discussed below.

3.3.3 The Navigation Trainer

The three types of material which could be trained in the navigation trainer are:

A  How to use the navigation aid - what it does and how to control it;
B  How to navigate the vessel using the navigation aid;
C  How to navigate the vessel without using the navigation aid.
Navigation is a critical task: it cannot stop in mid-ocean because a navigation aid, on which the navigator has come to depend, has developed a fault which cannot be fixed. He must be taught to navigate manually. The main problem here is caused by the ocean chart. Although the embedded trainer can set exercises in navigating with a paper chart and provide the necessary data, it has no direct way of telling what he has done, as it cannot read the paper chart. The training system can only assess the navigator's performance by requiring him to answer specific questions. It must use the answers to infer what he has done, including the cause of his errors. This is quite a hard problem. One solution would be to give navigators an occasional refresher course on manual navigation using a paper chart while they are on shore and human instructors are available. It is to be hoped that some transfer would take place between the manual and automatic versions of the task. The frequency of these refresher courses should take into account the rate at which the manual skills are lost, and an estimate of the probability of the navigation aid becoming unusable. It is unlikely that the navigation aid could be made so reliable that the probability of failure could be discounted.

As long as the navigation aid remains available, there are two aspects to the navigator's task. He must know how to operate the aid, and he must know how to employ the functions of the aid to navigate. As explained above, it is the second of these which is most difficult to train using an automated trainer.

The stringency of the training criteria will depend on the part of the task. In inshore waters where decisions must be made very quickly, and must be correct, training must be to a high standard. On the other hand, there will be plenty of times when the vessel is in the open sea and circumstances are more relaxed. Not only will the performance criteria be less stringent, as there is a greater margin of error, but training criteria can also be lower, as the cost of a wrong decision will generally be less. For example, visual fixes are often taken every five minutes in inshore waters, while they may be taken every half hour in the open sea. A trainee who takes more than five minutes to complete a fix will obviously not be capable of navigating in inshore waters, but could be allowed to practise in open water. This freedom can only be exploited if the training criteria are given separately for the various circumstances.
3.4 Candidate Description

Before selection criteria can be set, it is necessary to obtain a description of the candidates available for selection, as this will place limits on the selection criteria that can be imposed. The selection criteria can be set without knowing anything about the available candidates. However, if characteristics of the candidate group can be identified then the selection criteria can be geared to that group. Selection criteria can be raised or lowered to match the people who are actually available, and the training scheme itself can also be targeted at that group.

It is to be expected that the composition of the candidate population will change with time. Some changes will be cyclical - for example, the number and ability of candidates may be greatest just after the end of the school year when school leavers start looking for jobs. It may be possible to predict such changes, at least after a while once a pattern has emerged. Other changes may happen to no discernible pattern. It is therefore worthwhile to monitor the candidate population throughout the lifetime of the training scheme if possible.

3.4.1 Embedded Training

Even without knowing the specific task being trained, there may be constraints on the population of candidates for an embedded training system. Several of the motivations for using embedded training depend on the fact that a student can use the system, at least for short periods, without supervision. If that is to be the case, it is unlikely that the system will be suitable for use with complete beginners. Neither a rigid system nor a more sophisticated trainer can guarantee to cope with all the difficulties a beginner might encounter, since initial training must work from what he already knows on related subjects, which could be anything at all. In specific cases this may not be a problem, but as the initial training period is crucial in learning any subject, it is probably unwise to train complete novices with an embedded trainer.

3.4.2 The Navigation Trainer

It has not been possible to collect information on the population of candidates for the navigation trainer because the intended use of the system has not been
established. The proposed trainer may be so different from current training options that it will be used differently from them and so it may not be useful to draw parallels from existing training schemes. However, in the absence of better information the best that can be done is to assume that the target population will be similar to that for current navigation training schemes in the Royal Navy.

It has been assumed that trainees come to the navigation trainer with the basic skills for navigating using a paper chart, such as drawing bearing lines and measuring latitude and longitude. These skills are not common in the general population, even within the Royal Navy, so we cannot insist they are present in trainees simply by setting suitable selection criteria. Instead it will be necessary to ensure their presence by training them explicitly, by sending potential trainees on another training course prior to selecting them for the navigation trainer. The nature of this course remains to be decided, but there should be no problem in providing it as the Royal Navy has plenty of experience in such courses. The acceptance criteria of the pre-requisite course will provide fairly detailed information about the candidate population: if we assume only trainees who reached the acceptance criteria will be considered for the navigation trainer, then the acceptance criteria form part of the candidate description.

Useful as this will be in the long run, at the moment it leaves us knowing very little about the candidate group. We know they will have basic navigation skills; from their naval background we can expect a certain level of education - probably to 'O' level or equivalent, and we may also assume they have some familiarity with the objectives and constraints of ship-board life. We will therefore have to continue with the design process without the advantages of a good candidate description.

3.5 Selection Criteria and Tests

In an ideal world, selection tests would select from the candidate population those individuals who are most likely to do well on the job, or who would take least effort to train. All selection tests measure some aspect of a candidate’s performance on a chosen activity. The tests may be functional, where they are chosen purely for their ability to predict job performance, or they may measure traits - aspects of personality and intelligence - which have been identified as important.
Functional tests are empirical. There is no concern for what they might be measuring so long as the results of the tests can be shown to correlate with job success. However, there are significant problems in measuring job performance in most cases (see section 3.6). Even where job performance can be adequately measured, data is only available (except in highly unusual circumstances) from those candidates who were deemed suitable, and not from those who were rejected by the tests. Where the number rejected is large, the error in calculating the correlation will be high. Moreover, between the selection process and performance of the job, selected candidates undergo a training programme which separates them in time and social, educational and economic circumstances from the original candidate group - a training programme which may be in flux as it is refined in the light of experience.

For these reasons trait measurement, although less direct, may be preferred. Using this approach, traits are first identified which it would be desirable for the candidates to possess. In the chosen design approach the task description provides a basis for identifying desirable traits. These form the selection criteria.

The object is to set the selection criteria low enough that a sufficient number of people take the course, and yet we would normally seek to minimise the effort involved in training the students to the required standards. It may be possible to identify some basic abilities or skills which all students must have, either because they cannot be trained (eg good reaction times) or because the effort involved in training is prohibitive (eg reading). The task description can be used to identify the skills necessary for the task, and a minimum standard can be set for each.

If little is known about the candidate group then there may be difficulty in setting the selection criteria, and a risk that candidates with the appropriate abilities do not exist within the group.

The selection criteria are of two kinds: essential and non-essential. Essential criteria include those basic abilities which trainees must have in order to learn the required skills, and those skills which it would be prohibitively difficult to train. Non-essential criteria are used to select the best of the candidates who fulfil all the essential criteria. They can be used to set a certain standard which is then assumed as the entry point for the course. Changes in the candidate population with time may
make it necessary to adjust non-essential criteria if the number of trainees must be maintained.

Selection tests are devised to identify those students who meet the designated selection criteria. There is a wide variety of possible tests, including exam-type tests, psychometric tests, interviews, observation of candidates at work, etc, and they may be more or less formal. Unfortunately, as Singleton says (1974), although various selection methods exist, the more intellectual the criteria, the more fallible will the tests be in choosing students who satisfy the selection criteria.

The tests chosen must be reliable; the results must be the same when the same person is tested under different conditions - by a different tester, or on a different day, for example. This can be demonstrated by carrying out trials of the tests under various conditions.

The tests must also be valid; they must effectively measure what they claim to measure. Firstly, they must have face validity to involved parties - they should be acceptable to the candidates themselves and members of the organisation concerned with the training programme. If these people do not perceive the selection tests to be valid, they may not take the selection process seriously or give it their support.

The test should also have content validity - they should test behaviours relevant to the job for which the candidates are being selected. In the case of a training scheme, candidates are obviously not expected to possess all the skills necessary to perform the final task, but the tests should examine all the skills and abilities identified in the selection criteria.

The tests should have predictive validity. This is the extent to which a test score predicts performance on the job. However, for the reasons given above, performance on the job is difficult to measure, so although predictive validity is the tightest form of validity a selection process can have, it is unlikely to be achieved.

Where an independent measure of a particular trait is available, or if a different selection method (eg a more expensive one) is already in operation, selection tests can be validated by comparing them with these other measures.

Even where a test has been shown to have some form of validity, it can only select,
at best, candidates with a better than random chance of success. No selection test can guarantee that an individual candidate has the skill tested for, will succeed in the training course, or can do the job - or conversely. This is important when it comes to designing the training material (see section 3.8).

3.5.1 Embedded Training

The process of choosing selection criteria is not much different for embedded training from other training situations. If trainees are to use the system unattended it is particularly important to ensure that they are not complete beginners, but have a secure grounding in the basic skills. There should be little difficulty in specifying criteria to achieve this.

If an embedded trainer is to be used as part of a wider training scheme it would be wise to perform some additional tests (possibly informal) before putting students on the embedded trainer, as any who have not got a good grasp of previous material could be further confused if the system is not able to cope properly with their difficulties.

There may be parts of the task which, for practical reasons, it is not possible to train using an embedded trainer. These task elements may be precursors, in which case they should form part of the selection criteria for the embedded trainer.

3.5.2 The Navigation Trainer

In choosing selection criteria for the navigation trainer, we have access to an incomplete task description, and very little information about the population of candidates. We know one of the selection criteria is the ability to perform basic navigation using a paper chart. This criterion needs to be specified more precisely; however, the exact criteria will depend on the training scheme in which the trainees learnt these skills. There is no point in specifying selection criteria which none of the trainees will be able to meet because they have not been taught the required skills.

Other essential criteria are to show that trainees have the abilities necessary to learn the task. However, as some parts of the task, mainly the decision-making elements, are quite intellectual, it will not be possible to guarantee that all selected trainees
have the required abilities even if they pass the selection tests. The selection criteria for these parts of the task should try to establish whether or not candidates can make the sorts of discriminations necessary as the basis for the decisions.

Selection criteria for navigation would try to find out whether candidates have the mathematical skills they will need to understand the various vector calculations involved in navigation.

For the purposes of this thesis, we shall neglect the process of devising selection tests for the navigation aid, since it requires the selection criteria to be specified in detail.

3.6 Acceptance Criteria and Tests

Acceptance criteria are used to demonstrate the acceptability of the training scheme to the organisation by showing that it meets the objectives set down for it. One of the objectives might be that a percentage of graduates are able to perform the specified task to a certain standard. Unfortunately, in many cases the real task is not subject to control. Some aspects of the task may not be required for days or years. Where the task includes emergency procedures the problem is particularly severe. Emergencies are typically infrequent, and yet it is vital that graduates are able to cope with them. Deliberately creating emergencies to test a graduate may place him and others in needless danger, which would be immoral. Evaluating performance on the real task may also be expensive, if it takes a long time or ties up resources such as equipment and staff. For these reasons, acceptance tests are frequently carried out using the results of the training scheme rather than a separate evaluation of graduate's performance on the real task. This approach assumes that graduates are able to transfer their learning from the training situation to the real task. Where expense is the only obstacle to evaluating real task performance, a few graduates may be evaluated on the real task to show that transfer does seem to take place.

The acceptance criteria will usually specify the percentage of students who must successfully complete the training course. Criteria will therefore be derived from the task description. However, although the success of trainees is an important source of information for the acceptance tests, the criteria apply to the scheme as a whole. Acceptance criteria should also take into account all other objectives, and any
constraints such as cost, time scales, how the scheme is received by instructors and trainees, and how graduates of the scheme are received by the rest of the organisation. It is important to remember that the training scheme takes place within the organisation, and not in isolation (see section 2.1.3).

Two points are worth making about acceptance criteria in general. Firstly, the criteria relate to the organisation’s objectives for the training scheme, and since these can vary widely, so can the acceptance criteria. Secondly, although the acceptance criteria are used to try to demonstrate the acceptability of the training scheme, there is no guarantee that the scheme will be acceptable even if it meets the criteria, because the organisation may have other objectives which it neglected to state. This is the reason why it is so important to spend time determining the objectives at the beginning and to try to leave none unstated. It is hard to design a training scheme to meet unknown criteria.

Acceptance criteria which relate to performance of the task translate into acceptance tests which show how many graduates are able to perform the task. In testing aspects of the task for acceptance purposes, trainees must be tested in problems they have not encountered before. This not only ensures trainees do not remember the solution from a previous occasion, but also helps to show that they are able to transfer from familiar to novel situations. The most common way to achieve this is to set some problems aside for acceptance testing. This also promotes standardisation of acceptance tests. Other methods are to keep a record of the problems given to each trainee and check this record for repeats, or to generate problems with random variation so that the likelihood of a repeat is infinitesimal.

Devising acceptance tests for the less specific criteria is not easy. This is often the intention of the various questionnaires which students of new training schemes are asked to complete. While not providing rigorous proof of acceptability, the results of a questionnaire may serve to demonstrate to those in charge that the scheme is a success.

3.6.1 Embedded Training

Just as there is very little to be said in general about the objectives of embedded training systems, so the acceptance criteria are similar to those for other kinds of
In principle an embedded training system ought to make it more likely that transfer takes place between the training situation and the real task because training takes place on the actual equipment, and so the difference between the training conditions and the final performance conditions is less than it would be for a separate trainer. The training environment can be made identical with the operational environment by removing all additional facilities provided by the training system. The problem of transfer can never be eliminated, though, since the situations used in the training system are at best only a subset of what can happen in real life. Moreover, there is not the same stress on a trainee dealing with an emergency during training as there would be if he were actually in danger.

3.6.2 The Navigation Trainer

The objectives for the navigation trainer are to produce personnel trained in navigation to the extent that they can navigate adequately at sea in both normal and emergency situations, both with and without the navigation aid. Navigation is certainly an example of the class of task where performance cannot be thoroughly evaluated in a real situation due to the need to deal with occasional life-threatening emergencies. Over a period of time it is possible for a knowledgeable observer to make a subjective assessment of the competence or otherwise of a navigator, but it is only when a specific emergency presents itself that the navigator’s ability to deal with it can be known for sure. Acceptance testing of the navigation trainer will therefore be largely based on the trainees’ performance on the training scheme.

It is not known what percentage of trainees must pass the course for the trainer to be acceptable. However, it will be bound up with the cost: unless the scheme can produce graduates who are significantly more competent than those of previous training schemes there will be no justification for it to cost more per graduate than they did. The cost must be measured in terms of time, staff and other resources as well as capital outlay.

Once the objectives for the navigation trainer are known in detail it will be possible to draw up a set of acceptance criteria which will show what percentage of students must pass the course and what constitutes a “pass” (this depends on the training
criteria: students must achieve the training criteria in all aspects of the course).

3.7 Training Requirement

The training requirement indicates how much of the task description has to be trained. Students can be assumed to possess some skills at the beginning of the training scheme because the selection tests have selected for them. The training requirement is the difference between the skills specified in the task description, with their performance criteria, and those specified by the selection criteria. In other words it is the difference between what the students can do at the beginning of the training scheme, and what they must be able to do at the end. Where possible the vague, general criteria mentioned above should also be included in the training requirement. The construction of the training requirement is a simple matter if the task description and selection criteria have been developed as suggested here.

The training requirement constructed in this way must be treated carefully. In practice the training requirement will be different for each student, and none is likely to have the training requirement calculated. This is because the selection tests are distinctly fallible, and because each student will have skills in addition to those required by the selection criteria. For these reasons the calculated training requirement must be regarded as a tentative hypothesis about the training requirement for each student. As long as the training requirement is used on this basis, it can serve as a useful first guess about what needs to be trained.

3.8 Training Material

The training material itself is next designed and created. The training requirement is used as the basis: each element of the training requirement must have corresponding training material which will train its skills. Various writers (eg Singleton - see section 2.1.3.2) have produced lists of factors to consider and decisions to make during this process, but the advice usually seems to be couched at too high a level to be really helpful. Stages 2 to 6 of Singleton's six-stage procedure come under this heading, but we do not particularly follow them. It would seem that there is no substitute for experience of training design and that the best help the designer can have is an explicit, clear statement of the training requirement.
The systems approach would lead the designer to be very goal-directed, taking each part of the training requirement in turn and deciding how best to train it. By paying attention to the way in which task elements relate, the designer is given some guidance on the order in which he might train them. This process still relies a great deal on the experience of the designer. However, keeping closely to the stated training requirement, and hence to the task description, does make sure that the material is relevant.

An essential part of designing the training scheme is to design the performance tests which will establish whether each student has achieved the required standard at the end of the scheme. Generally the way these criteria are stated will determine the way they are tested. In an industrial setting where a specific task is to be mastered, the only true test of mastery is to perform the task itself (using the operational equipment rather than a training device) to the standard stated in the training criteria. However, as explained in section 3.6, this is often not practical, and so the evaluation contained in the training scheme will be the only measure of the trainees' performance.

The design of the training scheme should be consistent with the general model of instruction described in section 2.3. The discussion in that section shows that a flexible adaptive computer-based training system needs to contain more than just lesson material and exercises for the student to practise. In the model, the first step in a training session is to select an element for instruction. This selection can be done when the training scheme is designed - in which case the next element for instruction is simply the next element on the list - or it can be done "on the spot" by considering what the student has learnt so far, the structure of the task, how quick a learner he is, and possibly his learning style. In a computer-based system, if selection of the next element is to be done during the training session, the system will need to have some basis for making the selection. A flexible system would need explicit information about the student and the task, and some expression of training knowledge which will tell it how to use that information to select the next element.

The element to teach having been selected, the next step is to adjust the training conditions to suit the student. Obviously, not all training conditions can be controlled in every training situation; in some computer-based systems there may be
very few conditions which are subject to control. However, it is desirable that the intro-
ductory material and the practice conditions should be adjusted to suit the student. The provision of supporting features - performance aids and knowledge of results extrinsic to the task - is an important part of teaching. A computer-based system must include the means to decide what supporting features should be provided for each exercise, and how they should be removed as training proceeds.

The element of instruction begins with the student being prepared for practice by receiving the introductory material, or pre-instruction. This successfully completed, practice begins. Different methods of practice will require different facilities, from pencil and paper to detailed simulators. The choice of method will depend on cost, available resources, and the degree of fidelity deemed necessary to ensure that transfer takes place between the learning environment and the workplace. Whatever method is chosen, an appropriate learning environment must be provided.

The student’s performance during practice is monitored, and the results used to update the model of the student; this is either a mental model of the student in the case of a human instructor, or some kind of model built and maintained by a computer-based system. (Many computer-based training systems do not contain explicit student models, but this results in a much less flexible system than that considered here.) In turn, the student model is used to select the next element for instruction and the training conditions to apply to it.

Design of the training scheme includes identifying a structure which can provide this flow of instruction. In a computer system, the design will include the representation of the student and task models, and of the teaching knowledge. The practice environment and support features must be specified. Design of the training material, including how it is to be presented, in what order and under what conditions, is only one aspect of the design.

Creation of the training material should follow from the design. It may be possible to take advantage of material which is already available in the form of books, films, computer programs, and so on.

The design and creation of the training material for both a general embedded trainer and the navigation trainer are discussed in some detail in chapter 6.
3.9 Results and Assessment of the Training Scheme

Eventually the training scheme can be run. A group of candidates takes the selection tests in the usual way, is enrolled on the scheme, trained, and takes the performance tests at the end. The results of the training scheme, in the first instance, are the performance tests of a set of trainees. These can be analysed as a group to show how many trainees are succeeding, how many do not complete the course, how long each student is taking to complete the course (if it is variable), and so on, with a view to showing that the acceptance criteria have been met. These results may occasionally lead to a scheme being assessed as an outright failure or success, but it is more likely that it will be judged a qualified success and that the assessment can be used to make improvements to the scheme next time round. This feedback is a vital part of the design process which increases the chance of the training scheme being acceptable to the organisation. It must occur throughout the lifecycle of the scheme if it is to adapt to changes in its conditions, such as the candidate group, management practices, the task, and so on. Feedback may affect the design process at any stage.

3.10 Conclusions

The design approach described above is a disciplined method of developing a training scheme, allowing for the specification of assessment criteria based on identified objectives, and including the selection of trainees for the scheme. The assessment criteria, being relatively independent of the content of the training scheme, can be used to show how well the training scheme meets the objectives laid down for it. If the scheme has shortcomings, the assessment process will indicate where these lie so that they can be remedied. It has been stressed that identification of all the objectives which the organisation has for the training scheme is vital. Those objectives which are initially hidden are most important; if they are not discovered until the scheme has been implemented and the assessment carried out, the designers will have no chance of meeting them and the scheme is likely to fail.

When it comes to designing and creating the training material itself, the design method outlined above is no longer sufficient on its own. Although it is certainly necessary to keep the objectives of the training scheme clearly in mind at all times,
another guiding principle is needed at this stage to structure the material. If an explicit model of the task to be taught is included in the training system, this can be used to structure the system and the training material. Here the task model is derived from a description of the task constructed using Hierarchical Task Analysis. The design of the navigation trainer, including the task analysis, is described in chapters 4 to 6.

Throughout this chapter, the design method has been described as it applies to embedded training, and particularly to the navigation trainer, as this has been the emphasis of the project. However, the same method, including the use of Hierarchical Task Analysis, has subsequently been applied to a project for the European Space Agency which designed and built a demonstrator for an information retrieval training system. This is a quite different application area, but the design method was applied successfully, giving an indication of its generality. The information retrieval trainer was a much simpler system than the navigation trainer, but shares a number of features. It is described in chapter 7.
4. DESIGN CONCEPT FOR AN EMBEDDED TRAINER

Chapter 3 began to discuss some of the issues involved in designing training material for a training scheme within the design approach laid out in that chapter. Along with design of the training material must go design of the learning environment, delivery mechanisms, and support facilities. In the case of a computer-based training system the whole computer package is designed at this stage.

This chapter continues the discussion, considering in more detail the design issues involved in both embedded training in general and the navigation trainer in particular. A top-level, or "architectural" software design for the navigation trainer is developed.

4.1 Embedded Training

The chief feature of an embedded training system is that the actual equipment that trainees are learning to use is available for use during training. It seems only sensible to take advantage of this feature by allowing the student to use the equipment as much as possible.

4.1.1 Practice and Simulation

If the environment does not provide suitable problems for the student to practise on, it may be necessary to simulate environmental data for training. This may be necessary if the environment is uneventful, or if it contains situations which are too difficult for the trainee to cope with. Alternatively, in the second case, it may be possible to provide the student with sufficient support to bring the task within his means. This could be achieved by the system (or an instructor) doing some parts of the task for him, by giving him advice, by providing memory aids, etc.

Another circumstance in which simulators are useful in embedded training is if the task has elements which are needed only rarely but which must be performed well when they are required. The most common example of this is emergency procedures. By simulating emergencies and other rare events, the trainer can ensure that the student has practised them thoroughly. Ideally the simulator will use the navigation aid equipment as much as possible and be well integrated into the rest of
the system. The training system must be able to give instructions to the simulator to provide the training examples it requires. The simulator can work in a variety of ways. If it generates data on demand, flexibility can be built in so that the trainer can also control the conditions of training, such as the speed at which data is presented to the student, errors in the information, and so on. Stored examples will be less flexible and may be repetitive, but may be the only practical method in some cases.

4.1.2 Other Training Material

The other kinds of training material required in the embedded trainer are introductory material for each topic, and knowledge of results beyond that provided by the task. For example, a simple exposition of terms and basic principles may be useful at the beginning of a new topic. Diagrams, either static or dynamic, models of various kinds, mnemonics and procedural guides, advice-giving systems such as expert systems, question-answering "help" facilities, and any manner of other computer-based training material could be built into an embedded trainer. Knowledge of results may need to include remedial material.

4.1.3 Controlling the Presentation of Material

Whatever training material is used, a mechanism must control how it is presented to the trainee. If there are any decisions to be made about how the material is presented, the control mechanism must be provided with the information it needs to make the decisions. The two main sources of variability on which it can base its decisions are the environment and the student. Information about the environment is collected by observing the incoming environmental data; that about the student by observing the student’s interactions with the system. No other dynamic sources of information are ordinarily available to the training system. The amount of information it needs from these sources depends on the flexibility in the training material. The amount it can obtain depends on the bandwidth of the incoming data. The amount of information it has about the student is more likely to be constrained by the conclusions it can draw from the incoming data than by restrictions on the nature of its interaction with the student. Different methods for compiling information on the student are described in section 2.2.2.2 on user models.
A helpful method for obtaining information about a student is to start the first session with a new student by setting him a series of exercises and questions designed exclusively to test his knowledge. It may even be possible, if the complete set of selection tests is not too large, to add further tests which have nothing to do with selection but provide information about the prior skills and knowledge of individual students for use in the training scheme.

4.2 The Navigation Trainer

The navigation trainer places emphasis on practice, using the navigation aid in which it is embedded. A simulator is provided to allow students to practise emergency procedures, which are an important component of the task. However, the structure of the trainer permits any available training material to be incorporated. Because the trainer is designed to be used without supervision, a large part of this section concentrates on how presentation of the training material is controlled by the trainer, rather than on the content of the training material.

4.2.1 Controlling the Presentation of Material

Ohlsson argues (see section 2.1.2.5) that the training program should have overall structure. To provide this, the system constructs a tutorial plan. The plan shows how the system intends to teach the remaining material to the student. It is based on the remaining training requirement: that is, that part of the task description which has still to be taught. To determine this, the system keeps a record of what the student knows. This is called the student model, and is based on the task description.

The tutorial plan has to be flexible, as the training program is unlikely to unfold in quite the way that was expected. It must be continually compared with the student model to make sure it is still valid. The student model must therefore be kept up to date in the light of new information gathered by observing the student.

The system must report the progress of the training program to the program supervisor. Otherwise there is no-one other than the student who knows what he has been taught.

So the system must have these functions:
i. Monitor the student’s performance;
ii. Build a model of the student;
iii. Construct a tutorial plan;
iv. Execute the tutorial plan;

This is a slightly different grouping of functions from that of the model of instruction in section 2.3, due to the addition of the tutorial plan, which is not an essential feature of a training system. Constructing the tutorial plan involves selecting the sequence of elements for instruction and specifying the training conditions for each. Executing the tutorial plan involves preparing the student for practice, providing practice, aiding performance, and providing knowledge of results. The functions of constructing and executing the tutorial plan are deliberately kept separate. The planning function is concerned with giving structure to a series of lessons by planning them as a unit. The execution function is only concerned with the one lesson which it is in the process of teaching.

Executing the tutorial plan entails communication with the student. It requires simulation of environmental data since an important part of training for the navigation aid is learning to cope with emergency procedures. If the system is to continue to model the student when he is working on real environmental data, it must interpret the environmental data and classify the situations which face the student.

The functions of monitoring the student’s performance and building the student model are carried out together, and can be amalgamated.

Consequently, the training system needs three additional functions:
vi. Communicate with the student;
vii. Simulate environmental data;
viii. Classify environmental data.

4.2.2 Components of the Trainer

In order to carry out these functions, the training system has the following
components:
Model Builder: builds a model of the student by observing his behaviour;
Plan Builder: builds the tutorial plan;
Executor: executes the tutorial plan;
Report Generator: provides reports on the students, either individually or as a group;
Tutor Interface: handles communication with the student;
Scenario Generator: simulates environmental data as instructed by the Executor;
Classifier: classifies the environmental data and describes the situation.

The functions of observing and modelling the student have been combined into a single component, the Model Builder.

The function of each component is described in general terms below.

Model Builder

The Model Builder builds a model of the student by observing his performance. In terms of the model of instruction in section 2.3, this is two functions: monitor the student’s performance, and update the student model.

The bulk of the model is based on the task: it records which parts of the task he can perform, according to the evidence observed by the Model Builder. This is basically an overlay model (see section 2.2.2.2). The HTA of navigation is used for this, together with rules telling the Model Builder what conclusions it may draw. In addition, the model will contain non-task-specific inferences, such as how much guidance the student needs or wants, the level to which tasks must be broken down for him, his ability to use learning strategies, etc.

The Model Builder is not generally interested in the detail of the student’s actions: it wants to know that the student has successfully completed an exercise, not the fact that he mistyped a number. It therefore receives most of its information from the Executor, interpreted in the light of the current lesson. However, the student may also ask questions during a lesson, which may or may not be related to the current lesson; these are reported to the Model Builder directly. The Model Builder must exploit both positive and negative evidence about the student’s performance - what he can do as well as what he cannot.
Beyond the task description provided by HTA, the model builder can be told about the similarity between operations so that when it observes the student performing one operation it registers evidence that he can also perform similar operations, assuming that a certain amount of transfer will take place between the operations. In practice, although experiments on transfer of training show that transfer takes place, and give some grounds for predicting when it will happen, the evidence is not clear-cut, and any prediction of transfer effects must be tested. This can be handled in various ways. The simplest is to initially disregard transfer effects in the Model Builder, but to record whether each student learns the second task more quickly than the first. Over time, this record will suggest which pairs of tasks show transfer effects.

The student model is central to the system; the Model Builder must be able to provide information from the student model to almost every other component.

Plan Builder

The Plan Builder builds the tutorial plan. In the Plan Builder's terms, the aim is that the student should be able to perform all aspects of the task to the specified criteria; the operators are instructional elements. The Plan Builder contains rules which govern how the remaining tasks should be trained, both in terms of their order and of the training material to be used. Where the plan contains instructions for practising parts of the task, it must specify the conditions for the practice, such as the aid the student is to be given, the difficulty of the situation presented to him, and so forth.

An example of a rule which could be used by the Plan Builder may amount to: "If the student has previously had difficulty doing tasks at the required speed, then start by practising the task slowly".

The Plan Builder does not attempt to fill in all the detail for the whole plan at the start; this would be a waste of effort, as the plan is sure to need amendment at some stage. Rather, detail is filled in for the early parts of the plan, and the later parts are only specified in high-level terms. Particularly in the early stages in instruction, the student model will contain many gaps. If such a gap means the Plan Builder is unable to provide sufficient detail for the early part of the plan, it can add an item to
the plan whose purpose is to gather information about the student by asking him questions, or presenting him with a new situation to see whether he can deal with it. The plan is examined after each instructional element has been taught. It is amended if necessary, and further detail added, based on the updated student model. If the student model shows that the instructional element was not successful in teaching its task element, then some remedial tuition must be added to the plan.

An important feature of the Plan Builder is that it is prepared to give up. If many lessons are failing to achieve their objective, the Plan Builder may have to decide that it has encountered a student it cannot cope with for some reason. To struggle on would risk confusing and demoralising the student.

Executor

The Executor executes the tutorial plan. It communicates with the student via the Tutor Interface. An element in the tutorial plan is an introductory lesson and some exercises. The Executor is responsible for executing a whole element, including setting suitable exercises. The plan will show the parameters within which the exercises must be set. The Executor must provide support facilities and knowledge of results as directed by the plan.

The Executor will give the student as many exercises as he requires to achieve mastery of the instructional element. It therefore needs to judge when the student's performance reaches the standard required by the plan. A plan element can be judged to have failed if, over a number of exercises, the student seems to be making no progress or is getting worse.

The Executor interprets the student's responses and provides information to the Model Builder (eg the student failed an exercise because he was not sufficiently accurate).

Report Generator

The Report Generator provides reports on the students, either individually or as a group. This is a database access function which allows the instructor to retrieve the information he requires from the student model.
Tutor Interface

The Tutor Interface handles all communication with the student. It answers the student’s questions where it is able to. The range of questions the student can ask is limited by the syntax provided by the Interface; it will not be equipped to understand general English input. The Tutor Interface may request information from the Model Builder to help it interpret a question, or it may need to ask the student a question for clarification. Responses to the student must not use concepts which the student has not learnt yet.

The Tutor Interface reports the student’s behaviour and responses to the Executor and the Model Builder.

Scenario Generator

The Scenario Generator simulates environmental data as instructed by the Executor, to provide the student with practice exercises appropriate to the tutorial plan. It frees the system from having to rely on real environmental data for the student to practise on. That would be inadequate for the navigation trainer, where emergency procedures are an important part of the material to be trained.

Classifier

The Classifier classifies the environmental data and describes the situation. This enables the system to continue to model the student when he is working with real environmental data. With the Classifier, the system becomes a genuinely useful continuation trainer. It can observe a navigator at work, and subsequently provide him with examples of kinds of situation which he has not been exposed to recently. However, the method used by the Classifier has not been defined, and it is not clear that automatic classification of environmental data is possible. If not, the function of the main trainer is not impaired.

4.2.3 Structure of the Trainer

When the system is being used for training, it has the structure shown in figure 4.1. The structure is determined by the communication needs of each component. The Report Generator, for instance, need only communicate with the Model Builder,
since that has all the information which might need to be included in a report.

The Plan Builder also needs to know the state of the student model to build its plan. The plan can be made to an arbitrary level of detail. Executing a fully-detailed plan, with exercises specified completely, would be a trivial task. However, to achieve this level of detail the Plan Builder would need to know exactly what the student did during the previous exercise, and it would need to communicate with the navigation aid. Rather than involve the Plan Builder in these complications, the planning task was divided into two stages. The Plan Builder is responsible for deciding the order in which task elements will be taught, and it contains information about how to teach each one. From this, for each task element it constructs a corresponding plan element specifying introductory material and the general conditions of exercises for the student to practise the task element. It sends each plan element to the Executor as it becomes current. The Executor is made responsible for delivering the introductory material to the student, and for setting each exercise, including controlling the level of difficulty. The Executor sets each exercise by issuing instructions to the Scenario Generator.

The Executor reports the student’s responses to the Model Builder. The Model
Builder is the authority on whether the student has successfully learnt to perform a task, since it has more information than the Executor. However, the Executor is responsible for increasing the difficulty of the exercises within the parameters specified by the plan, as the student improves.

The Scenario Generator creates an exercise by simulating environmental data. The data is passed to the student rather than directly to the navigation aid, although that would make more sense in an operational system. The reasons for that decision are given in section 5.3 below.

The other major influence on the structure of the training system has been the provision of adaptive "help" or "explanation" facilities to answer the student's questions. These were initially seen as very important, since the trainer is intended for use in conditions where there might be no-one available to answer questions for the student. However, adaptive explanation facilities are difficult to provide. Most of those which exist are not particularly useful even to an expert user, and would certainly not help a student. Research is continuing, with Swartout's work (see 2.2.2.1) looking promising, but useful adaptive explanation systems still seem to be some way off. An alternative is to concentrate on providing well-worded explanations which are specific to the current context but not to the individual student.

When the navigator is using the navigation aid with real environmental data, the structure shown in figure 4.2 applies.

Between training sessions the navigation aid is an operational piece of equipment. When a navigator is using it for real the training package must not interrupt him. It must not take the initiative in communicating with the navigator except in training mode. However, if the Classifier can identify the situations with which the navigator is presented, the Model Builder can continue to build its model of him. If the navigator takes the initiative in communication, by asking a question, then the training package is available to reply. Should the available processing power be insufficient to perform both navigation and training functions fast enough to keep pace with events, training functions must of course be sacrificed.
4.2.4 Sequence of Events

Before teaching can take place, a plan must be devised. Unless the instructor provides information about the student in advance (perhaps from acceptance tests), the system initially knows nothing specific about the student. The first step is therefore for the Model Builder to create a new student model containing the student’s name or other identification, and some tentative default values.

The Plan Builder proceeds to build a tutorial plan. The plan is structured by the task model, so it is hierarchical. Tasks can be taught as a whole, or they can (usually) be broken down into subtasks, often in several ways (top-down or bottom-up, for example). At each stage, and each hierarchical level, the Plan Builder needs to know enough to decide how to construct the next plan stage according to its rules. If it needs more information about the student, it can set some exercises or questions to provide itself with that information.

It is not necessary for the system to build a plan for teaching the whole navigation task at the beginning. The plan will have to be changed, perhaps significantly, as new information arrives about the student, and even if this were not so, the details

Figure 4.2: Structure of the Navigation Trainer, with real sensor data (Arrows show lines of communication between the components.)
and later parts of the plan can be added as teaching progresses.

The student model and tutorial plan now being ready, the Executor executes the first

![Diagram: Sequence of Events once Training Session has started]

plan element. Figure 4.3 shows the sequence of events once the training session is underway.

The Executor carries out the instructions in the plan element; it prepares the student for practice by giving him introductory material, it generates an exercise for the student to practise on, with practice conditions (exercise conditions, support facilities and knowledge of results) as specified by the plan. As the student responds to the element, by performing the exercise he has been set, answering questions, and so forth, his actions are relayed to the Model Builder which updates the student model. There are two reasons for updating the model at this stage: firstly the model will be out-of-date because the student will now know more than he used to; secondly, it may now be possible to draw further conclusions about what the student knows, or to revise previous conclusions. The student model is never totally complete or accurate.

When the plan element has been executed, the Plan Builder is asked to revise the plan in the light of the changes made to the student model. Generally the revision will be slight; perhaps an element further on will be removed because there is now evidence that the student can do that operation. Sometimes the student model will change so much that the plan has to be totally reconstructed.

The sequence shown in figure 4.3 is repeated until either the plan is complete or the student wants to end the session. At this point the student model and the plan are stored for use in the next session. When the student arrives for another session, the model and plan are retrieved. The plan is checked before use; if some time has
passed between sessions, the student may have forgotten some of what he knew, or he may have learnt from other sources. The Model Builder will recognise that its information may be out-of-date and therefore reduce the confidence it has in it. The Plan Builder will abandon the previous plan on the same basis, and build a new one. If the previous model and plan are recent, they will be used as they stand.

Between training sessions the student may use the Aid for actual navigation. The purpose of the Classifier is to provide the Executor with a classification of the situation the navigator faces. The Executor compares the student’s actions with the classification to provide information for the Model Builder to use to update the student model between training sessions.

The instructor can request reports about the students, either as a group or as individuals. These are produced by the Report Generator on the basis of information provided by the Model Builder at its request. This is purely a cosmetic exercise to translate information contained in the model into terms comprehensible to the instructor; the Report Generator adds nothing itself.

4.3 Conclusions

This chapter has shown how the Navigation Trainer was designed to perform the various functions required for an adaptive training system which is sufficiently flexible to be used without supervision. The design of the Trainer was greatly influenced by the desire to include a Plan Builder to give overall structure to the training sequence. This has caused the Navigation Trainer to have a flow of instruction which is a more complicated version of the general model of instruction described in section 2.3.

Central to the design of the Navigation Trainer is the student model, which in turn is based on the task description for navigation. Although this was carried out before the Trainer was designed, it has not been described yet. Chapter 5 which follows explains how the task description was obtained using Hierarchical Task Analysis, and describes the results.

Chapter 6 describes each component of the Trainer in more detail, including how the components communicate with each other and with the student, and how the
task model is incorporated in the Trainer.
5. HIERARCHICAL TASK ANALYSIS OF NAVIGATION

5.1 Conduct of the Task Analysis

The description of the task of navigation was obtained using Hierarchical Task Analysis (see section 2.1.3.3). To familiarise myself with the task I attended the POOW (Petty Officer of the Watch) Submarine Navigation course, a week-long course held at HMS Dolphin to train Petty Officers in elementary navigation skills. On the basis of this course I attempted an initial hierarchical description of the task. This was amended and improved during discussions with navigation instructors from HMS Dolphin, themselves experienced navigators. I was only able to run one major interview, lasting about two hours, with my chief informant, Lt Royle, because of the distance between HMS Dolphin and Teddington, and the demand on his time. This was supplemented by telephone calls with Lt Royle and others. Lt Royle also checked and corrected several versions of the task description. He had been introduced to the format of the description during the interview, and had no difficulty working with it.

In talking to navigation instructors, it appeared that there was considerable consensus about how navigation is taught. When instructors were shown the final version of the task description, they would sometimes suggest an alternative breakdown of some part of the task. However, there was no occasion when they were not able to agree on what that part of the task involved, and they could accept that the description given was an acceptable alternative. This is presumably due to the degree of standardisation imposed by the Navy on its training procedures, and the stability of the current navigation training courses. As I was not able to observe navigators at sea, it was not possible to judge how well they keep to the procedures they were taught. However, it appears that departures from the procedures are seen as transgressions, and that navigators intend to follow them.

5.2 Scope

The task analysis reflects the two stages of navigation: planning the exercise and carrying it out. Only the second of these two stages has been analysed. Similarly, the first and last stages of executing the plan, briefing and debriefing, have not been
analysed on the grounds that they are not suitable tasks for the embedded trainer to tackle.

It is not possible to say with complete certainty that the task analysis meets the criteria mentioned in section 3.3.1: ie, that the hierarchy can account for all the operations and combinations of operations which are involved in performing the task, and that the plans should be suitable for training and preferably efficient. Experienced navigators have confirmed that it accounts for all the operations, and combinations of operations, which are involved in performing the task. The plans would appear to be suitable for training in that they relate closely to the way navigation is taught and performed. They have been formed with attention to efficiency, and redundancy has been avoided.

In analysing the automated task, Shepherd's *stopping rule*, which says that analysis should stop when it reaches the operations which interface with the system, has been applied in general. However, there have been exceptions. In some cases the navigator is required to perform checks or take decisions which are not fully specified. These are examples of the situation described in section 3.3.2 above, where the automated trainer cannot teach the decision-making task because it does not possess a full description of how it is carried out. In these cases a more pragmatic stopping rule has been applied: analysis was stopped when it could go no further. These items need training differently and are recognised as points where problems could arise with the trainer.

5.3 Assumptions

Several assumptions have been made for the sake of the task analysis itself. The main assumption concerns the definition of the *plan*. We have taken the plan to be *what the navigator intends to do*. Hence if he changes his mind during the exercise, deciding to decrease the fixing interval because of hazards, for example, this constitutes a change in the plan. Similarly, setting a TSU interval for a vessel (this is the interval at which the periscope is trained on the vessel to check its bearing and speed), which cannot be done during the planning stage, is still deemed to be a change in the plan. This is a device which simplifies the description of the task considerably, since it means decisions taken during plan execution do not have to be
treated as special cases. A clearer, simpler task description should lead to smoother action in the trainer. However, the wording is not entirely consistent with navigators' usage. A change of wording in the trainer ought to overcome any problems this gives rise to: failing that, the task may need to be redescribed.

The functions of the navigation aid are assumed to be open to inspection by the trainer and therefore are not analysed separately. They are described in the DCRS documentation. This is an important feature of the embedded training system as it enables the training package to keep track of changes in the main system automatically. The impact of this on the design of the main system is described in section 8.4.5.

For the purposes of the navigation trainer we assume that the basic skills involved in navigating using a paper chart - such as drawing bearing lines and measuring latitude and longitude - do not need to be taught by the trainer. We assume that these skills have not only been taught to students before they come to the trainer, but also that any necessary rehearsal is done elsewhere. We also assume that this aspect of the navigation task can be trained separately and that the transition from the navigation aid to paper charts in emergency conditions can be made provided the navigator is capable of both parts of the task. This is an assumption which will need testing: if it is not the case then the embedded trainer cannot train navigators to do without the navigation aid (type C material in section 3.3.3). This would be a serious restriction.

All sensor data and communications beyond the navigator/navigation-aid system are assumed to be transmitted using the current manual methods. For example, the navigator receives fix information aurally and types it into the navigation aid. Similarly he makes his status reports orally. This assumption makes little difference to the task analysis or to the tutor (it gives the navigator slightly more to do): it was made so that the navigation aid could fit directly into the current command structure. We would expect much of the communication, particularly the input of sensor data, to be automated if the navigation aid were used for real.

5.4 Results

The diagrams which follow show part of the HTA of navigation. A navigational
exercise can be analysed into two subtasks: plan the exercise, and execute the plan. Only the execution of the plan has been analysed further. The first diagram contains those elements of the task which are common to both manual and automated navigation. Following this are supplementary diagrams showing the break-down of those subtasks which are carried out differently depending on whether or not the navigation aid is in use. So, for example, when the learner is being trained to use the navigation aid, the appropriate set of subtask analyses is added to the main task analysis.

The lettered boxes refer to the plans for each task. They describe how the subtasks are combined to form the task. Together with the criteria for each task, and the input, action and output for those tasks which are not subdivided, they are shown in tables 5.1 to 5.3.
Figure 5.1: Main HTA for Navigation

Figure 5.2: Manual version of task 0.221 Get data for next step
Figure 5.3: Manual version of task 0.231 Receive Input

Figure 5.4: Manual version of task 0.23223 Allow for tide

Figure 5.5: Automatic version of task 0.221 Get data for next step
The tables below show each of the tasks in the diagrams. The number of each task is obtained from the diagram by tracing back up the hierarchy to the main task ("Perform Exercise", task number 0). For example task 0.23, "Process Data", is the third subtask of task 0.2, "Execute Plan". Task 0.23 itself has three subtasks, numbered 0.231, 0.232 and 0.233. In this way each task has a unique number. In the tables the tasks are ordered in a depth-first fashion, so that after each task are listed all of its subtasks (and their subtasks, if any) before any other tasks on its own level or the levels above it. Consequently the tasks are listed in numerical order. Each task shows the criterion for its success. This is not the training criterion, which would show how well the trainer must perform the task to be considered trained, but shows how the navigator may decide whether the task has succeeded or failed. If a task is subdivided, the criterion is phrased in terms of the subtasks. The word iff is used to indicate that a task succeeds or fails according to the success or failure of a particular subtask. A question mark in this column shows that the task has not been analysed.

The third column shows either the subtasks or, if the task is not subdivided, its input, action and output: the data required to perform the task, the action performed on the data, and the results of the action. An asterisk against an input or output item indicates that it is an item communicated outside the navigator/navigation-aid system, such as sensor data, or an instruction for a fix, rather than simply a result of the task.
Table 5.1: Main tasks for navigation

<table>
<thead>
<tr>
<th>Task</th>
<th>Criteria for success</th>
<th>Subtasks/Input-Action-Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (plan A)</td>
<td>If 2 succeeds</td>
<td>1 Make plan</td>
</tr>
<tr>
<td>Perform exercise</td>
<td></td>
<td>2 Execute plan</td>
</tr>
<tr>
<td>Plan A, to perform exercise: When instructed: well before execution is due, make plan (1). On schedule, execute plan (2).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Make plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2 (plan B)</td>
<td>No steps remain</td>
<td>1 Brief command team</td>
</tr>
<tr>
<td>Execute plan</td>
<td>(Fails if 3 fails)</td>
<td>2 Do next step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Process data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 Debrief</td>
</tr>
<tr>
<td>Plan B, to execute plan:</td>
<td>Brief command team</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1). If event is due, do next step (2); if incoming data, process it (3). Repeat 2 and 3 until no more steps. Debrief (4).</td>
<td></td>
</tr>
<tr>
<td>0.21</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Brief command team</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.22 (plan C)</td>
<td>3 is complete</td>
<td>1 Get data for next step</td>
</tr>
<tr>
<td>Do next step</td>
<td>(Fails if 2 fails)</td>
<td>2 Confirm step</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Send instruction</td>
</tr>
<tr>
<td>Plan C, to do next step:</td>
<td>1, 2, 3 in order</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.221</td>
<td>see tables 5.2, 5.3</td>
<td></td>
</tr>
<tr>
<td>Get data for next step</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.222 (plan D)</td>
<td>If all parts succeed</td>
<td>1 Check step required</td>
</tr>
<tr>
<td>Confirm step</td>
<td>(Fails if 2 fails)</td>
<td>2 Check pre-conditions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Check risks</td>
</tr>
<tr>
<td>Plan D, to confirm step:</td>
<td>All three, in any order</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2221</td>
<td>If step is still required</td>
<td>1: Data on next step</td>
</tr>
<tr>
<td>Check if step required</td>
<td>A: Check outcome of step is still required</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O: -</td>
<td></td>
</tr>
<tr>
<td>0.2222</td>
<td>If pre-conditions are satisfied</td>
<td>1: Data on next step</td>
</tr>
<tr>
<td>Check pre-conditions</td>
<td>A: Check whether pre-conditions are satisfied</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O: -</td>
<td></td>
</tr>
</tbody>
</table>
0.2223 (plan E)  
Iff 2 succeeds  
1 Ascertain risks  
2 Justify risks  

Plan E, to check risks: Ascertain risks (1), then justify risks (2)  

0.22231  
see tables 5.2, 5.3  

Ascertain risks  

0.22232  
Risks of next step are justifiable  
A: Decide if justifiable  
O: -  

0.223  
Instruction is sent  
I: Data on next step  
A: Compose and send instruction  
O: *Instruction  

0.23 (plan F)  
3 is complete  
1 Receive input  
2 Analyse data  
3 Report  

Plan F, to process data: 1, 2, 3 in order  

0.231  
see tables 5.2, 5.3  

Receive input  

0.232 (plan G)  
Subtask 1 fails  
1 Find problem  
2 Find solution  

Plan G, to analyse data: Repeat 1 and 2 until either fails  

0.2321 (plan H)  
Iff all tests succeed  
1 Apply pool of errors  
2 Test if plan is safe  
3 Test if on track  

Plan H, to find problem: 1, 2, 3 in order  

0.23211  
see tables 5.2, 5.3  

Apply POE  

0.23212  
see tables 5.2, 5.3  

Test if plan is safe  

0.23213  
Iff ownership is on track  
I: EP  
A: Assess position relative to planned track  
O: Discrepancy between fix and plan  

101
Plan I, to find solution: If hazard, avoid it (1). Review intervals (2). If off track, allow for tide (3). If 1 or 2 fail, abort exercise (4).

Plan J, to avoid hazard: Do any one

Plan K, to review intervals: Do 1, 2 or 3, as necessary
Abort exercise
Exercise is aborted
I: Hazard details
A: Bring emergency procedure into effect
O: Plan for aborted exercise

Results reported
I: Results (fix, fix calcs, tgt track, etc)
A: Report results
O: Report

Debrief complete
I: EP, fix, ARL + TSU intervals
A: Find first of: fix, ARL, look, event
O: Next step plan data

Table 5.1: Main tasks for navigation (end)

The following table (5.2) shows the analysis of those subtasks which only occur, or are performed differently, in the manual task (ie when the navigation aid is not used). The columns show the same is in Table 5.1.

Table 5.2: Supplementary tasks for manual navigation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.221M (plan a)</td>
<td>2 is complete</td>
<td>1 Decide on next step</td>
</tr>
<tr>
<td>Get data for next step</td>
<td>2 Adjust data</td>
<td></td>
</tr>
<tr>
<td>Plan a, to get data for next step: Decide on next step (1), then adjust data (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2211M</td>
<td>Next step chosen</td>
<td>I: EP, fix, ARL + TSU intervals</td>
</tr>
<tr>
<td>Decide on next step</td>
<td>A: Find first of: fix, ARL, look, event</td>
<td>O: Next step plan data</td>
</tr>
<tr>
<td>0.2212M</td>
<td>Next step adjusted for reality</td>
<td>I: Next step plan data, EP</td>
</tr>
<tr>
<td>Adjust data</td>
<td>A: Adjust data to account for reality</td>
<td>O: Data on next step</td>
</tr>
<tr>
<td>0.22231M</td>
<td>Risks found</td>
<td>I: Data on next step</td>
</tr>
<tr>
<td>Ascerten risks</td>
<td>A: Extrapolate courses</td>
<td>O: Risks</td>
</tr>
<tr>
<td>0.231M (plan b)</td>
<td>1 + one other complete</td>
<td>1 Record input data</td>
</tr>
<tr>
<td>Receive input</td>
<td>2 Process fix</td>
<td>3 Process ARL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 Process TSU</td>
</tr>
</tbody>
</table>

Plan b, to receive input: Record input data (1). If input is fix data, process fix (2); if ARL, process ARL (3); if TSU, process TSU (4)
0.2311M  Data recorded  I: *Fix, ARL, or TSU data  
Record data  A: Record data  
    O: *Record of input data

0.2312M (plan c)  3 is complete  1 Plot fix  
Process fix  (Fails if 2 fails)  2 Check fix  
    3 Do fix calcs

Plan c, to process fix: 1, 2, 3 in order

0.23121M  Fix has been plotted on  I: *Fix data  
Plot fix  the chart  A: Plot position on chart  
    O: Fix

0.23122M (plan d)  Iff both parts succeed  1 Apply rules  
Check fix  2 Check cocked hat

Plan d, to check fix: 1 and 2 in any order

0.231221M  Iff fix matches all fix  1 Find tidal drift  
Apply rules  rules  2 DR on  
    3 EP on  4 Calc speed made good

0.231222M  Iff cocked hat OK  1: Fix  
Check cocked hat  A: Check cocked hat looks OK  
    O: -

0.23123M (plan e)  4 is complete  1 Find tidal drift  
Do fix calcs  2 DR on  
    3 EP on

Plan e, to do fix calcs: 1, 2, 3, 4 in order

0.231231M  Previous DR has been  1: Previous DR, fix  
Find tidal drift  subtracted from fix  A: Subtract DR from fix (position vectors)  
    O: Tidal drift

0.231232M (plan f)  2 is complete  1 Construct ship’s heading vector  
DR on  2 Add heading vector to fix

Plan f, to DR on: 1 then 2

0.2312321M  Ship’s heading vector  I: Log speed, compass bearing  
Construct ship’s  has been constructed  A: Construct vector with log speed, compass  
heading vector  bearing
    O: Ship’s heading vector

104
Heading vector has been added to fix.

Tide vector is added to DR EP on.

2 is complete.

Distance travelled calculated.

Speed made good calculated.

Each target seen in ARL has had TSU.

TSU interval is set if target is under way.

POE applied.

If plan is safe.

4 is complete.

Plan h, to allow for tide: 1, 2, 3, 4 in order.
0.232231M  
Decide on course

Course decided  
I: Discrepancy between fix and plan
A: Decide on course
O: Course to make good

0.232232M  
Find tide effect

Tide found  
I: Tide, course to make good
A: Multiply tide by duration of course
O: Tide effect on course change

0.232233M  
Find course to steer

Course change found  
I: Tide effect, course to make good
A: Add tide effect to course to make good
O: Course to steer

0.232234M  
Change course

Course changed  
I: Course to steer
A: Compose and send instruction
O: *Course change instruction, compass bearing

Table 5.2: Supplementary tasks for manual navigation (end)

The following table shows the analysis of those subtasks which only occur, or are performed differently, in the automatic task (ie when the navigation aid is in use). The columns show the same as in Table 5.1.

Table 5.3: Supplementary tasks for automatic navigation

<table>
<thead>
<tr>
<th>Automatic Methods</th>
<th>Criteria for success</th>
<th>Subtasks/Input-Action-Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.221A (plan i)</td>
<td>2 is complete</td>
<td>1 Read next step message</td>
</tr>
<tr>
<td>Get data for next step</td>
<td></td>
<td>2 Adjust data</td>
</tr>
<tr>
<td>Plan i, to get data for next step: 1 then 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 0.2211A          | Message read         | 1: *Next step message      |
| Read next step message | | A: Read next step message |
| O: Next step plan data | | |

| 0.2212A          | Next step adjusted for reality | 1: Next step plan data, EP |
| Adjust data | | A: Adjust data to account for reality |
| O: Data on next step | | |

| 0.22231A          | Risks found          | 1: Data on next step       |
| Ascertain risks | | A: Extrapolate courses |
| O: Risks | | |

106
Table 5.3: Supplementary tasks for automatic navigation (end)
6. DETAILED DESIGN OF THE EMBEDDED TRAINER

This chapter further describes the design for the embedded navigation trainer. Each of the components from chapter 4 is described in more detail, including its communication with the other components and with the student. The data the component uses, and the actions it carries out are given.

Because the project was curtailed prematurely (see section 1.1), some gaps remain in the design. Where gaps occur, they are noted in this chapter. The work which was planned to complete the design is explained in section 8.1. More detail would need to be added throughout if the design were to be implemented by a programmer with no experience of computer-based training systems and artificial intelligence; providing that extra detail would require time and patience but no additional research.

The description of the components shows how the task model fits into the design. The complete design (apart from the known omissions) is this chapter and chapter 5 (the task description) taken together. Chapter 4 gives an overview of the components and describes the sequence of events during training.

6.1 Model Builder

Input from Tutor Interface (question and answer) and Executor (performance data, context).

Output to Report Generator, Tutor Interface and Executor (model data), and to Plan Builder (model data and task analysis elements).

INPUT

The Model Builder receives reports of the student’s questions and answers from the Tutor Interface. The Executor tells it about the student’s performance relative to the scenario. It also reports the context, ie the scenario or lesson (including teaching method) the student is tackling. The Model Builder compiles this information into a student model, using as its base the task analysis (not shown as an input).
DATA

The model has two parts. The major part is based on the task analysis. The model for each student consists of evidence of whether the student can or cannot do each element of the task. The second part of the model contains any other information about the student which is found to be useful. This information is called the "personality model" for convenience.

Task analysis element

The task analysis contains the following information:

Name                  uniquely identifies the task
Part of               supertask(s)
Has Parts             subtasks
Method                subtask plan, or actions to achieve the task
Succeeds              how to tell if the task is successful
Fails                 how to tell if the task fails
Criteria              performance (eg speed, accuracy) required for proficiency
Inputs                data needed to perform the task
Outputs               data provided by performing the task
Uses skills           other than those in the subtasks
Pre-reqs               tasks which must be learned before this one
Enables               tasks for which this is a pre-requisite
Proof                 evidence which will prove student can do task
Exercise              features of an exercise which will practise task
Required              circumstances when the task must be done
Achieves              why the task is done
Similar to            other similar tasks

Most of these items can be lists.

The method for a task which has been broken down into subtasks is a plan by which the subtasks achieve the task. If the task has not been subdivided, the method is described in terms of objects and actions the Aid recognises (perhaps after translation into Aid terminology).
For example, the task analysis (see section 5.4) contains an element "DR on" (task 0.231232M) in which the navigator predicts positions of ownship in the near future by "dead reckoning".

**Example task analysis element: DR on**

<table>
<thead>
<tr>
<th>Name</th>
<th>DR on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part of</td>
<td>Plot fix</td>
</tr>
<tr>
<td>Has Parts</td>
<td>1 Construct heading vec.</td>
</tr>
<tr>
<td></td>
<td>2 Add heading vec. to fix</td>
</tr>
<tr>
<td></td>
<td>3 Mark DR</td>
</tr>
<tr>
<td>Method</td>
<td>1,2,3 in order.</td>
</tr>
<tr>
<td>Succeeds</td>
<td>$&lt; n%$ error on DR mark</td>
</tr>
<tr>
<td></td>
<td>(could have formula for n)</td>
</tr>
<tr>
<td>Fails</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>(no legal way to fail)</td>
</tr>
<tr>
<td>Inputs</td>
<td>speed, bearing, fix</td>
</tr>
<tr>
<td>Outputs</td>
<td>heading vec., DR, DR mark</td>
</tr>
<tr>
<td>Uses skills</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>(only subtask skills)</td>
</tr>
<tr>
<td>Pre-reqs</td>
<td>Plot position</td>
</tr>
<tr>
<td>Enables</td>
<td>EP on</td>
</tr>
<tr>
<td></td>
<td>Tidal drift</td>
</tr>
<tr>
<td>Proof</td>
<td>student plots DR mark OK</td>
</tr>
<tr>
<td></td>
<td>(no evidence if Aid plots DR)</td>
</tr>
<tr>
<td>Exercise</td>
<td>automatic DR off</td>
</tr>
<tr>
<td></td>
<td>(student must do it)</td>
</tr>
<tr>
<td>Required</td>
<td>Plot position complete</td>
</tr>
<tr>
<td></td>
<td>(need a fix to plot DR)</td>
</tr>
<tr>
<td>Achieves</td>
<td>EP on</td>
</tr>
<tr>
<td></td>
<td>Find tidal drift</td>
</tr>
<tr>
<td></td>
<td>Report fix</td>
</tr>
<tr>
<td>Similar to</td>
<td>EP on</td>
</tr>
</tbody>
</table>

**Student model element**

Each task analysis element has a corresponding element in the student model. It has the same name, and contains the following items:

<table>
<thead>
<tr>
<th>Name</th>
<th>same as task analysis element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evidence</td>
<td>list of the evidence that has been found</td>
</tr>
<tr>
<td>Total value</td>
<td>model builder’s decision about the value of the evidence: does</td>
</tr>
</tbody>
</table>
Difficulty: level reached, i.e. level of competence

Evidence

Each evidence entry has the following parts:
Source: description of the evidence, including context
Time: when the evidence was seen
Value: weight (and direction) of the evidence

Observing and evaluating different kinds of evidence is the main function of the Model Builder.

The definition of the difficulty level is described in the section on the Executor, below.

Personality model

The personality model contains information about a student which is not specific to a task. This information is recorded because it is needed in another part of the Tutor. Items needed include:

a. A record of what he has been taught, and by what methods;
b. The types of task at which he is good (or poor), in terms of the level to which tasks must be broken down to teach to him, the difficulty level he can cope with at first, his learning rate, etc.;
c. How much guidance he needs or wants.

It would be possible to look for evidence of whether a student is using learning strategies, but that would probably be too complex for the first version of the Tutor.

ACTIONS

Positive evidence

Evidence that the student can do a task is of various kinds. Firstly, the Executor may observe the student performing the task in appropriate circumstances. This evidence is fairly strong, except that the student may still not be able to cope with variations on the task. In the example above, the Aid normally calculates DR's, and it is only
when it has been specifically instructed not to (by the Executor) that the student needs to do this task. If the student plots a DR the Executor will report the accuracy of the plot to the Model Builder. If the accuracy meets the specified criterion for success, then the Model Builder has evidence, strong but not overwhelming, that the student can plot DR’s.

The model may show that the student can do another task which is similar, or two similar tasks which together cover the ground of another task, or some other combination. In our example, the student may be able to plot EP’s. This kind of evidence is suggestive rather than certain. The fact that a student has been taught about a particular task is slender evidence that he can do it, but there is much stronger evidence if he successfully performs exercises involving the newly taught task.

The Tutor Interface reports the student’s questions and answers to the Model Builder. It is not possible to glean very much positive evidence when a student asks a question. To some extent any question reveals what he does know, since he presumably asks about things which are at the limit of his understanding, not way beyond it. It would take a very sophisticated analysis to draw this kind of conclusion in general. However, it may prove possible to conclude something cruder when the student asks a question about an area not yet covered. For example, if in practising DRs the student asks “Is this similar to EPs?” then there is weak evidence that he knows about EPs. Whether this will be possible, or useful, remains to be seen.

Negative evidence

Negative evidence is similar. If a student does not perform a task when circumstances dictate he should, this suggests he either cannot do it or does not know when it is required. If a student tries to perform a task and makes an error, then he is not fully competent at the task. This assumes it is possible to tell what the student was trying to do, which will not always be the case. In the DR example, the Executor may report that the student has plotted a DR inaccurately, or in another case that he has not plotted one when he should have. Here it is easy to tell whether the student should have plotted a DR because it is a task normally carried out by the Aid. It may be difficult to tell that the student was trying to plot a DR if he
marks it as an EP, for example.

Incompetence at a similar task suggests that a student cannot do the task, as does incompetence at a pre-requisite skill. If a student asks a question about a task, it is not possible to conclude he does not understand it, as the answer may conceivably have made everything fall into place. However, if the Tutor Interface cannot deal with a question, perhaps because he asked it while using the Aid operationally, or because it was too large a matter to deal with as an aside, then there is evidence that the student does not understand the task. An outstanding issue is how to determine what task a student’s question refers to, which may not be always be obvious (see section 8.1.2).

Compiling the model

The use the Model Builder makes of evidence and the way it compiles it is a matter for further investigation. The Model Builder should err on the pessimistic side in first building the model. Since the evidence for each model element is recorded, it will be possible to judge how reliable the data is. The remedy for incompetence at a task is, initially, more practice at it. This is also the response to gaps in the student model, so there is no penalty (in teaching terms) for assuming the student cannot do a task. To overestimate the reliability of negative evidence would be more serious, as it might persuade the Planner to give a remedial lesson where one is not needed, to which the student might react unfavourably.

Example student model element: DR on

Corresponding to the “DR on” element in the task analysis is a student model element. If the Model Builder sees the student plot a DR mark within the specified error margin (and there is no other evidence for this task), the student model element might look something like this:

<table>
<thead>
<tr>
<th>Name</th>
<th>Evidence</th>
<th>Source</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR on</td>
<td>DR mark</td>
<td>OK</td>
<td>0012</td>
<td>OK, very strong</td>
</tr>
</tbody>
</table>
Total value | OK, strong
---|---
Difficulty | Speed met criterion
           | Accuracy met criterion

The evidence value is "OK" because it seems the student can do the task. The piece of evidence received is "very strong": marking DR accurately is very strong evidence that the student can mark DR. However, overall it is only marked "strong" because only one such piece of evidence has been received. It could be that the student cannot mark DR in all circumstances. (This is an example of the kind of reasoning which will be performed by the Model Builder: how evidence values and leanings will be recorded and combined has not yet been decided.) If further evidence were received there would be two entries in the evidence list. The Total Value of the evidence would change accordingly. The "difficulty" entry shows that the student demonstrated all the speed and accuracy criteria stipulated as necessary to do the task proficiently.

Providing data

The Model Builder must be able to answer all the queries put to it by other parts of the Tutor. It must have procedures for extracting from the model all the information described in Output.

OUTPUT

The primary output from the Model Builder is the student model. This is not output wholesale, however. Other parts of the system requiring data from the student model must ask specific questions of the Model Builder. This saves other parts of the system from having to know the format of the model.

Some of the data the Model Builder must supply are:

a) A student’s strengths, described at the highest possible level.
b) A student’s weaknesses, described at the highest possible level.
c) Gaps in the data on a student, described at the highest possible level.
d) The evidence (with times) for any element in any student’s model.
e) All evidence for elements in a student’s model.
f) The status of any element (ie learnt, not learnt, no evidence).
Subtasks of a higher-level task.

Elements similar to a particular element.

The connection between two elements.

Which elements in a part of the model have a given status (learnt, not learnt, no evidence).

"Personality" information.

Method used to teach an element.

Pre-requisites of an element.

Any tasks learnt during the previous lesson/exercise.

Any new evidence gathered during the previous lesson/exercise.

How to prove the student can do a task.

How to exercise a task, etc.

The Model Builder also supplies task analysis elements to the Plan Builder.

6.2 Plan Builder

Input from Model Builder (model data and task analysis elements).

Output to Executor (plan element).

INPUT

The Plan Builder (or Planner) operates in three situations: it builds a plan for a student when he first starts to use the system; it checks the plan after each element has been executed and re-plans if it considers that the plan is no longer workable because the student model has changed greatly; it builds a new plan when the previous one has been completely executed. It takes data solely from the Model Builder. It may want to know:

Any gaps in the model for a student.

Any weaknesses the student has.

The evidence for any element.

"Personality" information.

Method used to teach an element.

Any tasks not practised recently.

The Model Builder also supplies task analysis elements to the Plan Builder.
DATA

The data used in the Plan Builder are the model data, received from the Model Builder (qv) and the plan itself. The plan contains three different types of element, each of which has the following information:

Next  next element in the plan
Tasks  the tasks taught or practised in this element
Type   lesson/question/exercise

Exercise definition

Lessons and questions are not defined yet. Exercises are defined by the following details:

Difficulty    two difficulty levels: to start with, and to aim for
Chart         which is to be used
Destination   of ownship
Objective     for the mission
Start point   for the exercise
Plan          mission plan - or partial mission plan
Vessels       other vessels in the area, and their behaviour
Ownship       type of ownship
Errors        magnitude of errors in instruments, tides, etc
Hazards       any uncharted hazards
Time          any time constraint on the exercise
Accuracy      any accuracy constraint on the exercise
Feedback      amount and type
Functions     Aid functions to be performed by the student.

Difficulty level

The difficulty of an exercise is affected by such things as:

a Speed required of the student;
b Accuracy required of the student;
c Complexity of the exercise;
d Amount of feedback and help the student gets;
e How many extraneous factors are involved (other than the task the student is to practise).

The complexity and extraneous factors are specified by the Planner as part of the exercise, and cannot be adjusted by the Executor to affect the difficulty. Feedback from an exercise is still a problem (see section 8.1.3). Therefore the only factors included in the difficulty measure at the moment are speed and accuracy (although there is no reason why other factors should not be added). The difficulty measure could specify these constraints in several ways. The simplest is for it to contain two separate figures, one for each. For example, speed might be a percentage of real time, and accuracy a percentage of the value. The exercise definition contains two difficulty measures: one to start from, and one to aim for. Difficulty is therefore defined something like this:

<table>
<thead>
<tr>
<th>Start</th>
<th>Speed</th>
<th>speed constraint to start with</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>accuracy constraint to start with</td>
</tr>
<tr>
<td>Finish</td>
<td>Speed</td>
<td>speed to aim for</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>accuracy to aim for</td>
</tr>
</tbody>
</table>

**ACTION**

**Building a New Plan**

The first action of the Plan Builder is to consider what is known about the student. If this is a new student, his model probably contains very little information. Even when a student has used the system before there are likely to be gaps in the model. The Planner must first ensure that it knows enough about the student to make a plan; if not it must first identify any important gaps in its knowledge and put those tasks on a list of tasks to be practised. Only early parts of the plan need to be completed in full detail at first, so some gaps in the student model can be tolerated. If there is sufficient information to make a plan, the Plan Builder will compile one using the evidence available to it.

**Using the evidence**

Where the model contains information regarding a task, the evidence may suggest one of three things: the student cannot do the task adequately; he can do the task,
but he has not practised it for some time; or he can do the task and has practised it recently. In the last case no action is necessary. Any out-of-date tasks must be put on the list of things to practise. Where there is evidence that a student cannot do a task the problem is more tricky. The Planner must decide how good the evidence is. If the evidence is very strong it may decide that remedial teaching is called for. If the evidence is less strong it needs to find out more by calling for the task to be practised. The threshold for this decision should be set by trial and error once the system is operational. In the extreme case where there is a great deal of weak evidence that a student cannot perform a task, and no very good way of getting more conclusive evidence, the Planner may call for him to be asked a specific question. It may be either a test question, or else a straightforward choice of whether he would like the task to be re-taught. The use of questions in the plan needs to be considered further. (During execution of the plan, additional information of this sort is incorporated when the Executor checks whether a plan element is still necessary before executing it. If it proved necessary, the plan could have branches.)

Abandoning training

If many unrelated tasks need remedial teaching, the Planner should consider giving up on the student. Either the student is genuinely inept, and needs substantial re-teaching which the trainer is not designed to give, or there is something about the student with which the trainer cannot cope. For example, it is conceivable that a student who has been taught using a completely different approach to that assumed by the trainer may be quite competent at navigation but operate in such a different way that the trainer cannot recognise what he is doing. Hopefully this risk is slight because the student model is based on observable features of the task rather than some inferred model of the way the students will think, but there will always be a student with which the trainer cannot deal. The trainer should be prepared to recognise its own limitations.

Teaching a task

If a task needs to be taught to the student, a lesson must be put in the plan. Although flexibility would be increased if lessons were constructed by the Planner, they were not seen as a major concern and so "canned" lessons were planned to
allow effort to concentrate on exercise generation. In fact early versions of the Tutor may not teach at all, but merely report the need for teaching to the instructor.

Assuming the Planner is to include lessons, it must choose how much to break the tasks down to teach them. Sometimes the tasks in the task analysis will be too detailed for a student. For example, some students will be able to construct the ship's heading vector when instructed, even though they have never done it before, whereas others will have to have it explained to them. The rules used to do this are not obvious. Possible rules are:

a. Always use the maximum level of detail available in the task analysis. The task analysis can be altered if students seem to find it too hard or too easy.

b. In the "personality" model, store a measure of the amount of detail the student seems to need. Being explicit, the value can be changed according to the student's rate of progress.

c. Store a measure of detail for each task. If the student does well at a task, reduce the level of detail for all similar tasks.

More complicated methods can be devised. The one(s) actually used must be chosen, either by an experiment, or by assuming that the simplest rule is adequate until proven otherwise. To implement any of these rules (including the first), a measure of "ease" is necessary, which can record how easy the student is finding the task. This would be based on such factors as speed, number of errors, questions asked by the student, etc. If the measure is wrong, or is not used correctly, there is a risk of the level being set too high, so that students have difficulty learning a task, or too low, so that they are not fully stretched.

Tasks to be taught must be added to the list of tasks to be practised.

Practising a task

At this stage the Planner has a list of tasks to be practised, and a list (hopefully short) of tasks to be taught. In many cases it will be possible to practise several tasks simultaneously by the judicious choice of an exercise. This is appropriate if an exercise is to gain information, or check that a student has not forgotten a task he once knew. If a task has just been taught, though, it will be better to first practise it in isolation before combining it with other tasks. The task analysis describes the
features of an exercise which will practise each task. Where these features do not conflict, the Planner can group several tasks together for practice. Each task or group of tasks is allocated a minimally-defined exercise containing just those features needed to practise them. The Planner does not specify any more details than it needs to ensure the task(s) will be practised.

The difficulty levels must be set. Obvious levels to aim for are the criteria set in the task analysis. The levels to start from are defined to save time in practising a task: the exact method used is not critical. A sensible method is to establish the levels of speed and accuracy the student can usually achieve on the first practice. They could also be defined for individual tasks in the same way as the task breakdown level, if this was found to be helpful. A default level will need to be used until the student's own difficulty level has been set.

The Executor has freedom to alter the parameters of an exercise, particularly those that govern the difficulty, within the limits specified by the Planner. One thing the Planner must always specify exactly, however, is which tasks are to be included in the exercises. The student cannot be said to be competent at a task unless he can do it in combination with other tasks. However, the Executor cannot add extraneous tasks to an exercise without considering how well the student can do them. This is not its job, and may interfere with the plan. Therefore the plan must always include combined practice of tasks the student can do separately to make sure he can do them together.

Ordering the plan elements

The two lists need to be combined and ordered. Some orderings are obvious: a task should be practised after it is taught. Other orderings will be preferential. Suppose task A is a pre-requisite of task B, and both are to be practised. If task A (the pre-requisite) is practised first, and it fails, then task A should be sorted out before task B is practised. On the other hand, if task B is practised first, and it succeeds, then there is no need to practise its pre-requisite, which the student can be assumed to know. The choice between these orderings will depend on the evidence for and against each task. The Planner will need a series of rules (similar to an expert system) to enable it to decide which orderings to apply when. Possible rules are:
a Gather data before teaching or practising anything.
b Gather coarse data before fine.
c Practise a task after teaching it.
d Do similar tasks together.
e Do related tasks together.
f Do tasks using the same skills together.
g Do similar tasks apart.
h Do related tasks apart.
i Do subtasks of a task before the task itself (bottom-up).

A choice needs to be made between these rules, as some are contradictory. There is no need for the choice to be final: the rules could be qualified, so that different ones apply in different circumstances.

Having made both obvious and preferential orderings of the tasks to be taught and practised, the remaining ordering decisions (if any) can be made arbitrarily.

Filling in plan details

The plan is now nearly complete. It consists of an ordered list of groups of questions to be asked, tasks to be practised and lessons to be given. The final stage is to fill in any remaining details for the plan element which is about to be executed. The sort of details which may need to be added are the choice of teaching methods for a task, if more than one is available. If the student only needs a recap, a short summary will be adequate. There may be information in the "personality" section of the student model which will help the Planner choose between methods. For example, he may previously have shown he can learn quickly.

Revising an Old Plan

The Planner is called on to revise the plan at the beginning of a training session and after each plan element has been executed. At the start of the session it checks the time which has elapsed since the last training session. An old plan must be discarded because the student model may be out-of-date. The student may have practised
navigation elsewhere, or have forgotten some aspect. Since the Planner considers the age of evidence, calling for a new plan automatically takes this into account.

After each plan element has been executed, the Planner is invoked to check the plan to make sure it is still workable. Several things may have happened. Hopefully, the student model now shows that the student has learned the object of the previous plan element. The evidence may also now show that he knows some other task, either because he has learned or remembered it along the way, or because the Model Builder has gathered more evidence about it. If this is the case, the Planner may be able to prune one or more elements from the plan.

Another possibility is that the evidence is growing stronger that the student cannot do a task. This may reflect two problems: either the student never could do the task, and the evidence was wrong, or he could do it once but has forgotten it or become confused. The Planner must consider how strong the evidence was to distinguish these two cases. If the evidence was initially weak, the Planner may assume that it was wrong. Following the maxim that weak evidence is not used as the basis for important decisions, it may be that no harm has been done.

If the initial evidence was strong, the Planner must assume that it was correct: the student can no longer do the task. This is a much more serious problem. In either case, a new plan will have to be drawn up. (This also applies if the task in question is the object of the previous plan element.) The matter must be recorded so that the system supervisor becomes aware of it.

Replanning

If a previous plan has been abandoned by the Executor, some special considerations apply. It may be possible to use the old plan as the basis for the new one, to save planning time. Until this proves to be necessary, the old plan will be discarded, for simplicity.

A task may have previously been taught unsuccessfully (this is recorded in the student model). The Planner needs rules to tell it what to do. Possible rules are:

a. Reteach the task using the same teaching method.

b. Reteach the task using a different teaching method (if there is one).
c Reduce the task breakdown level (see Teaching a task, above).
d Don’t reteach the task.

To start with, the Planner should use rule a in all cases, to avoid complicating the system unduly. Later, if necessary, other rules can be brought in. The circumstances for each rule to apply will then need to be established.

The plan may have been abandoned because a task which the model once showed as “learnt” has now changed to “not learnt” (see Executor). If the evidence for the task was initially weak, the Planner will assume the student never knew the task after all. It will simply produce a new plan on the basis that the student cannot do the task. It is possible, though, that the student could do the task once and has forgotten it or become confused. The lack of strong evidence does not make this impossible. The new plan will treat the task as any other failed task and re-teach or re-practise it. If the student really is deeply confused this will be discovered next time round when more evidence will be available.

Conversely, if the evidence that the student could do the task was initially strong, the Planner will assume that the student is confused, even if this is not the case. Here the risk is that the Planner may overreact to the crisis, re-teaching the task when in fact all the student needs is a little more practice. This is unsatisfactory: the solution is for the Model Builder to be conservative (and accurate) and for the thresholds used by the Planner to be set very carefully.

OUTPUT

When complete, the plan element is passed to the Executor.

6.3 Executor

Input from Model Builder (model data), Tutor Interface (answer), Plan Builder (plan element), Classifier (scenario) and Aid (action, results, scenario details).

Output to Model Builder (performance data, context), Tutor Interface (context, question, lesson, feedback), Aid (instruction) and Scenario Generator (instructions).
INPUT

The Executor receives the plan element from the Plan Builder. A plan element (see Plan Builder) is a question, lesson or exercise.

When the Aid is being used operationally, the Executor receives scenario data from the Classifier instead of generating it itself. This enables it to continue to report performance data to the Model Builder at all times.

Throughout its operation, the Executor continually queries the Model Builder for the status of the operation being taught, to see if the plan element is succeeding.

The Executor receives answers to its questions from the Tutor Interface. These have been answered by the student, interpreted and sent to the Executor.

It receives reports on the student's actions from the Aid, as they occur. On the occasions when the Executor has instructed the Aid not to show the results of a certain task to the student, they are sent to the Executor instead. As the scenario unfolds, the Aid sends an outline of it to the Executor. This allows the Executor to keep track of what is going on. In calculating feedback the Executor may need to know further details, such as the bearing of a hazard. It gets these from the Aid by specifically asking for them.

DATA

Most of the data used in the Executor has been sent to it from other places. The plan element is the main data item: this is sent from the Plan Builder, and is described in that section. The only type of plan element which is altered to any extent by the Executor is an exercise. This is transformed into two things: instructions for the Scenario Generator and context details for the Tutor Interface and Model Builder. The types of instruction the Scenario Generator understands are described in the DCRS documentation. They fall out of the exercise definition quite naturally. The context details required by the Tutor Interface and Model Builder are the "essential" parts of the exercise, which are:

- **Tasks**: the tasks taught or practised in this element
- **Difficulty**: difficulty level of this exercise
- **Objective**: for the mission
Plan (whether there was one)
Vessels other vessels in the area, and their behaviour
Ownership type of ownership
Errors magnitude of errors in instruments, tides, etc
Hazards any uncharted hazards
Time any time constraint on the exercise
Accuracy any accuracy constraint on the exercise
Feedback amount and type
Functions Aid functions to be performed by the student.

In order to complete each exercise, the Executor contains a selection of general exercises which it can use, and definitions of constraints, such as the types of vessels which are available. This enables it to construct several different exercises for each plan element if necessary. The general exercises are defined in terms of:

a chart;
a destination;
a mission plan or partial mission plan with which to start;
other vessels, with their behaviour already defined;
uncharted hazards;
a text description of the exercise, which is shown to the student;
the tasks specifically exercised.

The other vessels and uncharted hazards may or may not appear in the final exercise, according to the specifications in the plan.

**ACTIONS**

The Executor executes the plan element it has been given. Questions are passed to the Tutor Interface for phrasing. Lessons are fully defined and the Executor need only carry them out, passing lesson material to the Tutor Interface which will forward it to the student. Exercises are minimally defined in the plan: the Planner describes the exercises only just sufficiently to ensure they will practise the required tasks. This enables the Executor to generate as many exercises as are necessary for the student to achieve mastery. It observes the student's responses, compares them with the current lesson or exercise, and sends details of the student's behaviour to
the Model Builder.

When the Aid is being used operationally the Executor cannot attempt to teach, but it still observes the student's behaviour, compares it with the scenario, as described by the Classifier, and passes the information on to the Model Builder.

Lessons

The lessons given by the Tutor are simple expositions. They may include text, diagrams, examples, questions for the student to answer, etc. Questions are dealt with by the Tutor Interface. The Executor ceases trying to teach while the Tutor Interface operates. The Executor reports to the Model Builder on the lesson it is giving, and the teaching method used (as specified in the plan). It reports enough of the lesson content to the Tutor Interface to enable it to understand the student's questions; it remains to be established what is required for this (see section 8.1.1). Again, this information is provided in the plan.

When a lesson includes a question, the answer ought to affect the way the lesson continues from there on. The method for this has not been worked out.

Exercises

The Executor sets exercises by sending instructions to the Scenario Generator. Each practice element in the plan is an incompletely defined exercise. The Executor must finish defining the exercise by choosing values for the undefined terms. It can do this several times in different ways, so providing the student with several exercises on the same theme if necessary. The exercise definition provided by the plan will be in terms of the tasks the student must practise. It may call, for example, for the student to be given an exercise where the tidal stream varies greatly from the predicted tide. The Executor can select any exercise at all so long as it instructs the Scenario Generator to allow a large variability in the tide.

Two factors affect the way the Executor generates exercises. Firstly, it needs to have a large number to choose from so that a student can use the system for some time without doing the same exercise twice. Secondly it must only give the student exercises which are not only possible to do, but plausible. To satisfy these constraints, the Executor has a library of exercises to choose from, in which some
parameters are defined in terms of limits, rather than a single value. The Executor can choose a value between the defined limits. In this way a large number of different, plausible exercises can be generated by the Executor. Not all exercises will be suitable to practise every task. The library will include information about the kinds of task an exercise is suitable for.

Exercise difficulty

Sometimes the parameter values chosen will affect the difficulty of the exercise. The plan specifies the levels of difficulty to start at and to aim for. The Executor should not add extraneous factors to the exercise: these will always be specified in the plan (see Plan Builder). The student’s ability at the task, and his rate of progress, are determined from the student model between each exercise. Since the difficulty is specified for each factor separately, the Executor can increase the difficulty of each factor as the student improves at it. For example, as the student’s speed improves, the Executor can increase the rate at which fix information arrives until the student is working in real time.

Translation for the Scenario Generator

Having filled in the details of the exercise, the Executor then proceeds to translate the exercise into the terms used by the Scenario Generator. These are those an instructor would use if he were controlling the Scenario Generator himself (see DCRS documentation). They include such factors as:

a  The number and behaviour of other vessels;
b  The information which is available (due to weather conditions, etc);
c  Error variability of equipment, tide, etc;
d  Equipment failure;
e  Uncharted hazards.

The Scenario Generator must be told the initial position and any events, with times.

Instructions to the Aid

In order to give the student practice at tasks the Aid normally does, the Executor occasionally sends instructions to the Aid telling it to let the student carry out a
particular task for himself. For example, to allow the student to practise calculating DRs, the Executor would instruct the Aid not to show DRs on the chart. The DRs input by the student are relayed to the Executor along with all his other actions.

Feedback

The Executor compares the report of the student’s actions with the scenario. It tells the Model Builder what tasks the student has been seen to do, how well he did them, and under what circumstances. It reports on tasks the student should have done but did not. The Executor does not draw its own conclusions about whether the student has learned the task: it always relies on the model, which has all the evidence.

The Executor also provides the student with feedback on the exercises. Ideally plenty of feedback should be supplied at first. As the student improves, the feedback is reduced until finally only task-intrinsic feedback remains. (This is the feedback the student will get when he performs the task for real.) The difficulty is that a comprehensive method of assessment, on which feedback could be based, is not available for navigation. Certain overall measurements can be made, such as the amount of periscope exposure time, the closest approach to a hazard, and the amount of deviation from the mission plan (measured in some way). Although these measurements can be reported to the student at any time they do not constitute immediate feedback. Further work is needed.

If suitable feedback measures can be found, then it would be possible to have thresholds in the student model to determine how much feedback the student receives. The thresholds and the levels of feedback would require consideration. In any case, feedback is one of the factors governing the difficulty of an exercise: the way difficulty levels are dealt with overall may affect the way feedback is treated.

When the Executor has instructed the Aid to let the student perform one of its calculations, the Aid sends back the value it calculated instead of showing it to the student. The Executor therefore knows when the student should have calculated a value and what it should have been. It can give the student (and the Model Builder) much more concrete feedback, including the percentage error in his calculation.
If the Aid is being used operationally the Executor receives scenario data from the Classifier rather than generating it itself. In this case it does not provide feedback: in fact, its only function is to provide the Model Builder with performance data on the student.

Abandoning a plan element

In order to set the difficulty level for the next exercise, the Executor checks the student model to see how the student is progressing. A problem can occur if the student appears to be getting worse at the task, or at least not getting better. Eventually there will come a time when the student model shows very strong evidence, compiled over several exercises, that he student cannot do the task. The plan element is deemed to have failed and is abandoned.

OUTPUT

The principal outputs of the Executor are lessons, as described above, which are sent to the student via the Tutor Interface, and instructions to the Scenario Generator. It reports the current scenario or lesson to the Tutor Interface and the Model Builder. Any questions it has for the student it sends to the Tutor Interface. It tells the Model Builder how the student’s actions compare with the scenario he was presented.

6.4 Report Generator

Input from Model Builder (model data) and external (instructions).

Output to external (report).

INPUT

The Report Generator produces reports on students according to external instructions (from the instructor, presumably). There should be a grammar for specifying what is to be reported, and a facility for setting up standard reports (and possibly also some standard reports set up in advance). The interface does not have to be immediately obvious, as instructors will presumably have time to become familiar with the system, but it should not be too difficult to use.

The Report Generator is liable to request the following data from the Model
Builder:

a. A student's strengths, described at the highest possible level.
b. A student's weaknesses, described at the highest possible level.
c. Gaps in the data on a student, described at the highest possible level.
d. The data source (and date) for any element in any student's model.
e. All data sources for a student.
f. The status of any element.
g. Subtasks of a higher-level task.

**ACTIONS AND OUTPUT**

From this the Report Generator can compile various data:
- Strengths and weaknesses of a student;
- Whether a student is competent (ie OK at all major tasks);
- A student's rate of progress (ie the number of tasks learned per session);
- Group statistics (eg 20% of students can't do task A, average rate of progress);
- Reliability of a data item (ie the evidence for it);
- Success rate (% of students who have improved, % who have achieved competence, etc.)

**6.5 Tutor Interface**

Input from Model Builder (model data), Executor (context, question, lesson, feedback) and external (question and answer).

Output to Model Builder (question and answer), Executor (answer) and external (question and answer, lesson, feedback).

**INPUT**

The Tutor Interface is the means of direct communication between the student and the Tutor. The interface with the Tutor and the interface with the Aid are kept separate to remind the student which functions belong to which system. Inputs from the student may be questions, or answers to previous questions the Tutor Interface has put to the student. The Executor also sends the Tutor Interface any questions it wants to ask the student. The context, ie the exercise or lesson the student is
currently doing, is reported automatically by the Executor. From the Model Builder the Tutor Interface requests data on the student to help it decide what his query is, and to answer the question.

Data the Tutor Interface may want from the Model Builder include:

a. The status of an element in the student model.
b. Elements similar to a particular element.
c. The connection between two elements.
d. Which elements in a part of the model have a given status (learnt, not learnt, no evidence).
e. The evidence for any element in a student's model.

ACTION

The function of the Tutor Interface is in principle straightforward - to handle communication between the Tutor and the student - and in practice complex. The student's questions are in a restricted format (yet to be defined), but even so they need to be interpreted in context. A mechanism which is likely to be useful is hypertext. With hypertext, certain words or graphical items are visually labelled in some way (eg by their colour) as hypertext. The user can select these items to further information about them. For example, a student could select a hypertext word which he does not understand in order to receive a definition or some teaching material. This is a valuable, simple device which enables the student to ask questions.

Answering the student's questions

It is in the realm of answering the student's questions that most work remains to be done. Section 8.1.1 describes the investigations which would have been carried out if circumstances had permitted.

The answer which the Tutor Interface must give to a student's question depends on several things. The most important factor is whether the Aid is being used for training or operationally. If it is in operational use the role of the Tutor Interface is clear-cut: it must answer the question as concisely, clearly and rapidly as possible. To do this, it may need to know about the current situation, the task analysis, or a
combination of both. It may be necessary to have the answers to certain standard questions, or question types, ready and waiting. (This assumes that standard questions can be identified.) If the question is graphical, the answer should be graphical if at all possible.

The Tutor Interface may decide not to answer a question at all; the basis for this decision remains to be found. The answer may be too large a matter to deal with as an aside, particularly if the aid is being used operationally. If it is at all possible, the Tutor Interface should recognise whether it knows the answer to a question and not try to reply if it does not.

If the Tutor Interface does decide to answer a question directly it should take care to give the answer in terms of things it can assume the student knows.

Training responses

If the Aid is being used for training, the Tutor Interface has more time to play with. Firstly it must consider whether the question the student asked makes any sense in the context of the current exercise, and whether it is the question the student really wants the answer to. A student may not know exactly what question it is he wants to ask. By comparing the student's apparent query with the student model to find out what he is likely to want to know, the Tutor Interface can hope to be more helpful in answering the student's question. If there is any doubt about what the student's query really is, the Tutor Interface can itself ask a question of the student. This must be carefully phrased to clear up the doubt. The response, when it arrives, is then used to refine the original question into a more precise query.

Indirect responses

During training, the student's question need not be answered directly. Human teachers sometimes respond to a question with a hint or example which causes the student to work out the answer for himself. Further research might show that the student model can provide enough information to allow the Tutor Interface to do this.
Asking questions

The Tutor Interface asks questions of the student when it needs to have more information about his query. It also asks them when the Executor wants to query the student. It may do so either as part of a lesson or to improve the information in the student model. Asking such questions is a matter of translating the executor’s query into an intelligible question for the student to answer. The answers to such questions are translated back into the Tutor’s internal format and forwarded to the Executor.

OUTPUT

The Tutor Interface passes its questions and answers back to the Aid, which forwards them to the student. (The Executor ceases trying to teach while the Tutor Interface operates.) It reports the answers to the executor’s questions to the Executor. It reports the significant points of the exchange to the Model Builder as it may be possible to use them as evidence in the model.

6.6 Scenario Generator

Input from Executor (instructions) and Aid (commands).

Output to external (scenario data).

INPUT

The inputs to the Scenario Generator are of two kinds: firstly, the Executor mimics the inputs an instructor would make to control the scenario, controlling targets, tides, etc; and secondly, the student makes command inputs to the Aid which are forwarded to the Scenario Generator so it knows when he has issued commands to ownership.

ACTIONS

The actions of the Scenario Generator are described in the DCRS functional specification and design report. The Scenario Generator simulates all the sensor data which would be obtained if the Aid were in operational use. The Scenario Generator is only used during training.
OUTPUT

The output from the Scenario Generator is not fed via the Aid, but is displayed on a separate screen (or window: see DCRS documentation). This is to allow it to simulate the environment data more closely, as sensor data does not come from the Aid.

6.7 Classifier

Input from Aid (environment data).
Output to Executor (scenario).

INPUT

The role of the Classifier is to classify the situation with which the student is being presented when he uses the Aid operationally rather than for training. It takes environment data from the Aid. When this is all fed in by the user then he may introduce errors, but if the environment data is fed directly into the Aid, as is eventually intended, the Classifier sees all the information the student sees. The Aid receives environment data in the form of fix information, and other sensor data.

ACTIONS AND OUTPUT

From the environment data the Aid calculates the most likely position of ownship and other vessels, with headings and speeds, and tidal flow, etc. This is the information which is passed on to the Classifier. The Classifier compares this description with the kinds of situation the navigator is called on to deal with and decides which one it is. (The method of classifying navigation situations has not been established - see section 8.1.4.) For example, the relative positions and headings of ownship and another vessel may indicate a near-miss situation. The Classifier will pass this conclusion to the Executor so that it knows what the student is having to deal with when it interprets his actions.

Difficult classifications

Some kinds of situation will be fairly clear-cut, but many more will be quite difficult to classify. It should be possible, though, for the Classifier to spot when a
situation has aspects of certain classifications, and in particular when circumstances are such that the navigator should take action to avoid getting into a situation. It will be much more difficult to spot when a navigator did take action and so avoided getting into tricky situation before it even became apparent. Such a loss reduces the information in the student model, but it does not mean the need for teaching or practice will be missed.

Classification system

A classification of different kinds of situation does not exist yet. It should be possible to derive it from the task analysis, which is the basis of the student model. In this way the conclusions drawn by the Classifier may be useful to the Model Builder. The information passed to the Executor will show how the situation has been classified, either as a particular type of situation, as a combination of two or more types, or not at all (ie the Classifier has failed).

Training mode

The Classifier has no function when the Aid is being used for training, as the Executor knows exactly what situation the student has to deal with.

6.8 Aid

Input from external (action (data, instruction, request for data), command, q/a) and Executor (instruction).

Output to external (calculation results, data, feedback from instructions), Executor (action, results, scenario details), Scenario Generator (commands), and Classifier (environment data).

INPUT

The Aid receives the usual operational inputs from the student and (if environment data is input automatically) from the environment. These are either data, instructions to the Aid, requests for data, or navigation commands. It may also receive instructions from the Executor to tell it not to perform one or other of its usual functions, so that the student can practise doing them himself.
The Aid processes the operational inputs in the normal way, absorbing the data, doing the calculations and carrying out the instructions. The results are shown on the display. Navigation commands (eg change of speed) are relayed to the Scenario Generator so it knows what ownership is doing. The Aid also echoes some of these inputs to the Tutor: actions and commands are always forwarded to the Executor; environment data is forwarded to the Classifier if the Aid is being used operationally. The operational functions of the Aid are described in detail in the DCRS functional specification and design report.

So that the student can cope if the Aid ever breaks down, he is also given practice at the tasks the Aid normally performs. To do this, the Executor sends instructions to the Aid telling it not to do the relevant task. Although this is an input from the Tutor, it affects the "system" functioning of the Aid. A message relayed from the Executor will tell the student that he is expected to perform the task himself. He performs the task using the normal facilities for overriding the Aid. His actions are forwarded to the Executor as usual. In fact, the Aid carries out the task in the normal way, but instead of passing the results on to the student it sends them to the Executor. This enables the Executor to compare the student's results with those of the Aid to see how accurate he is.

For example, in order to give the student practice in calculating DR's, the Executor instructs the Aid not to show DR's on the chart in the normal way. The student must then calculate his own DR's and mark them on the chart. The Aid actually calculates the DR's as well, but does not mark them on the chart. DRs from the Aid and the student are forwarded to the Executor whenever they occur.

Other inputs from the student are forwarded to the Tutor Interface. Similarly, questions and answers from the Tutor Interface, and lessons and feedback from the Executor are forwarded to the student.
7. The ESA-QUEST TUTOR: A DEMONSTRATION OF THE APPROACH

The ESA-Quest (or Quest) Tutor was built as a demonstrator for the European Space Agency (ESA). It was built after the work on the navigation trainer described in previous chapters. Having had the experience of designing the navigation trainer, it seemed natural to use the same design approach, and to re-use as much of the design structure as would comfortably suit the new system. It turned out to be a straightforward task to adapt the design method and structure for the Quest Tutor.

The Quest Tutor demonstrates some possible methods of teaching information retrieval, particularly to casual users requiring continuation training. It is not an embedded trainer; the purpose of including it here is to give an indication of the flexibility of both the design approach and the structure used for the navigation trainer. Information retrieval is a completely different kind of task from navigation. The only major similarities between the two systems are that they are both intended to be used without supervision, and they are both primarily continuation trainers.

The Quest Tutor was built by a team of three, as one of a set of four demonstrators. Detailed design and coding of the components was carried out by team members to my overall, "architectural" design, described here.

7.1 Design Approach

The Quest Tutor was built using a rather cut-down version of the design approach described in chapter 3. There were two reasons why the full design approach was unsuitable. Firstly, ESA wanted a demonstration system to show them some of the features which could be put into a tutorial system, to help them formulate their objectives for such a system. This meant the objectives for the demonstrator were extremely vague and there was no formal acceptance procedure. Secondly, the system was intended for the use of any person with a licence to use Quest. There was therefore no selection process - more accurately, students were self-selecting - and the candidate population was so diverse that only the vaguest description of them was possible.
7.1.1 Objectives

ESA-Quest is an on-line information retrieval service operated by ESA. Those using the service access it via a computer link and can obtain information on publications, etc, from any of several databases. The charge for the service used to be high, and in the past searching was almost always done by information specialists. Charges have fallen over time, and increasingly researchers and others are doing their own searches as casual users. Local branches affiliated to ESA run courses and provide training material, and various manuals are available. However, there is a need for flexible continuation training for non-specialists who use Quest only occasionally. The Quest Tutor was required to provide an adaptive tutorial system available to users on-line.

The tutor was intended to teach users about information retrieval in general, and ESA-Quest in particular. It was intended to be used without supervision by users with a wide range of experience. It was especially intended to be a continuation trainer for use by non-information-specialists who use Quest only occasionally and may need reminding of aspects they have forgotten.

7.1.2 Task Description

The Quest Tutorial uses a Task Model which was generated by Hierarchical Task Analysis of on-line searching and Quest. By reading about the subject, and from interviews with staff at IRS Dialtech, the separate tasks and topics and the dependencies between them were identified. This is represented explicitly in the demonstrator as a task model. It can be seen in tree diagrams presented to the student for him to select a topic to be taught. The tree diagrams are created directly from the task analysis, although not every topic is shown. The task analysis determines the order in which topics are presented, when the student does not specify this.

7.1.3 Candidate Description

It was not possible to compile a detailed description of the candidates, as they constitute anyone who may choose to do their own on-line search using ESA-Quest. It was expected that they would be mostly casual users, who would be subject
matter experts rather than information specialists. A number of them would presumably be academics. Many would have little or no experience of computers. Although other training material is available for complete beginners, some might try to use the Quest Tutor; at the very least they should be directed to more suitable material.

7.1.4 Training Material

The training material is the Quest Tutor, which is described in the sections which follow. It was based around the task model for information retrieval, as was the navigation trainer.

7.2 Overview of the Demonstrator

The tutor was built as part of a study into the use of expert systems for information retrieval. It has been demonstrated to students at IRS Dialtech, which is ESA's outlet for Quest in the UK, and to ESA staff in Italy. It was well-received in both cases, and important suggestions were made for its further development. A full operational training system for Quest based on the demonstrator is now being developed for ESA.

7.2.1 Methods of using the Quest Tutorial

The Quest Tutorial can be used in three ways. A student can choose:

- General tuition on on-line searching and Quest;
- A lesson on a subject of his choice;
- Advice on a particular type of search problem.

If he chooses the first option, the Tutorial system selects a topic to teach him; if he chooses the second, he chooses his own topic from a tree diagram representing the subject. If he chooses the third option, the system presents a list of common difficulties which researchers encounter, and suggests remedies for them. The remedies are associated with topics in the Tutorial and he will be given a lesson on a remedy if he asks for it. In effect this allows the student to choose a topic without knowing what it is called.
7.2.2 Lessons and Exercises

Each implemented topic has one or more lessons associated with it. In contrast to the navigation trainer, lessons are of greater prominence than exercises. The lessons are of different types and the system has rules which govern when the different types of lessons are taught. For example, some lessons are taught only if the student already knows some specified other topic. This means that different lessons can be given depending on the student’s previous history and state of knowledge. Several lessons are often given in teaching a topic. A common pattern is to give an introductory lesson, teach one or more subtopics, and then give a summary lesson.

The main element of most lessons is a piece of text. Many lessons have one or more exercises for the student to perform. The system assesses the student’s response and tells him if he is correct or, if not, what his error is. Some exercises refer to the student’s own search query (if he has one) and in this case the system will usually not be able to say whether the answer is correct; the student must decide for himself. The exercises in the demonstrator are few and provide limited feedback to the student.

Lesson material was adapted from the following sources:

- Quest User Manual, 1988;
- ESA-Quest Mini Manual, 1987;
- IRS Dialtech Beginner’s Notes (IRS Dialtech is the UK outlet for ESA-Quest);

7.2.3 Hypertext

The Quest Tutorial system is written in *KnowledgePro*, which is a commercially available hypertext tool and expert system language. This gives the student an alternative method of moving round the system, and allows him, in effect, to ask questions. Throughout the Tutorial, various words are highlighted. Such words are hypertext, and have code associated with them. If the student selects such a hypertext word some action will take place, usually the display of further text explaining the word. That text itself may have hypertext in it, and so forth. Hypertext is particularly used in the Tutorial system for words from previous
lessons which the student may need to recap, or just for terms which he has been assumed to know. In many cases selecting a hypertext word results in both an explanation of the word and an option to be given a lesson on it. This gives the student additional control over the order of presentation of lessons. In effect, it also gives him a restricted means of asking questions.

7.3 Structure of the System

The system has the following components:

Interface: handles communication with the student;
Sequencer: selects the next lesson to teach;
Tutor: teaches each lesson;
Student Model: builds a model of the student by observing his behaviour.

The structural relation between the parts is shown in Figure 7.1.

7.3.1 Interface

The Interface handles all communication with the student. Generally speaking, the Interface is controlled by the Tutor, but the student is always able to interrupt a lesson, in which case the Sequencer will take control.

7.3.2 Sequencer

The Sequencer is responsible for selecting the next topic or lesson. If the student asks for general tuition on on-line searching and Quest, the systems will choose the
order in which to present topics to him. Before teaching any particular topic the system makes sure that the student knows all other topics which are required to learn it. Many topics are divided into subtopics which must be known before the main topic can be learnt. The system will give an introductory lesson, if it has one, before teaching the subtopics. When it has taught as many of the subtopics as are required, it will resume teaching the main topic. This process is recursive, which means that during the teaching of a topic, some or all of its subtopics at every level may also be taught.

If the student selects a topic from the task model diagram, the system will teach that topic in the same way, so that it will teach any pre-requisite topics first, and all the subtopics along with it.

The system does not attempt to teach any topic which it believes the student knows (see section 7.3.4 below). (However, it will teach any topic if the student particularly asks for it.) If the Student Model does not show whether the student knows a topic or not, the system will ask the student whether he knows it before teaching him. Since the system starts with no information about the student, the result of this strategy is that it starts the first session by asking the student a series of questions (usually no more than four or five) to find out how much he knows before giving him his first lesson.

7.3.3 Tutor

The Tutor is responsible for delivering the selected lesson to the student. The Tutor does the bulk of the work of the system. It schedules lessons and exercises, and gives the student feedback on his answers.

At the beginning of a session, the system asks the student if he has a particular search query in mind. If he does, it may later give him exercises based on his query. It records his answers, and gradually builds up a record of his query which it can use for further exercises or for any other purpose. This information is not recorded from one session to another.

7.3.4 Student Model

When the student starts to use the Quest Tutorial he is asked to type a name to
identify himself. This is so that information about the student can be stored from one session to another. The student’s permission is sought before information is stored.

Most of the information kept about the student is a reflection of the task model. The system records which topics it believes the student knows. It infers that the student knows a topic if he says he knows it, if he has been taught it, or if he knows another topic which requires it. It infers the student does not know a topic if he says he doesn’t, if he asks for a lesson on it, or if he does not know another topic which is required by it. If the system has no information regarding a topic, it may ask the student whether he knows that topic or not, if it needs to know.

The Student Model can be revised at any time. For example, if a student is taught a topic, it will be recorded in the Student Model that he now knows that topic. If later he asks for another lesson on the topic, the Student Model will be amended to show that he does not know the topic after all. It would be possible to teach the student a different lesson in this case, but no such lessons have been provided in the demonstrator.

7.4 Sequence of Events

At the beginning of a training session, the Quest Tutor asks the student to give his name (see section 7.3.4 above) and asks him whether he has a particular search problem of his own that he would like to consider (see section 7.3.3). Now the student is asked to choose the type of tuition he would like:

- General tuition on on-line searching and Quest;
- A lesson on a subject of his choice;
- Advice on a particular type of search problem.

If the student chooses the first option, the training system selects a lesson to teach him (see section 7.3.2). Otherwise, the student also has a hand in selecting the lesson. In either case, he is taught the chosen lesson, and the system observes whether he completes the lesson, and how he carries out the exercises (if any) before updating the student model. The student model is updated on a very simple basis: if the student successfully completes the lesson and any exercises, he is deemed to have learnt the topic. This sequence is illustrated in figure 7.2.
The sequence is repeated until the student interrupts it or all the lessons have been taught. The student may interrupt a lesson at any time, to end the session or to select a different teaching option. He may also make temporary excursions into other topics during a lesson, by calling up hypertext words and phrases (see section 4.1.4 above).

7.5 Implementation

The Quest Tutorial was implemented on an IBM-compatible PC using Knowledge-Pro.

7.5.1 Problems Encountered

Speed

Development of the Quest Tutorial demonstrator was generally trouble-free. The only problem encountered was a tendency for KnowledgePro to be rather slow. This did not cause great problems, as the display of straightforward text is relatively fast, and speed is not crucial in this system. However, because of this in some cases maintainability of the code which processes student inputs has been sacrificed for efficiency.

Graphics

Some of the lessons in the Tutorial were intended to contain diagrams. Although KnowledgePro can use graphics in principle, in practice the graphics version takes up too much memory and places unacceptable constraints on the rest of the system. Therefore no diagrams were included.
7.5.2 Testing

The lesson material in the Tutorial system is separate from the control mechanisms, so testing was relatively straightforward. The following were tested:

- All hypertext links
- Content of each lesson
- Updating of the student model
- Automatic sequencing of topics
- Saving and restoring of student models
- Operation of each exercise.

The content of each lesson was checked for spelling and general presentation on the screen, not for its value as teaching material.

Exercises were not tested exhaustively.

The flexibility of the system means that lessons may be presented in many different orders. All combinations of lessons have not been tested to make sure they form coherent sequences, although it is intended that they should.

7.5.3 Demonstration at IRS Dialtech

On 23rd October 1989 the Quest Tutorial demonstrator was taken to IRS Dialtech in London where it was shown to some of the students from their Quest course. The students were given a brief introduction to the project and the demonstrator, and then allowed to use it without interference. This was before the demonstrator was completely finished, although most of the material was in place and it was quite robust. The demonstrator was well-received, and, as the team had hoped, various constructive comments were made by the students. One of the objectives was to find any difficulties which the students had in using the system.

None of the students had used a mouse before and they all found it quite hard to use. Although this is a skill which people seem to learn quickly, changes were made to the system so that it can be used either with or without a mouse.

At several points the wording of instructions and lessons was not clear, and several small bugs were found. These were subsequently corrected.
Three useful suggestions were made which the team were unable to act on.

Lesson Packages

The first suggestion, that lessons should come in packages of about half an hour, is more difficult to achieve than it seems at first sight because the student is able to choose a combination of specific topics and general tuition.

Student “Map”

The second suggestion was that there should be a “map” to show students what they have been taught. The task model diagram from which students select topics seems to be an excellent basis for such a “map”, but there was not enough time to implement it during the project.

Global Exercise

The third suggestion which arose from the demonstration was that there should be a “global exercise” which takes the student through the entire task. This could be implemented within the current structure in two ways. The exercises which are currently given could be related to a single search problem, stated at the beginning. Unrelated exercises could still be included. In addition, exercises could be associated with lessons higher up the task model tree than at present. For example, the task “prepare search” could have an exercise attached to it which requires the student to go through the entire process of preparing a search. At the moment only relatively low level topics have exercises attached to them. It would be difficult to give specific feedback to the student on such an exercise, but useful comments could probably be made. Again, there was not enough time to implement these options.

7.6 The Full System

7.6.1 Demonstration at Frascati

On 11th January 1990, the Quest Tutorial demonstrator, along with three others built during the same project, was taken to ESA at Frascati in Italy. A formal presentation of the project results and of the four demonstrators was given. ESA staff were given the opportunity to use the demonstrators after the presentation. The
demonstrators were well-received. The Quest Tutorial demonstrator attracted considerable interest. Comments were not recorded. Since then, the Quest Tutorial demonstrator has been made available to interested parties within ESA.

After examining the demonstrator, ESA decided to have built a full ESA-Quest Tutorial system based on it. Some additions and amendments are described below.

7.6.2 General Feasibility

The experience of building the Quest Tutorial system gives no reason to doubt that it could be extended to a full tutorial system. Such a system would have a full set of lessons covering all the areas contained in the current task model. Although KnowledgePro is slow, its performance is expected to stay near its current level as more material is added. Steps taken to improve the speed of the current system have caused it to be less easy to maintain than it ought to be, but not to the point of impracticality.

Besides the three suggestions made by Dialtech students (see section 7.5.3 above) there are several important ways in which the tutorial system will be developed.

To encompass these amendments, three new components are needed; an interface to Quest (section 7.6.7), a lesson editor (section 7.6.10), and a component to print out test results (section 7.6.8) which is essentially a report generator. Each of these components is simple in itself and can be added to the present structure without requiring any changes.

7.6.3 Improving the Lessons

There is no doubt that the lesson material could be improved. Material from the Quest documentation was used without making much allowance for the difference in media. The material for a full tutorial system should be created in cooperation with an experienced Quest instructor.

The structure of the system allows for different types of lessons to be provided. Other types could easily be added; remedial lessons and repeat lessons are just two.
7.6.4 Adding Exercises

One of the principal benefits of the Tutorial system is its ability to set exercises for the students. This gives them a chance to try out what they are learning, gives a focus to the lesson so that they are forced to think through what they have been told, and also breaks up the monotony of reading page after page of text. However, the number of exercises in the demonstrator is not large. More exercises could be added, and the feedback given to the student could be improved. This can all be done within the present structure.

The results of some exercises will be suitable for submission to Quest via the new link (see section 7.6.7).

7.6.5 Extended Exercises

Exercises will be provided for higher level subtasks. This will enable students to practise combining several search steps, and act as a recap. Travelling up the task hierarchy, exercises will require the student to carry out more steps without guidance. This increases the difficulty of the exercise. Only an advanced student would be asked to perform an exercise relating to a task at a high level in the task hierarchy; a less advance student would be led through intermediate steps.

7.6.6 Student’s Own Problem

The student can choose to be led through the search process by the system. This is intended to help the student to carry out his or her own search problem. Although the system will not know what problem the student is trying to solve, it will be able to adapt the search process to the particular problem according to the answers the student gives to the various questions. As questions are answered the system will build up information about the problem, and tailor subsequent questions to suit it. This feature was included in the demonstrator, but only a few questions were asked, and they were scattered throughout the lessons, which was not found to be satisfactory as continuity was poor. The resulting search strategy can be submitted to Quest via the new link (see section 7.6.7).
7.6.7 Link to ESA-Quest

By providing a link to Quest (eg via a modem) it will be possible to submit queries to Quest on-line and observe the results.

7.6.8 Adding Test Material

A series of test exercises will be added which the student can use as self-test if he chooses. The results can be printed out for the student’s benefit.

7.6.9 Improving the Student Model

The student model in the demonstrator is built up in a very simple fashion (see section 7.3.4), and uses only three values: known (the system believes the student knows the topic), unknown (it believes he doesn’t know it) or no information (the system doesn’t know whether the student knows the topic or not). By using a more sophisticated scoring method, more use could be made of available information. For example, if a student can complete a simple exercise, but not a more difficult one, the system could make some deduction about how well he knows the topic, and probably on the nature of his difficulty.

There is a limit to the sophistication which can usefully be added to the student model, however. The purpose of the model is to help the system to make teaching decisions. With a system based on fixed lessons written in advance, the teaching decisions it is able to make are necessarily limited, and therefore a very detailed, sophisticated student model would be inappropriate.

7.6.10 Lesson Editor

An editor will be provided to make it easy for ESA staff to change lesson content.

7.7 Conclusions

Construction of the ESA-Quest Tutor has tested both the method and the structure to a limited extent. The Quest Tutor has rather different requirements from the navigation trainer, as it not an embedded training system and was built as a demonstrator. Not being an embedded trainer means that some parts of the structure are not required. Being a demonstrator means that it is not actually intended to be
used as part of a training program. In fact, due to the strictures of the project, much of the method described in this document was not followed. However, the central procedure, Hierarchical Task Analysis, was used very successfully and is an integral part of the tutor.

7.7.1 Comparison of the Structure

The Quest Tutor has four components:

Interface: handles communication with the student. This is exactly comparable with the Interface component in the navigation trainer.

Sequencer: selects the next lesson to teach. This corresponds to the plan generator in the navigation trainer.

Tutor: teaches each lesson. This corresponds to the Executor in the navigation trainer.

Student Model: builds a model of the student by observing his behaviour. This is equivalent to the Model Builder in the navigation trainer.

There are no components corresponding to the Classifier or the Scenario Generator, because the Quest Tutor is not an embedded trainer. A Report Generator was not provided for the demonstrator, although the full system will have a limited version.

The Sequencer and the Tutor components, although equivalent to the Plan Builder and the Executor in the navigation trainer, are actually much simpler, and closer to the model of training given in section 2.3. The Quest Tutor does not construct a proper tutorial plan: the sequencer simply chooses the next lesson to teach when it is asked. This is appropriate for short teaching sessions, or where the student chooses lessons for himself. However, students appeared to feel a lack of structure in the teaching during the demonstration at IRS Dialtech (see section 7.5.3) when they suggested that lessons should come in "packages". It was not possible to include a plan builder in either the demonstrator or the full version of the Quest Tutor due to resource constraints.

Similarly, teaching each lesson is a simpler matter in the Quest Tutor than in the navigation trainer, as the tutor is not embedded in any other system. Exercises in the Quest Tutor are administered directly by the Tutor component. This is a direct result of the Quest Tutor not being an embedded trainer.
Much use was made of hypertext in the Quest Tutor. Students seemed to respond well to this. It gave them an opportunity to request more information on particular terms, including complete lessons if they chose. Some students appeared to get sidetracked and started to browse through the hypertext. Unfortunately, when they returned to the lesson they had usually forgotten what they were doing. If the tutor had had a planning component it would have been able to replan at that point and adopt a new course suited to the new circumstances.

7.7.2 Lessons for the Navigation Trainer

The Quest Tutor was built rapidly as a small demonstrator. It is not nearly such a sophisticated system as the navigation trainer was designed to be. Nevertheless, the same structure (simplified because the Quest Tutor is not an embedded trainer) provided a useful start to designing the system. An adaptation of the design approach was also helpful. Although much cut-down from the full version described in chapter 3, it retained the essential feature of basing the training scheme on the task description. It is particularly interesting that Hierarchical Task Analysis was so successful, as that technique was originally developed for process control which is a quite different kind of task from information retrieval.

The Quest Tutor relied heavily on hypertext. Students found this acceptable, and it proved to a useful medium for communication with them. A version of hypertext extended to include graphics would probably be useful in the navigation trainer.

Experience with the Quest Tutor suggests that it was correct to include a planning component in the navigation trainer. Students using the Quest Tutor complained of a lack of structure; the intention was that the use of a tutorial plan would provide that for the navigation trainer.
8. DISCUSSION AND CONCLUSIONS

This report has described a design approach for training systems, which it proceeded to apply to embedded training, and in particular to the design of a trainer to be embedded in a navigation aid. The structure of the trainer will generalise to many embedded training systems, if the requirements are not too dissimilar from those of the navigation trainer.

Chapter 6 described the design of the embedded navigation trainer in some detail. Throughout the chapter various design decisions were noted as being unresolved. If the project had gone according to its original plan, these design decisions would have been resolved by additional work, including some small-scale experiments. The start of this chapter summarises the outstanding decisions and briefly describes the work that would have been done if circumstances had permitted.

The later parts of this chapter discuss some of the conclusions which can be drawn about embedded training.

8.1 Unresolved Issues

Most of the outstanding design decisions noted in chapter 6 refer to the Tutor Interface and concern how it is to answer student’s questions. The other main issues are the details of how evidence is used to build the student model, what kind of feedback should be given to the student, and how real navigational situations can be classified and identified. Of these, the last may turn out to be too difficult for practical use: in that case the trainer will not be able to add to the student model while the navigation aid is used for real.

It had been planned to carry out some experiments during the project to answer some of these questions. In particular, it was planned to build a mock-up of the trainer on the navigation aid to find out the types of questions that students ask, to try out alternative schemes for answering them, and to compare different rules for building the student model. Due to the circumstances at ARE (Teddington) described in section 1.1, none of these experiments could be carried out.
8.1.1 Interpreting and Answering Students' Questions

A number of issues remain to be resolved in the Tutor Interface (see section 6.5), concerned with interpreting and answering students' questions. The first to be tackled should be to identify questions which students ask frequently. A practical way to do this is to have students try out the mock-up (or a version of the full system) with someone on hand to answer questions. Answers could be supplied either by a knowledgeable system developer or, better still, by a navigation instructor. If there are completely standard questions, standard answers can be provided. If there are standards types of questions, if might still be possible to provide standard answers with blanks to be filled in by the system as appropriate. In any case, valuable information about the types of questions students ask will be obtained by this method.

Since the navigation trainer is not to have a general facility for understanding English, a format needs to be worked out which will allow students to ask questions. The format, or formats, must be easy for students to learn, and cover as many types of question as possible. Hypertext was used for this purpose in the Quest Tutor (see chapter 7) with some success. A graphical extension of hypertext would be a candidate for the navigation trainer as well, although other formats may be required.

Issues concerned with interpreting a student's questions are partly dependent on the question formats chosen. The system must be equipped with the knowledge of how to decide what task a question refers to, and how to use the context and the student model to interpret questions. Probably the best approach is to supply the system with a simple first attempt (e.g., assume the task is always the current task) and expand on that as required by taking advice from instructors. If the system cannot disambiguate the question, it may ask a question of the student to help it. This would have been beyond the scope of the project except in some simple cases, such as where the system has identified two alternative interpretations.

In answering students' questions a trial-and-error approach will also be required, although the explanations given by certain expert systems (mostly experimental systems) will give some guidance on how to construct answers. The particular issues to be resolved are the use of graphics in answering questions, how to decide whether to answer a question or not, and how to construct answers using only terms the
8.1.2 Building the Student Model

Outstanding issues in the Model Builder (see section 6.1) are how to use and compile evidence, and how to set the value of various types of evidence. Alternative methods would have been evaluated on the mock-up, and the chosen method subsequently fine-tuned once the system was operational.

8.1.3 Building the Tutorial Plan

The mock-up would not be so suitable for investigating issues in the Plan Builder (see section 6.2), as it relies heavily on a student model being available, which requires a fairly complete version of the trainer. One of the decisions to be made is when to decide that remedial teaching is required, based on the strength of evidence in the student model. This is a crucial decision; students will not appreciate remedial teaching they don’t need, but may become confused if remedial teaching is withheld. The point at which remedial teaching should be given may vary from student to student. Initially the advice of instructors would have been sought in deciding how much evidence is required. Trials with students would have been used to fine-tune the method.

Specifying what extrinsic knowledge of results students should receive is another important function of the Plan Builder. Once again, the advice of instructors would have been used to provide methods for specifying when to provide and remove knowledge of results, followed by trials with students to fine-tune the methods.

8.1.4 Other Issues

The Report Generator (see section 6.4) provides reports with content and layout specified by the supervising instructor. Interviews with instructors would have helped define an interface and grammar for specifying reports and whether any standard report formats would be useful.

The Classifier (see section 6.7) needs to be able to classify navigation situations. This means that a classification of navigational situations must exist, together with a method for assigning actual situations to the classification. It is not clear that either...
of these can be achieved. The help of experienced navigators would be required to provide a classification.

8.2 Training Supervision

Embedded training seeks to make cost or resource savings in training people to use computer-based equipment by using the equipment itself for training. Many embedded training systems simulate incoming data about the environment so that users can be trained when the equipment is not in operational use. They are not intended to be used unsupervised. The navigation trainer is not in this class: it must be able to train navigators without supervision. Much of the effort of this study has gone into considering how to compensate for the lack of human supervision, and many of the conclusions drawn here do not apply to embedded trainers designed to be closely supervised.

An unsupervised training system must be able to cope with the student who, despite all precautions, comes to the system lacking a vital element of skill or knowledge. One solution is simply to make sure that trainees know how to get help when they do not understand what the system is telling them or asking them to do. This may not always be practical, and may still result in the system confusing a student further. If abandoning students in this way is not possible, the system must be prepared to teach students the things they do not know. A student may be able to diagnose when he needs such remedial teaching, or the system may need to do it for him. Having identified the need, in one way or another, the system must provide the teaching. Here again, the absence of supervision means that the teaching must be specific and effective: the student may have no-one to fall back on. However, designing a system which is capable of teaching a reasonably complicated skill with any hope of adapting to the individual student’s problem is a far from simple task.

The difficulty arises from the subject matter: for the material to be taught by a computer system it must be described explicitly. For this reason it will never be possible to build a trainer which replaces the human instructor totally and can cope with every training situation. However, probably long before this limit is reached, a practical limit to the detail, or depth, of the description contained in the system will be met. A student with a difficulty in a particular area may find the system cannot
explain things fully enough for him to grasp. These limits are reassuring in a way, as they imply that the human instructor can never be satisfactorily removed from the training loop.

Although this limitation is potentially quite a problem, in practice it may not be so severe. The process of transfer of training enables students to perform tasks they have not been taught but which are similar to what they already know. It is not necessary to teach students everything; they can often work things out for themselves. Even an experienced human instructor does not try to teach students everything in any case, due to time constraints, and because novel situations can always occur. By the same process, the automated system need not teach everything to the students.

Secondly, it is extremely unlikely that there will be no human supervision at all, although it could well be intermittent. A valuable result may be for the system to identify a student's weaknesses without trying to remedy them.

Where the student must learn to take strategic decisions which cannot be described in the training system, it may be possible to give the student exercises defined in advance, with answer and explanations similarly pre-specified. The system cannot generate exercises or apply its knowledge to real-life situations in this case.

8.3 Continuation Training

These objections to embedded training arise because we have considered an unsupervised system which may need to teach confused beginners. However, we would probably not seek to train complete beginners using a computer-based trainer: beginners, who have not yet grasped the basics of a subject, may have a variety of conceptual problems which are outside the scope of the most sophisticated computer-based trainer, but which a human instructor can easily understand and remedy. Instead, we would use such a system for more advanced students: In fact, the more advanced the better, as they are less likely to have conceptual problems. The embedded trainer would provide a useful means for such students to exercise the skills they have been taught in theory. The simulator can provide examples of the various situations they need to be able to cope with. By observing a student's performance it can generate extra examples for the aspects at which he is weak.
Other sources of information about the student's abilities are available to the trainer: advanced students are often able to say what they find difficult.

The embedded trainer can provide carefully tailored practice for students who have already been taught the theoretical material, and can evaluate their performance. Better yet, it can provide continuation training for graduates. Frequently not all aspects of a task will be required equally often; the more important aspects, such as emergency procedures, may be required least of all. An embedded trainer can simulate such situations to make sure that the graduate does not forget how to respond, or lose his facility. It may be as continuation trainers that embedded training systems have most to offer. A continuation trainee may still need to be taught some aspect of the task he has forgotten, but he is not nearly so likely to be confused by an explanation because his basic concepts should be sound.

8.4 Advantages and Disadvantages of Embedded Training

All of the advantages of embedded training are equivocal. Some of the issues are discussed here.

8.4.1 Cost and Fidelity

The advantages usually put forward for embedded training are fidelity and cost. Students have no problem transferring from the training to the operational environment because they are one and the same thing. The effect is to produce a very high-fidelity training simulator while cutting costs by using the same equipment as the operational system.

The disadvantage connected with fidelity is one of the least severe. Fidelity is put forward as one of the advantages of embedded trainers because it reduces the problem of transferring skills from the training to the operational situation. However, quite often in a training system it will be desirable to simplify the training environment, and so reduce the fidelity, in order to teach students the overall task without their being bogged down by detail. This is not easy with an embedded trainer.

The argument for the cost of embedded training is initially persuasive: an embedded training system may be cheaper than a high-fidelity simulator. However, in many
tasks a low-fidelity simulator is just as good for training, and other methods such as part-task training, or a model of some kind, may be more useful. Even if a high-fidelity simulator is essential, embedded training may not be a cheap option. As software costs become increasingly significant in building systems it may even be cheaper to build a high-fidelity simulator from scratch, including equipment costs, than to try to embed a training package in a system which was not designed for it.

8.4.2 Availability of Training Facilities during Operation

Another advantage of embedded training is that the training system can be available when the equipment is being used for real. If appropriate, the user may be able to invoke the training system at any time to ask a question, get some extra training on a particular skill, or whatever.

This advantage may be totally offset, however, if the user has no time to make use of such facilities or if it is vital that he is completely trained before he is allowed to use the system operationally. Also the most likely facility for him to want is the help facility, which answers his questions. This is probably the most difficult facility to make truly useful, since the range of questions it needs to cope with is so vast, and the context affects the answer which should be given.

Where the training system models the user, it can continue to collect information about him when he is using the system operationally. This would be an extremely useful facility for a continuation trainer, as it could tell which situations the user had had to deal with recently and not bother to practise those. The problem is two-fold. Firstly the training system must be given the information it needs about the environment. How hard this is depends on the way environmental data is collected. Secondly the training system must be able to classify the situation represented by the environmental data. This is a problem of far greater complexity than choosing a situation and simulating data to represent it. It will not be possible in every case.

8.4.3 Continuation Training

There are obvious benefits to having a continuation trainer in the place of work so that users can practise emergency procedures during quiet shifts and other times
when no instructor is available.

However, there is potential for confusion if this is allowed. If the system’s outputs are at all dangerous, it could be catastrophic if the user does not keep clearly in mind the distinction between the operational system and the training system.

8.4.4 One-to-One Training

If the system is adaptive and suitable for use without supervision (see section 8.2 above) then there are further advantages. The student gets some of the benefits of one-to-one training, as the system adapts to his particular training needs, while at the same time the instructor is released to concentrate on another student’s training problems.

The disadvantages of this approach are discussed in section 8.2.

8.4.5 Updating of Training Material

An interesting advantage of an embedded trainer is that, if it is well-integrated with the operational system, the training material will be automatically updated when the operational system is changed, so that the training package can never get out of step with the task.

On the other hand, a poorly-integrated embedded training system may make matters worse. If a separate training system is used, the consequence of it getting out-of-date is that students learn to perform a task which is no longer quite correct. Since they have to make some adaptation in transferring to the real task anyway, the effect may not be significant. If an embedded trainer becomes inconsistent with the main system, the results could be complete nonsense, with the trainer telling the student to use a facility which is no longer available, for example.

In order to build a well-integrated embedded training system, the operational system must be open to inspection by the trainer. This means that the training system can directly use information in the main system and so automatically keep up-to-date with changes to that information. If the main system cannot be made open to interrogation in this way, the next best thing is to make the links to the training system explicit in the main system, so that a programmer revising the main system
will know where to change the training system at the same time.

8.4.6 Conclusions about Embedded Training

Embedded training is a technique which applies to training computer-based tasks. The term covers a wide variety of systems from straightforward systems which simulate input data to sophisticated systems containing dynamic student models. Section 8.4 showed some of the disadvantages of embedded training, which are many, and some severe. With all these drawbacks, it is clear that embedded training should be employed only after careful thought. Taking the drawbacks into account, it seems that the most likely place for embedded training to be useful is in the continuation training of a computer-based task where there is no danger of the training and operational situations being confused by the operator. It would be better if the operational system could be easily interrogated by the training system so they can be well-integrated, and of course embedded training should be a desirable, cost-effective training method for the task. Given this rather stringent set of conditions, embedded training may have something worthwhile to offer. It should be noted that the navigation trainer fulfils all of these conditions except the last; no consideration has been given to whether there are more desirable, cheaper methods of training navigators because the system was convenient to take as an example for embedded training.

8.5 Generality of the Design

There are two senses in which the design of the navigation trainer has some generality: the design process and the output of that process, the design itself.

8.5.1 The Design Process

The navigation trainer was designed using a method (see chapter 3) derived partly from Singleton, who uses a systems approach. There are certainly other ways to design training, but Hinrichs shows (see section 2.1.3.1) that the systems approach can be a useful way to view both organisations and man-machine systems. The method outlined in chapter 3 proved useful in the design of the navigation trainer. Its two chief advantages are the way in which it clearly relates the training objectives to the training scheme, focusing the design process, and the way it
encourages the designer to consider all aspects of a training scheme, not just the production of the training material itself.

As far as the generality of the overall method is concerned, there naturally are limitations. The method was specifically developed for training, rather than teaching, which, following Hinrichs' definition (see section 2.1.2) refers to procedures initiated by an organisation to foster learning among its members which are intended to contribute to its objectives. It would be rash to apply the method developed here to a non-organisational teaching situation. The method refers to a task to be trained, because generally speaking organisations are interested in enabling their members to perform tasks for them. Any learning of facts or theories is usually in support of a task. Where the training objective is not naturally seen as a task, the method described here may not be the best approach.

Training objectives differ greatly, and are not easily categorised. It is difficult to say precisely what the limits of applicability of this method are: each case must be considered separately. However, there is no obvious reason why the method would not be useful in the design of a program to train any task within an organisation.

The navigation trainer was designed using Hierarchical Task Analysis (HTA - see section 2.1.3.3). This produced the task description called for in the method, and was also used as part of the navigation trainer itself (see section 8.5.2 below). HTA was originally developed for process control, and it is well-established that it has general applicability to process control tasks and other tasks of a similar nature. Its use for navigation and information retrieval (see chapter 7) shows that it can be applied to other tasks. The adaptations of HTA suggested by Shepherd and Kontogiannis make it especially suitable for computer-based training, but should it not prove applicable to a particular task, other methods of task analysis are available.

8.5.2 The Structure of the Trainer

The structure of the navigation trainer is described in chapter 4, and the components are described in detail in chapter 6. As a brief recap, the components are:

Report Generator: provides reports on the students, either individually or as a group;
Model Builder: builds a model of each student by observing his behaviour;
Plan Builder: builds a tutorial plan;
Executor: executes the tutorial plan;
Classifier: classifies the environmental data and describes the situation;
Tutor Interface: handles communication with the student;
Scenario Generator: simulates environmental data as instructed by the Executor.

These components are all embedded in the navigation aid.

Of the components, some are entirely specific to the navigation trainer. The aid itself, and the Classifier are clearly not general. The Scenario Generator is basically a simulator, and hence also specific to the navigation aid. The Model Builder and the Plan Builder are both based on the HTA, but the rules they use could in principle be general. In practice it might be convenient to make some of their rules navigation-specific, but the internal structure of both of these components need not be. The Executor's actions are controlled by the plan. The Executor could probably be made general by storing all the specific parts of the lessons in the plan, so that the Executor acts rather like an interpreter for a computer language. However, the navigation aid has a rather specialised graphical interface, and so any communication between the training system and the student will be specific to the navigation aid. Consequently the Tutor Interface and at least some parts of the Executor will not be general. To make the Report Generator general would make it difficult for the instructor to use.

However, the most important re-usable part of the navigation trainer is its structure. This provides a tried system architecture which was put together with the intention of making the most of the results of Hierarchical Task Analysis, which is an important part of the design method. It has sufficient flexibility to cope with both adaptive and embedded training, which are both complex. The requirements of embedded training, in particular, make the structure fairly complicated, although a trimmed-down version is possible for non-embedded training systems, as the design of the Quest Tutor shows.

8.6 Final Remarks

This thesis has addressed the problem of how to build computer-based training systems, especially embedded training systems, in a methodical fashion. By
emphasising the training objectives two main features of the proposed method arise: a clear development from the training objectives to the training material, and the use of Hierarchical Task Analysis. These are not specific to embedded training, but apply also to other computer-based training systems. The thesis does not claim that it is the best of all possible methods, nor that the method will apply to all such systems. It has not even been possible to investigate the classes of training problem to which it will apply. It does claim that the method is a useful one which, if it is judged to be applicable to a particular case, will provide a procedure and a structure for designing a training system. This claim is made on the basis of the experience of designing a complex embedded trainer for navigation, and of designing and building a relatively simple trainer for information retrieval.
9. REFERENCES


IRS Dialtech Beginner’s Notes.
Additional References


