British Journal of Educational Technology *Vol 39 No 2 2008* 311–318
doi:10.1111/j.1467-8535.2008.00819.x

*A new intelligent tutoring system**

Lindsey Ford

At the time of writing, Dr Lindsey Ford was with Logical UK Limited in London

Abstract
Early in 1984, Logica started an ICAI research contract with MoD (Procurement Executive), which was to be undertaken in collaboration with the technical authority at the Royal Signals and Radar Establishment, Malvern. A prototype system, which has become known as TUTOR, was scheduled for delivery to the client later in 1985. Initially, and for evaluation purposes, the system ran on VAX 11/780 hardware under VMS. The target subject for TUTOR was the field of emergency procedures in flight safety regulations as taught to air traffic controllers. Initial development of TUTOR, however, has been aimed at the Highway Code, a domain which we, as well as the client and other observers, are familiar with. Evaluation of TUTOR would be based on both domains. Prolog was chosen as the implementation language because ICAI, as we shall see, makes use of rich data structures that can easily be expressed in Prolog.

Introduction

Here are some of the cost-effective features of CAI systems that are frequently cited:

- courseware establishment;
- provision of training;
- less training time required; and
- more trainees can be trained;

and one would not wish to quarrel with them. Improvements in authoring languages over the last decade enable courseware to be established in a routine and technically undemanding way. Once established, courseware is administered by machine at considerably less cost than its human counterpart. Most studies indicate that less training time is required when CAI systems are used, the rationale being that individualised tuition does not penalise faster learners. Finally, CAI can, in principle, accommodate any number of course members and indeed any number of courses. Class size and curriculum constraints of traditional training are therefore less critical in CAI.

*This paper was first published in *Interactive Learning International* Vol. 3, no. 4, 1986, 23–26, by John Wiley & Sons Ltd.

A number of CAI/CBT systems currently on the market advertise most of the features above (they are, after all, good selling points), but desirable though they may be, they are worthless if trainees do not acquire the knowledge or skills that such systems attempt to impart. Paradoxically, training is then cost-ineffective as the operational goals of training will not be met.

An awareness of this situation has led to the emergence of a new style of educational technology that seeks to raise the level of confidence that a system is imparting the training effectively, ie, that the trainee learns what is taught. In this paper, we shall look at a system called TUTOR, which is an example of this technology. The first section addresses the background of the TUTOR project, and in the second section, we discuss the design principles on which the system is based. The architecture of the system is revealed in the third section, and finally we speculate on the roles that such systems can play now and in the future.

Design principles

The major aim of TUTOR is to provide effective instruction. As we have said, this requires that a trainee learns what is taught. Now, it is a matter of fact that human teachers are, by and large, extremely skilful at their trade, and one is hard-pressed to find any evidence that CAI systems outperform them in terms of transferring knowledge and skills to students. Why should this be? Why is it that after some 20 years or more of CAI that there is not an abundance of evidence?

The answer, of course, is that CAI does not outperform its human counterpart in the way mentioned. I shall suggest that there are three main reasons for this state of affairs, and that these can be derived by a simple comparison of CAI and human teaching.

Three problems of CAI

First, the human teacher or trainer has knowledge of individual students that he refines and builds on. He maintains an internal model of, among other things, what a student knows. There is ample evidence to support this idea. A teacher is able to predict with some accuracy how students will perform when asked a question or in an examination. CAI systems are not in this happy position.

They may well record what training a student has received, but this is a far cry from having an idea of what he knows. CAI systems have difficulty in recording this knowledge because, in essence, it is a model-building activity for which a prerequisite is knowledge of the subject on which the model is based. And this brings us to the second point of comparison.

Human teachers understand the subject they are teaching. They are able to answer questions about the subject and solve problems in the subject domain (where this is appropriate). Conventional CAI systems cannot be claimed to have real subject understanding: the branching method by which they are implemented merely indicates possible routes through the subject material. The ability of a CAI system to provide the

answer to a task on demand is itself insufficient for real learning to take place, since what is crucial to a student who has provided a wrong solution is the precise identification of where he went wrong and why—not a global and uninformative 'incorrect'.

Analysis of teacher–student interactions reveals that a mixed initiative dialogue occurs—that is, a dialogue in which either the teacher or student at any moment has the initiative or control. This initiative is used in a variety of ways. Frequently, a student will not directly answer a teacher's question but ask a question himself, usually to clarify the original question but sometimes to determine some subject knowledge that he does not know but which he feels is needed to provide the required solution. CAI, on the other hand, is more authoritarian in approach, relegating the student to a passive role of answering questions. This denies the student an opportunity of engaging the subject in a meaningful way and can be a frustrating experience.

Possible solutions
We have suggested that three features of the human tutoring situation:

- knowledge of the student;
- knowledge of the subject; and
- mixed initiative dialogue;

are largely absent from conventional CAI and contribute to CAI's poor performance in terms of transferring knowledge and skills to students. What steps, then, can be taken to improve this situation?

Throughout the development of a number of TUTOR prototypes, we have attempted to provide these features as far as possible. The extent to which we have been successful will be gained from the following description below, but the real test concerns the ability of TUTOR to transfer knowledge, and this we shall only appreciate after formal evaluation trials.

Student model
TUTOR maintains for each student a model of his knowledge of the subject he is learning. The model is used to influence tutorial action (of which more later), and to provide the course administrator with a profile of the student. Typically, a profile will indicate the strengths and weaknesses of a student and his coverage of the syllabus. At the lowest level, the model has recorded in it details of concept learning. We regard a concept as a simple piece of declarative knowledge (eg, a motorway is a dual carriageway—or a step of a procedure (eg, indicate after checking mirror when overtaking) for which three types of learning evidence are recorded in the model:

- the number of times the student has received exposition for the concept;
- an aggregate task score; and
- a chronological history of correct and incorrect use of the concept in task performance.

The first of these is used to control the frequency of exposition, and the last the timing of exposition and presentation of remedial information. The aggregate task score is a function of the number of tasks performed requiring knowledge of the concept, the task performance (correct or incorrect), and the task difficulty. Each concept has a threshold associated with it, which when compared with the task score indicates whether the student has met some pre-established criterion. When a student's score hits the threshold, no further training will be provided for the concept unless some subsequent task performance causes the score to drop below the threshold.

Now each concept is associated with a particular topic in the syllabus, thus an 'interpretation' of the model allows TUTOR to draw conclusions about a student's mastery of a particular topic. The teaching strategy of TUTOR is therefore able, by reference to the model and the interpretation of it, to determine which topic a student needs further tuition on and, in particular, which concepts of the topic.

Subject knowledge

At the heart of TUTOR is a knowledge base of rules and an interpreter of them. These rules represent the subject knowledge and expertise that TUTOR attempts to impart to a student. A rule, then, is a concept. The rule form of the two concepts mentioned earlier is:

IF ... THEN a motorway is a dual carriageway

IF the intention is to overtake and you have checked your mirror
THEN indicate to overtake

TUTOR uses rules such as these in two ways. First, it uses them to provide a solution to a task given to a student. This it compares with the student's solution in order to verify the correctness of the solution, and, where appropriate, provide a critique. Second, TUTOR uses the rules in a consultation mode to provide students with task competence information about the subject domain.

Mixed initiative dialogue

At each point in the dialogue with a student, TUTOR enables the student to take the initiative in a number of ways. He may select any topic from the syllabus and one of four types of training:

- novice;
- skilled;
- self-test; and
- examination.

Novice training consists of exposition followed by tasks, and, where appropriate, remedial feedback. Skilled training assumes that the student has already received novice training either from TUTOR or some alternate source. The self-test mode of training is useful to the student using the system for the first time, as it enables him (and TUTOR) to assess the level of training required during subsequent use of the system. TUTOR can

set formal examinations, the results of which are made available to the course administrator. The important point about these different training modes is that results from each of them are used to update TUTOR's student model, and this enables TUTOR to pitch its instruction at a level appropriate to the student concerned.

Other options available to the student include: WHY (or what is the justification or explanation of exposition or task answer), I DON'T KNOW (the answer to a task), ANSWER (the task yourself), CONTEXT (of the present situation, ie repeat the task or exposition), and HELP (indicate the options available to me and explain them).

The above features can be used by a student to enable him to control: what he receives training for, the type of training, and the information flow from TUTOR to himself.

TUTOR architecture

Figure 1 shows the main elements of TUTOR's software architecture.

Imagine that a student has been presented with a task to perform and has provided a solution. The administrator records the student's response in the student history database (TUTOR may make use of this information later), and then presents the task for the expert system to perform. A comparison of the two solutions—student and expert—enables the student model to be updated appropriately. In the case of a wrong solution, the administrator may decide to invite the student to try again. In this situation, the teaching strategy component is passive, but the administrator is activated again. It

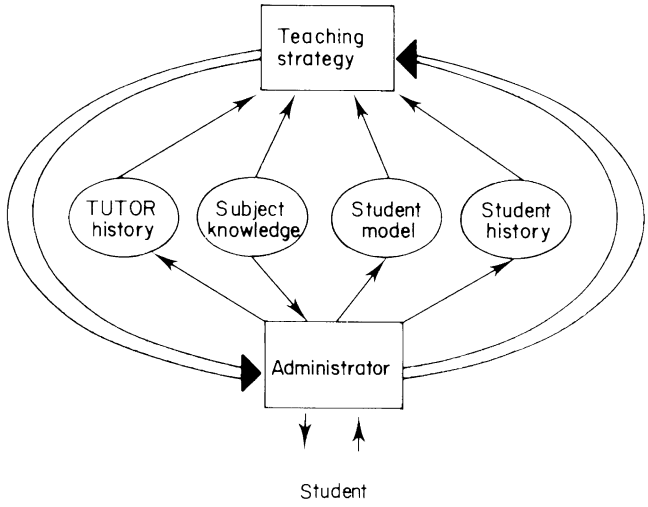

Figure 1: The main elements of TUTOR's software architecture

updates the TUTOR history database (to indicate that the student should repeat the same task and that the student should repeat the same task) and outputs a message to that effect to the student.

A correct solution activates the teaching strategy. The teaching strategy is a rule set that examines the four databases shown in the diagram to determine what best should next be done to increase the student's knowledge. The repertoire of possible actions includes:

- providing (remedial) exposition;
- setting another task (perhaps related to the previous one);
- progressing the student to a new subtopic; and
- regressing the student to a previously taught subtopic (where this is appropriate).

The decision is communicated to the administrator, which performs an update of this information to the TUTOR history database and then outputs a natural language sentence of it to the student. A new dialogue cycle is thus initiated.

The important point of this method is that the teaching strategy uses all the information available to it to reach a decision and that decisions are taken not at the end of some preconceived lesson plan, but on each interaction cycle. Sensitivity to a situation (which includes historical information as well as the current state of the student model) and immediacy of response to it, both characteristics of human tutor–student interaction, are features of this approach.

The future

TUTOR can be used effectively now to provide training in the manner described. However, the modularity of its construction and explicitness of its implementation enable us to envisage its refinement to meet more complex and sophisticated training requirements.

TUTOR and video

The use of video in training has been in evidence for many years, but it is only comparatively recently that its value in interactive training has been exploited. Systems are now available which engage the student in a quiz about a displayed video sequence. Of a more sophisticated nature, there are systems that enable graphics to be superimposed on the video film, sometimes under the control of the learner, eg, entering (graphic) cash information on a (video) balance test.

These uses to date, however, have been limited from an educational point of view, with the trainee having little control over what is presented to him beyond 'freezing' the film or requesting a section to be played in slow motion. By incorporating in TUTOR descriptions of what video teaching material is available, the system would have some understanding of what is being shown to the student. This would enable TUTOR to answer a student's questions about a film clip and even use other film clips to clarity any miscon-

ceptions he may have. The important point here is that the student is not a passive onlooker, as it is his interactions with the system that enables it to orchestrate its video and verbal output to best foster learning.

Embedded TUTOR

There is a wide spectrum of functions that can be undertaken to improve operational performance, and there is no better case in point than when the operational system involves the use of humans interacting with a computer application.

The fortunate novice user of a computer application may receive 'off-line' training about it (CBT or otherwise), but will not be allowed to use the live system as part of the training as this could have disastrous operational consequences. When trained, the user—again if fortunate—will find the system has a HELP facility that may give the desired operational help (some HELP facilities may themselves require training prior to use). Between these two extremes—formal training and HELP—a system could potentially provide its users with a consultancy or guidance function, but this is rarely, if ever, to be found.

There are a number of problems associated with the two extremes. 'Off-line' training admits the possibility of incompatibility of training courseware and operational practice: a certain amount of retraining thus becomes necessary. Frequently, CBT 'off-line' training utilises non-operational (and different) hardware that requires the trainee to undergo another environment change. Both factors are stressful to the trainee and inevitably result in degraded operational performance. Most HELP facilities are primitive—taking no account of the user's state of knowledge or the context in which help is requested—and involve laborious searching of the HELP database for the required information by the user.

With the increase in frequency of interaction with computer systems by an ever larger percentage of the public and private sector, a more enlightened view is required.

It is an important insight that both operational and training aspects of a computer application can be fully integrated. The knowledge base used by TUTOR for training can, as it is competent in the subject domain, be used operationally. Contrariwise, a knowledge base developed to become operational can serve in a training capacity. As the knowledge base can thus perform both functions, it ensures that training material and operational practice are not just equivalent but the same.

Now, if a computer application is to be at all sensitive to its users, it must maintain user models of them. But these models are the student models we have earlier referred to. They contain beliefs about users' knowledge—in this case, of the application. These models would be used by both the training and operational system. The training system would perhaps create the initial model to be later used by the application. But the application should also update the model—after all, it can provide actual information

about the users' performance and knowledge—which in turn would be used by the training system to provide refresher training where necessary.

We can now envisage a much more capable HELP facility. It has at its disposal a model of the user (what he knows and what features of the application he has frequently used), and a base of knowledge about the application and its situational state when help is requested. These it can use to determine the precise nature of the help that is needed, and the level at which to pitch the help information. Also we can imagine a more subtle HELP system that is initiated by the application rather than the user. This would, for example, be activated by an error, eg, invalid user input, and be interpreted and translated to a more convenient form before being outputted to the user.

While we can foresee the integration of training and operational practice at system development time to provide the features discussed above, there is an interim problem concerning systems that have already been implemented (in the conventional way), but for which these features are desirable. Logica is presently examining practical ways by which TUTOR and intelligent help can be embedded in them.

Acknowledgements
Financial support of the TUTOR project by the Ministry of Defence (Procurement Executive) is gratefully acknowledged. A number of people have influenced work on TUTOR. Nigel Davies, Simon Dickens, and Rod Rivers are codevelopers of the system, and technical assistance has been provided by Steve Bevan, Dave Lowry and Philip Wetherall. Consultations with Bran Boguraev, Rod Johnson, Tim O'Shea and John Self have been invaluable.