Rapid Prototyping of an Intelligent Tutorial System

Pamela J. Woods and James R. Warren
Advanced Computing Research Centre
University of South Australia
pam.woods, james.warren@unisa.edu.au

Abstract

Today there are few Intelligent Tutoring Systems (ITS) available that can teach outside a specialised domain. In the 90's, the hardware limitations of speed, memory capacity and cost are reducing, but the software still needs to be streamlined. We need to build tools that enable rapid prototyping of tutoring systems over a range of domains. We argue that such systems can be built effectively from the combination of electronic books as the interface medium, driven by knowledge-based systems.

Early generic systems were not effective teaching systems. We need to link the independent layers of the domain and the basic teaching strategies by providing a meta-strategy layer. Different teaching styles may be used both for different domain areas and for different students. Systems must allow flexibility in teaching styles as well as in domain material.

This paper describes the process of rapid development of an intelligent tutoring system in ToolBook. It uses dynamic links to Word for Windows and Excel files to automate the electronic book creation for a given domain. The teaching strategies and student progress are also communicated as tables in Excel, enabling the tutor to teach intelligently according to the teacher's prescribed paths. We have thus developed a generic framework into which the teacher supplied domain material and tutoring strategies can be linked. The prototype generates an ITS for tutoring loop structure in Pascal with various strategies. Our aim is to build a tool that can be used to generate tutoring systems with versatile teaching strategies over a range of hierarchically organised domains.

Keywords

rapid prototyping, adaptive, intelligent tutor

1. Introduction

Today there is universal demand for cost effective instruction. Computer assisted learning in higher education is one aspect of this that has considerable interest, both locally to enable remote students to access courses and globally, for example the TILT (Teaching and Learning Technology Initiative) programme launched by the University Funding Council in Britain 'to encourage the widespread expansion of technology-supported learning in the British university sector' (Boyle et al, 1994).

Despite considerable interest over the last thirty years in the development of intelligent tutoring systems, which were hoped to be as effective as a human tutor, few systems are in practical use today.

However, the factors such as machine speed, cost and memory capability which limited the development of systems last decade are now vanishing. There is still a considerable cost in the development of domain material, but this task has been simplified by modern authoring software. Research has proceeded in the cognitive theory of learning, but we need now to focus on putting the pieces together to allow alternative and adaptable tutoring styles. Existing shells that enable a teacher to enter domain and teaching strategies are becoming available, but are not easy to use and do not teach well.

This paper describes a rapidly prototyped intelligent tutoring system for introductory Pascal that allows comparison of alternative teaching strategies. The knowledge base was created using the Microsoft Windows authoring package ToolBook (ToolBook is a trademark of Asymetrix Corporation) to create an electronic book.

We will initially discuss the evolution of intelligent tutoring systems and the research into teaching programming, then look at the components needed in today's intelligent tutors. Strategies for teaching are reviewed, then the methodology for creating alternate teaching strategies and for recording learning in an electronic book environment is discussed.

2. Background

2.1 Review of Intelligent Tutoring Systems (ITS)

Computers have been used in education for almost thirty years. There has been considerable interest in using the computer as a private tutor to provide individualised, dedicated teaching. Sokolnicki (1991) comprehensively reviews ITS and concludes that an intelligent system must be a flexible system that can adapt its teaching rules to the individual's performance. Mandle and Lesgold (1988) defined an ITS by comparing the student's knowledge of the domain to that of the domain expert: an ITS reduces the difference between the competence of the learner and the expert by the application of various teaching strategies.

Since the late eighties, the barriers to building ITS have been reduced by the development of a cognitive theory of learning (ACT*) and the wider availability in the community of computers with adequate speed and memory to run an ITS. However, few commercial systems exist today. Those that are in use, e.g. The LISP Tutor, took considerable resources to build the knowledge base. (Each hour of instruction cost 200 hours work.) Attempts to use existing expert systems to provide the domain knowledge as in developing a microbiological identification tutor, GUIDON, based on MYCIN, lost the explaining ability which was compiled in the expert system (Clancey, 1987). In the nineties we are seeing the development of domain independent (generic) tutors but these have not been easy to use or accepted by teachers.

Our system, RAPITS (Rapid Prototyping Adaptive Intelligent Tutoring System) is based on such a generic system, COCA-1, developed by Nigel Major. We are particularly interested in developing a practically useful ITS in a non-specific domain that can teach intelligently, i.e., adapt its style of tutoring to the student and the topic. Currently we are looking at using such a system to teach programming Pascal loop structures, but will also consider using it to teach in other domains, e.g. electronic communication. Although we find the approach of Major in COCA-1 very reasonable, its practical implementation is not adequate in a text-only based environment as the author is well aware, as he is also moving to a multimedia environment.

2.2 Review of teaching programming

PROUST (Spohrer and Soloway, 1985) and LispTutor (Anderson and Skwarecki, 1986) both involved students entering code to solve a given problem. PROUST attempted to estimate the students' plan intentions, while LispTutor guided their left-to-right, top-down attempts in a highly directed fashion by interpreting their code as a correct or a buggy solution and advising when errors occurred. However, this methodology is very specific to the programming domain. PROUST's inference cannot readily be generalised. LispTutor's effectiveness depends on an extensive library of bugs. Boyle et al's(1994) learning in context approach, where small programming tasks can be viewed as part of a complete program and the acquired skills are detailed to the user, focuses on learning from examples and refinement. Can we find other teaching strategies that teach programming skills to novices almost as well as PROUST and LispTutor, but which can also be used in domains other than teaching programming? We aim to make a generic tutor with a range of alternative teaching styles, available eventually to a range of domains, but we will use the extensive literature on teaching programming to compare the effectiveness of our system with that of specialised programming tutors.

3. Creating the Domain

We have been developing a generic intelligent tutoring system based on an electronic book using Asymetrix ToolBook (ToolBook is a trademark of Asymetrix Corporation), that allows rapid prototyping and use of various teaching strategies as alternate paths through the book. One way to speed up and simplify the domain creation is to use the 'electronic book' as a front end to display the tutoring material. This can also be developed to include multimedia material to make the domain much more interesting to the student. The various teaching strategies can then be created as alternate paths through the book, rather than the 'dumb book' sequentially displayed, perhaps with some 'hotwords', or indexed look-up of topics. Alem and Lee (1993), Meyerowitz (1995) and Boyle et al (1994) have demonstrated practical success in creating electronic books as the basis of tutoring systems using ToolBook and Guide authoring packages in a Microsoft Windows environment and Orey , Trent and Young (1993) have shown that it is cheaper than creating a system in C.

4. Teaching Strategies

Research in the domain of teaching programming has suggested a variety of models. There is no one best way, and so a range of teaching strategies is needed.

The generic tutoring approach, used by Major and Reichgelt (1991), involves teaching a rule, demonstrating an example, then testing the student with a similar example. Although COCA's teaching seems to be system directed, the student may have the option to choose her own path. Our system gives the student the opportunity to exit or to choose another topic from the menu, or elucidate a 'hotword' in the text, updating the history accordingly.

Another problem with the COCA system is that the student rating is used to guide the meta-strategy, rather than using the finer tool of the complete student history. This delays feedback and appears to make the response less sensitive. However, COCA is itself much more sensitive than DOMINIE (Spensley et al, 1990) which could only change from one overall teaching style to another (Major,1993).

We aim to build an adaptive system in the style of Benyon and Murray (1993) which can compare student and domain models, and can automatically change its teaching style.

Humans also learn readily by analogy, so tutoring systems can ‘fast-track’ through teaching a topic by comparing a task to one with which the student is already familiar. The basic style of tutoring is teaching by analogy in that the student has to solve a problem similar to that demonstrated. We can take this a step further, analogous to human teaching as the student learns the initial concepts, we can teach subsequent concepts built on these. Even this approach is controversial as several authors including Soloway, have shown that students are likely to prefer a particular looping construct, often one they were first taught. So too Escott and McCalla (1988) who show that some students misunderstand analogies and their use can cause even more errors. However, a system will only approach a human tutor’s effectiveness if it responds to individual students intelligently, i.e., it has a range of tutoring strategies that it applies according to the student’s response (Spensley et al,1990).

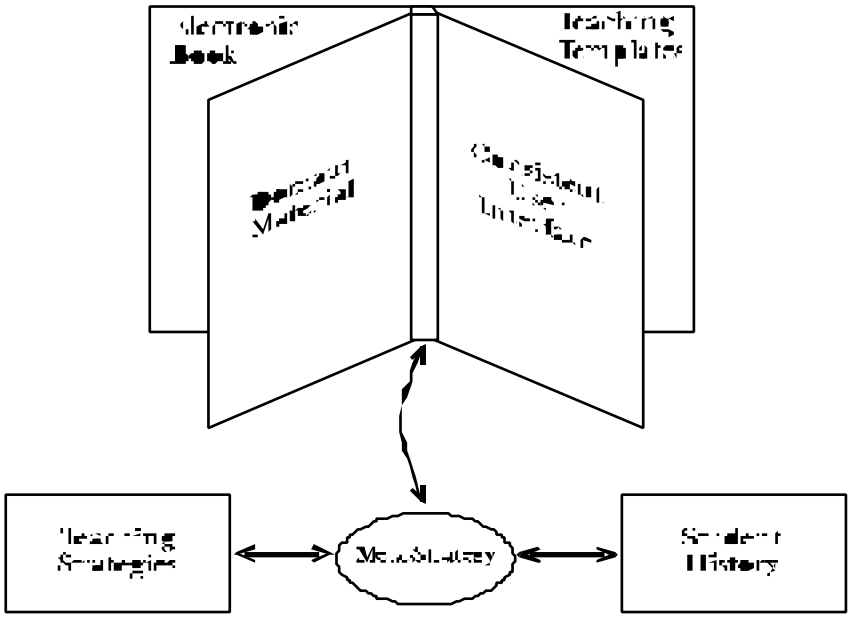


Figure 1. The components of an ITS interfaced via an electronic book

5. Components of our ITS

We believe that the use of an electronic book to create the domain material will also provide a friendly interface to the student in a consistent instructional environment (Figure 1). The teaching primitives are built into the way the domain knowledge is created. Although earlier systems tried to keep domain knowledge and teaching strategy independent in order to maximise re-use of the tutorial system in other domains, they found this didn’t teach very well (Murray and Woolf,1992) and Major(1993). Using the electronic book the cost of creating domain material in a tutoring setting can readily be minimised by using template pages for various styles of teaching.

Another problem, however, is how to index the knowledge so it can readily be accessed by the teaching strategy and can update the student model. Schank (1994) also sees the indexing of knowledge representation as a problem in his discussion of case based teaching by examples. For now, a simple first approach using ‘page numbers’ in the ‘book’ will provide a rapid way of locating and recording relevant knowledge and has adequate grain size, provided that each concept or example is located on a new page, although it may be better to reduce the granularity

of the index to an object on the page, if, for example, the page contains a rule, a demonstration and an example.

Grain size has been a problem in COCA-1, where several examples must be completed before feedback is given, so feedback is not immediate (ACT*) and errors are not so clearly identified. Major (personal communication) has also suggested that if an electronic book is used to enter domain material it should have each alternate teaching task on a new page.

Major and Reichgelt (1991) use a strategy interpreter to drive alternate teaching strategies from the meta-strategy. It interprets the student history and decides when to apply which teaching strategy, and what to teach next. In RAPITS the student history is simplified to a vector of page numbers, rather than a list of concepts, but these can readily be linked back to concepts via the book indices.

So, to create an ITS in an electronic book environment the author needs to:

1. Create an electronic book with each teaching primitive as an object on the page.

2. Generate the teaching strategies for the topic.

3. Create the meta-strategy as to what teaching strategy should be used for what student history.

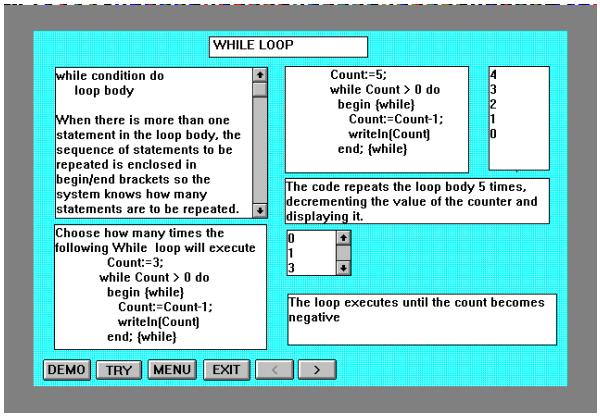
6. Implementation and Comparison of Teaching Strategies in an Electronic Book.

6.1 The prototype

In RAPITS (Rapid Prototyping Adaptive Intelligent Tutoring System), our basic teaching ‘page’ is based on:

- display the rule being taught,
- display the code of a demonstration example,
- demonstrate execution of the example,
- display the code of a problem for the student to solve,
- allow the student to enter a response,
- explain the point of the examples.

The general screen layout has the form shown in figure 2. An example of a teaching screen is shown in figure 3.



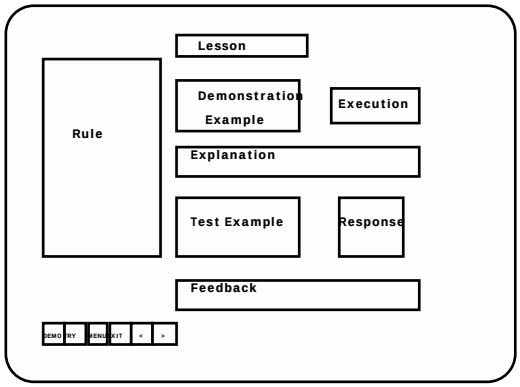


Figure 2: The Screen layout Figure 3: Example of a Teaching Screen

Lesson material is stored as Microsoft Word for Windows files, one file per electronic book page, bookmarked for each object on the page. The list of files needed for the book is exchanged from Microsoft Excel, with both applications linking to ToolBook via the DDE (Dynamic Data Exchange protocol).

In our system, the student will not enter code, but will enter the expected results when a block of code is executed, in the style of COCA, which allows a range of assessments, including multiple choice tests, fill in the blank, test by categorisation. However, we would like to incorporate Boyle's approach of putting code into context, so we would allow the student to view the code as part of a complete program which could be executed from the DEMO button.

The student history is stored initially as an array, and on exit as a spreadsheet table which enables the student to return to the position she left the book when next she accesses the system. The alternate paths are set up in another spreadsheet table, the meta-control table, pointing the student to the appropriate page according to her success so far.

In the prototype, the rules for the meta-strategy are written in the language OpenScript (OpenScript is a registered trademark of Asymetrix Corporation, similar to Microsoft VisualBasic), accessing the meta-control table and the student history tables in Microsoft Excel via the DDE. However, it is envisaged that these will subsequently be built in a Microsoft Windows oriented expert system with authoring facility, similar to the use of M.4 (M.4 is the registered trademark of Cimflex Tecknowledge Corporation) by Alem and Lee (1993) in TOITS, to make a system that can be used by teachers to enter teaching meta-strategies.

7. Evaluation of RAPITS

This project is currently at a stage where we are refining the prototype 'loop-tutor'. The need for such a system arose from the diversity of our student population, some of whom prefer self-paced learning. We plan to trial our prototype initially with a pilot study in the second half of this year.

One advantage of building the system in the Microsoft Windows environment with the authoring package ToolBook is that it forms the free distribution vehicle by which the 'electronic book' can be made available to students to use on their off-campus machines. This is very important, since at least thirty percent of our first year students have their own personal computer, but few can afford expensive software and many live a considerable distance from campus.

Next year we plan to try the tutor in another domain, teaching parts of Information Systems, particularly electronic communication interacting with other information systems from 'the book'. So students can experience 'hands-on' an information system, as well as having access to conventional rules, demonstrations and tests.

8. Summary and Future Work

This paper has described the development of a prototype generic intelligent tutoring system, using 4GL in the Microsoft Windows environment. It's difference from existing systems lies in its ability to build the components rapidly, using commonly used software such as the spreadsheet and the word processor, into an intelligent electronic book. One of the tedious tasks in creating domain material is the transfer of teaching material. Today, text is usually available in an electronic form, derived from a wordprocessed source and most teachers can use a word processor. Our system, RAPITS uses the Microsoft Windows DDE to facilitate this transfer of material to an electronic book. This is necessary at this stage since authoring systems are not yet able to import wordprocessed files, conserving layout and graphic material.

We have at this stage a very simple interface, aiming to be consistent and easy to use, but we expect to refine it following evaluation of the pilot study. A gauge as displayed in Corbett Anderson and O'Brien (1993) to illustrate students progress with the task seems not too distractive, but is able to reassure progress and may enhance learning.

Although the creation of the meta-strategy needs only a table in a spreadsheet, we believe there is still work to be done in making a user friendly authoring system for meta-strategy entry and editing. A graphical interface to map alternate tutoring paths would clarify teaching strategy.

So our system, RAPITS enables teachers, who can use a word processor and spreadsheet, to build an intelligent tutoring system rapidly and experiment with different teaching paths, without having to program in Prolog or Lisp.

9. References

Alem, L. and Lee, M. (1993). A task oriented intelligent tutoring system. In *Proceedings of the First International Two-Stream Conference on Artificial Neural networks and Expert Systems,* Dunedin, pp. 196-200.

Anderson, J. and Skwarecki, E. (1986). the automated tutoring of introductory computer programming. *Communications of the ACM,* Vol. 29, pp. 842-849.

Benyon, D. and Murray, D. (1993). Adaptive systems: From intelligent tutoring to autonomous agents. *Knowledge Based Systems,* Vol. 6, pp. 197-219.

Boyle, T., Gray, J., Wendl, B. and Davies, M. (1994). Taking the plunge with CLEM: The design and evaluation of a large scale CAL system. *Computers Educ* Vol. 22, pp. 19-26.

Clancey, W. J. (1987). Methodology for building an intelligent tutoring system. In G. Kearsley (Ed.), *Artificial Intelligence and Instructions - Applications and Methods,* Addison-Wesley.

Corbett, A. T., Anderson, J. R. and O'Brien, A. T. (1993). The predictive validity of student modelling in the ACT programming tutor. *Proceedings of AI-ED 93, Edinburgh* , pp. 457-464.

Escott, J. A. and McCalla, G. I. (1988). In *Proceedings of ITS-88,* Montreal, pp. 312-319.

Major, N. (1993). Restructuring teaching strategies with COCA. *Proceedings of AI-ED 93*, Edinburgh, pp. 66-73.

Major, N. and Reichgelt, H. (1991). Using COCA to build an intelligent tutoring system in simple algebra. *Intelligent Tutoring Media* Vol. 2, No. 3 / 4, pp. 159-169.

Mandle, H. and Lesgold, A. (1988). In *Learning issues for intelligent tutoring systems*, Springer-Verlag.

Meyerowitz, J. (1995). Experiences with PasTIL: an interactive Pascal tutor. In P. M. Alexander (Ed.), *Proceedings: Computer-Assisted Education and Training in Developing Countries*, pp. 67-172.

Murray, T. and Woolf, B. (1992). Results of encoding knowledge with tutor construction tools. *Proceedings of the 10th National Conference on A. I.*, MIT press, pp. 17-23.

Orey, M., Trent, A. and Young, J. (1993). Development efficiency and effectiveness of alternative platforms for intelligent tutoring. *Proceedings of AI-ED 93,* Edinburgh, pp. 42-49.

Schank, R. C. (1994). Tractor factories and research in software design, *Communications of ACM,* Vol. 37, pp. 19-21.

Soloway, E., Bonar J., and Ehrlich, K. (1983). Cognitive Strategies and Looping Constructs: an Empirical Study. *Communications of the ACM ,* Vol. 26, No. 11, pp. 853-860.

Sokolnicki, T. (1991). Towards knowledge based tutors: a survey and appraisal of Intelligent Tutoring Systems. *The Knowledge Engineering Review*, Vol. 6, pp. 59-95.

Spensley, F., Elsom-Cook, M., Byerley, P., Federici, M. and Scaroni, C. (1990). Using multiple teaching strategies in an ITS. In C. Frasson and G. Gauthier (Eds.), *Intelligent tutoring systems: At the crossroads of AI and education*. Norwood N.J. Ablex.

Spohrer, J. C. and Soloway, E. (1985). Putting it all together is hard for novice programmers. In *IEEE Proceedings of the International Conference on Cybernetics and Society, Tucson*, pp. 728-735.