Int. J. Engng Ed. Vol. 12, No. 5, p. 365–372, 1996
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1996 TEMPUS Publications.

X Windows on the World: A Systems Approach to Designing CBL Tutorials

RUSSELL KEENAN
KEVIN FORWARD
Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, Australia
CARMEL McNAUGHT
Academic Development Unit, La Trobe University, Victoria 3083, Australia

This paper describes a system which uses an X Windows environment to provide a system which can be used to develop tutorials which use computer-based learning (CBL) The system can be used to development tutorials which can provide tuition to, and assessment for, small groups of students. Hence it provides a means of overcoming some of the problems associated with the provision of small group tuition and the marking of assignments. When these tutorials are developed, in addition to relieving tutors from some of their duties we explain how they provide an opportunity for CBL to provide an improvement in the quality of teaching. In highly technical subjects computer-based tools are available which can be used by the students during problem solving sessions. Simple authoring systems do not provide access to such tools. Here we report upon more sophisticated authoring systems running on a multitasking operating system such as Unix which permit the tutorial to concurrently access tools running in other windows. Example tutorials are described which illustrate how the development system is used. The need for evaluation techniques which can be used to determine the effectiveness and student reaction to this style of CBL are also discussed.

EDUCATIONAL FRAMEWORK

Context of this project

THE DEPARTMENT of Electrical and Electronic Engineering at the University of Melbourne has approximately 400 students in years 2–4 of the degree. All students have about two tutorials a week throughout this time. At second year level, the traditional tutoring system has operated with one tutor (often a higher degree student) and a class size of about 30 students. In third and fourth years, the tutorial sizes are much larger—essentially the same as the lecture group, which can be as high as 140 students; tutors are often members of the lecturing staff. Sessional tutors (higher degree students) undergo a half-day training workshop and have regular interaction with the lecturing staff. Little marking of student problem sheets is done because of financial constraints.

The problems of this style of tutorial system are clear:

- Insufficient individual assistance for students, either in tutorials or as written feedback on attempted problems
- Lack of interactive probing question/answer sessions between tutors and students which can enable problems to be explored and extended
- The promulgation of a view that there are set answers to problems which can be learnt as a generalisable strategy
- Limited development of student–student interaction and communication skills.

In order to improve the quality of teaching and learning in the department, increasing use of computer-based learning (CBL) has taken place. In this paper the development of a suitable tutorial environment is described.

Theoretical framework

The justification for developing a suite of computer-based tutorials, whilst often couched in terms of the capabilities of computers, is actually underpinned by theories of learning. This is because the choice of design features seen as necessary for effective learning are determined by whatever understanding of learning is held. Broadly speaking, ideas in keeping with the constructivist model of learning have been used as a basic philosophy in this project. These ideas are that the tutorials might be used to ensure that each student interacts with the course material, thereby integrating new knowledge and developing problem solving skills.

Constructivism is a theoretical framework that has been extensively explored in recent literature. Current constructivist views of learning see individual learners as building up their own internal conceptual maps or models as a result of interactive processes between each learner and her/his environment. Learning outcomes are dependent not only on the environment but also on the state of the learner with her/his existing conceptions and motivations. Learning occurs as an

* Accepted 15 July 1996.

active construction of meaning as a result of refection on experiences (e.g. [1]).

'Reflection' is one of those concepts embodied in educational theory which deserves to be reflected upon. It does not simply mean thinking over an experience, but implies a conscious integration of experience into an existing model. Our mental frameworks are influenced by our sociocultural environment as well as our physical environment.

Modelling often involves the construction of mental images, of visual metaphors. Language metaphors are an integral part of our way of understanding the world [2]. The concept of visual metaphors—graphical or visual ways in which links are made between existing and new knowledge—can also trigger the production of useful questions which enable the learner to progressively explore, predict, test and refine models. In CBL, extensive use is made of visual metaphors; the means of ensuring students interact with their course material has traditionally been done through visual and verbal analogies, exercises, such as performing calculations and plotting graphs, and answering questions related to and extending material which has been presented in lectures.

It is this framework of progressively developing models through a process of metaphorical association, asking questions and testing ideas that has driven the development of the CBL material described below. The process of constantly referring to students' interpretations of the material is an integral part of the development of each package.

This approach to educational design is described well by Laurillard [3]. She outlines a framework for analysing educational media in terms of opportunities for:

- discussion and negotiation between the learner and the teacher;
- interaction between the learner and the educational context designed by the teacher;
- adaptation by the learning of the material provided in order to construct his/her personal understanding;
- reflection on the learner's performance by the teacher and the learner.

Educational goals

The goal of producing these tutorials has been to improve the quality of education of electrical engineering undergraduates. The second and third year tutorials were identified as the best area for such work to focus on.

The primary educational objective of this project is to deliver tutorial material to students on an individual basis, using relatively new technology to develop a suitable CBL system. The second objective, which is a consequence of the first, is to free up staff to run discussions and practice classes for small groups, where material which requires intense tuition, but which is not amenable to CBL, can be effectively dealt with.

The most suitable way of improving the department's tutorials seemed to be the development of a suite of computer programs, comprising a shell, specific tutorial modules and general tools. It was felt that this approach would utilize many of the department's strengths, especially as staff and students already had some familiarity and expertise with the department's large and growing computer network infrastructure. The goals of such a suite of programs would be to:

- Aid students' learning of the subject matter.
- Develop students' problem solving abilities.
- Develop students' skills in using appropriate computer tools.
- Cater for individual student's approaches to learning.
- Cater for individual student's rate of learning.
- Foster interactive learning.
- Increase the relevance of the tutorials by incorporating practical engineering problems.
- Provide feedback to both students and staff about how much students are learning.
- Provide evidence about the teaching effectiveness of the course.

It is hoped that these tutorial modules will aid students both in the learning of subject matter and about the kinds of tools used to solve the relevant problems. This could give a greater understanding of the computer resources available for the rest of the students' studies. Many of these tools would also be used by students later in their professional lives and this will better equip students for the workplace. This, however, does create a tension in the planning and design of tutorials, as time constraints make it difficult to cover both aspects well. To resolve this, specifying goals for each tutorial is essential.

Another advantage of a computer-based approach was thought to be that students might be able to take their tutorials at the times that suit them. This would allow for greater flexibility for students, giving them more control over the time, place and pace of their tutorials.

One great advantage of computer-based tutorials is the possibility of making problems interactive, especially those requiring much calculation. Such interaction tutorials allow students to actively explore models. By changing parameters and seeing results instantaneously, students are able to develop a better feel for what are important factors in a model, and what are incidental. Interactive tutorials might then be used to help bridge the gap between understanding of theory and practice [4–6].

There is also the possibility for the student of being able to focus more upon the actual material being presented. Since simple, repetitive calculations can be performed by an online tool, and graphs can be drawn automatically by any number of drawing or maths packages, doing many of these sorts of exercises manually is no longer relevant in a computer-based engineering environment.

Problems can also be implemented with some form of random factor so that the numerical context changes. This would prevent the most basic forms of cheating in any assessment. It also allows problems to be attempted more than once for additional learning reinforcement.

To ensure these other goals are fulfilled, adequate feedback is vital. This feedback is required on many levels; the performance of students, their perception of course and tutorial presentation and the relevance of the tutorials to students' needs. Feedback from teaching staff is also vital, as it ensures that the goals of the tutorial correspond to those in the curriculum.

Therefore it has been a goal of preparing these tutorials to rethink the kinds of exercises that students are required to do. This has required clear thinking on the part of the relevant lecturers about the educational goals of each tutorial, and how they might best be achieved. This has brought about a fresh focus on student learning in some of the subjects for which tutorials are being prepared.

THE TUTORIAL ENVIRONMENT

The tutorial environment provides a consistent way of interacting with the computer. It exists within the existing software base and adds new features.

Brief X Windows outline

X Windows is an extremely flexible graphical environment that is standard on Unix platforms and is available within many other operating systems. Since X Windows runs with Unix, it allows many applications to run simultaneously and seamlessly in a networked environment. This flexibility is available at the user and tutorial designer levels. It allows the design of tutorials to be quite modular and self-contained.

The issue of security is important within such a tutorial environment. Users of the X Windows system have a lot of control over their operating environment, making security difficult where user's freedom is not to be too limited. In order to prevent doctoring of results, some level of security is needed to prevent access to those areas of the system where results and attendances are logged in some form. The tutorial software must also protect students' privacy in the results stored and the confidentiality of responses to questionnaires.

Outline of tools

There are many applications available within the X Windows Unix environment. They range from quite general-purpose tools, such as Matlab (a maths package), through circuit simulators, such as Spice, to quite specific applications like Mosaic (a hypertext viewer).

The X Windows environment is also quite open to the development of custom applications. We have developed these where available tools were too general, too difficult to use or simply not in existence yet. So far two custom applications have been written—a neural network simulator and a step response (pole-zero) simulator. These tools have been able to be integrated within the tutorial framework despite being written in various languages and within different design philosophies and having quite different goals.

TUTORIAL SYSTEM CONSTRAINTS/ REQUIREMENTS

Importance of visual design

The aspect of visual design, though originally seen as slightly peripheral to the overall success of the project, has now assumed a much more central role. Driving this is the realization of the need to focus development of the tutorial module system on the way in which students and lecturers, though particularly students, will interact with the software. While it may be that engineering students have experience of coping with poorly designed interfaces and some ability to figure out arcane command sets, the time spent doing this can be boring, discouraging and wasteful of time better spent doing the tutorials themselves.

Visual cues have been used where possible to focus attention upon relevant information. There is a tension between wanting visual variety to keep students' interest, and the consistency required for ease of use. Visual design also addresses the difficult in working out exactly the right amount of information to present to students, so that all relevant information is able to be found. In most cases it is not possible or desirable to present all the relevant information on the screen at one time. Instead information should be available at a level of effort proportional to the importance of that information, i.e. varying from on-screen to nested deep within menus. It is undesirable to make the path to information too obscure, or else it might never be found [7].

Administration of the tutorial system

Planning the administration of the tutorial system has involved addressing two main criteria. The first is that the system should ease administration of tutorials, by doing such things as reporting the progress of students, both collectively and individually. The second is that the system be as self-contained as possible, so that the need for supervision may be kept to a minimum for the running of the tutorials. Both these issues may be addressed by working towards making the tutorial software behave as intelligently as possible [8].

In order to ease the administration of tutorials, such features as automatic marking and reporting of students progress seem to be desirable. Feedback forms can be tedious to collate by hand, but this is easily done online. The time taken to answer questions, and the various navigation paths students take through the tutorials may also be used to improve the tutorials and identify areas in the course with which students are having difficulty. The possibility is thus open for computer-based tutorials to be more responsive, and better suited to the needs of students.

Automatic correction, recording of results and/or answers and reporting of students' progress ease the administration load. This necessitates the requirement that the tutorial shell either be intelligent or be able to be manipulated by intelligent general purpose programs. Intelligent systems tend also to be more flexible, aiding responsiveness. In order to progress this aspect of this project we have developed an equation solving program which is used in conjunction with a question presentation program and a full graphics editor to provide tutorials on topics such as network analysis and electronic circuits.

Features such as online, context-sensitive help and consistency in the user interface reduce the need for supervision. By reducing the need for staff to provide the help necessary to those using the tutorial system some of the burden on system administrators is relieved. This allows that supervision that does occur to be based more on the course material. Such help screens may be implemented using hypertext style links on keywords and diagrams, with the ability to call up further help if that is required.

Finally, the tutorials need to be accessible by the students. Planning for this raises further questions as to the desirability and/or need of supervision in the tutorial computer labs. The availability of other computer labs on campus and their supervision is an issue that needs to be resolved.

TUTORIAL IMPLEMENTATION

Tutorial system

Figure 1 shows a diagram of the tutorial system. The lines of interaction required to create and use a tutorial are shown as full lines. From this diagram it can be seen the person writing a tutorial does so through the use of Windows 4GL and Mosaic. These are the shell or glue programs which tie the tutorials together. The variation possible in navigation through tutorials places some demanding requirements upon these shell or glue programs. Such a program must be able to interact with others, be able to call other tools and applications transparently to the user.

The program must also allow for multiple approaches to a particular problem. It must be able to cater for students who are new to the X Windows environment and present an integrated environment to them. It must also allow students with some experience to explore the environment and let them interact with the utilities and specialized modules at a more advanced level. These programs are covered in more detail below.

Another requirement of the tutorial system is that it provides automatic assessment of the students performance. In Fig. 1 the activities of the Intelligent Tutorial Software are shown as dashed arrows. From a database of student activity this system assess the student performance in a number of possible different ways. It can, for example, determine how far through the tutorial the student has progressed, how many hints of what level they used and the time taken to perform tasks assigned by the tutorial.

The authoring system

When looking for a suitable program to use as a shell to glue together the tutorial modules and utilities, a number of features were quickly identified as being desirable or necessary. They were:

- The ability to incorporate text and graphics within a unified framework.
- The ability to display equations.
- Hypertext capabilities.
- The ability to call outside programs.
- The need for the interface to have some form of intelligence or programmability, to allow for automatic correction.
- Some form of tool to author and layout windows and screens.
- Easy administration of the system wherever possible.

The initial systems reviewed were specifically designed hypertext viewers. Of these, a large number were still in the vapourware stage. The others were generally unable to function interactively with other programs. Existing shell programs were able to function fully in the multitasking X Windows environment, but did not possess good authoring tools and did not come with pre-existing hypertext facilities, making it necessary to write everything from scratch.

Ingress Windows 4GL has proved itself a suitable tool to use as a tutorial shell, though it was originally designed as a database front-end. It possesses an authoring tool, has a general purpose scripting language and is able to call both C functions and exterior programs. It is also able to display text and graphics in various formats. Hypertext functionality is able to be programmed in using transparent buttons. Similarly, equations are able to be typeset using other tools, and are able to be incorporated seamlessly into the text as a piece of graphics.

Other benefits gained by using a database front-end are security and the ability to record and store any information required, in a form amenable to further processing (such as for marking or evaluation). The security features are standard requirements for databases. They were relatively well tested and so did not require any development

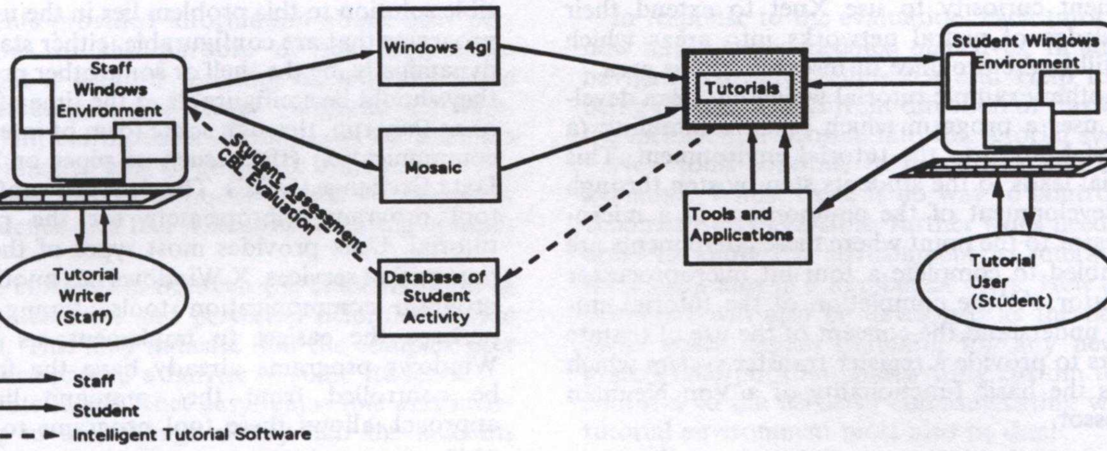
Staff Windows Environment
Windows 4gl
Mosaic
Student Assessment CBL Evaluation
Tutorial Writer (Staff)
Database of Student Activity
Tutorials
Tools and Applications
Student Windows Environment
Tutorial User (Student)
Staff
Student
Intelligent Tutorial Software

Fig. 1.

effort on our part. The ability to store data easily and securely were not originally identified as being of high priority. However, with hindsight it seems that these are vital to the system we are developing.

Reuse continues to gain prominence within the software industry. It saves time in code writing and debugging. Within the environment of tutorial development it gives the additional benefit of consistency in the user interface. An example of this is in the use of Mosaic—another package which can be used to perform some similar functions to Windows 4GL. At present it is used for the online help system. Not only does the use of Mosaic give a consistent interface to the help information, it also exposes students to Mosaic, which is gaining rapid acceptance in the user community as a hypertext browser for the Internet. Mosaic has been developed as a means of navigating the World Wide Web (WWW), a massive, decentralized and growing hypermedia system. The WWW allows the viewing of text, graphics, and files. Sound and animation are also possible (where available). Links may be made between documents stored at the same site, or in another country, without the user needing to worry about the details. A local Mosaic server has the potential of providing a consistent, easily used and extended hypertext help system.

In a tutorial, students therefore have access to a large number of utilities and programming tools that they might use to solve any particular problem. A tutorial thus becomes a braided thread of execution through the students' desktop environment. A typical session may vary from just stepping through some information pages to navigating through a host of text, utilities, graphs and editors.

Custom tools developed

Several custom tools have been used as part of tutorials in the first year of the project. They are a neural network simulator, a step response simulator, a transformer equivalent circuit and an RLC circuit simulator. All of these were written and used in the tutorials because no other pre-existing tool was available.

The Xnet program is a neural network simulator. It was developed within the department as part of research into the functioning of neural networks, and has been modified for the teaching of a fourth year course. Xnet was designed to provide a general interactive graphical interface to neural networks. The program stores networks and input/output as loadable files, avoiding the time wasted by students writing custom neural network simulators from scratch. It was originally developed for people who already had some familiarity with the X Windows environment. Therefore, it is an intricate tool, allowing the graphic adjustment of most of the parameters of a neural network.

The step response program was developed specifically for teaching second year control theory. It was designed to highlight the correspondence between the common graphical representation, abstract description and real world behaviour of a continuous control system. Students are able to interactively adjust the amount of feedback in the control system and observe the effect of this. The plant and controller may also be changed, and the results of this seen immediately.

The interface of the step response simulator was consciously kept simple. This was done to keep attention focused firmly on the interaction of the control parameters involved, rather than on learning to use the system. A graphic designer was consulted for discussion on layout and colour to increase the clarity of the user interface.

Example tutorials

As an example of the use of these tools a tutorial has been developed which enables students to explore the training application of neural networks. An introductory tutorial is designed to teach the students how to use the Xnet tool and then subsequent tutorials encourage the students to use the Xnet tool to explore the training and operation of neural networks. The aim of the tutorials is to encourage the students to develop sufficient curiosity to use Xnet to extend their knowledge of neural networks into areas which are still in the province of research in this area.

Another example tutorial which has been developed uses a program which links in Simulink (a part of Matab) to the tutorial environment. This tutorial leads to the students step by step through the development of the components of a microprocessor to the point where these components are assembled to complete a four-bit microprocessor simulator. At the completion of the tutorial students understand the concept of the use of tristate buffers to provide a register transfer system which shows the basic functionality of a Von Neuman processor.

FUTURE DIRECTIONS

As the work has progressed it has become clear to us that it is important to develop and set standards for the development tools and environment for CAL packages [9].

We have been most impressed by the work of Thomas *et al.* [10] who have developed INTERACT. This is a project that uses Mosaic as a front end to a group of interacting programs. Mosaic forms the hypertext glue to link a number of other programs together. Very small modifications to Mosaic have made two way communications possible, and a C++ toolkit library exists to aid the development of programs that communicate together. There are a number of sites in the UK already developing CAL packages that use this environment.

REFLECTIONS

Integration issues faced

While the aim of the project has always been to develop a suite of programs, comprising a shell, specific tutorial modules and general tools, the majority of development time spent in the first year has been directed toward the production of custom tools. The initial reasoning for this was both to gain some initial momentum and to try to avoid becoming locked into an appropriate design. Because this was done before the tutorial shell had been put in place, an attempt was made to make these programs relatively stand-alone. This proved difficult, due to the large number of features required for such a program. The experience gained through this approach has reinforced awareness that the programs used in a tutorial setting need to be closely integrated set, reusing wherever possible.

One way in which close integration was perceived to be desirable was in the initial loading of programs into the student's work space. Some of the earliest tools used within trial tutorials were quite difficult for students to configure, even with handouts and instructions on whiteboards. A possible solution to this problem lies in the use of tool programs that are configurable, either statically or dynamically, by the shell or some other program—they should be configurable at the time of loading, or as they run, through some form of inter process communication (IPC), such as pipes or Dynamic Data Exchange (DDE). The shell can configure the tool program appropriately for the particular tutorial. Unix provides most types of these communication services. X Windows has another set of program communication tools. Piping data is perhaps the easiest to implement, as many X Windows programs already have the facility to be controlled from the command line. This approach allows these tool programs to be used as-is.

Another issue becomes apparent when communication between processes is considered. This is how visible such communication should be. Inexperienced users may be uncomfortable or confused watching or participating in low-level program manipulation. However, experienced users might like the extra control available from a more open system. Again the question arises as to whether a particular tutorial is designed to teach the tool or the subject matter.

Integration highlights certain aspects of visual design. The environment needs to be integrated both at the level of user interaction, and at the level of IPC. This requires the consistent use of symbols and graphical devices to display and navigation information. Otherwise the programs may be too enigmatic for student use.

Evaluation of the experience thus far

In educational evaluation it is essential to seek evidence of success by several means; the validity of any approach comes from the coherence between the interpretations of several measurements [11]. In this project, four main mechanisms were used to evaluate the success of the work being done:

The use of a balanced reference group. The reference group members were electrical engineers, programmers, computer systems experts, and an educational consultant. The constant dialogue between these professionals, both at an informal and formal level, ensured that the final decisions being reached considered engineering content, hardware and software factors and educational considerations. The inclusion of a graphic designer enhanced this team.

Interviews with students. Students' perspectives and interpretations were constantly sought throughout development. The formative evaluation enabled the decisions of the reference group to be fine tuned.

Questionnaire given to students. A formal summative evaluation was done on the Xnet custom tool with 15 students. A summary of the results is:

- Students provided information about their perceptions of the purpose of the tool; some thought it was for teaching content and some for training in the use of computer software. Constant clarification of objectives for students is an integral part of any CBL system.
- Students have acceptable levels of computer confidence and feel relaxed when using computers.
- Some correlation between the students' level of computer skills and perceived helpfulness were noted. This may indicate that the complex user interface proved a barrier to some students.
- Students found Xnet easy, enjoyable and interesting to use, though nearly half the students were unsure about how useful it was.
- Students did not use the 'Help' information and this may account for some students not seeing the full potential of the tool. The one student who admitted to using 'Help' claimed to have done so by accident. This incomplete use of a CBL tutorial system is a common problem [12] and overcoming this will be part of future work.
- All students were satisfied with the screen design and layout and found it intuitive to use; this validates the increasing emphasis being placed on visual design.

Discussions with all staff members in the Department of Electrical and Electronic Engineering. The development of the CBL system and the assurance of its use in a wide variety of contexts has been enhanced by the outreach made by project members to other members of the department. The support and interest shown by colleagues has been a very positive part of the project.

Further work

Work continues in several main areas. The first is the continuing design of the tutorial environment itself. The visual design of the environment needs constant revision as more functionality is built into the tutorial system and any shortcomings in the current design become apparent. The programs that comprise the tutorial environment are also being upgraded fairly regularly, and with that come new design possibilities and new ways of interacting with the shell. The planning for such change will also become an issue as the tutorial environment stabilises.

In response to the evaluation, each tutorial will now have clearly specified objectives. In addition, navigation towards online help and 'Help' itself are being redone. The current direction of this work is the inclusion of Mosaic into the tutorial system.

New tools continue to be written and become available. Whilst there is no way to control what becomes freely available, further work needs to be done to allow the environment to interact with these programs in a number of ways. New custom programs will also be developed as the need for them arises. Again planning for any new program's incorporation and writing custom tools to conform to the needs of communicating with the tutorial environment must also be done.

As the project shifts toward developing a seamless tutorial environment, previously written programs will need further modifications to allow them to fully interact with the rest of the tutorial system. These programs must also be updated to conform to any new standards that are set, to try to prevent them being orphaned.

As we have stated above, we are keen to encourage the development and setting of standards for CAL tutorials and see the INTERACT project as a step in the right direction towards achieving this goal.

Finally, and most importantly, is the continued work with the academic staff postgraduate tutors and undergraduate students. The academic staff are vital in the development of new teaching material. The goals, possibilities and limitations of the project need to be communicated in order to ensure continued feedback, new ideas and teaching material. Communication is necessary to ensure an accurate understanding of what is possible, so that realistic expectations are maintained. Student feedback will play a more prominent role as more of the tutorials come online. Their use of the tutorial environment will provide the impetus for much of the change as the system matures.

Acknowledgements—This project has been made possible by the financial support of a grant from the Committee for the Advancement of University Teaching and subsequent financial backing University of Melbourne. The authors of this paper acknowledge the contribution of Catherine Gleeson of the CHSE Graphic Information Design Dept. to the design of the project's graphical interfaces. Finally, we acknowledge the staff members of Electrical Engineering who have helped through their support and ideas, especially Dr M. Palaniswami and Dr C. Zhang.

REFERENCES

1. R. Driver and B. J. Bell, Students' thinking and the learning of science: a constructivist view, *School sci. Rev.*, **64**, 443–456 (1986).
2. C. Sutton, Metaphorical imagery: a means of coping with complex and unfamiliar information in science, *Durham and Newcastle Res. Rev.*, **11**, 216–222 (1981).
3. D. Laurillard, *Rethinking University Teaching: A Framework for the Effective Use of Educational Technology*, Routledge, London (1993).
4. H. Diab and I. Demashkieh, A Computer-aided teaching package for microprocessor systems education, *IEEE Trans. Ed.*, **34**, 179–183 (1991).
5. A. Read, Computers and computer graphics in the teaching of field phenomena, *IEEE Trans. Ed.*, **33**, 95–103 (1990).

6. F. Stremler, S. Klein, F. Luo and Y. Liao, Numerical solutions and mapping of electrostatic fields using the Apple Macintosh computer, *IEEE Trans. Ed.*, 33, 104–110 (1990).
7. J. Phillips and M. Crock, Interactive screen design principles, in B. Chia, R. Pennell and R. Sims (eds), *ASCILITE '92: A Promised Future*, Sydney, pp. 237–250 (1992).
8. B. Antao, A. Brodersen, J. Bourne and J. Cantwell, Building intelligent tutorial systems for teaching in engineering education, *IEEE Trans. Ed.* 35, 50–56 (1992).
9. R. Keenan and K. Forward, Standards for the CAL Environment, *Proc. Pacific Region Conference on Electrical Engineering Education*, Marysville, Victoria, Australia, pp. 84–87 (1995).
10. R. Thomas, I. Neilson, A. Slater and C. Smeaton, The INTERACT Project—an integrated engineering simulation environment, *Proc. Computer Aided Learning in Education*, University of Sheffield, England (1994).
11. S. Wills and C. McNaught, Evaluating computers in learning in tertiary education, in B. Lo (ed.), *Reaching out with IT. Proceedings of the Australian Society for Computers in Learning in Tertiary Education '93 conference*, Centre for Computing and Mathematics, the University of New England, Northern Rivers, Lismore, pp. 691–703 (1993).
12. R. Day, C. McNaught, J. Tonkin and P. Sluczanowski, Teaching population dynamics at tertiary level, in B. Chia, R. Pennell and R. Sims (eds), *ASCILITE '92: A Promised Future*, Sydney, pp. 53–63 (1992).