Australian Journal of Educational Technology
1997, 13(2), 98-114.

A knowledge-based computer instruction system

Joshua Poh-Onn Fan
Tina Kwai-Lan Mak
Li-Yen Shue
University of Wollongong

The purpose of this paper is to introduce the features of the Knowledge-Based Computer Instruction System (KBCIS), which was designed to assist and enhance the teaching-learning function for basic accounting classes with large number of students. The most important feature of this system is the incorporation of the knowledge-based approach in the development of the system. This approach makes it possible to store problem solution knowledge in the system, and use it to mark problems which are of the same nature but with different settings. This approach also makes it possible to implement the flexible marking scheme for providing partial credit for partially answered questions. More importantly, this approach allows the system to provide advice on wrongly answered questions.

Introduction

The problems with the conventional teaching approach to large classes, which relies mostly on the text books and the questions of the review exercises at the end of each chapter, are well known. Apart from the fact that face-to-face consultation time between students and lecturers can never be sufficient, the amount of time required to mark a large number of assignments has usually prevented a lecturer from providing detailed feedback. In most instances, the detailed comments for each individual question is not provided, instead, an overall remark at the end of an assignment is given; which may not provide any direct assistance to students in problem solving. Very often, for the wrongly answered problems, it is mostly up to the students themselves to find out the correct problem solving procedures. One of the consequences of these is the fact that some students tend to relate solution approaches with particular exercise problems only, and find it difficult to extend the problem analysis logic and solution approach to problems of the same nature but with different settings (Nachouki & Gouarderes, 1994). Another well known problem with large classes is the tendency for some students to plagiarise and hence reduce the learning effectiveness. In the past, the only way to reduce these problems is through providing more teaching assistants, which is generally very costly and has proved ineffective in many cases (Flechsig, 1989; King & McAulay, 1992). Recently, the promotion of the Computer Assisted Instruction (CAI) has targeted these areas.

The major classifications of CAI designs include tutorials, drill and practice, simulations and instructional games (Alessi and Trollip,1985). Each of these designs provide basic method for using the computer to teach, reinforce, practice, or apply information. Most of these CAI systems (Bourne, 1990; MacKnight & Balagopalan, 1989; Pogue, 1985) need to incorporate a large quantity of pre-prepared questions in a database. These questions, when needed, will then be retrieved either through random selection or according to some form of selection index. This approach may prove to be valuable for the kinds of topics, whose contents are universally valid with minimum variation between lecturers. Consequently, it is easy to see the problems one may encounter in applying present CAI in a university environment.

The problems are two-fold. Firstly, it is well known that the development of large pools of questions for a topic is very costly and time-consuming. Secondly, there usually exists a degree of variation between lecturers in terms of emphasis and the depth of a subject. And it is likely that the question developer is not the lecturer himself/herself. This second problem highlights the lack of control by the lecturer over the contents and the nature of questions with the CAI approach (Hannafin & Peck, 1988; Lockard, 1992), which will certainly lead to the mismatch of pre-prepared questions and the contents designed by a lecturer (Nachouki & Gouarderes, 1994). In addition, a common criticism of present CAI is the fact that the marking mechanism of most systems is carried out through matching users' answers with stored answers. Hence, the result for a question can either be correct or wrong, and it is only the final answer that counts. The intermediate steps which are correct can not be awarded with partial credits (Hannafin & Peck, 1988; Lockard, 1992); unless each intermediate step is designed as a separate question.

Another major drawback of most CAI systems is that, it is difficult for most of them to provide meaningful feedback to guide students in problem solving processes (Bourne, 1990; Cook & Kazlauskas, 1993; MacKnight & Balagopalan, 1989; Milheim, 1993; Pogue, 1985). Since the proper feedback, which highlights the parts that are not well understood by students and points out the sources of correct solution approach, is regarded as one of very important elements in achieving maximum effect of learning, the lack of this function was considered as a major deficiency of the traditional CAI.

The development of the Intelligent CAI (ICAI) is an attempt to further advance the capabilities of CAI by applying various techniques in Artificial Intelligent (AI). While the traditional CAI was mainly developed by educational researchers, who tried to solve their practical problems through non-AI techniques, ICAI was initiated by computer scientists, who tried to explore the capability of AI techniques in the process of learning and teaching (Kearsley & Seidel,1985). As a result, the focus of ICAI has been more on the technical aspects of the system. Specifically, ICAI adopts the "leaning-by-doing" as the basic instructional approach (Dewey, 1910; Sleeman & Brown, 1982). In this approach, students are required to engage activity in the instructional process to formulate and test their ideas and witness the consequences resulting from the system's reaction to their behaviour (Brown et al, 1982). A typical ICAI system contains most or all of the following: a domain expert, a teaching expert, a diagnostic expert and a student model. A domain expert provides the knowledge of both procedural and factual that students need to learn. The diagnostic expert uses rules to analyse student responses. The teaching expert determines the strategy for teaching the student based on the current state of the student model.

Mandl and Lesgold (1988) advances the concepts further by emphasising the importance of providing flexible and adaptive user interaction, providing different viewpoints for users to access information, delivery of contents according to users' knowledge and skill level, and providing updated assistance to students according to user's learning progress. With these expectations in mind, we developed the Knowledge-Based Computer Instruction System (KBCIS), which was designed as an effective teaching-learning and examinations tool for large Accounting classes. The design of this system incorporates the knowledge-based system approach, which enables this system to store problem solution knowledge and utilise it for marking problems of the same nature but of different settings, and with a great flexibility. This ability makes it possible to achieve to some degree the above mentioned expectations.

Specifically, this system provides lecturing material, exercise problems, help, hints, and solution procedures, which can be accessed by students according to their ability and the criterion set by the lecturer-in-charge. Based on a representative problem, the system is able to generate similar problems with different settings and parameter values, and mark the answers from students accordingly. Feedback for each problem could be provided if needed. The system can access the student log to monitor the performances of each individual student in a class, and help the lecturer to determine the proficiency status for each topic. This real time analysis also makes it possible for the lecturer to pre-set performance criterion to determine the status of a student, and hence provide appropriate topic and exercise problems.

Current practices and objectives of KBCIS

Currently, the Department of Accountancy and Finance of University of Wollongong offers a number of elementary subjects, each of which takes in more than two hundred students. The particular subject that uses this system has more than seven hundred students. Before the development of KBCIS, this subject was taught the traditional way. In addition to the normal classroom teaching, tutorial classes are provided to allow students to interact with tutors and to practice what is taught in the classroom for problem solving. In order to provide maximum fact-to-face discussion opportunities between students and the tutor, each tutorial class has only twenty students. Thus, for a class of seven hundred students, more than thirty tutorial classes will have to be arranged, and a number of tutors will be required to cover the whole class.

Over the years, the Department has to employ students with higher degree or even outside part-timer to help run the tutorial classes. In principle, the lecturer of a subject is in charge of all aspects of teaching as well as the well-being of students in the class. In practice, a lecturer can only be responsible for the development of the teaching material, the tutorial material, and the co-ordination of all tutors. For those students who need to talk to the lecturer after the class, each lecturer is required by the Faculty to provide four hours of consultation times each week. Obviously, four hours of consultation times in a week are not sufficient for a large class, hence most of the questions are expected to be handled and answered in the tutorial classes by tutors.

One problem with the current system is the qualification standard of tutors, which is sometimes beyond the control of a lecturer. Associated with this is the problem of coordination of tutors, which should ensure that all tutors are familiar with the lecturing material of every week, and apply the same standard in marking both exercises and examinations. In addition, different degree of communication skills of tutors often led to complaints from students in their ability to answer questions properly. Complaints about different standard in marking with different tutors are very common. As a result, there are still a large number of students who like to talk the lecturer, if they can.

The arrangement for conducting an examination for the whole class is another serious problem. Firstly, it is hard to find a room large enough to fit the whole class, unless the examination is conducted within the normal teaching hours. Secondly, it is almost always true that a supplementary examination will have to be provided for those students who were not able to attend the original examination, the number is not small for a large class, and the supervision must be also provided for.

It was against these background that the development of the Knowledge-Based Computer Instruction System (KBCIS) was conceived. The principle objective in designing the KBCIS is to enhance and extend the current teaching-learning function of large Accounting subjects, so that the traditional constraint of teaching and learning through personal contact between students, lecturers, or tutors can be minimised. It is envisaged that through this system, the requirements of locality and timing for delivering subject material can be greatly reduced or even eliminated in some cases. The goals in developing this system encompass three aspects of teaching: lesson delivery, evaluation, and feedback.

Under lesson delivery, it is intended for the system to lessen the dependency of students on formal classroom teaching and tutorial exercises. The present and past lecturing material and tutorial material should be made available to students for retrieval at anytime. In addition, it is vital for the system to be able to automatically generate exercise problems for a given topic, and have the capability to mark the answers properly and instantly. In this way, the system will allow students to practice with a wide range of problems to raise their proficiency and hence alleviate the demands for consultation requests. The specific goals in this area are to: (1) make available lecturing and tutoring material for retrieval by students at anytime, (2) generate exercise questions upon requests from students, and (3) mark answers from students properly and instantly.

Under evaluation, the system must be able to facilitate the development of examination questions with flexibility and efficiency. The types of questions which may be developed include the traditional stand alone questions, and the questions with more than one level of structure. In addition, regardless of the types of questions, the system must be able to mark answers from students with any scheme as is currently used by lecturers. One particular emphasis in this part is the ability for the system to provide partial credit for the portions of a question which are correctly answered. For personal evaluation, the system must be able to determine, just like a lecturer, if a student is good enough to go to the next topic or needs to practice more with the current topic, or needs to go back to an earlier topic. The specific goals for this part are to: (1) provide a flexible and efficient platform for developing examination questions, (2) provide an evaluation platform for developing flexible marking schemes, and (3) provide a monitoring mechanism to measure progress of students.

Under feedback, the system must be able to explain why an answer is wrong, and indicate the source material for students to refer to, and, if necessary, provide the step-by-step problem solving procedures for those who repeatedly fail the same question. As advocated by Hannafin (Hannafin & Peck, 1988) in catering for the needs of each individual student, the system must maintain a complete operation and performance records for every student, which includes the amount of time each student spends for each topic, the number of times a student tries a topic and the marks for each problem. With these, the system can identify those students who are underperforming, or are not participating enough, and notify the lecturer to initiate necessary actions before it becomes too late. The specific goals for this part are to: (1) provide reference information for problems which are wrongly answered, (2) indicate problem solving steps if necessary, and (3) collect performance statistics of students to help lecturers to identify problem areas or individual students who need help.

The backward chaining referencing approach

The main focus for achieving the goals stated above is the selection and implementation of an appropriate technique, which can utilise the given problem solution knowledge to solve a problem through major solution steps. This is a significant deviation from the traditional solution matching process, which tries to match given answers with stored answers. It is only when this part is accomplished satisfactorily, then the automatic problem generation, flexible marking scheme, and other related monitoring and supporting mechanisms of the system will become meaningful.

From the available techniques, we decided that the backward chaining approach is the most appropriate technique for this development. The backward chaining approach is one of the main approaches used in the development of knowledge-based systems (Durkin, 1994; Luger & Stubblefield, 1993; Medsker & Liebowitz, 1994). This approach is normally applied in conjunction with decision rules in the development of knowledge-based systems. In running a knowledge-based system, upon a request to find a solution for a given set of conditions, this approach takes each possible answer one at a time as the assumed answer, and work backward through inter-mediate decision processes to prove the assumed answer is a right answer. An assumed answer is a right answer if its required initial conditions can be met with the given conditions; otherwise it is a wrong answer, then this approach takes the next possible answer as the assumed answer and go through the same process. Using the following example for illustration, this approach will try to find a solution for X by, firstly, assuming A being an answer. From the rule set , it then finds out that B must be true for A to be a right answer, and C must be true for B to be true. C is the initial condition and is compared with the conditions to be given by the users. If C is true, then B is truce, and X=A is true; otherwise solution A will be discarded and F will be tested.

```
FIND  X

rule 1      IF B is true
               THEN X=A

rule 2      IF D is true
               THEN X=F

rule 3      IF  C is true
               THEN B is true

rule 4       IF G is true
               THEN D is true
```

In terms of the step-by-step algorithmic procedures, this approach takes A as the assumed final goal, and follows the decision rules to take B as its subgoal, and C as B's subgoal. In this way, the correct answer of a question can always be derived as long as the solution procedures of each subgoal can be expressed in rules. For the development of this system, a subgoal, for a given problem, represents a major solution step toward its final solution and the final solution is the final goal. The backward chaining approach can be implemented in the following recursive algorithm.

```
Search for the final goal of a question
While contained unknown subgoals
    Search for the unknown subgoals
    If no more unknown subgoal
         carry out prescribed instruction to
         obtain values of subgoals
    EndIf
EndWhile
```

Automatic problems generation and flexible marking

For this system to be able to generate exercise or examination problems of a given topic automatically, which are of the same nature but with different settings, a representative problem must be provided by the lecturer at the topic design phase. In the representative problem, there must be designated parameters whose values are subject to changes. These parameters can either be numerical or alphabetical, and their ranges of values and the formulas for selecting alternative values must also be specified. The system will, upon a request, select a new value for each designated parameter to produce a new problem. The following example is an illustration.

> *Example*. {AnyName} Ltd uses a Replacement Price Accounting System, and depreciates long term assets at 20% per annum - straight line depreciation. The replacement cost of a plant in new condition was ${Amount1} at January 1996 and increased by ${Increased-Amount} at January 1997. What is the amount of depreciation on plant for the period?

In this question, the variables inside bracket {} are the designated ones whose values are subject to changes. The variable AnyName can have its value chosen randomly from the list provided by the lecturer, for example it may include such names as IMB Tool, RCA Machine, Steel Maker, or any other names. The value of Amount1 is assumed to vary from $100,000 to $900,000. The Increased-Amount is assumed to vary in the range from $10000 to $90000. Upon a request, the system will randomly choose a name for the AnyName, and randomly selects a value for Amount1 and Increased-Amount from their respective ranges. As a result, every student will be seen as giving a different question, which will result in a different answer; although the solution procedures remains exactly the same.

In marking a problem, the system will first retrieve the solution knowledge of its representative problem from the database, and then invoke the backward chaining process to trace through each major solution steps to derive an answer, and use this answer to mark the answer from students. We use the above question to demonstrate the implementation of the backward chaining approach.

The solution procedures for the above question are to calculate, firstly, the cost of a new plant {$Amount2}for 1997, which is ($Amount1 + Increased_Amount), then find the final result by applying the formula: ((Amount1 + Amount2)/2*(20/100). The backward chaining approach works from the final solution of the problem and takes it as final goal of the question, which is ((Amount1 + Amount2)/2*(20/100). For this final goal to be verified, the values of Amount1 and Amount2 must be known. Hence, Amount1 and Amount2 become two subgoals which are to be verified. The subgoal Amount2 can not be known until both Amount1 and Increased_Amount are known, hence Increased_Amount is another subgoal. Amount1 can be obtained by random selection between $100000 and $900000, and Increased_Amount can be obtained by random selection between $10000 and $90000. Once both values are known, the subgoal Amount2 will be known, and then the final goal can be verified. The solution knowledge of this question is:

```
AnyName: randomly chosen from a given list
Amount1: ((random 9 +1) * 100000)
Increased_Amount: ((random 9 +1) * 10000)
Amount2: (Amount1 + Increased_Amount)
Solution: ((Amount1 + Amount2)/2*(20/100).
```

These solution knowledge is then translated into the following rule set to facilitate the application of the backward chaining approach. The input variable for students to enter their answers is "Reply", which is compared with the value of Solution calculated by the system.

```
Increased_Amount = (random 9 + 1)*10000
Increased_AmountFlag = ok
Amount1 = (random 9 +1)*100000
Amount1Flag = ok

IF Amount1Flag = ok AND Increased_AmountFlag = ok
THEN  Amount2 = Amount1 + Increased_Amount
             Amount2Flag = ok

IF Amount1Flag = ok AND Amount2Flag = ok
THEN Solution = (Amount1 + Amount2)/2*(20/100)

IF Reply =Solution
THEN Answer = Right
ELSE  CALL Wrong-Answer-Message
```

The contents of the Wrong-Answer-Message could range from the pages of the textbook which explain the solution approach in general to the solution steps of this particular problem.

From the above example, it is clear that part of this system is to translate the solution knowledge of a question into a set of rules, and carry out the solution steps through backward chaining process to find answers for each step. This will make it possible for the system to mark answers from students with great flexibility. In particular, it is possible to apply the widely accepted marking scheme which provides partial credits for partially answered questions. The process of verifying subgoals during the backward chaining process allows the system to assign different weights of credits to different parts of a solution. For example, with the above example, the correct calculation of the cost of a new plant {$Amount2}for 1997 may receive 40% of the total credit, and the final solution 60%. These can be built into the solution knowledge. In this case, there will be two answers “Reply1” and “Reply2” from students, and, assuming the total mark for the question is 10, the rules are changed as following:

```
IF Amount1Flag = ok AND Increased_AmountFlag = ok
THEN  Amount2 = Amount1 + Increased_Amount
             Amount2Flag = ok

IF Amount1Flag = ok AND Amount2Flag = ok
THEN Solution = (Amount1 + Amount2)/2*(20/100)

Mark=0
IF Reply1 = Amount2
THEN Mark = 4
            CALL Right-Answer-Amount2
ELSE   CALL Wrong-Answer-Amount2

IF Reply2 =Solution
THEN  Mark=Mark+6
             Answer = Right
ELSE  CALL Wrong-Answer-Solution
```

The mark is 4 if the answer “Reply1” is correct, and this mark will be added to the mark of the answer “Reply2” as the final mark if it is correct; otherwise the final mark will only be 4.

The following is the type of questions with hierarchical structure, which can be easily handled by this system. This type of questions, depending on the answers from students, can continue to different levels of questions.

> *Example*. {Company1} Ltd owns {Ownership}% of the shares in {Company2} Ltd. {Company2} is one of the 50 largest publicly listed companies. The remaining shares are widely held. Does <Company1> control <Company2>? (yes/no/do not know)

The system will randomly choose names from the given company list for the parameters “Company1” and “Company2”. The ownership is given between 20% and 60%, which is calculated by the system with the formula: Ownership = (integer)(20 + ((random mod 3) * 10) + (random mod 10)). The following rules are provided for the system to match the “Reply” from the user.

```
If Reply=“yes”
Then      Call  Question-Part-2
If Reply=“no”
ThenCall Wrong-answer-1
If Reply=“do not know”
ThenCall Wrong-answer-1
```

Based on the answer “Reply”, the above rule set will call the corresponding module to provide explanation or continue the next level of questions. For module Wrong-answer-1, the system calculates Remainder=(integer)(100 - Ownership), then it displays the message:

```
You are incorrect.
Since {Company1} owns {Ownership}% and {Remainder}% of the
shares are diffused, {Company1} is in a position to
dominate decision making.
```

For module Question-Part-2, the system calculates Remainder = (integer)(100 - Ownership), then it displays the following message and call the subsequent question (Question-Part-2):

```
You are correct.
Since {Company1} owns {Ownership}% and {Remainder}% of the
shares are diffused, {Company1} is probably in a position
to dominate decision making. {Company1} controls {Company2}
because the remainder of the shareholding is diffused.
Therefore {Company1}'s {Ownership}% block of shares is
larger than anybody else and will allow {Company1} to
outvote everyone else.   (true/false)
```

The rules provided for the system to mark the Question-Part-2 are:

```
If Reply=“true”
Then      Call  Question-Part-3
If Reply=“false”
ThenCall Wrong-answer-1
```

Depending on the answer, the system can then call up the following question Question-Part-3, or display the message for wrong answer.

The structure of the system

The structure of this system can be best explained by using Figure 1 and Figure 2. Figure 1 illustrates the major subsystems involved in receiving and interpreting inputs from a lecturer and the subsequent processing of the inputs. This system contains a set of macro commands for executing built-in subsystems, which was designed to facilitate lecturers to develop subject contents, solution knowledge, and control environment for exercises and examinations. The inputs from a lecturer may consist of commands for setting up the contents of specific topics, representative problems for different topics, solution procedures of representative problems, and other instructions for controlling the access of material by students. In addition, commands are also available for real time information collection and analysis, which will allow the system to initiate necessary actions just like a human lecturer would do.

The text inputs prepared by lecturers are screened by the Lexical Scan program of the system first, which can pick up key words and identify different type of information, and channel them to appropriate subsystems. These inputs are then processed and become accessible to the Agenda Scheduler, which serves as the controller of the system in coordinating the execution of various subsystems to carry out necessary functions.

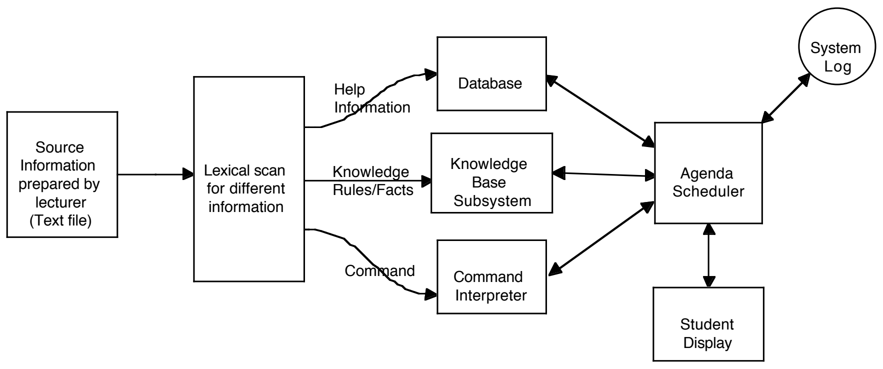

Figure 1 System configuration for processing inputs from a lecturer

The Database module contains materials for weekly class lectures, weekly tutorial sessions, representative questions for selected topics, solution procedures for each representative question, solution guidance for each representative question, and/or hints for each representative question. Help related information is also stored in the Database module. In application, students interact with the Agenda Scheduler, their request commands will be interpreted by the Command Interpreter module, and executed accordingly by invoking various subsystems. When the system is used in the examination mode, questions will be retrieved from the database module. The Agenda Scheduler at the same time will retrieve the solution procedures and invoke the Knowledge Base module to convert them into a set of solution rules for marking purpose. Alternatively, the solution rules can be provided as parts of initial inputs.

The system structure which allows students to interact with the system through Agenda Scheduler is shown in Figure 2.

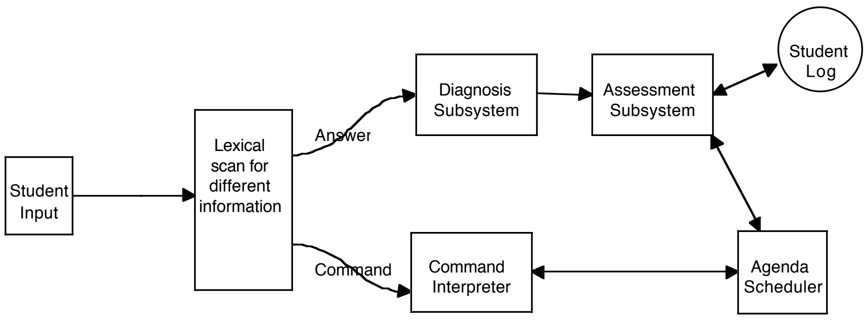

Figure 2 System configuration for processing inputs from students

In Figure 2, the inputs from students will be read from terminals and checked by the Lexical Scan program for identification of either system commands or answers to problems, and the inputs will be channelled to either Command Interpreter or Diagnosis subsystem for processing. If the input from a student is an answer to a problem, then the answer will be passed to the Diagnosis subsystem. This subsystem, through Agenda Scheduler, interacts with the Knowledge Based Module to determine if an answer is correct, partially correct, or incorrect. It also provides the feedback information for wrongly answered problems from the Database module.

The results from the Diagnosis subsystem will be passed to the Assessment subsystem, which determines the level of assistance to be provided to a student, if he/she has not achieved the pre-set standard. The level of assistance is pre-determined by a lecturer, and may include the option of showing the contents of topic material, textbook reference, or even solution approaches. For example, a student may be presented with the textbook reference for a problem for the first failure, and the solution approach for the second failure. When the performance of a student is below a satisfactory standard, the system may even suggest a topic of lower level unit to work with. The Agenda Scheduler will also take command inputs from students, and carry out the requested function accordingly, which may include the need to retrieve lecture notes, practice exercise problems of a particular topic, or request advice or guidance when in doubt.

The system keeps a student Log File which records the usage and performance of each individual student, and it can be accessed only by the subject lecturer through the online system. The Log File chronologically records usage history of each student, which includes number of attempts made for each topic, answers for each question, and time spent for each problem. This information is analysed by the Assessment Subsystem for determining the performance of students, and is updated on a real time basis. This performance analysis can also help lecturers to understand the learning behaviour of students, their attendance, and the progress they are making. An email facility is included in this subsystem, which, on detection of unusual behaviour or under-performing students, will automatically send students' names and ID numbers to the lecturer.

Conclusions

This system has been in operation for three years for one Accounting subject with seven hundred students. The responses from students as well as lecturers are very encouraging. In terms of the enhancement of the teaching-learning function, based on the experience of one lecturer who used the system, the improvement is noticeable through the substantial reduction of the failure rate. It seems that the flexible contents retrieving and problem marking capability of the system has created an environment, which encourages students to learn without the limitation of locality and without the need of presence of the lecturer or tutors. In terms of the marking process, which is the most dreaded aspect for a lecturer to take up a large class, her experience showed that the mid-term examination, if properly structured, can be marked instantly by the system. For the major assignment, which requires the use of the system as well as other supporting documents, she took on the average takes three minutes to mark one assignment, that compares very favourably with thirty minutes without the system before.

The major problem for any lecturer to use this system, according to her experience, is the fact that the first time user need to spend a fair amount of time, much more than the amount needed for the traditional approach, to learn the software and develop the contents. Like any software, a period of learning is required before one can gain familiarity of the software and use it effectively. The development of the contents, which includes the systematic structuring of a subject and the design of representative problems along with the procedures of their solution knowledge, takes some considerable amount of effort. In her first year, this lecturer spent at least five times the effort in preparation compared with the traditional method. However, in her subsequent years, she only needed to spend less than half the time compared with the traditional method.

The general conclusion that can be drawn from this experience is that, the system has demonstrated to be a very effective teaching aid. The effectiveness of this system depends heavily on the ways contents and questions are structured and designed. A strong commitment from lecturers is required so that contents can be developed with depth. At present, the system is used mostly for drills and tutorial developments, which is the easiest and the least time consuming; the potential of the system has not been fully utilised which can allow the design of instructional games and simulation environment. Finally, since the amount of effort required for the first time preparation tends to deter most lecturers from using the system, it may be necessary for the Department to provide some kinds of incentive by relieving them from other duties, so that they will have more time to spend with the system.

References

Allessi, S.M. & Trollip S.R. (1985). *Computer-Based Instruction: Methods and Development*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Bourne, D.E. (1990). Computer-assisted instruction, learning theory, and hypermedia: An associative linkage**.** *Research Strategies*, 8(4), 160-171.

Brown, J.S., Burton, R.R. & de Kleer, J. (1982). Pedogogical, natural language and knowledge engineering in SOPHIE I, II and III. In Sleeman, D. & Brown, J.S. (Eds.), *Intelligence tutoring systems*. New York: Academic Press.

Carbonell, J.R. (1970). AI in CAI: An artificial intelligence approach to computer assisted instruction. *IEEE Transcations on Man-Machine Systems*, 11, 190-202.

Cook, E.K. & Kazlauskas, E.J. (1993). The cognitive and behavioural basis of an instructional design: Using CBT to teach technical information and learning strategies. *Journal of Educational Technology Systems*, 21(4), 287-302.

Dewey, J. (1910). *How we think*. Boston:Heath.

Durkin, J. (1994). *Expert Systems Design and Development***.** Englewood Cliffs, New Jersey: Macmillan Publishing Company.

Flechsig, K. (1989). A knowledge-based system for computer-aided instructional design (CEDID). *Proceedings of the International Congress in Education and Informatics*, 400-403.

Hannafin, M.J. & Peck K.L. (1988). *The Design, Development, and Evaluation of Instructional Software*. New York: Macmillan Publishing Company.

Kearsley, G.P. & Seidel, R.J. (1985). Automation in training and education. *Human Factors*, 27, 61-74.

King, M. & McAulay, L. (1992). Simple expert systems for computer assisted instruction. In Doukidis, G.I and Paul, R.J. (Eds.) *Artificial Intelligence In Operational Research*. London: The Macmillan Press Ltd., 105-114.

Lockard, J. (1992). *Instructional Software: Practical Design and Development.* Debuque, Iowa: Wm. C. Brown Publishers.

Luger, G.F. & Stubblefield, W.A. (1993). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 2nd ed. RedWood City, California: The Benjamin/Cummings Publishing Company, Inc.

MacKnight, C.B. & Balagopalan, S. (1989). An evaluation tool for measuring authoring system performance. *Communications of the ACM*, 32(10), 1231-1236.

Mandl, H. & Lesgold, A. (1988). Preface. In Mandl, H and Lesgold, A (Eds.), *Learning Issues for Intelligent Tutoring Systems*. New York, New York: Springer-Verlag, pp v - xiv.

Medsker, L. & Liebowitz, J. (1994). *Design and Development of Expert Systems and Neural Networks*. New York, New York: Macmillan College Publishing Company.

Milheim, W.D. (1993). Using computer-based instruction with adult learners. *Journal of Continuing Higher Educational*, 41(3), 2-8.

Nachouki, M & Gouarderes, G. (1994). A knowledge acquisition system to resolve ambiguous solution in intelligent tutoring systems. In Liebowitz, J. (Ed.), *Moving Towards Expert Systems Globally in the 21st Century,* Elmsford, New York: Cognizant Communication Corporation, 534-541.

Park, O., Perez, R.S. & Seidel, R.J. (1987). Intelligent CAI: Old wine in new bottles, or a new vintage? In Kearsley, G. (Ed.), *Artificial Intelligence and Instruction: Applications and Methods*. Reading, Massachusetts: Addison-Wesley Publishing Company.

Pogue, R.E. (1985). Authoring systems: The key to lesson development. *Journal of Educational Technology Systems,* 13(2), 75-81.

Sleeman, D. & Brown, J.S. (1982). *Intelligent tutoring systems*, New York: Academic Press.

Wallach, B. (1987). Development strategies for ICAI on small computers. In Kearsley, G. (Ed.) *Artificial Intelligence and Instruction: Applications and Methods*. Reading, Massachusetts: Addison-Wesley Publishing Company.

Joshua Poh-Onn Fan, Department of Business Systems
University of Wollongong
Northfields Avenue,
Wollongong, NSW 2522 Australia
Tel: +61 42 214041 Fax: +61 42 214474
Email: joshua@uow.edu.au

Tina Kwai-Lan Mak, Department of Accounting and Finance
Li-Yen Shue, Department of Business Systems