Transactions on Information and Communications Technologies vol 20, © 1998 WIT Press, www.witpress.com, ISSN 1743-3517

Web-based Delivery of AI/Expert System Courses in Engineering Education

Prof. Paul J. Nolan
Department of Mechanical Engineering,
National University of Ireland, GALWAY

E-mail: paul.nolan@ucg.ie
fax: 353-91-563991

ABSTRACT

This paper describes on-going work at University College Galway on the use of Web-based interactive courseware which, in addition to the usual situation of presenting courses as static hypertext documents including text and graphics allows the students to access AI languages and expert system components directly from their web browser. The system is at beta test stage and is being used in courses on AI languages (LISP and Prolog) to engineering and computer science students.

The overall approach used in reformulating existing lecture material and associated computer laboratory demonstrations and exercises into web-based interactive courseware is set out and the rationale for the choice between server side CGI scripts, plug-ins, and Java applets is briefly discussed. Sample courses are described and future plans outlined.

1 Introduction

In the last couple of years there has been a gargantuan increase in the amount of teaching material being made available on the World Wide Web (WWW). For example, the "World Lecture Hall" now contains hundreds of courses in a number of disciplines and there have been a number of concerted initiatives to promote use of web-based courseware. As an example see the Teaching & Learning Technology Programme in Doughty[5] for enabling "open courseware" in UK. A number of course development and delivery systems incorporating features such as student registration, access tracking, homework submission, self assessment, grade accumulation and so on, are

emerging. The proceedings of the CTC on the Teaching of Computing[4] provide a comprehensive recent reference. Generally, such systems focus on the development of (or translation from previously existing word-processor) documents and on their subsequent delivery as static documents to the student. The pedagogically important facility of creating examples 'on the fly', as would be done in a real lecture by an enthusiastic lecturer, is generally not included. Furthermore, the opportunity of allowing students to learn by posing "WHAT-IF" questions and seeing the results of evaluating an expression or running a small program, is generally not included in the interactive courseware.

It is often argued that the World Wide Web represents the third phase of digital electronic teaching and learning technology. The digital electronic information age opened with the arrival of the microprocessor-based personal computer in the late 1970's. The second phase came with the introduction of multimedia objects such as digital images, digital video and sound. The third phase, the World Wide Web has become the medium for rapid dissemination of multimedia learning resources. The web technology also provides for instantaneous interactivity, essentially eliminating the traditional space and time barriers. The web technology is likely to have a significant impact in many areas. Web sites can serve as communication hubs for student-teacher, student-student and teacher-teacher interactions. By providing web-based instructional material, teachers can structure the out-of-the-classroom time so the students can use it more productively. Web sites can provide a virtual setting for collaborative peer instruction which has proven so effective in the classroom. All the instructor must do is provide links between the classroom work and the web work. Web sites can also be used by teachers from geographically separated institutions to collaborate on instructional material development and educational research.

2 IMPLEMENTATION DETAILS

As is well known, the Web offers tremendous advantages in making available electronic documents world-wide. There is a gargantuan growth in the number of Web sites and the number of people accessing the web is increasing at a phenomenal rate. In addition to text, graphics, sound and video are being distributed. At the most basic level, the Client-Server architecture allows users to retrieve existing

electronic documents from Web servers. The ability to link from one document to others, possibly at different sites is the overwhelming feature. Java Scripts on the client machine can be used to perform some processing which permits more sophisticated interaction with the Web server. As an example, checking for reasonableness in client request can be performed before transmission to the server. JavaScript at the client end can also be used to do local processing.

Figure 1 shows the interaction of the various web technologies we use in the delivery of the courseware material. Java Scripts on the client machine can be used to perform some processing which permits more sophisticated interaction with the Web server. As an example, checking for reasonableness in client requests can be performed before transmission to the server. Java Script at the client end can also be used to perform local processing. CGI (Common Gateway Interface) scripts on the Web server allow serious processing to be done. CGI scripts can invoke routines in other languages (including compiled languages) so computationally demanding programs can be run. Python, described by Lutz[9], was used to implement our CGI scripts. Our current web server is a Pentium PC running LINUX.

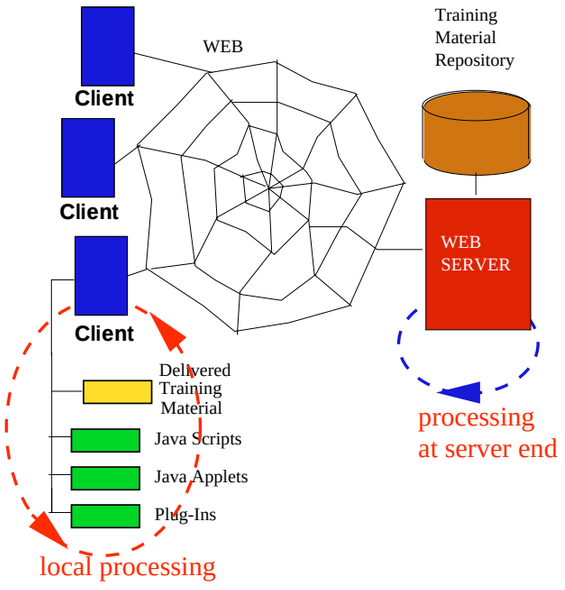


Figure 1 Architecture for Web-based Course Delivery

CGI scripts can be used to create Virtual Documents (i.e. a document is created 'on the fly') to reflect the specific request by a user. This adds a completely new dimension as the interaction becomes much more dynamic. Careful use of server side processing is important for speedy responsiveness. Java Applets are compiled programs sent on demand from the server and run on the client machine. They persist only as long as the client is connected to the server. As they are compiled, they run relatively fast. The code is not visible to the client. Java Applets are particularly good at graphics applications (including animations) and number of serious applications are emerging. The fact that the program is run at the client end means reduced load on the Web server. Plug-ins further extend the capability of the browser by allowing the client to run other specialised programs. The plug-in can be downloaded from a website, stored locally and loaded and unloaded from memory as required. There are a growing number of plug ins for multimedia etc.

The choice between implementation among the various alternatives depends on a number of factors including security, download time, processing requirements, amount of interactivity required, amount of feedback response and client machine specification. Rather than promoting any particular approach we have provide alternatives where possible.

The configuration described above is for out-of-classroom, self paced or distance education. In some situations it is useful to use the underlying technology in a classroom environment where a lecturer controls the presentation of the material but where students still have facility to interact with the on-line examples. The proposed approach is depicted in Figure 2.

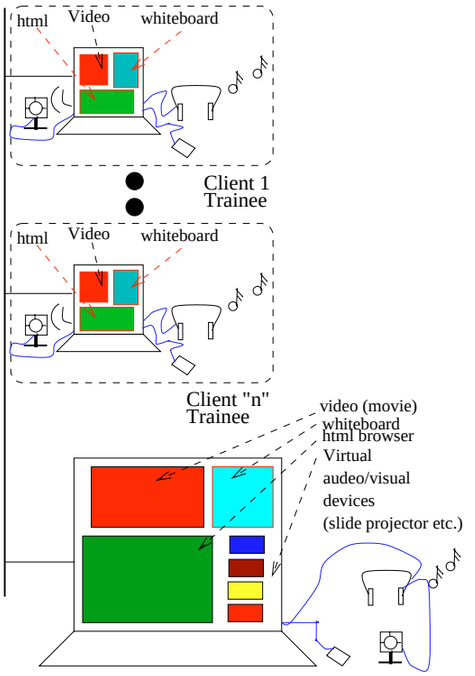


Figure 2 Architecture for instructor mediated session

3 SAMPLE COURSES

3.1 General

The approach taken was to break down each course into a series of sections with each section further subdivided into subsections corresponding approximately to traditional lectures. Each lecture comprises a number of linked HTML documents. Translation utilities such as TROFF2HTML and LATEX2HTML, Drakos[6] were used in the translation. Control panels (HTML forms) are used in the interactive examples such as simulations. Output from AI programs are returned as HTML documents and GIF images.

As mentioned earlier a very important feature of our approach is the facility for students to write, edit, debug and run programs on their browsers. Plug-Ins for LISP and Prolog are available. As an example,

the development of the SICStus Prolog Netscape[12] Plugin was motivated by the need to make the expert systems available on the web. It is available only on Windows 95 and NT platforms only.

At the time of writing we did not source a suitable applet for LISP. There are in fact Java implementations of Scheme interpreters. For example, the Kawa[2] system provides a compiler from Scheme into Java byte-codes. Further Possible to use Kawa to build applets using the Java AWT so examples can be run directly from web page. Kawa is a full Scheme implementation. It implements almost all of R5RS and Scheme functions and files are automatically compiled into Java byte-codes, providing reasonable speed. Kawa provides the usual read-eval-print loop, as well as batch modes.

Another Scheme interpreter applet in Java has been implemented which has been extended to allow for easy access to the Java AWT windowing and graphics facilities. Since this new language combined features from Scheme and Java, we it was called Jscheme[8]. Depending on how the applet was invoked on a web page it would either create a software development window for writing, editing, and debugging your program, or it would execute a pre-existing program given either the code itself or the URL of the code. Thus students could visit the course home page and have access to the development environment. They could also add the applet to their own web page and specify their own program in the applet parameters. When used this way the applet would run their program and hence allow them to create applets on their own pages.

The only hardware/software requirement is that they have access to the Internet and have a Java-enabled web browser on a Mac, PC, or Linux box. Jscheme can also be used as a scripting language, like Java Script, to create applets which appear automatically on a web page.

3.2 Introductory Course

The core objectives of a course in Artificial Intelligence might be expressed as follows:

- to introduce to the students the guiding concepts, such as search and productions, which structure the entire field.
- to communicate such concepts in terms of specific search and production strategies

- to ground the student's understanding of these concepts via practical course-work and problem-solving particularly in engineering applications
- To introduce the student to the theoretical basis of artificial intelligence (i.e. Search theory)
- To educate the student in the methodology of problem formulation for automatic solution by machines (e.g. ranging from everyday puzzles to complex expert applications)
- to sharper problem formulation skills (e.g., the ability to recognise areas where search techniques are applicable and the ability to reformulate a problem specification to accommodate a search perspective).
- to heighten programming skills in an A.I.-friendly language such as PROLOG and/or LISP.
- to develop an awareness of the potential for intelligent behaviour in everyday domains and applications.

As a flavour of the course material we briefly describe some of the language element in the courseware, Lisp and Prolog are the most frequently used symbolic programming languages for artificial intelligence. They are widely regarded as excellent languages for "exploratory" and "prototype programming". Figure 3 shows a partial view of the Table of Contents page of the module on LISP of the introductory course. Examples and sample exercises are given on line. Students' LISP code can be run on the courseware server and results displayed on the students terminal. Jscheme, which has already been described as a Java Applet implementation of Scheme, which includes impressive GUI facilities, can be used to run small LISP code at the client end. A typical screen shot is shown in Figure 4.

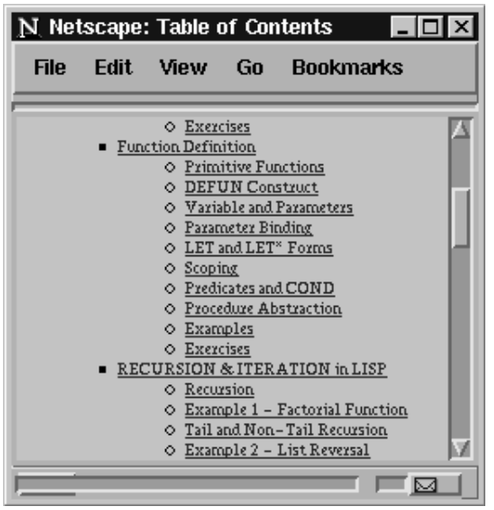


Figure 3 Screen capture showing table of contents for LISP module

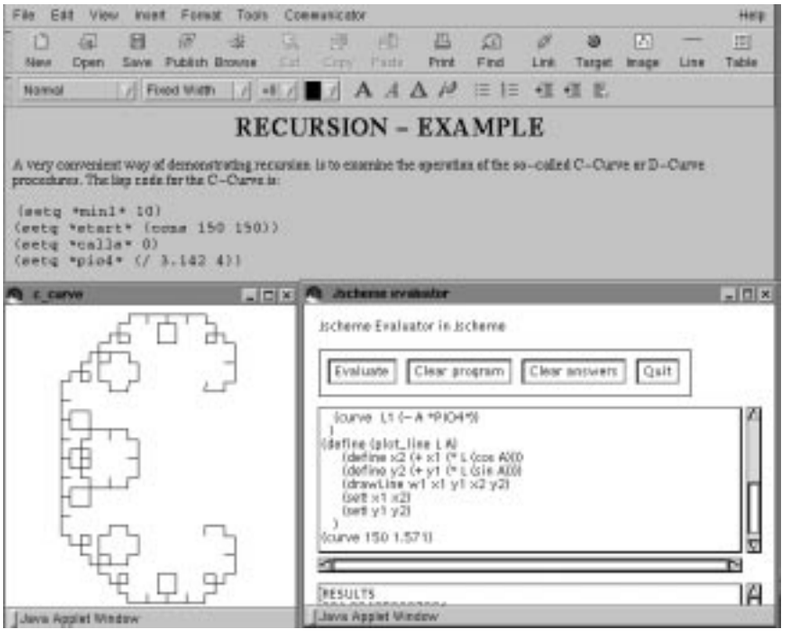


Figure 4 Screen capture showing interaction using Applets

The usual examples on recursion and iteration e.g. Tower of Hanoi, Factorial and Fibonacci Number generation, pattern matching etc. are explored. A number of more advanced applications are also included. These include computer algebra examples (e.g. differentiation and integration of polynomials and inverse kinematics in robotics) and graph theoretic applications such as AND/OR tree search, graph search (including reliability, project activity and road network search) and state-space search. A simple planning application based on the so-called "Blocks World" is also presented.

Figure 5 show a typical output from a roadmap search implemented in LISP. Selection of the origin and destination, search strategy, number of solutions sought etc. are implemented from a control panel embedded in HTML form.

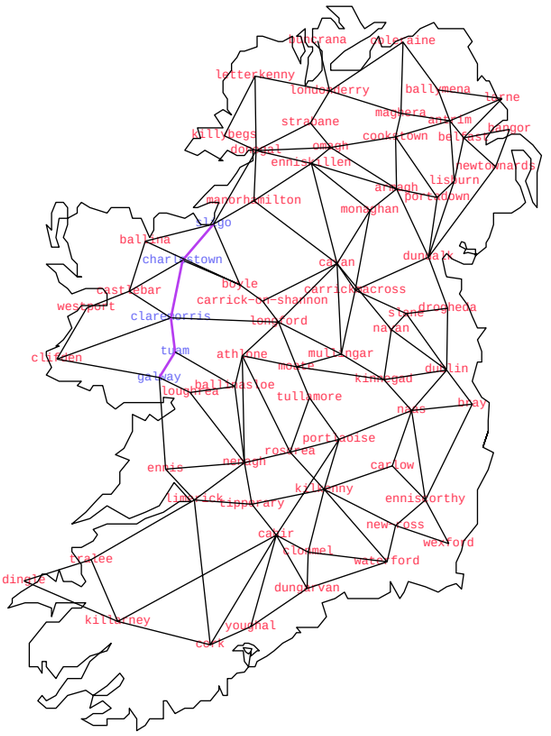


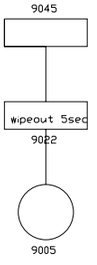
Figure 4 Map Search Example

The second main language introduced is Prolog. All the expected introductory material is included e.g. Prolog syntax, built-in predicates of Prolog, declarative and procedural meaning, pattern matching, data structures, unification, backtracking, the "cut" etc.

Example include those typically found in elementary text books e.g. N-queens problem, 8-puzzle, symbolic differentiation, graph search, sorted tree dictionary and so on. Many of the sample problems (e.g. the roadmap search problem) are implemented in both languages.

The classic textbook by Clocksin and Mellish[3] is used as reference.

As a specific example we consider the diagnosis of hardware logic circuits. Our overall objective is to be able to infer what combinations of inputs result in specified main or alarm events. The basic operations of the search algorithm is a extension of AND/OR graph search and has been previously reported by Howie & Nolan[10]. This problem is considerably more difficult than the usual AND/OR search because of the temporal character of the input signals. As reported the approach adopted was to consider a time window for each signal. It must be emphasised that we are attempting to infer the inputs for specific outputs. A more complicated example involving a large number of AND and OR nodes is presented in the advanced course.



node_list	type_list	time_list	logic_list
9045	main	[0..0]	1
9022	delay1	[0..0]	1
9005	basic	[-5..0]	1

node_list	type_list	time_list	logic_list
9045	main	[0..0]	0
9022	delay1	[0..0]	0
9005	basic	[-5..0]	0

Figure 5 Example in Logic Circuit Temporal Reasoning Application

Each of the logic circuits can be represented using the so-called truth-functional-calculus and has an associated truth table. As Prolog is a Logic Programming language we might expect it to be particularly suited in this application. However, its real power is its ability to unify the input variables for specified outputs rather than the more usual situation of simulating the output for given inputs.

The diagnosis problem can be viewed as an extension of a basic AND/OR graph search. One of the most straightforward ways of solving an AND/OR graph is to use Prolog's inherent search capability via backtracking. The following are the rules:

- if a node is a basic node then it is inherently solved.
- if a node represents an OR gate then solve one of them. Prolog will automatically try out alternatives until an appropriate one if found.
- if an node represents an AND gate, then solve all of them.

The Prolog presented is based on the above ideas. Operators are defined to specify successor and input relationships to assist in representing the graph.

3.3 ADVANCED COURSE

In the advanced course the material in the introductory course is reviewed and subsequently extended to include new topics. As a specific example, the search algorithms in the first course are based on blind search (depth first, depth first, bounded depth first and so on). Heuristic search is introduced in the advanced course. Similarily uncertainty and fuzziness concepts are introduced in the production systems.

In the advanced course other examples such as an algorithm for a graph-theoretic approach to Mason's Rule and minimum cut set in reliability networks employ image graphs for student interaction. The algorithms are run in response to specific user input data and graphical output, including animations showing the operation of the algorithms, are generated. Figure 6 (a) and 6 (b) shows a block diagram and the corresponding signal flow graph for a control system problem. The input output pairs are selected by the user. Figure 6 (c) depicts some of the closed paths determined using the search algorithms developed

as part of the course. Table 1 shows the loops and loop products. Processing is done on the server and the overall transfer function is determined in symbolic form.

Another practical engineering example of network search is in determining the reliability of electrical networks. The transmission system depicted in Figure 7 is used as an example and the minimum cut-set is determined using LISP and/or Prolog programmes. Processing is again done on the server side and results displayed on the browser. These larger examples used code fragments from earlier exercises in the introductory course and hypertext links are provided as appropriate.

Development of simple expert systems is an important part of the advanced code. CLIPS 6.0[7] (C Language Integrated Production System) is an OPS-like forward chaining production system written in ANSI C by NASA is used. COOL (CLIPS Object-Oriented Language) which is directly integrated with the inference engine is used to demonstrate object oriented features and DYNACLIPS (DYNAamic CLIPS Utilities) is used to demonstrate blackboard, dynamic knowledge exchange, and agents. FuzzyCLIPS 6.02 is a version of the CLIPS rule-based expert system shell with extensions for representing and manipulating fuzzy facts and rules.

In general the expert system shell is run on the server; although code can be edited on the client side. However, facility is included to run Jess[13] (a Java version of CLIPS) at the client end. Jess is a clone of the core of the CLIPS expert system shell. It is written entirely in Java. It is downward compatible with CLIPS, in that every valid Jess script is a valid CLIPS script. Like CLIPS, Jess uses the Rete algorithm to process rules. Jess contains only the essential features of CLIPS.

The textbook by Giarratono and Riley[7] is an excellent reference on CLIPS which is used as the reference text.

In the advanced course, a number of advanced examples on Prolog programming are also presented. One of the main projects is to take some of the ideas on logic circuit diagnosis and apply them to industrial sized projects. The logic sheet shown in Figure 8 depicts an alarm circuit used in power plant monitoring. This system has been described in detail by Nolan & Howie[10].

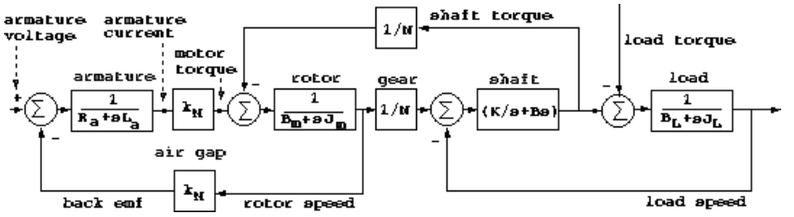


Figure 6 (a) Servo Block Diagram

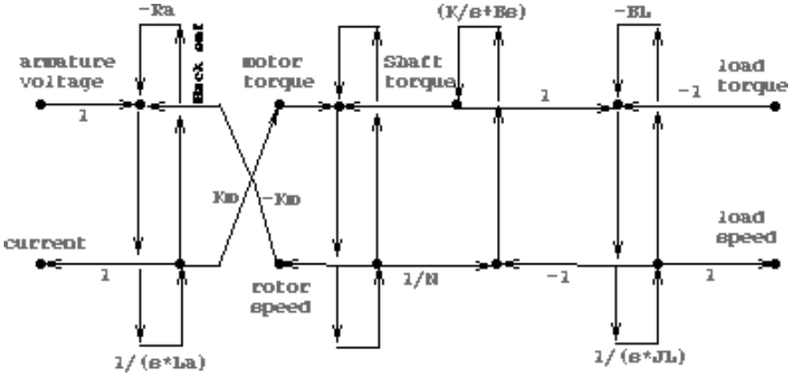


Figure 6 (b) Servo Signal Flow Graph

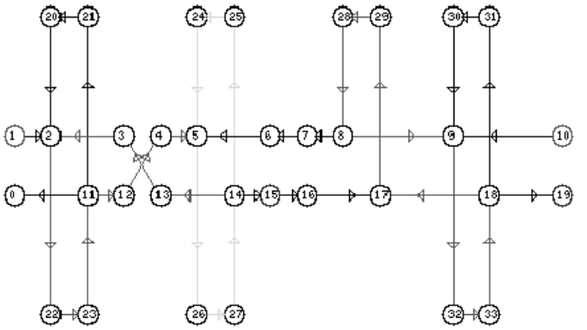


Figure 6 (c) SFG with paths identified

paths: ((N1 N2 N22 N23 N11 N12 N4 N5 N26 N27 N14 N15)) gain: ((T6 T4 T2))

0	(L6)	(- T10 T11)	(N9 N32 N33 N18 N31 N30);	((1))
1	(L5)	(- T5 T6)	(N5 N26 N27 N14 N25 N24);	NIL
2	(L4)	(- T1 T2)	(N2 N22 N23 N11 N21 N20);	NIL
3	(L3)	(- T11 T9)	(N17 N29 N28 N8 N9 N32 N33 N18);	((1))
4	(L2)	(- T6 T7 T9 T8)	(N15 N16 N17 N29 N28 N8 N7 N6 N5 N26 N27 N14);	NIL
5	(L1)	(- T3 T6 T4 T2)	(N2 N22 N23 N11 N12 N4 N5 N26 N27 N14 N13 N3);	NIL
6	(L1 L3)	(T3 T6 T4 T2 T11 T9)	(N2 N22 N23 N11 N12 N4 N5 N26 N27 N14 N13 N3 N17 N29 N28 N8 N9 N32 N33 N18);	NIL
7	(L1 L6)	(T3 T6 T4 T2 T10 T11)	(N2 N22 N23 N11 N12 N4 N5 N26 N27 N14 N13 N3 N9 N32 N33 N18 N31 N30);	NIL
8	(L2 L4)	(T6 T7 T9 T8 T1 T2)	(N15 N16 N17 N29 N28 N8 N7 N6 N5 N26 N27 N14 N2 N22 N23 N11 N21 N20);	NIL
9	(L2 L6)	(T6 T7 T9 T8 T10 T11)	(N15 N16 N17 N29 N28 N8 N7 N6 N5 N26 N27 N14 N9 N32 N33 N18 N31 N30);	NIL
10	(L3 L4)	(T11 T9 T1 T2)	(N17 N29 N28 N8 N9 N32 N33 N18 N2 N22 N23 N11 N21 N20);	NIL
11	(L3 L5)	(T11 T9 T5 T6)	(N17 N29 N28 N8 N9 N32 N33 N18 N5 N26 N27 N14 N25 N24);	NIL
12	(L4 L5)	(T1 T2 T5 T6)	(N2 N22 N23 N11 N21 N20 N5 N26 N27 N14 N25 N24);	NIL
13	(L4 L6)	(T1 T2 T10 T11)	(N2 N22 N23 N11 N21 N20 N9 N32 N33 N18 N31 N30);	NIL
14	(L5 L6)	(T5 T6 T10 T11)	(N5 N26 N27 N14 N25 N24 N9 N32 N33 N18 N31 N30);	NIL
15	(L2 L4 L6)	(- T6 T7 T9 T8 T1 T2 T10 T11)	(N15 N16 N17 N29 N28 N8 N7 N6 N5 N26 N27 N14 N2 N22 N23 N11 N21 N20 N9 N32 N33 N18 N31 N30);	NIL
16	(L3 L4 L5)	(- T11 T9 T1 T2 T5 T6)	(N17 N29 N28 N8 N9 N32 N33 N18 N2 N22 N23 N11 N21 N20 N5 N26 N27 N14 N25 N24);	NIL
17	(L4 L5 L6)	(- T1 T2 T5 T6 T10 T11)	(N2 N22 N23 N11 N21 N20 N5 N26 N27 N14 N25 N24 N9 N32 N33 N18 N31 N30);	NIL

Table 1 Results from Masons Rule

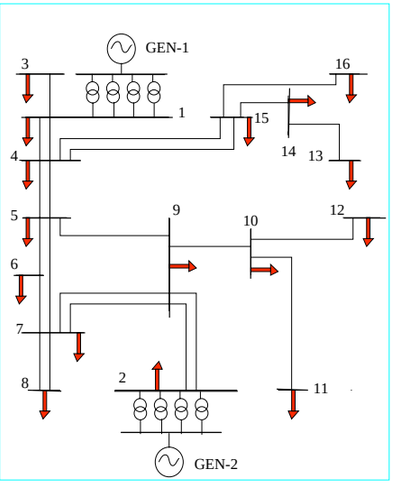


Figure 7 Min Cut-set Problem

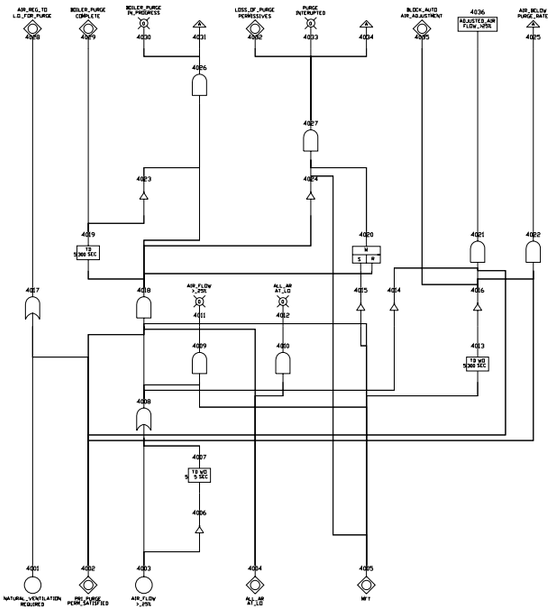


Figure 8 Alarm System Logic Circuit used as AND/OR Graph Example

4 EXTENSIONS

The web-based courses presented have been developed very much on an ad-hoc basis and approximately mirror the traditional classroom lecture presentation and computer demonstrations. Future work is aimed at (1) reengineering these courses taking into account student feedback and pedagogical issues and (2) adding some element of ITS (Intelligent Tutoring System) or ICAI (Intelligent Computer Aided Instruction).

With regard to the pedagogical issues, we hope to get some input from an educational psychologist. Important research such as that performed by Gagni[12] and other relevant theories will be reviewed and applied as appropriate. In particular Gagni suggests nine universal steps in an instructional context:

- Gain attention e.g. present a good problem, a new situation or use a multimedia presentation.

- Describe the goal: e.g. describe the goal of a lesson, state what students will be able to accomplish and how they will be able to use the knowledge, give a demonstration if appropriate.
- Stimulate recall of prior knowledge e.g. remind the student of prior knowledge relevant to the current lesson (facts, rules, procedures or skills). Show how knowledge is connected, provide the student with a framework that helps learning and remembering. Tests can be included.
- Present the material to be learned e.g. text, graphics, simulations, figures, pictures, sound, etc. e.g. follow a consistent presentation style, chunking of information (avoid memory overload, recall information)
- Provide guidance for learning e.g. presentation of content is different from instructions on how to learn. Should be simpler and easier that content.
- Elicit performance "practice", let the learner do something with the newly acquired behaviour, practice skills or apply knowledge.
- Provide informative feedback show correctness of the trainee's response, analyze learner's behaviour (or let him do it), maybe present a good (step-by-step) solution of the problem.
- Assess performance test, if the lesson has been learned. Also give sometimes general progress information
- Enhance retention and transfer inform the learner about similar problem situations, provide additional practice. Put the learner in a transfer situation. Maybe let the learner review the lesson.

The possibility of applying AI in the course delivery of the subject which so happens to be AI is a very exciting possibility. All the main components are already there. We already are employing an expert system shell (CLIPS) on the server side to enable students to perform experiments. At the client side i.e. at the web browser we already have a Java Applet based CLIPS system. Consequently we have considerable flexibility in implementing an expert system with separate knowledge bases for instruction in general, the individual student and the domain per-se. The fact that all important student actions (e.g. pages accessed, response to multiple choice questions etc. are logged means that a student model (e.g. so-called overlay model) could be

developed. Many of the ideas in O'Sheas[11] earlier work should be revisited in the context of current software tools. This is the basis of on-going work.

5 DISCUSSION & CONCLUSIONS

A brief outline of the approach taken in the delivery of web-based courseware has been presented. The usual hypertext training material (text and graphics) is reinforced by the ability of students to run AI programs (LISP and Prolog) on line.

The web technology allows built-in interactivity via JavaScript embedded in the HTML documents and via submissions to and responses from a web server. Lessons can be structured so that students can proceed along a path of their choosing, at their own pace.

The locations of the materials is transparent to the students, and the ability to access materials from anywhere opens up enormous possibilities for education. How these capabilities can best be used by students and educators to transform education is a question we in the education community have only begun to explore.

In general the possibilities include:

- An easily maintained and updatable set of links to the huge amount of
  up-to-date information.
- Self-contained interactive tutorials and review materials with individualized feedback provided by JavaScript embedded in the document.
- Virtual labs to extend or complement the hands-on classroom experience.
- Independent web project assignments, with the web resources providing an alternative to the traditional library materials.

As already discussed, future work includes addition of ICAI (Intelligent Computer Aided Instruction) elements and further investigation of pedagogical issues in courseware design.

ACKNOWLEDGEMENTS

The author would like to acknowledge the assistance of David Lane in implementing the web site. Useful discussions on web teaching with Dr. Peter Corcoran of NUI, Galway's Electronic Engineering Department are also acknowledged.

REFERENCES

[1] Aronson, D. T. Et Briggs, L. J. (1983). Contributions of Gagni and Briggs to a Prescriptive Model of Instruction. In Reigeluth, C., (ed.), Instructional Design Theories and Models: An Overview of their Current Status. Prentice-Hall, Hillsdale, NJ.

[2] Bothner, P "Kawa - Implementing Scheme in Java", Cygnus Solutions, 1325 Chesapeake Tce, Sunningdale CA, 94089. http://www.cygnus.com/~bothner/kawa.html

[3] Clocksin, W.F & Mellish, S.C., Programming in Prolog, Springer-Verlag, New York, 1984.

[4] Daughton, G & Magee, P (Editors), "Integrating Technology into the Computing Curriculum", Proceedings of 5th Annual Conference on the Teaching of Computing, Published by Dublin City University, Aug 1997.

[5] Doughty, G., Pollock, M, McAteer, E and Turner I, "Conversion of a mathematics course to tutor-supported computer-assisted flexible learning", in Open and distance Learning in Industry and Education, chapter 8, 99-110, ed S Brown, Kogan Page, 1997.

[6] Drakos, N., The Latex to HTML Translator, Computer Based Learning Unit, University of Leeds, URL: http://cbl.leeds.ac.uk/|nikos/tex2html/doc/latex2html/, Jan 1994.

[7] Giarratono, J and Riley, G., Expert Systems: Principles and Programming,
PWS Publishing, ISBN 0-534-93744-6, 1994.

[8] Hickey, T "Jscheme: an applet for teaching programming concepts to non-majors", 1998
http://www.cs.brandeis.edu/~tim/Packages/Jscheme/Papers/jscheme.html

[9] Lutz, M., Programming Python, O'Reilly & Associates, Sebastopil, CA, (ISBN 1-56592-197-6), 1996

[10] Nolan, P,J and Howie, N, "Prolog Based Diagnosis of Power Plant Alarm Systems", Proceedings of Tenth International Conference on Application of Artificial Intelligence in Engineering, pp 199-208, Computational Mechanics Publications, Southampton, 1995

[11] O'Shea, T & Self, J, Learning and Teaching with Computers, Harvest Press, John Spiers Publishers, Sussex, 1983.

[12] SICStus Prolog Manual, Published by Swedish Institute of Computer Science, PO BOX 1263, SE 164 29 Kisti, Sweden,

[13] Watson, M "Jess" in book Intelligent Java Applications for the Internetand Intranets, published by Morgan Kaofman, ISBN 1558604200, 1998