TEACHING AN APPLIED EXPERT SYSTEMS COURSE: A CONTENT OUTLINE

Dr. Jay Liebowitz
Professor of Management Science
School of Business and Public Management
George Washington University
Washington, D.C. 20052

ABSTRACT: As expert systems become more commonplace in the business curriculum, it is helpful to learn from others in teaching the expert systems course. Over the past ten years, the author has been teaching an Applied Expert Systems course in the School of Business and Public Management at George Washington University. This paper shares some of the lessons learned from these experiences in terms of administrative-, content-, and project-related issues and considerations associated with teaching the expert systems course.

KEYWORDS: Expert Systems, Expert Systems Education, Knowledge Engineering, Applied Artificial Intelligence, Knowledge-based Systems

INTRODUCTION

Expert systems courses are regularly being taught in many business, engineering, and liberal arts universities worldwide. In business schools, expert systems were originally introduced as part of a course in decision support systems. Gradually, expert systems courses emerged as standalone courses in business schools. Now, universities, like George Washington University and others, have introduced several applied artificial intelligence (AI) courses into their undergraduate business program as well as a Master of Science program in Applied Artificial Intelligence and Expert Systems in the graduate business curriculum.

This paper will address some lessons learned through the author's ten years of experience in teaching an Applied Expert Systems course in the business program. The lessons will be grouped into the following categories: administrative, content-related, and project-related lessons. Hopefully, these lessons and guidance will help those teaching or considering to teach an expert systems course in the business program.

LESSONS LEARNED

Administrative Lessons

The expert systems course in the business program should serve several purposes:

1. to expose the student to the knowledge engineering methodology in building expert systems
2. to provide an awareness of terminology, concepts, advantages, applications, limitations, and trends of expert systems
3. to allow the student to gain hands-on experience in building an expert system prototype to reinforce the lecture material.

To meet these goals, it is helpful to have examinations to test the student on the knowledge engineering concepts in the expert systems field and to require the development of an expert systems prototype by the student with a presentation on the topic [a possible breakdown of the grade could be 30% for the midterm, 30% for the project, 30% for the final examination, and 10% for the presentation]. It should be encouraged that the examinations be applied-oriented, such as showing a transcript between a knowledge engineer and an expert and then asking some comprehensive questions to the student relating to the knowledge engineering methodology (which will be explained in the next section).

The applied expert systems course should require that the student build an expert systems prototype and interact with a domain expert throughout its development. On the administrative side, an expert system shell should be used to facilitate the construction of the expert systems prototype. A number of books are packaged with expert system shells [1-4,13] or there are foundation texts [5-10,14] that could be used if the university already has access to expert system shells or the student could pay a small price to procure a shell (for example, the Exsys Professional shell demo, which can save up to 50 rules, can be purchased for $25 including the manual). Whatever strategy is used in the course in regards to the shell, it is essential that the student have easy access to the shell and the shell should not be difficult to

use. Since the student will typically have only 7 to 8 weeks to develop the expert systems prototype (discounting introductory lectures and examinations), the student should not be bogged down with the mechanics of working the shell. Instead, the student should concentrate on the knowledge engineering life cycle involved in building the expert system, with using the shell as an easy means to implement some of the strategies learned. The expert systems prototyping project is an important ingredient in reinforcing the concepts presented in class.

Besides some administrative lessons learned, we can now look at course content-related issues.

Course Content-Related Lessons

An essential component of the applied expert systems course is a knowledge engineering methodology used for developing expert systems. The professor should present a methodology for expert systems construction. The iterative, rapid prototyping approach that the author uses is: problem selection, knowledge acquisition, knowledge representation, knowledge encoding, knowledge testing and evaluation, and implementation and maintenance [5]. Each of these steps, with helpful hints, will be presented in turn [11,12].

First Step: Problem Selection

The first step, and the most important, in knowledge engineering is selecting the ''right problem.'' This is particularly important for the first expert system that you develop because if the application is inappropriate, then you will frustrate not only yourself but also your ''client'' (in the classroom environment, the professor). For classroom purposes, it is very helpful to keep the following tips in mind:

- pick a problem in which you are already familiar with the domain: by doing this, you will save alot of time by not having to first learn the general ideas, terms, and acronyms used in the domain. Since you may only have about 7-8 weeks in class to actually develop your expert system prototype, familiarization with the problem domain will save you much time during the knowledge acquisition step.
- pick a problem that is causing a large number of people, a fair amount of grief: by selecting a task for expert system development that warrants a better solution than what is currently being done, you may find alot of support and enthusiasm for the project. Don't pick a problem that seems too trivial because it might show that expert system technology is uninteresting.
- select a ''doable'' problem as the first expert system project to win the support of management: the general rule is that you should pick a problem that you think is too small to handle, and you will probably have to scope that down further to develop your prototype.
- get an expert who is willing to work with you: the commitment from the expert is one of the most important factors in developing a successful expert system.
- have a backup expert: in case something were to happen to the expert, it might be useful to have a ''backup'' expert. You will need another expert anyway to validate your prototype.
- find out from the expert what is the expected accuracy rate of the prototype: this will help you later during your validation phase.
- do not overpromise: expert system technology and the shell that you are using have various limitations, such as the inability for the expert system to learn, a limitation on the number of rules that it can handle, limitations on the user interface, restrictions on the different ways the expert system can handle uncertainty, etc. When you are speaking with the expert, point out some of these limitations but also discuss the advantages of expert systems, such as building the corporate memory, freeing up the expert's time to pursue other areas of professional interest, savings in cost and time, supplementing the expert's decision making, allowing for training, and improving one's productivity.
- use the rapid prototyping, iterative approach: build a little, test a little. This approach will allow you to incrementally refine the knowledge in your expert system.

The next step in the knowledge engineering process is knowledge acquisition.

The expert systems prototyping project is an important ingredient in reinforcing the concepts presented in class.

Second Step: Knowledge Acquisition

One of the most difficult tasks to accomplish in the expert systems development life cycle is knowledge acquisition. In order for the student to obtain a feeling of the difficulty doing this process, the student should be required to use a domain expert during the development of the expert system prototype. This will allow the student to gain a better appreciation for why knowledge acquisition is considered the biggest bottleneck in expert system development. The following are some useful tips for the student during this knowledge acquisition process:

- before interviewing the expert, make sure that you are familiar/ comfortable with the domain: this may allow you to ask intelligent questions and understand what the expert is saying to you.
- the first session with the expert should be an introductory lecture on the task at hand: by having the expert provide an introduction to the

expert system task, you will realize several points: (1) you will realize the bounds on the task and if you need to further scope the problem; (2) you will see if you have enough initial knowledge to understand what the expert is describing to you; and (3) you will see if the expert is the "right expert" for your task--is he/she articulate and organized?

- have a systematic approach to acquiring knowledge: it is very helpful for you, as the knowledge engineer, to use some framework, like an attribute hierarchy [5], for structuring the expert's acquired knowledge. An attribute hierarchy provides a pictorial representation of the important characteristics that lead up to the goal of what the expert system is supposed to do. This then serves as the framework for structuring the knowledge base.
- incorporate the input and feedback from the expert into the expert system: get the expert enthusiastic about the project and show each version of the system to the expert (one caveat is if you show a version of the expert system too early in its initial development, the expert may think that there is not much there!); try to make the expert feel that the system is his/her "baby"--you can do this by even naming the expert system after the expert.
- pick up manuals and documentation on the subject matter: by doing this, you will acquire more background knowledge about the task and this will also help you formulate questions for future knowledge acquisition sessions.
- tape the knowledge acquisition sessions, if the expert allows: you won't be able to write down everything during the knowledge acquisition sessions, so a tape recorder will aid you; also, there are certain intonations by the expert which are important and a tape recorder will allow you to pick up on these.

After performing an initial set of interviews with the expert, the next step in the knowledge engineering process is knowledge representation.

Third Step: Knowledge Representation

After acquiring the knowledge, the next step in the knowledge engineering process is knowledge representation. This involves representing the knowledge in the knowledge base as rules, frames, scripts, semantic networks, or some hybrid. There are some useful tips to remember when considering knowledge representation:

- try to use the representation method which most closely resembles the way the expert is thinking and expressing his/her knowledge: if the expert is talking in terms of if-then clauses (as in this particular sentence!) then you might think about using rules to represent that knowledge. If the knowledge is more descriptive and context-dependent, then the use of frames and semantic networks may be a favored representation scheme to use.
- consider whether uncertainty should play a part in your system: most expert systems handle uncertainty, but it is not essential to include it if it is in appropriate for your application. Again, the expert should provide some insight into this subject.
- consider if the expert reasons in a data-driven manner or a goal-directed manner, or both: this will tell you, as the knowledge engineer, if you need forward chaining (data-driven), backward chaining (goal-directed), or both control strategies.

After representing the knowledge, the next major step in the expert systems development life cycle is knowledge encoding.

Fourth Step: Knowledge Encoding

The next step in the knowledge engineering process is knowledge encoding. This step entails using your expert system shell/programming language to encode the knowledge that you just represented. There are a few helpful points that come under this category:

- remember the motto, "For every shell there is a perfect task, but for every task there is not a perfect shell": remember to first determine the requirements of the task, instead of force-fitting a shell to a task (the author realizes that there might be only a limited number of shells that the university has for the expert system course, but this problem requirements analysis should still be performed).
- try to develop the knowledge base in a modular format for ease of updating: just as you might use subroutines in a program for ease of development and maintenance, you should similarly construct the knowledge base to facilitate ease of updating and maintenance.
- you need an appropriate and adequate user interface, but concentrate on the knowledge base: the power of the expert system lies in the knowledge, although ultimate acceptance of the expert system is heavily dependent on its user interface; for the classroom, however, the development of the knowledge base should be more important than the user interface (accept the user interface of the shell as is).
- use an incremental, iterative approach to developing your expert system prototype: again, the "build a little, test a little" method of refining the knowledge is a useful way of building expert systems.

An expert system shell is used in the class, instead of programming from scratch in Lisp, Prolog, C, or some other language, because the shell greatly facilitates the development of the expert system prototype within the constraints of a semester course. The professor should, however, include a lecture on Lisp and Prolog so that the student

is at least aware of the usefulness of each language. By using a shell, the student can concentrate on the expert system life cycle development methodology instead of being engrossed in the simple mechanics of programming. If the student is very interested in artificial intelligence/expert systems after this introductory course, then it is advisable for that student to take next a Lisp and/or Prolog programming course.

The next major step after knowledge encoding involves the testing and evaluation of the knowledge and resulting expert system prototype.

By using a shell, the student can concentrate on the expert system life cycle development methodology instead of being engrossed in the simple mechanics of programming.

Fifth Step: Knowledge Testing and Evaluation

After knowledge encoding, the next important steps in the knowledge engineering process are knowledge testing and evaluation. Some useful guidelines are outlined below:

- for verification and validation of the expert system prototype:
  - -- check for consistency of the knowledge/logic
  - -- have a representative set of test cases (hard and soft cases and special subcases)
  - -- perform backcasting: run the expert system against documented cases and compare the results with historical results
  - -- use blind verification tests
  - -- have the expert and other experts test the system.
- for evaluation of the expert system prototype:
  - -- have users evaluate the human factors design of the system (i.e., instructions, free-text comments, ease of updating, exiting capabilities, response time, display and presentation of conclusions, ability to restart, ability for user to offer degree of certainty, graphics, ability to back up to a previous question, etc.).

After performing knowledge testing and evaluation, the last step in the expert systems development life cycle is implementation and maintenance.

Sixth Step: Implementation and Maintenance

Even though your students may only be developing a prototype, they still should know about the implementation and maintenance step of expert systems. A few points are highlighted below:

- train the users on how to use the system.
- deliver good documentation on the system.
- assign a person or group to maintain the system.
- instruct the maintenance team on how to maintain the system/use the shell.
- don't forget to consider the cost of a runtime license so that multiple copies of the expert system could be used throughout the organization.

Project-Related Lessons

In order to reinforce the concepts presented in the lecture, each student is required to build an expert system prototype using an available expert system shell [11]. Since the course stresses the knowledge engineering process of expert systems and since it is an applied course, a microcomputer-based expert system shell is used to facilitate the development of the expert system prototype. Shells that have been used for class projects include Exsys, VP-Expert, Guru, Level 5, and others.

Through the hands-on experience in developing a prototype, the student really gains a great appreciation for the expert system development process. In particular, each student is required to use a "domain expert" during the knowledge acquisition step, as well as the other steps in the expert system life cycle, and the student quickly discovers the difficulties in trying to acquire knowledge from the expert. Appendix A shows a sample of student expert system prototype projects developed during the Applied Expert Systems class.

There are six assignments relating to the construction of the expert system prototype. The six assignments include:

1. Write a one or two page proposal on the topic for expert systems prototyping and discuss why it is an appropriate subject for expert systems development (PROBLEM SELECTION ASSIGNMENT).
2. Meet with the domain expert, and develop an initial attribute hierarchy based on the discussion(s) with the expert. An attribute hierarchy is a pictorial representation of the relationships between the important characteristics that lead to the goal of the expert system. This attribute hierarchy later serves as the framework on which to construct the knowledge base. Also, the student turns in a tape of the knowledge acquisition session (which the professor critiques during the class) and a write-up of the successes and failures of the first knowledge acquisition session (KNOWLEDGE ACQUISITION ASSIGNMENT).
3. The third project involves writing out the rules on paper based upon the attribute hierarchy (KNOWLEDGE REPRESENTATION ASSIGNMENT).
4. Encode the rules using the expert system shell and turn in a printout of the knowledge base and two sample

user sessions (KNOWLEDGE ENCODING ASSIGNMENT).

5. Further refine the knowledge base, and incorporate human factors features into the expert system (such as instructions, free-text comments, notes and references on rules, etc.). Turn in the knowledge base and two sample user sessions of the expert system prototype (KNOWLEDGE ENCODING ASSIGNMENT CONTINUED).
6. Last, perform formal testing and evaluation of the expert system prototype. Perform verification and validation of the expert system prototype in order to check for logical consistency and accuracy in advice, respectively. Use a naive user, a knowledge user, your expert, and another expert in the testing and evaluation of the expert system prototype. Evaluation includes an analysis of the human factors features built into the expert system prototype. Turn in a four to five page write-up of the testing and evaluation results, and a disk of your final knowledge base.

These assignments have worked quite well because it forces the student to keep up with his/her expert system prototyping and the professor can also make sure, at each step, that the student is heading in the right direction.

SUMMARY

This paper provides some useful lessons for those involved in contemplating or already teaching an applied expert systems course in the business school. The expert systems project that the student selects should greatly reinforce the concepts and methodologies presented during the lectures. In fact, many students have mentioned that this hands-on experience was extremely valuable for convincing their organizations to pursue expert systems technology. This expert systems course should also highlight the management and institutionalization issues relating to expert systems programs and projects. For a business school course, in particular, it is important to emphasize the "technology" and the "management of the technology" as vital complementary components necessary for the successful development and deployment of expert systems within an organization. If these guidelines are followed, the expert systems course will surely be enjoyable to both the student and professor.

Appendix A: Sample of Student Expert System Projects:

- bid advisor for requests for proposal
- tax form selection advisor for individuals
- graduate course advisor
- network fault diagnosis advisor
- individual's investment portfolio advisor
- advisor to train military officers in appropriate courses of action for battle management
- advisor to classify error conditions in the tax forms for IRS
- expert witness selection advisor in food and drug law cases
- credit collection advisor
- computer configuration advisor
- advisor for diagnosis of and treatment for periodontal gum disease
- DOS Help advisor
- new venture capital investment advisor
- real estate selection advisor
- disaster management staffing advisor
- bankruptcy advisor

REFERENCES

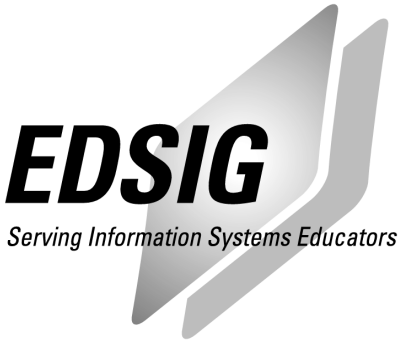
1. Mockler, R. and D. Dologite, Knowledge Based Systems: An Introduction to Expert Systems, Macmillan Publishing, New York, 1992.
2. Turban, E., Applied Artificial Intelligence and Expert Systems, Macmillan Publishing, New York, 1992.
3. Mockler, R., Developing Knowledge Based Systems Using an Expert System Shell, Macmillan Publishing, New York, 1992.
4. Dologite, D., Developing Knowledge Based Systems Using VP-Expert, Macmillan Publishing, New York, 1992.
5. Liebowitz, J., Introduction to Expert Systems, McGraw Hill, New York, 1988.
6. Liebowitz, J. (ed.), Expert Systems for Business and Management, Prentice Hall, Englewood Cliffs, NJ, 1990.
7. Liebowitz, J. and D. DeSalvo (eds.), Structuring Expert Systems:Domain, Design, and Development, Prentice Hall, Englewood Cliffs, NJ, 1989.
8. Turban, E. and J. Liebowitz (eds.), Managing Expert Systems, Idea Group Publishing, Harrisburg, PA, 1992.
9. Liebowitz, J., Institutionalizing Expert Systems: A Handbook for Managers, Prentice Hall, Englewood Cliffs, NJ, 1991.

10. Liebowitz, J. (ed.), Operational Expert System Applications in the United States, Pergamon Press, New York, 1991.
11. Liebowitz, J., "Introducing An Expert Systems Course into the Operations Research Curriculum," Computers and Operations Research, Vol. 17, No. 6, Pergamon Press, Oxford, 1990.
12. Liebowitz, J., "A Sample of Projects for an Applied Expert Systems Course," Interface: The Computer Education Quarterly, Vol. 10, No. 4, Mitchell Publishing, Watsonville, CA, Winter 1988/1989.
13. Liebowitz, J., A Seven Week Classroom Quick Guide to Developing Your First Expert System Prototype, Macmillan Publishing, New York, in press.
14. Medsker, L. and J. Liebowitz, Expert Systems and Neural Networks, Macmillan Publishing, New York, in press.

AUTHOR'S BIOGRAPHY

Dr. Jay Liebowitz is Professor of Management Science at The George Washington University. He is also the Program Director for the Applied Artificial Intelligence and Expert Systems Program at George Washington University. He has gained experience in expert systems technology through his work at NASA Goddard Space Flight Center, Navy Center for Applied Research in Artificial Intelligence, MCI, ICF, Encore Computer Corporation, American Management Systems, Inc., and others. Before joining George Washington University, he worked at NASA Goddard; Computer Sciences Corporation; Peat, Marwick; and the Program Analysis and Evaluation Division at the Pentagon. He is the Editor-in-Chief of Expert Systems With Applications: An International Journal and the Associate Editor of the International Journal of Telematics and Informatics. He is on the Editorial Advisory Boards of the International Journal of Robotics and Computer-Integrated Manufacturing, International Journal of Computers & Operations Research, International Journal of Computers & Education, International Journal of Computers & Industrial Engineering, AI Abstracts, Heuristics, and CETTICO AI Journal. He is a member of Sigma Xi, Tau Beta Pi, Omega Rho, Omicron Delta Kappa, AAAI, IEEE, ACM, DPMA, and TIMS. He has published 12 books and over 100 journal papers, mostly in expert systems. He is the Congress Chairman of the first ever World Congress on Expert Systems.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096