STUDENT LEARNING PROBLEMS IN A COURSE ON EXPERT SYSTEM SHELLS

by Dr. Fritz H. Grupe
Department of Accounting and Information Systems
University of Nevada, Reno

ABSTRACT: This paper describes salient problems encountered by students enrolled in a course on the use of expert system shells in business that was offered at the University of Nevada-Reno. The paper identifies the major problems encountered by students as being: A) The learning of new programming concepts; B) "Freeing" the mind from procedural thinking; C) Becoming clear about the use of certainty factors; D) Adapting to incremental design; E) The selection of an appropriate problem; F) Thinking like an expert; G) Lack of models and limitations of tutorials;H) Tracing and debugging. These problems can be addressed by instructors in various ways. Several instructor-oriented recommendations are made to address these problems, although simply being aware of some of the problems students in these courses will face will be of assistance to instructors preparing class lectures and materials.

KEYWORDS: Expert systems, artificial intelligence, computer information science, student learning.

INTRODUCTION

The Course

Expert systems are computer programs which incorporate human knowledge and reasoning to emulate the behavior of a human expert. The use of expert system shells is one of the fastest growing areas for the application of computers today, accounting for much of the growth of interest in artificial intelligence. (1) It was felt that a course that considered the historical and contemporary development of expert systems in the modern business environment was needed, especially one that stressed the actual use of expert systems technologies. The need for artificial intelligence and expert systems courses has been cited frequently in the literature regarding computing in the liberal arts,(2,3) in business,(4,5) as well as in engineering and computer science.(6) The course discussed in this article, "Using Expert System Shells in Business," has been offered twice at the University of Nevada-Reno. It was designed to address the DPMA model curriculum(7) and includes:

1. A review of the major artificial intelligence projects that preceded contemporary expert system shells.
2. An understanding of the basic concepts and techniques underlying the development of expert systems.
3. An exposure to the nature and availability of artificial intelligence languages, tools, environments and shells.
4. The modeling of problem-solving activities in the context of several expert system shells.
5. Problems and opportunities that attend the implementation of expert systems in corporate settings.
6. The management structures used to implement artificial intelligence activities in various types of organizations.

Instructional Materials

The basic text was Expert Systems Tools and Applications.(8) Supplementary readings were assigned. Software manuals and tutorials were available for the software. Texas Instruments has broadcast four satellite seminars on artificial intelligence. Students attended or viewed selected segments of these programs.

Software

The primary software used, Personal Consultant Easy (PC Easy), is sold by Texas Instruments and is sufficiently full-featured to give students an exposure to a strong, commercially viable expert system. This software is not so overwhelming as to require too much of the course to be devoted to learning the shell's features. It should be said that while PC Easy is easier to learn than PC Plus, a more sophisticated shell, it is not as simple as many other shells. It does provide many of the features cited by Sondack(9) as being desirable for shells used in a business course. Among the features are:

1. Backward and forward chaining,
2. Ability to incorporate graphics and color in user prompts, questions and conclusion screens,
3. Ability to access and update dBASE files, Lotus files, external programs, etc.

4. Heuristic drawing of conclusions with predicate functions, a CONCLUDE operator and uncertainty factors.
5. Varied parameter types,
6. Why and prompt screens,
7. Conclusion screens with use of text functions,
8. Multiple goals,
9. Multiple rule groups,
10. An intelligent editor,
11. Good tutorial and reference manuals.

The students who enroll in this type of course are likely to be totally unfamiliar with expert systems technologies and how they are used, at least for a while.

Of these features, the most important was considered to be the use of certainty factors and the operations that allow the knowledge worker to manipulate them. The relatively low cost for a site license ($2,500) made the software readily available for this and other courses, as well as for research projects.

Copies of VP Expert, Auto-Intelligence, Exsys and First Class were placed in a microcomputer laboratory. Usage of these shells was limited to the tutorials in their manuals. They were used to give students a feeling for some of the other systems available. The student "deliverable" was a product evaluation.

Semester Projects

Each student completed one independently identified expert system application which exhibited diverse use of the features available under PC Easy. An in-class demonstration of the operation of the system was given. Early emphasis was placed on the selection of an appropriate problem domain for the students' expert systems. The instructor obtained periodic submissions of projects to assure that progress was being made and that students were grasping expert system programming insights.

Projects that were submitted originated in such domains as welding problem diagnosis, racing bicycle configuration and selection, logging advisement, mining road ramp construction and wild fire fighting recommendations.

STUDENT PROBLEMS ENCOUNTERED

The development of applications for student programs appeared to present several key problems which instructors of this type of course should anticipate and, as possible, redress before they become too severe. Programming with an expert system shell is not the same as programming in COBOL, FORTRAN or Pascal. The students who enroll in this type of course are likely to be totally unfamiliar with expert systems technologies and how they are used, at least for a while. They almost certainly have never constructed one. Even students who have been exposed to LISP and Prolog will have only a limited perception about how to utilize a shell like PC Easy effectively. Consequently, students will need assistance in finding their way through these problems. Although I have crystallized the primary problems for this paper, it will be obvious that there is a significant amount of interaction between them.

Problem 1: Learning new programming concepts.

There is, of course, the need for the students to obtain new insights into programming with this technology. Distinguishing between an expert systems' parameters and the variables used in traditional programming languages, and understanding backward and forward chaining are among the core concepts that require students to temporarily "unlearn" notions that don't hold the way they did in their earlier programming experiences.

Also, the absence of familiar control structures like the DO-WHILE and the CASE statements cause students to have to re-think how to solve their programming problem. For instance, PC Easy does not have a method of directly assigning a value to a parameter. All assignments have to be made in a rule.

Problem 2: "Freeing" the mind from procedural thinking.

Students at this level have been thoroughly pressured to build programs that are heavily modular. Each module is to have one entry and one exit. Students can visualize data moving through their programs in a flow chart-like fashion. They can trace the movement of data and actions through their programs.

Knowledge bases built with PC Easy are less easily modularized. The nature of backward chaining is such that while rules can be grouped there is no guarantee that they will be tested in any particular order or that they will be tested at all. Rules may also have consequences that have wide ranging effects. For example, a rule that assigns too much certainty to a single valued parameter may terminate a consultation before many other rules are tested.

Problem 3: Becoming clear about the use of certainty (confidence) factors.

As noted, PC Easy allows extensive use of certainty factors (CFs). CFs come into play in the premises of rules, in the consequences of rules, in the interaction of CFs assigned in different rules, in user input and in a range of predicate functions used to examine and test CFs. Because many persons cite the ability of expert systems to reach conclusions with unknown and uncertain data as being among their more important features for particular classes of problems,(10) the effective use of CFs was treated as an important component of the course project.

The production of the students' expert systems went through several phases. In the first phase students tended to try to write rules with excessive numbers of multiple conditions and one conclusion. For example, a rule might be in the form,

IF W AND X AND Y THEN Z

In the early stages of the development of an expert system this form of rule is satisfactory, but as new parameters are added the combinatorial explosion of rules becomes clearly unmanageable. More importantly, with respect to CFs, the *weakest* certainty attached to the parameters w, x and y will be assigned to z, a result not generally contemplated by the students. ORs assign the strongest CF.

Another area where a CF problem appeared was in the tracing of rules. Students expected that if a positive response was given to a question (i.e. a positive CF was assigned to a value in a property list), then a test of that parameter with the equal sign should always succeed. It doesn't. The equal sign in PC Easy translates to, "has a CF in the range of 21 to 100," consequently the rule,

IF W THEN Z

fails if the CF for w is 20. Space does not permit the expansion of this problem, but suffice it to say that students have to go through a learning curve to comprehend the ramifications of the use of CFs.

<u>Problem 4: Adapting to incremental design.</u>

The predominant view of expert system development is that systems must be developed incrementally. Top-down design, decomposition of a problem, modularization, etc. are difficult to implement in expert systems written with PC Easy, although some other shells encourage this type of design. Each phase of a model's existence is used to nurture the development of an expanded model. Early mistakes may require that the system under development be thrown out and started again. The use of CFs makes design especially difficult.

<u>Problem 5: Selecting an appropriate problem.</u>

To build an expert system, one must be expert in some problem domain. Some of the students, in fact, were experts in some aspect of their lives while others felt they did not know enough about any field to write an expert system. The quality of the projects were significantly affected by the initial problem selected. The necessity for selecting an appropriate problem was also cited by Sondack.(11)

<u>Problem 6: Thinking like an expert.</u>

Again, many of the initial rules written by students were of the form,

IF W AND X AND Y THEN Z

Experts also reach conclusions that are less than certain. Students tend to write rules that are completely certain.

Experts tend to reach many conclusions not just one. Consequently, many rules should be of the form,

```
IF W THEN Z1 CF 30 AND
        Z2 CF -30 AND Z3 CF 100
```

in which multiple conclusions about the same or different parameters are reached simultaneously. Experts also reach conclusions that are less than certain. Students tend to write rules that are completely certain.

Another aspect of expert decision-making is the use of multiple means of obtaining and inferring conclusions. Most students tend to have one rule that concludes a value for a parameter. Still another aspect is the expert's ability to reach conclusions with functions like,

IF X IS NOT DEFINITE THEN Z

Students tend to rely exclusively on "equal to" or "not equal to," even though these are restricted in their value. Similarly, yes/no questions are over used and inadequate use is made of more symbolic parameters.

<u>Problem 7: Lack of models and limitations of tutorials.</u>

Students benefit from reading the programs of others in order to see what their final products should look like.(12) Texas Instruments staff were helpful in this respect by providing a copy of the Ford Motor Company Robot Maintenance Expert System for examination. The instructor provided several sample systems and the availability of student-developed systems in the second offering of the course was helpful.

PC Easy provides a book, Getting Started,(13) that walks the learner through a series of tutorial exercises. This tutorial is instructive, but limited. There are many features available that students do not use in the tutorials. Specific examples of the more advanced, more difficult to apply techniques are only discussed in the reference manual. Although college-level text books written around PC Easy are promised to be available shortly, they did not exist at the time this course was offered.

<u>Problem 8: Tracing and debugging.</u>

Expert system shells provide minimal levels of assistance in debugging programs. PC Easy offers an on-screen facility which indicates which parameters are being traced and which rules are being tested and fired. A review facility will identify how a conclusion was reached. But since this facility may display a large number of rules that had an effect on the conclusion, the precise identification of the program's semantics was difficult for some students. The usefulness of these utilities comes with time, however, and the absence of some of the traditional snapshot routines and debugging facilities make the understanding of how the program is operating difficult.

ADDRESSING THESE PROBLEMS

Student problems attend the offering of any programming course and not all of these can or should be the responsibility of the instructor to eliminate. Instructors can provide some assistance that will allow students to focus on the problem more directly, and to write more thorough expert systems, however. The following recommendations are offered for consideration:

A. Provide the students with completed, non-trivial models to review and analyze. PC Easy and many other shells provide some examples of expert systems, but most of these are unlikely to be enlightening to computer science majors.

B. Although expert systems applications can be classified into many different categories such as interpretation, prediction, design, monitoring, debugging, repair, instruction, planning, and diagnosis/prescription,(14), the easiest form for most students to grapple with is the diagnostic prescriptive model. This form begins to suggest modularization at least at two levels: rules for diagnosing the nature of the problem and rules for prescribing a solution.

C. The class enrolls students who are already knowledgeable about computers even though they may not be "experts." The one discipline that most of them can grapple with at an advanced level is computer science. The instructor provided the students with a list of about 40 types of hardware devices and software packages and suggested that the students obtain their expertise from computer literature that reviewed specific products in each category and from interviews with persons knowledgeable about their product category. The consultations would assess the nature of a user's needs for a category of product such as a modem or laser printer and then would recommend a specific modem or laser printer by brand name and model along with information about that product.

D. Early in the exploration of the problem, students were encouraged to think about the categories of rules that were needed. PC Easy supports rule groups that assemble all rules dealing with a parameter into one location. This is not equivalent to a module that forces consideration of all of the rules nor does it guarantee that the rules will all be used. It does stimulate students to see if they are accounting for the major parameters. It also suggests how they can enter rules in sets that can be tested before the entire rule base is attempted.

E. Specific assistance must be given to students on a one-on-one basis to identify a problem that can effectively be coded in the shell being used and that can be extended to permit usage of advanced programming features. Some instructors may prefer group projects of larger scope.

F. Provide adequate laboratory instruction in debugging and tracing techniques.

G. The major project should be submitted for instructor evaluation several times as it is being developed. The students need milestones against which to assess their progress. More importantly, since each project is unique, individualized assistance must be given to students so they can understand how their programs can be revised to include advanced features and to go beyond trivial implementations.

H. Students demonstrated their projects to the class at several points during the semester. This gave them milestones to guage their progress and it gave the instructor a firm point where constructive assistance could be given.

I. Project grade points were assigned to reflect a student's usage of a variety of functions, rules, operators and screen formatting techniques. High points could not be earned simply by amassing rules of a trivial type. For instance, one project that had the largest number of rules received the lowest grade because the rules were all of the same type. Instead, to earn points students had to use operators like IS MIGHTBE or ISDEFIS and they needed to utilize a wide variety of features such as multiple conclusions in the action part of rules, user entry of uncertain input, rule descriptions, instances of backward chaining that went more than one level deep and certainty factor calculations that involved calculations from a variety of rules with different parameters in the premise and both forward and backward chaining rules. The point assignments were made known in advance of project submission to direct student energies into utilizing the functions.

J. Finally, although it was expected that CF calculations had to be employed, it was not required that the CF calculations had to be fully defensible. Students cannot be expected to have sufficient command of the problems they selected to validate CF assignments and manipulations. There appears to be evidence that the logic of the program is more important than the precision with which CFs are calculated.(15)

REFERENCES AND FURTHER READING

1. Feigenbaum, E.; McCorduck, P. and Nii, P. The Rise of the Expert Company, Times Mirror Books, 1988.
2. Skala, H. "An Artificial Intelligence Course for Liberal Arts Majors," Collegiate Microcomputer, Vol. 6, No. 4, November 1988, pp. 311-316.
3. Liebowitz, J. "Artificial Intelligence is for Real: Undergraduate Students Should Know About It," Collegiate Microcomputer, Vol. 6, No. 1, February 1988, pp. 87-91.
4. Eliot, L. "Expert Systems in a Graduate Level Course: A Business-Oriented Approach," Association for Information Decision Sciences,

Proceedings, 1986, pp. 22-24

5. Sondack, N. E. "Laboratory Support for Business-Oriented AI Courses," Proceedings, Western Educational Computing Conference, California Educational Computing Consortium, 1986, pp. 101-105.
6. Medsker, L. R.; Morrel, J. H. and Medsker, K. L. "Expert Systems: Meeting the Educational Challenge," Collegiate Microcomputer, Vol. 5, No. 3, August 1987, pp. 223-229.
7. Data Processing Management Association Model Curriculum for Undergraduate Computer Information Systems, (CIS 86), DPMA, 1986.
8. Harmon, P., Maus, R. and Morrissey, W. Expert Systems Tools and Applications, Wiley & Sons, 1988.
9. Sondack, N. E. "Laboratory Support for Business-Oriented AI Courses," Proceedings, Western Educational Computing Conference, California Educational Computing Consortium, 1986, pp. 103.
10. Holsapple, C. W. and Whinston, A. B. Business Expert Systems, Irwin, Homewood, IL, 1987, pp. 121.
11. Sondack, N. E. "Laboratory Support for Business-Oriented AI Courses," Proceedings, Western Educational Computing Conference, California Educational Computing Consortium, 1986, pp. 103.
12. Weinberg, G. M. The Psychology of Computer Programming, Van Nostrand, New York, 1971, pp. 6.
13. Personal Consultant Easy: Getting Started, Texas Instruments, Austin, TX, 1986.
14. Hayes-Roth, F.; Waterman, D. A. and Lenat, D. B. Building Expert Systems, Addison Wesley Publishing, 1983, pp. 14.
15. Hayes-Roth, F.; Waterman, D. A. and Lenat, D. B. Building Expert Systems, Addison Wesley Publishing, 1983, pp. 93.

AUTHOR'S BIOGRAPHY

Fritz Grupe is an Assistant Professor of Computer Information Systems at the University of Nevada, Reno. He has taught courses in computer science, computer information systems and computers in education. He served as the Coordinator for Campus Computing Services at the University for six years and is the author of a series of books on microcomputer applications in business (W. C. Brown Publishers).

ISSN 1055-3096