The Graduate Student Advisor (GSA): An Expert System for SAN Graduate Student Advising

Jiazhu Zhang
Miami University, commons-admin@lib.muohio.edu

TECHNICAL REPORT: MU-SEAS-CSA-1992-016

The Graduate Student Advisor (GSA): An Expert System for SAN Graduate Student Advising

Jiazhu Zhang

The Graduate Student Advisor (GSA): An Expert System for SAN Graduate Student Advising

by

Jiazhu Zhang
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-016 December, 1992

The Graduate Student Advisor (GSA): An Expert System for SAN Graduate Student Advising

Jiazhu Zhang

Department of Systems Analysis
Miami University

Abstract

The Graduate Student Advisor (GSA) is an experimental expert system that advises graduate students in systems analysis. It simulates a faculty advisor in suggesting the schedule of courses a student should take based on the student's background and interests. It is implemented in NEXPERT Object. This report first describes the task, knowledge engineering and solution approach of the GSA. The report then gives a sample session to illustrate how to run a consultation. It also includes some maintenance notes about how to modify the knowledge about courses and related rules if necessary. Finally, it discusses some lessons learned.

1. Introduction

The Systems Analysis Master's Degree Program at Miami offers a blend of computer science, information systems and operations research. The program curriculum consists of four elements: Foundation Courses, Core Courses, Elective Courses and Graduate Research Courses. A graduate student must complete at least 4 courses(12 credit hours) from each of Core and Elective elements and at least 2 graduate research courses (6 credit hours). Successful completion of a minimum of 30 credit hours is required for the degree. Additional foundation courses may be necessary depending on the student's different interests and background.

Currently, there are approximately 40 graduate students enrolled in the Systems Analysis Graduate Program. The number is expected to be increasing in the coming academic year. Due to the conversion nature of the program, the students come from a variety of backgrounds. Some may have backgrounds in a systems-related field such as systems analysis or computer science. Some may have backgrounds in a non-systems field such as geology or geography. There are many other different possibilities in between these two extreme cases. According to the program, students must have had: 1 communication course, 2 calculus courses, 2 probability and statistics courses, and 1 computer programming course. Some students may have to make up some of the undergraduate deficiency courses before they take any core and/or elective courses. Also, most students pursue their graduate study on a full-time basis though there are some part-time students. Students are supported through various financial means (e.g. graduate assistantship, grant-in-aid or scholarship from an organization). The minimum number of credit hours for which a full-time student must register and the maximum number of credit hours for which the student may register in a regular term vary depending on the financial support means. All of these make advising students even more complicated.

Advising students is a complex and time consuming task. However, a large percentage of an adviser's time is spent with fairly repetitive activities. For example, the process for determining course prerequisites and planning programs of study may seem confusing to the individual student. The Graduate Student Adviser (GSA) is an experimental expert system that simulates a faculty adviser in suggesting the schedule of study a student should take for obtaining the degree. Its purpose is to provide students with the curricular knowledge and information on a uniform and consistent basis, thus making more efficient use of the graduate adviser's time.

2. Knowledge Structure and Components

GSA's approach is largely data driven; it begins with a set of "best" courses and tries to produce a schedule within the constraints imposed by the properties of the courses, relationships between them, and restrictions on the schedule.

The knowledge for GSA has been elicited from several experts including SAN graduate faculty members, professors of Mathematics and Statistics Department, and the Associate Dean of the Graduate School. GSA contains two kinds of knowledge:

* knowledge about courses -- course number, title, credit hours, prerequisites, offering time and so on;
* knowledge about constraints -- rules for determining course prerequisites, the "best" courses and rules for determining the schedule length and forming the schedule of study.

It is implemented in NEXPERT Object. The program currently has about 550 rules and 50 objects (and classes). Knowledge about courses is stored as objects (and/or classes). An object is the fundamental unit of knowledge representation in NEXPERT Object. A given object may be defined to belong to one or more classes, which determine the names and types of its properties. It may include subobjects which are in turn full-fledged objects with properties and subobjects of their own. A class is merely a grouping of a set of objects. The class definition may include any number of properties to be inherited by the objects belonging to the class (referred to as its instances or members). The class may have any number of subclasses, which will likewise inherit its properties and pass them on in turn on to their own instances. A given class may be a subclass of more than one other class (called superclasses), just as a given object may be an instance of more than one class or a subobject of more than one other object. Consider the following example, which describes the course SAN572.

Name	**SAN572**
Classes	Core_Courses

	San_Grad_Courses
	...
SubObjects	...
Properties	**credit_hours:** 3
	prerequisites: Unknown
	desc1: Systems life cycle; problem definition; gathering information;
	desc2: creative problem-solving; project management; feasibility study;
	desc3: alternative technology selection; system requirements;structured systems
	desc4: analysis tools; data flow diagrams and dictionary, algorithms
	desc5: specification; logical design; user procedures; review of translation of
	desc6: logical design into systems design. Normally students work on systems
	desc7: development project.
	no_of_descs: 7
	title: Analysis of Information Systems
	weight: Unknown
	when_offered: fall, spring
	...

Below is another example, which describes the category of Core Courses.

Name	**Core_Courses**
SubClasses	...
Properties	**credit_hours:** Unknown
	prerequisites: Unknown
	desc1: Unknown
	desc2: Unknown
	desc3: Unknown
	desc4: Unknown
	desc5: Unknown
	desc6: Unknown
	desc7: Unknown
	desc8: Unknown
	desc9: Unknown
	no_of_descs: Unknown
	title: Unknown
	weight: Unknown
	when_offered: Unknown
	...

Knowledge about constraints is represented by rules. A rule is the basic unit of inference and reasoning in NEXPERT Object. Every rule has three basic parts:

*One or more conditions;

*Exactly one hypothesis;

*Zero or more actions.

The following shows an English translation of a sample GSA rule:

```
IF      the student is a full time student
and
        the student is supported through a graduate assistantship
THEN    setting the max and min numbers of total term credit hours
ACTIONS     set the max number of total term credit hours to 16;
            set the min number of total term grad credit hours to 10.
```

Objects (and classes) in NEXPERT Object are essentially the same as frames suggested in an AI setting by Minsky [5]. Frames are formalized structures for representing knowledge. They are often linked together in a network, thus effectively handling the major inadequacy of production rules for defining terms and for describing domain objects and relationships among objects [1, 3]. A frame commonly consists of two parts: a name and a set of attribute-value pairs. It provides a structured representation of an object, or a class of objects. The use of frames increases the efficiency of the stored information processing by attaching procedures to nodes which know how to compute values of variables in response to queries and how to update values of variables in response to assertions [3]. However, the procedures attached to some slots of individual frames are insufficient to organize the whole computation. The rule representations play the major role in organizing the whole computation.

Objects represent the knowledge being reasoned on by the rules. Hierarchical relationships can be defined between objects to give rules greater reasoning flexibility over objects. The rules, objects (and classes) and relationships between them form the whole knowledge base. The rule and object (and class) relations in NEXPERT Object are illustrated in Figure 1. Note that the relations can be static or dynamic. For example, the relations between courses and their prerequisites are established according to the student's background and exist only for the duration of the advising session in which they are created.

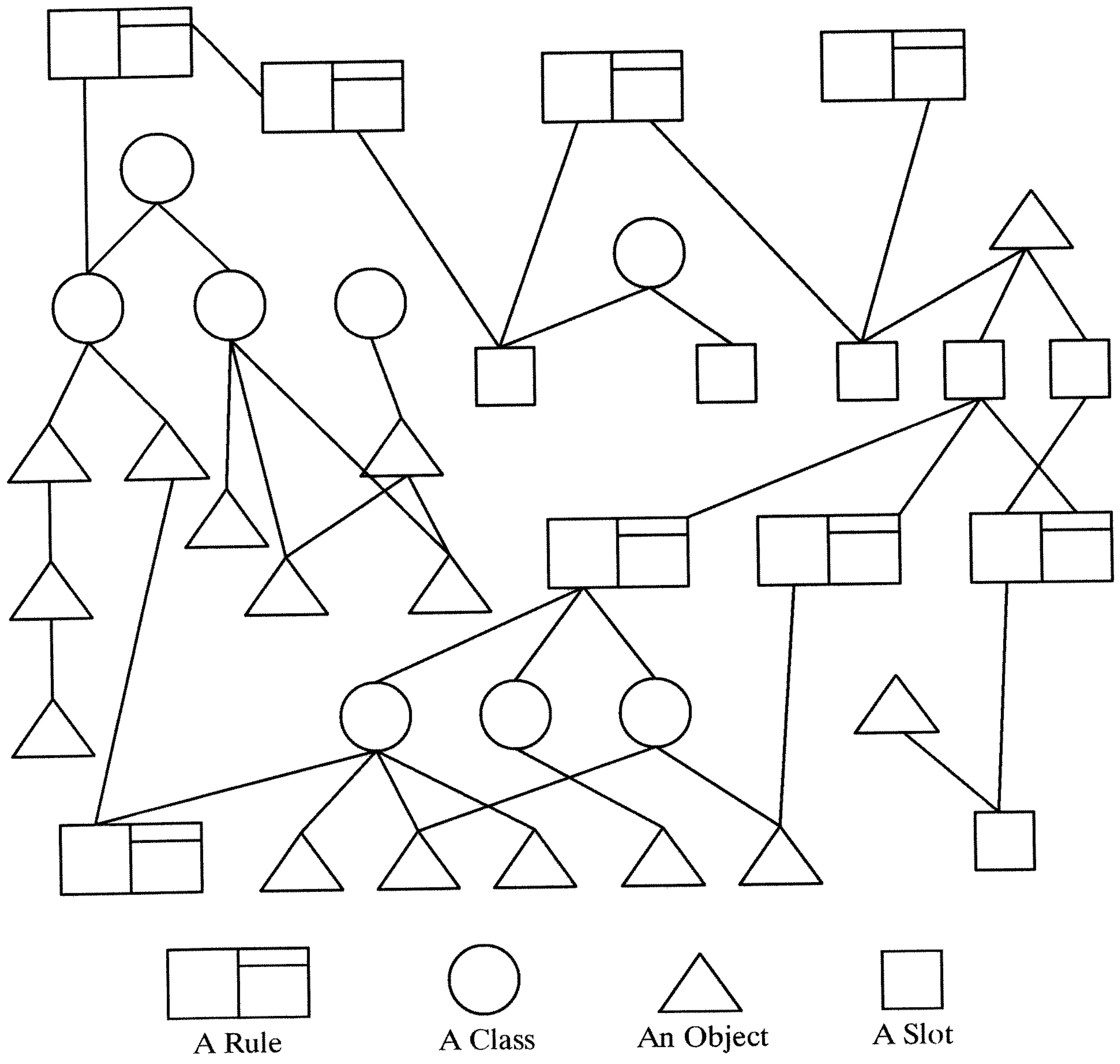

Figure 1. Rule and Object Relations in NEXPERT Object

3. Constructive Problem Solving

The distinguishing feature of constructive problem solving is that solutions have to be constructed rather than selected [3]. Typical tasks that require constructive problem solving are planning, design and certain kinds of diagnoses [3]. In each of the cases, it is infeasible to fix the solution set in advance. There are many different ways in which actions can be ordered, components can be assembled, and faults can co-occur.

Our student advising problem is essentially a constructive problem. Students come from different backgrounds and have different interests. There are too many possible programs of study in which courses can be planned.

Advising a graduate student can be considered as a multi-phase process [7, 8]:

(a) determine how many courses the student should take;

(b) determine which courses the student may take, based on prerequisite requirements and the student's academic history;

(c) determine the "best" courses based on the student's interests;

(d) generate a program of study from the best courses.

The first three phases determine the needs of the student, and are thus of a diagnostic character, while the last phase plans the student's schedule.

The determination of the "best" courses is based on the course weights. One course is said to be the "best" if it has the highest weight. Each course has a preassigned individual weight to each interest area. The weight of a course is computed by summing up the preassigned individual weights of the course to the interest areas of the student (See the Appendix A for the course individual weights).

The last phase is the main task: construction of the program of study. It can be divided into three subtasks:

(1) determine the schedule length;

(2) plan SAN790 courses;

(3) generate each semester's schedule from the best courses.
In the context of the last subtask, there are several constraints used to form each semester's schedule. Here are some examples:

* Each course must be offered in the current semester.
* The prerequisites of each course must be taken or planned prior to the current semester.
* The total number of credit hours planned in the current semester must not exceed the maximum number of credit hours a full time student may register in a regular semester.

Although GSA may have several attempts at planning a course, it never backtracks. In other words, it never makes a decision which it later has to go back and undo. At any point in the problem-solving process, it has enough knowledge to recognize what to do; this cuts down on trial-and-error scheduling. Backtracking is computationally expensive, especially in terms of run-time [3].

4. Running An Advisory Session

As described above, the main task of GSA is advising the student and drawing up a program of study based on the student's background and interests. Before the system begins to make any inferences for the case at hand, the student will be asked a series of questions. There are about 25 questions which may be put to the student. Which of these questions will be posed vary from case to case. In other words, a fixed set of questions is not asked in each case; rather the questions asked depend on answers to previous questions. All the questions are posed in its natural way. If a question requires an answer other than a simple "yes/no" answer, a pop-up list, choice list or selection table is attached to the question. For some questions, there will be a help information pop-up available.

As a result of the dialogue, GSA starts planning the student's schedule. When construction of the student's schedule is completed, the schedule will be presented to the student. The student could print out the schedule or look at the course descriptions. After the student looks over the schedule, he/she could decide to accept the suggestion or make any change under course constraints and restrictions on the schedule. The whole process is very straightforward.

The following is a self-explanatory example of a session with GSA (See the Appendix B for running the GSA). What would actually appear on the screen uses pop-up windows. It is very difficult to draw those pop-up windows here. Therefore, the format has been changed for the purpose of illustration.

```
--GSA: On what basis do you intend to pursue your graduate study?
             Full-time or part-time?
 USER: full-time
--GSA: How are you going to be supported financially in your graduate study?
             Graduate assistantship;
             Modified graduate assistantship;
             Graduate grant-in-aid;
             other support means.
 USER: graduate assistantship
--GSA: Please indicate the first semester of the study schedule you would like
       to let GSA create for you.
 USER: fall, 1992
--GSA: Is this going to be your first semester in the SAN graduate program at
       Miami?
 USER: y
--GSA: Have you ever taken any systems analysis classes (at the 500 or higher
       level) and/or statistics classes (sta563, sta663) at Miami before?
 USER: n
--GSA: The following is a list of current SAN graduate faculty research
       interests.  Please indicate which areas you would be interested in:
             Research Areas                              Graduate Faculty
```

```
           Artificial Intelligence                  M. Ozden, A. Sanders
           Database Systems                         Y. Uckan
           Software Engineering                     J. Kiper
           ...
           Undecided
USER: Database Systems, Software Engineering
--GSA: What is your undergraduate major?
           Computer Science;
           Decision Science;
           Management Information System;
           Mathematics;
           Operations Research;
           Statistics;
           Systems Analysis;
           Other.
 USER: Mathematics
--GSA: Have you ever taken any communication class before?
 USER: n
--GSA: Please choose one of the following suggested communication classes:
           com135      Essentials of Public Speaking;
           com138      Technical Communication;
           com231      Small Group Communication.
 USER: com135
--GSA: How many probability and statistics classes have you taken before?
 USER: 2
--GSA: Have you ever taken any class(es) in Engineering Economics, Linear
       Programming, and/or Stochastic Process?
 USER: y
--GSA: Are you familiar with any programming language?
 USER: y
--GSA: Do you have any background in Data and File Structures?
 USER: n
--GSA: Do you have any background in Computer Architecture and Assembly Language
       Programming?
 USER: n
```

```
--GSA: Based on the information you have provided about your background and
       interests, GSA suggests to you the following schedule:
      Fall, 1992                          Total Credit Hours: 13
      com135     3     Essentials of Public Speaking
      san586     3     Artificial Intelligence
      san603     3     Program, Data and File Structures
      san604     3     Computer Organization and Operating Systems

      Spring, 1993                              Total Credit Hours: 12
      san571     3     Simulation
      san572     3     Analysis of Information Systems
      san623     3     Data Communication, Networks and Distributed Processing
      san624     3     Expert Systems

      Fall, 1993                          Total Credit Hours: 12
      san573     3     Automata, Formal Languages, and Computabiliy
      san575     3     Structured Design and Implementation
      san612     3     Data Structures and Database Systems
      san790     3     Graduate Research

      Spring, 1994                              Total Credit Hours: 12
      san574     3     Language, Interface, and Their Processors
      san622     3     Advanced Database Systems
      sta663     3     Introduction to Applied Probability
      san790     3     Graduate Research

      Would you like to make any changes ? (Y/N)
 USER: y
--GSA: To which semester would you like to make changes?
 USER: Spring, 1994
--GSA: You may check any courses you would not want to take in Spring, 1994:
                 CRHR         COURSE TITLE
      san574     3            Language, Interface and Their Processors
      san622     3            Advanced Database Systems
      *sta663    3            Introduction to Applied Probability
```

```
     san790       3           Graduate Research
      You may choose the courses you would like to take in Spring, 1994:
              CRHR SEMESTER PREREQUISITES COURSE TITLE
     san582 3     Fall      none            Analysis of Inventory Systems
    *san584 3     Spring    none            Analysis of Manufacturing Systems
     ...
--GSA: On the basis of the information you have provided about your background
      and interests, GSA suggests to you the following schedule:
     Fall, 1992                          Total Credit Hours: 13
     com135       3     Essentials of Public Speaking
     san586       3     Artificial Intelligence
     san603       3     Program, Data and File Structures
     san604       3     Computer Organization and Operating Systems

     Spring, 1993                             Total Credit Hours: 12
     san571       3     Simulation
     san572       3     Analysis of Information Systems
     san623       3     Data Communication, Networks and Distributed Processing
     san624       3     Expert Systems

     Fall, 1993                          Total Credit Hours: 12
     san573       3     Automata, Formal Languages, and Computabiliy
     san575       3     Structured Design and Implementation
     san612       3     Data Structures and Database Systems
     san790       3     Graduate Research

     Spring, 1994                             Total Credit Hours: 12
     san574       3     Language, Interface, and Their Processors
     san622       3     Advanced Database Systems
     san584       3     Analysis of Manufacturing Systems
     san790       3     Graduate Research

     Would you like to make any changes ? (Y/N)
  USER: n
```

5. Maintainer's Manual

This section describes how to modify the knowledge about courses and the related rules if necessary. It is assumed that you know NEXPERT Object.

5.1 Update Offering Time of a Course

First, use Object Editor to find the object with the course number as object name. Then activate Meta-slot Editor to update the value of the property **when_offered.**

5.2 Update the Description of a Course

Updating the course description is similar to updating the course offering time. First, use Object Editor to find the object with the course number as object name. Note that there are 9 properties **desc1, ..., desc9**, which can be used to define up to 9 lines of course description. Each line can hold about 70 characters. Activate Meta-slot Editor to modify the values of properties **desc1, ..., desc9**. Then, use Meta-slot Editor to update the value of property **no_of_descs**, which hold the actual number of lines of the course description.

5.3 Add A New Course

Currently, GSA has the knowledge about all SAN graduate courses. In the case that a new course should be offered in the future time, you can add it to the knowledge base in NEXPERT Object. First, use Object Editor to create an object with the course number (e.g. san626) as the object name. Fill in the class field of the Object Editor window one of the class names (Foundations, Cores, Electives) to indicate that the course is a foundation course, core course, or elective course. The object created will have the same properties as other SAN courses belonging to the same class have.

Then use Meta-slot Editor to define the values of the properties: course number, course title, credit hours, offering time, course description, prerequisites. There are nine properties: **desc1, ..., desc9**, which can be used to define the course description. Thus, you can define up to 9 lines of course description. Put the actual number of lines in property **no_of_descs**. If the course does not have any prerequisites, assign "none" to property prerequisites. When it has prerequisites or is prerequisite of other courses, you need to update those rules dealing with course prerequisites. Here is what needs to be done. If the course has foundation course(s) as its prerequisites, use Rule Editor or Rule Notebook to find those rules with hypothesis **"set_grad_course_prereq"**. Then use Rule Editor to modify the action part of those rules. For example, say, san603 is a prerequisite of the new course san626. Add to the action part of those rules with condition **"Is student.san603_ok "n""** the following three actions:

```
Do    "san603"  san626.prerequisites
CreateObject   san603    san626
Do    MAX(san603.wgt, san626.wgt+1) san603.wgt
```

If the new course is a prerequisite of other SAN course(s), use Rule Editor or Rule Notebook to find the rule with hypothesis **"set_prereq_wgt"**. Then use Rule Editor to modify it. For example, say, the new course san567 is a prerequisite of san654. Add the statement:

```
CreateObject   san567    san654
```

to the condition part and the following two actions:

```
Do    "san567"  san654.prerequisites
Do    MAX(san567.wgt, san654.wgt+1) san567.wgt
```

to the action part of the rule. If the new course has as its prerequisites some course(s) other than foundation courses, similar modifications should be made to the rule with hypothesis **"set_prereq_wgt"**. Finally, you have to update the weight list of each interest area. First, determine the weight of the new course to each interest area. Then, Add the weight of the new course to the end of the **wgt_list** property value of the corresponding interest area. Find the rule with hypothesis **"init_dialogue"**. Then, add the new course number (e.g. san567) to the end of the String Value in the second condition.

6. Lessons Learned

(1) One of the major pitfalls to be avoided in developing an expert system is choosing an inappropriate problem [2, 3, 4, 9]. Here are some important criteria relevant to the selection of an appropriate problem for expert system development [2, 3, 4, 9, 10]:

*The application task must have a well-defined domain;

*One or more experts must have the knowledge required;

*Those experts must be able to verbalize desired task performance;

*The task does not depend heavily on common sense;

*The task is of managementable size.

Not picking the "right" problem can lead to complications, even failure in the subsequent development of the system.

(2) There is a fundamental difference between an expert system shell and a conventional programming language: shells are object oriented and knowledge intensive, while conventional programming languages are procedure oriented and code intensive [6]. However, it is not necessary that programming in shells should be any easier to debug and result in less effort. In conventional programming, there exist notions of what constitutes good programming practice [3]. Such is less the case in knowledge engineering [3]. My own experience of programming in NEXPERT Object suggests that the level of programming skill required by a shell should not be underestimated.

(3) Every aspect of the advising process has to be mapped out in detail, and every alternative has to be explored to the point of a conclusion. The expert system can start with only the intelligence

that is put into it. If that information is confusing or incomplete, then your "expert" will be also. A large part of the total effort in creating an expert system must take place before you even touch the keys of a computer.

(4) The process of building an expert system is inherently experimental [3]. It is reported that simple expert systems have been built in as little as 3 man-months [2]. The time for expert system development with present techniques appears to be around 5 man-years per system [2]. If you see a demonstration of a really good expert system that took three scientists years to develop using Prolog, Lisp or some expert language on a mainframe, do not expect to create a similar system in weeks on a microcomputer by using an expert system shell.

7. Conclusion

The process of building the Graduate Student Advisor was very educational and insightful. The GSA program performs well. The decomposition of student advising process into phases has been of help in the following ways: (1) it simplifies interactions with the domain experts; (2) it reduces the complexity; (3) it provides effective modularization. The GSA approach is not necessarily limited to advising graduate students. It could be applied in solving some other problems such as personal financial planning [7].

References

1. Fikes, R. and Kehler, T. (1985). The Role of Frame-based Representation in Reasoning. Communication of the ACM, September, pp. 904-920

2. Gevarter, W. B. (1990). The Basic Principles of Expert Systems. In Raeth, P. G. Expert Systems: A Software Methodology for Modern Applications. Los Alamitos, California: IEEE Computer Society Press. pp. 17-32

3. Jackson, P. (1990). Introduction to Expert Systems 2nd edn. Wokingham UK: Addison-Wesley

4. Liebowitz, J. and De Salvo, D. A. eds. (1989). Structuring Expert Systems: Domain, Design, and Development. Englewood Cliffs NJ: Yourdon Press

5. Minsky, M. (1975). A Framework for Representing Knowledge. In Winston, P. ed. The Psychology of Computer Vision. New York: McGraw-Hill. pp. 211-277

6. Raeth, P. G. (1990). Two PC-based Expert System Shells for the First-time Developer. In Raeth, P. G. Expert Systems: A Software Methodology for Modern Applications. Los Alamitos, California: IEEE Computer Society Press. pp. 2-6

7. Valtorta, M. G., Smith, B. T. and Loveland, D. W. (1984). The Graduate Course Advisor: A Multi-phase Rule-based Expert System. Report No. CS-1984-18, Dept. of Computer Science, Duke University

8. Valtorta, M. G. (1983). The Graduate Course Adviser. Master Project Report, Dept. of Computer Science, Duke University

9. Waterman, D. A. (1986). A Guide to Expert Systems. Reading, Massachusetts: Addison-Wesley

10. Williams, C. (1990). Expert Systems, Knowledge Engineering, and AI Tools: An Overview. In Raeth, P. G. Expert Systems: A Software Methodology for Modern Applications. Los Alamitos, California: IEEE Computer Society Press. pp. 2-6

Appendix: Survey Results on the Course Weights

Note: In the survey, SAN graduate faculty members were asked to evaluate the individual weight of each SAN graduate course to each research interest area using a scale from 0 to 5. Eight of them responded to the survey.

	SAN571	SAN572	SAN573	SAN574	SAN575	SAN582	SAN583	SAN584
Algorithms	1	1	5	3	2	1	1	1
Artificial Intelligence	1	1	3	3	1	1	1	1
Computer Assisted Instruction	3	3	2	3	3	0	0	0
Computer Graphics	2	1	1	1	1	0	0	0
Data Communication & Computer Networks	2	2	2	2	1	0	0	0
Database Systems	2	2	3	3	2	0	0	0
Expert Systems	2	3	2	3	2	1	1	1
Forecast Systems	3	2	1	1	1	3	5	3
Information Retrieval	2	4	2	2	1	0	0	0
Inventory systems	3	2	0	0	0	5	4	3
Machine Learning	2	0	4	3	1	0	0	0
Mathematical Optimization	3	1	2	1	1	3	2	3
Natural Language Processing	1	2	5	5	2	1	0	0
Operating Systems	2	1	3	3	1	0	0	0
Programming Languages	1	1	4	5	3	0	0	0
Queueing Systems	4	0	0	0	0	2	2	2
Simulation	5	2	0	0	1	3	2	2
Software Engineering	2	5	4	4	5	0	0	0
Systems Dynamics	1	3	0	0	0	2	2	1
Theory of Computation	1	1	5	4	2	0	0	0

(Continued)

	SAN586	SAN601	SAN602	SAN603	SAN604	SAN612	SAN613	SAN614
Algorithms	3	5	4	5	1	3	4	1
Artificial Intelligence	5	3	3	3	2	4	1	1
Computer Assisted Instruction	3	3	1	3	2	3	1	3
Computer Graphics	1	3	1	3	2	1	1	1
Data Communication & Computer Networks	1	3	3	3	3	2	2	2
Database Systems	3	3	1	5	3	5	1	1
Expert Systems	5	4	3	4	2	3	2	1
Forecasting Systems	1	2	4	1	0	0	2	2
Informational Retrieval	2	3	1	3	3	4	1	1
Inventory Systems	1	2	5	2	0	1	4	3
Machine Learning	5	3	2	3	3	2	1	1
Mathematical Optimization	3	4	5	2	2	1	5	3
Natural Language Processing	5	3	2	2	2	2	1	1
Operating Systems	1	4	1	4	5	3	1	1
Programming Languages	3	3	1	5	5	3	0	0
Queueing Systems	1	3	5	3	2	2	3	4
Simulation	2	5	5	3	2	1	3	5
Software Engineering	2	4	1	5	5	4	0	0
Systems Dynamics	2	2	3	1	0	0	2	4
Theory of Computation	2	3	2	2	3	2	2	1

(Continued)

	SAN621	SAN622	SAN623	SAN624	SAN625	STA563	STA663
Algorithms	2	1	1	2	2	1	1
Artificial Intelligence	2	2	2	5	1	1	1
Computer Assisted Instruction	2	3	2	3	1	1	1
Computer Graphics	1	1	2	0	1	1	1
Data Communication & Computer Networks	2	2	5	1	4	1	1
Database Systems	1	5	3	3	3	0	1
Expert Systems	2	2	1	5	2	1	1
Forecasting Systems	0	1	1	1	0	4	3
Information Retrieval	1	4	2	3	3	2	2
Inventory Systems	0	0	1	1	0	3	3
Machine Learning	1	1	2	4	1	1	1
Mathematical Optimization	1	0	1	2	1	2	3
Natural Language Processing	2	2	2	4	1	1	1
Operating Systems	2	2	4	1	5	1	1
Programming Languages	3	1	2	2	2	1	1
Queueing Systems	1	0	2	1	1	2	5
Simulation	1	0	2	2	1	3	4
Software Engineering	5	3	3	2	3	1	1
Systems Dynamics	0	0	0	2	0	2	2
Theory of Computation	1	1	0	1	1	1	2

Appendix B: Running the GSA

The GSA system consists of the following files (total 77 files):

```
     GSA.RTD
     GSA.TKB
     FRM1.FRM
     ...
     FRM69.FRM
     COURSE.TXT
     CRSEDESC.TXT
     SCHEDULE.TXT
     SEMESTER.TXT
     WAIT.TXT
     HLP20.HLP.
```

The file **GSA.RTD** is the Runtime Definition File which is used to start the GSA. The file **GSA.TKB** is the knowledge base developed under NEXPERT Object. The files with **.FRM** extension are the NEXPERT forms which are used to create customized interfaces. **COURSE.TXT**, **CRSEDEC.TXT**, **SCHEDULE.TXT** and **SEMESTER.TXT** are NEXPERT report files used to display course descriptions and schedules. **WAIT.TXT** and **HLP20.HLP** are just DOS text files used to display some information to the users.

Suppose that the directory \GSA contains all the files of GSA. In order to start NEXPERT Forms with GSA automatically, **NXPFORMS.EXE** should also be under the same directory \GSA. To start GSA on your system from the directory \GSA, type

```
     NXPFORMS /FGSA.RTD
```

at the prompt. This command automatically starts NEXPERT Forms with GSA.

If NXPFORMS.EXE is not under the directory, you have to start NEXPERT Forms environment by typing the command

```
     NXPFORMS
```

at the prompt. Then use the System menu to read the runtime definition file of GSA and start GSA.