XINGFENG DENG
Anesoft: An Expert System for Canine Anesthesia
(Under the direction of DONALD NUTE)

AneSoft, an expert system for canine anesthesia, has been programmed directly from human expertise and domain documents. The application is not large but somewhat complex, and a useful high performance system has been implemented with a friendly user interface. Designed both to provide expert recommendations and learner guidance, AneSoft can generate a set of consultations for making anesthetic plans based on the information about the patient. It can also assess anesthetic schemes made by the user, explaining why any part of that scheme might be contraindicated and providing suggestions for improving the scheme. In addition, AneSoft can produce any number of cases as specified by the user with or without system recommended anesthetic plan, in a well designed, consistent format. We take full advantage of LPA Win-Prolog to build efficient declarative knowledge base and to develop procedural control and user interface. The expert system will soon be ready for field-testing and we are optimistic that it will be of considerable practical use.

INDEX WORDS: Expert system, Computer-based instruction, Anesthesiology

ANESOFT: AN EXPERT SYSTEM FOR CANINE ANESTHESIA

by

XINGFENG DENG

B.A., Nanjing University, 1989

Ph.D., Nanjing University, 1994

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2000

ANESOFT: AN EXPERT SYSTEM FOR CANINE ANESTHESIA

by

XINGFENG DENG

Approved:

_____

Major Professor

_____

Date

Approved:

_____

Dean of the Graduate School

_____

Date

ACKNOWLEDGMENTS

I would like to acknowledge the following individuals for their help and support during the writing of this thesis:

Donald Nute, my major professor, who gave me the space, time and encouragement I needed in order to develop AneSoft from what started out as a vague germ of an idea;

Dr. James Moore, Dr. Cynthia Trim and Dr. Christine Egger, without whose help this project would have a much weaker basis in anesthesiology, and without whose suggestions the program would have had a very different, and most likely not nearly as efficient, control structure;

Dr. Suchi Bhandarkar, for his time, consideration, and decidedly helpful comments;

My wife, Yun Jia, and my parents, Deng Dongshan and Su Meifang, for their encouragement, ability to keep me on track, and willingness to listen to me ramble on;

My friends and fellow students, Yong Wei, Mingguang Xu and Shulei Sun, for their camaraderie, their willingness and ability to act as sounding boards, and their occasionally helpful comments on Prolog programming, the nature of expert system in general, and the acceptability of anesthetic plans recommended by my program in particular.

My sincere thanks and gratitude to you all.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this thesis, the development of AneSoft, an expert system for selecting canine anesthetic drugs and making anesthetic plans, will be discussed. There are four chapters in this thesis. Chapter 1 introduces the domain problems and motivations for AneSoft, reviews related systems, and outlines the advantages of AneSoft, the tools used for developing it as well as the basic guidelines for evaluating the system. Chapter 2 addresses issues regarding the preparation for knowledge acquisition, the sources of expertise, the process of knowledge acquisition, and other effective methods for facilitating knowledge acquisition. Chapter 3 describes the implementation of AneSoft. The evolution of AneSoft and the basic implementation process for AneSoft are discussed. Chapter 4 concludes the thesis by presenting the results of preliminary evaluation and suggesting future plans for AneSoft.

1.1 THE PROBLEM DOMAIN OF ANESOFT

Anesthesia is a state of unconsciousness induced in an animal. The three components of anesthesia are analgesia (pain relief), amnesia (loss of memory), and immobilization. The drugs used to achieve anesthesia usually have varying effects in each of these areas. Some drugs alone can achieve all three. Others have only analgesic

or sedative properties and may be used individually for these purposes or in combination with other drugs to achieve full anesthesia.

Anesthesia is not a matter that can be ventured into lightly, considering its profound effects on an animal's physiology. Research shows that the use of anesthetic drugs can significantly impact the central nervous system as well as other body systems. Certain drugs or combinations of drugs, if misused, could cause serious effects, sometimes to the extent of killing the animal (Paddleford 1999).

Anesthetic drug selection apparently plays a critical role in veterinary practice. However, expertise in anesthetic selection, particularly knowledge about how to choose appropriate anesthetic drugs for dogs in various conditions, is not always readily available when veterinary students or practitioners need it. It was in response to this need that the AneSoft project came into being.

At the outset of the project, it was determined that how to select anesthetic drugs for dogs during preanesthetic, induction, and maintenance stages would be Anesoft's primary domain. It was also expected that once the system became fairly stable, the domain might be expanded to encompass knowledge for making a complete anesthetic plan, with intraoperative, adjunct, and recovery drugs included, and knowledge for evaluating the plan.

1.2 Expert System Approach

The objective of AneSoft is to help students or practitioners to learn how to choose canine anesthetic drugs and make acceptable anesthetic plans in various

situations. Needless to say, this can be reached in more than one way. Below, I will discuss five common approaches in which the study of canine anesthesiology may be facilitated, focusing on the pros and cons of each approach.

First, the traditional lecture and/or test approach. With this approach, the professor imparts knowledge to the students in a classroom setting. The latter absorb the knowledge by taking notes and then demonstrate their mastery of the subject by passing the exams. The advantage of this approach is that students may be able to have face-to-face communication with their professor. The disadvantages are numerous, however. For one thing, it tends to encourage cramming on the part of the students and might prevent them from playing a more active role in the learning process. In addition, the students cannot consult their teacher at any time when they have questions. Further, even with the same instructor, the quality of the teaching might differ from session to session, depending on the mood and energy level of the instructor.

Second, textbooks. Textbooks can provide necessary information on making anesthetic plans. Students can access the knowledge presented in textbooks at any time. The disadvantage of learning from textbooks is that it can be very inefficient—students may need to browse through a whole textbook in order to find a solution to a specific problem. Another drawback of this kind of learning is that it is not interactive. Students do not get any feedback or evaluation on the anesthetic plans they made, making it difficult for them to verify their knowledge or to know what progress they have made in the subject area.

Third, the clinical training approach. Through clinical training, students can get real-world experience in making anesthetic plans. This approach has its distinct

advantages; nevertheless, these advantages do come at a cost. Potential risks include, for instance, the possibility that one small mistake made in selecting drugs might cause the death of the animal. In clinical training, one can seldom go back to an earlier case to try out a different plan. Nor is the availability of a wide range of cases guaranteed—more likely, one only gets to work with a very limited number of situations. It is almost impossible for learners to gain systematic knowledge in the subject area solely through clinical training.

Last, the courseware approach. This approach does not have the disadvantages of the first and second approaches, and yet it retains most of their advantages. One advantage of the courseware is its portability that allows domain knowledge to be transferred or reproduced easily. Transferring knowledge from one human expert to others is usually an expensive and somewhat tedious process. We must go through this process to build the courseware. However, once it is built, the knowledge it contains can be transferred from user to user simply by making copies of it. Second, With this approach, students can learn in an interactive way at any time at any place. The third appealing aspect of using courseware in anesthesiology is that it produces more consistent, reproducible results than does a human expert. An instructor may make different decisions with identical situations because of difference in attention, weariness, or other reasons. Courseware will always generate the same recommendations when provided with the same inputs. Thus the quality of the instruction delivered via courseware is consistent across sessions.

In addition, unlike clinical training, computer-based lessons generally entail few risks. A computer program that helps students to learn how to construct acceptable anesthetic plans is clearly a preferred choice over other approaches.

Yet even within the courseware approach, there are two main alternatives to consider. One is to use standard authoring tools, such as Macromedia Authorware or Director, to develop interactive lessons. The other option is to develop an expert system using programming languages suitable for representing declarative knowledge, such as Prolog, which was exactly what we did for AneSoft.

An expert system is a computer program that helps a user to solve a problem that normally requires special expertise to solve. It focuses on simulating human reasoning about a problem domain, rather than the actual domain itself, through representation of human knowledge. (Plant and Stone 1991, Stefik 1995). The advantage of an expert system lies in that it allows the learner to work through a problem in any of several different ways—a feat that regular authoring tools are ill-suited to perform. The main feature that distinguishes an expert system from the other tools is --its knowledge base, which contains rules for representing systematic domain knowledge and enables the expert system to analyze any choice the learner makes as he or she makes it. The second advantage of an expert system is its ability to keep track of how the learner interacts with the program; therefore later on the system can provide explanations for all the recommendations it has made. In addition, expert systems cost less than human experts equipped with similar expertise. As more and more advancements are being made in artificial intelligence and computer-based related systems, there should be new opportunities to apply expert systems technology to the area of anesthesiology. Once

AneSoft was established, students could use it to learn, practice or verify their knowledge in anesthetic selection at any time. Other users could include any other non-experts who want to know how to select anesthetic drugs and make appropriate anesthetic plans.

1.3 Related Systems

Although many expert systems have been developed for different applications, few focus on the area of anesthetic selection. No expert systems for choosing veterinary anesthetic drugs were found in a search of the literature. However, two systems should be noted here because the topic of anesthetics is addressed in both and both were intended for educational or training purposes.

ACCESS (The Anesthetic Computer Controlled Emergency Situation Simulator) was developed by Aidan Byrne in England. The simulator is set up in a vacant operating theater or a room usually reserved for resuscitation training. A resuscitation manikin is used as a patient, since it looks realistic and allows practical skills such as external cardiac massage and cardiac defibrillation to be performed. The manikin has a realistic airway and is connected to a standard anaesthetic machine, complete with all the necessary accessories.

Trainees are introduced to the scenario as if they were taking over a case from a colleague. The condition of the patient and the state of the anesthetic are explained.

An anesthetic chart is provided, completed up to the point where the trainee takes over. The trainee is then expected to manage the anesthetic as if the patient were real. The tutors take the part of an interested but totally unhelpful anesthetic assistant. The

simulations are realistic, in that the trainees have to physically perform any actions required. All the simulations are followed through until the patient becomes stable. The feedback available to the trainees is the same as what they would get during a normal working day, i.e., the clinical state of the patient, the readings of the instruments and the behavior of the equipment. The tutors do not provide any suggestions as to diagnosis or treatment. This stimulated our interest in developing an unsupervised learning module for AneSoft. The system is currently in use in ten hospitals in the UK. (Byrne 1994)

The other related system is AlgoSim (The Difficult Airway Algorithm Tutorial). Written by Rob Jones, M.D., this educational software was demonstrated at several anesthesiology conventions and has exerted profound influences in the anesthetic community. Customized versions of AlgoSim have been developed in other languages.

This interactive, hypermedia computer program for training medical students is designed to present the famous ASA difficult airway algorithm in anesthesiology (Benumof 1991) in an entertaining, yet supremely educational new format. The first part of the program takes the user sequentially through the algorithm, with extensive discussion of each step. The second section breaks the algorithm down into 11 discrete pathways, or branches, in order to facilitate rapid recognition of clinical scenarios. Finally, the simulator “puts it all together” by presenting the user with complex, real-life scenarios requiring prompt airway intervention skills. Success is measured, not by an arbitrary scoring system, but by patient outcome, as indicated by real-time pulse oximetry simulation. After the smoke has cleareds, the program enables the user to review his or her actions and compare them with the ASA algorithm standard. Since real-life airway disasters are rare, and long-term learning requires repetition, the program allows one to

review each individual airway scenario as many times as necessary to obtain mastery of the educational objective. The authors hope that the routine use of this training aid with interactive hypertext-based help will expedite the acquisition of the difficult airway algorithm. An impressive system, AlgoSim raise my interest in creating an HTML help file for AneSoft, and the possibility of developing a hypertext reference component for a later version of AneSoft.

The expert system discussed in this thesis, AneSoft, demonstrates the same advantages as the above-mentioned systems. It frees anesthesiology professors from routine instructions to tackle other anesthetic problems. It also allows the same staff to train more students in the same time, or improves quality of training. Like AlgoSim, AneSoft allows the students to practice at any time and at any place for as many times as desired. Using AneSoft in training also saves money in raising experimental animals.

AneSoft also differs from these two systems in several aspects. First, its proposed domain, which is the planning of anesthetic drugs rather than their actual administration, dictates that AneSoft should not be a "virtual" simulator, as ACCESS is. Indeed, throughout the program, students do not need to perform any physical action at all except for clicking with their mouse. Secondly, it is assumed that the user will have already been introduced to the fundamental concepts and rules in canine anesthesiology and will be using AneSoft primarily as an aid for practicing their knowledge. Thus, detailed discussion of the subject area is not included, the way it is included in AlgoSim. It asks the user a series of questions and has a knowledge base that contains almost all the heuristic rules for making anesthetic plans. The program has the ability to mimic human reasoning in applying these rules to search through the problem solution space. Once a

recommendation is made, it can give an explanation to it. As a teaching tool, AneSoft also differs from these two systems in several ways. First, it has a test case generator, which can generate cases with or without system recommended plans. Second, it allows the learner to choose between supervised or unsupervised learning mode. In supervised learning mode, users get immediate feedback when he or she selects a drug or finishes selecting drugs for a stage. While in unsupervised learning mode, the learner will not get any feedback until he or she finishes making his or her own anesthetic plan. In each mode, the program will compare the system recommended plan and the user's anesthetic plan, specifying which drugs should be selected or rejected and why. In addition, AneSoft evaluates the drugs the user has just selected. For example, it lists all the contraindicated drugs that the user has selected as well as all the indicated drugs that user has not yet selected. In addition to guiding students in their study in this area, AneSoft can be used to check or verify the anesthetic drugs selected in a clinic or hospital, where a professional anesthesiologist is not readily available.

1.4 TOOLS FOR ANESOFT

Several reasons make LPA WIN-Prolog for Windows the preferred tool for implementing AneSoft. (1) Prolog is an ideal language to represent knowledge and the human reasoning processes. Compared with other languages, it excels in logical reasoning programming thanks to its built-in inference engine and backtracking mechanisms. These characteristics also render Prolog most suitable for developing AI applications. (2) PCs running Windows operation system are widely used on the

University of Georgia (UGA) campus, and are easily available in computer labs at UGA's College of Veterinary Medicine. Using a Windows version of Prolog would minimize portability problems. (3) LPA-Prolog provides predicates to allow developers to build sophisticated graphical user interfaces (GUIs). This includes sets of predicates both for creating windows with various styles and for handling user actions or inputs when he or she interacts with the interface. LPA Prolog also comes with a dialog editor, a rapid application development tool. This utility allows drag-and-drop editing, enabling the programmer to create windows and controls quickly without writing a single line of code. Thus, in addition to being a language suitable for representing declarative knowledge, LPA Prolog is also capable of generating event-driven applications and of building graphical user interfaces, just as languages like Visual Basic are. LPA Prolog's functionality is further extended by the fact that it can be coded to call .dll files and execute external programs directly. (4) If AneSoft is distributed, there will be no need for a run-time license.

At the beginning of AneSoft's development phase, I considered taking advantages of both Prolog (efficient inference engine and declarative knowledge base) and Visual Basic (friendly user interface, rapid application development, and procedural control), using the LPA Prolog's Intelligence Server, a mechanism to provide an interface between a Prolog program and a front-end GUI written in a language other than Prolog. Later on when I finished implementing part of the program and tested it, I realized that this approach would increase the system overhead and slow down the consultation process, since information would need to be passed back and forth between Prolog and Visual Basic programs. I redesigned the system and decided to build the interface in Prolog.

Indeed, using “pure” WIN-Prolog, one can build fairly sophisticated user interfaces without resorting to external programs. Other programming languages or tools were used for enhancing AneSoft. These included CGI technology, HTML, JavaScript and Java Applets.

1.5 GUIDELINES FOR EVALUATING ANESOFT

Three evaluation criteria are used for AneSoft. These are the accuracy of domain knowledge, the completeness of domain knowledge, and the ease of use. The first two have been assessed by the domain experts --- Dr. Trim, Dr. Moore and Dr. Egger at University of Georgia’s School of Veterinary Medicine. Ease of use has been evaluated by experts and non-experts, most of whom are veterinary students at the University of Georgia. The results of these evaluations are presented in Chapter 4.

CHAPTER 2

KNOWLEDGE ACQUISITION

To obtain domain knowledge from an expert or other sources is a time-consuming task, but it is the core of expert system development because the accuracy and reliability of the system depends, to a large extent, on effective collection of the domain expertise. In this section, I'll discuss my firsthand experience with knowledge acquisition for developing AneSoft.

Knowledge acquisition can be viewed as the process of extracting, structuring, and organizing knowledge from one or more sources to the knowledge base, and sometimes even to the inference engine. Thus, although knowledge acquisition is usually considered a separate phase in the cycle of the expert system development, it actually occurs throughout the whole development process of an experts system as will be discussed below.

2.1 PREPARATIONS FOR KNOWLEDGE ACQUISITION

Preparations needed to be made prior to knowledge acquisition for AneSoft. The first preparatory task was to identify the sources of expertise. Within the field of anesthesiology, how to select an anesthetic drug for dogs during preanesthetic, induction and maintenance stages was defined as the system's initial domain. Then the domain was

extended to include knowledge for making a complete anesthetic plan (intraoperative, adjunct, and recovery drugs were included) and evaluating the plan. The sources of expertise would be identified after the domain was defined. AneSoft would be built upon expertise pulled from knowledge documents and human experts, which will be elaborated later in section 3.2.

The second task was for me to familiarize myself with the domain by reading relevant materials about the domain. It would be impossible to interview a veteran anesthesiologist efficiently and effectively without some background knowledge. These materials gave the developer a broad view of the domain. The reading helped me to analyze the domain knowledge and formulate the interview questions. It also served to enhance the dialog with the experts.

The last preparatory task was to acquaint the domain experts with expert system technology. I spent time explaining the concepts of expert systems. This started with a description of several existing systems in the application area of medicine, followed by a proposal of what the AneSoft system might look like. Later after having finished the prototyping, I demonstrated the system to the domain experts. The demonstration not only gave them an understanding of an expert system’s capabilities and its use, but it also stimulated their interest in developing an expert system.

2.2 SOURCES OF EXPERTISE

The sources of expertise came from books and domain experts. In this section, I will explain the type of knowledge obtained from each source of expertise and the techniques involved in obtaining the knowledge for AneSoft.

2.2.1 KNOWLEDGE ACQUISITION FROM KNOWLEDGE DOCUMENTS

Upon recommendations from domain experts, four books were used for knowledge acquisition for AneSoft. They are *Veterinary Anaesthesia* (Hall et al 1991), *Clinical Veterinary Anesthesia: A Guide for the Practitioner* (Short 1974), *Manual of Small Animal Anesthesia* (Paddleford, 1999), and *Veterinary Anesthesia* (Lumb 1973).

All of these books were most useful in providing a checklist of the various factors involved in recommending an anesthetic drug as well as a detailed analysis of each of these factors. As a valuable source of information, these books were put into the following use. Firstly, being a layman to the field of veterinary diagnosis and prescription, I used these books to familiarize myself with the various anesthetic terms and to get a general picture of the process of prescribing veterinary anesthetic drugs in various situations. Secondly, when preparing the questionnaires to be used at the interviews with the domain experts and structuring the information to be obtained from the interviews, I consulted these books for detailed accounts of the proposed issues, such

as the side effects of a particular agent, and the dosage needed for a patient at a particular age. The interviews with the domain experts focused on acquiring the general problem solving strategies and heuristics; hence, specific information had to be filled in by these books. Lastly, some indications and contraindications of anesthetic drugs from these books were transcribed directly into Prolog rules to provide the information to be used by the expert system.

As fundamental works for veterinary students, these books were undoubtedly capable of providing much of the needed expertise for AneSoft.

2.2.2 KNOWLEDGE ACQUISITION FROM EXPERTS

The second source was advice and experience from Dr. Cynthia Trim, Dr. James Moore and Dr. Christine Egger, all professors at the College of Veterinary Medicine at the University of Georgia.

At my initial meeting with these domain experts, they explained that the first step was to identify the factors that might affect the anesthetic selection. The way to do that, they suggested, was to ask a series of questions. Thus, they advised me to read some of the books in anesthesiology and formulate the questions to be asked which would be vital to both the problem domain and a computer program. This initial meeting set the tone for all later interviews to be conducted for the purpose of knowledge elicitation from the domain experts. From my experience, the interviews bore the following characteristics:

First, each of the interviews had what is considered a structured format, meaning that it was a systematic goal-oriented process. Before each interview, I would study all

available materials to identify topics for discussion with the experts. The topics, together with information that I had gathered on each topic, from the books above or a previous interview, and a specific set of questions that I might have with regard to a topic, were then recorded in a written document. During the interview, I would ask the expert to check off topics from the document to review and to critique the information I had prepared. Meanwhile, I would write down the opinions given by the experts. After the interview, I would review the notes, making sure that I understood the experts' views and their problem solving heuristics. If I had further questions I would note them down. Then, I would try to get answers from books or consult experts at the next interview session.

Second, the scheduling of the interviews could be affected by knowledge representation and system implementation. The processes of knowledge acquisition, representation and system implementation and testing are actually interwoven. An important advantage of expert system technology over traditional programming is its potential for prototyping with an initially small set of facts and rules in the knowledge base. This small set could be further improved or expanded as more knowledge was acquired through subsequent interviews. The knowledge acquired from one interview session could affect weeks of effort in trying either to represent and organize the knowledge properly to make it usable by the inference engine, or to modify the inference process so that it could make more efficient use of the knowledge. However, the knowledge gained from any interview would be of little use if not organized properly, or not integrated with the existing knowledge. The representation and organization of the knowledge would in turn enhance the quality of the next interview. Sometimes an interview could only be scheduled after certain "breakthroughs" had occurred in

development activities other than knowledge acquisition and new problems had resulted therefrom. For example, after figuring out all the common questions to be asked about a dog, I found that some of these questions had the same effect so far as the choice of anesthetic was concerned. So for example, instead of inquiring about several individual diseases, I combined categories of diseases into one generic question. This way, the interview could be conducted more effectively and the system could be implemented more efficiently.

Third, at every interview, I would report to the domain experts on the development status of the system and explain some of its design considerations in order to make them understand why some of the questions were being asked of them. This practice proved extremely beneficial since the experts could suggest solutions to problems related to the domain with the capabilities of the current system in mind, thus making it easy for me to compile the rules. Meanwhile, the experts would test and review the parts of the system that were currently functioning. Once they had a better sense of what the current system looked like, they were in a better position to provide appropriate solutions to problems. So at every step, I would seek suggestions from the experts before continuing the development.

2.3 TECHNIQUES TO FACILITATE KNOWLEDGE ACQUISITION

After several interviews, the experts sometimes felt that they had provided most of the heuristic rules. They said that this might be all that they could think of at that time.

At this stage, it is important to employ alternative ways to elicit more knowledge from the experts in order to maximize the system's performance.

2.3.1 TEST CASES

After I acquired and represented most of the domain knowledge from the experts and the knowledge documents, I implemented the prototype of the expert system and also a case generator for the system, even before the knowledge acquisition and knowledge base were complete. The computer randomly generated cases with system recommended plans were printed in a well designed format. These print-outs were then presented to the domain experts. This gave the experts an opportunity to see how well the current system behaved. The experts compared the system generated plans with their own plans. When discrepancies were detected, the experts provided additional rules or refined existing rules. They reviewed and verified the plans one by one. More knowledge was thus elicited when they detected there was something missing in the system generated anesthetic plans. Later on, case generator became a separate module in AneSoft.

2.3.2 EXPLANATIONS

Usually the explanatory facilities are implemented at the end of the development process for expert systems. However, during the knowledge acquisition for AneSoft, I felt a need to construct the explanations for the system concurrently with the knowledge acquisition. The reason is, when I presented the system with an incomplete knowledge

base to the experts and there seemed to be a problem in the system recommended plan, it usually took me much time to explain why the system generated this plan in everyday English. The domain experts were not expected to check the program to find what rules might be wrong since all rules were represented in Prolog clauses. After explanatory facilities were implemented, knowledge acquisition sessions became more productive. In addition, with all these explanations phrased in everyday English, the domain experts were able to run the program to verify the rules themselves. Soon they themselves were able to explore the current knowledge base, using the explanatory facilities to view the rules being used and the information in the working database. Sometimes this process triggered the experts into volunteering some additional information, exposing a whole dimension to the problem in hand that had previously been overlooked.

2.3.3 WEB SITE

It should be noted that knowledge acquisition is a time-consuming process and that interviews alone are not sufficient for the purpose. Experts usually have busy schedules, which can make the scheduling of the interviews problematic. Hence minimizing the meeting times is necessary from the domain experts' point of view. Often, the domain knowledge has become almost second nature to the experts, so much so that they need time to think about the reasoning processes used in problem solving in their domain. They also need time to think about the best way to present this knowledge so that it can be formulated and programmed by the knowledge engineer later on. All these problems require that interviews be supplemented with other means of knowledge

acquisition. With this goal in mind I set up a web site that included general information about anesthesiology and expert system technologies. The site also described the goal for AneSoft, the current status of AneSoft and problems encountered. Once a demo version was ready, it was posted on this site. The experts could download the current build anywhere at any time.

This web site proved to be useful for knowledge acquisition. First, it provided me a means to gain background knowledge of the domain and mitigated the need to allocate time at each interview session for that purpose. The interviews with the domain experts could thus focus more on the acquisition of heuristic rules. Furthermore, by reflecting on the issues posted on the site, both the experts and I could better prepare for the next interview session. In addition, the site stimulated the experts' interest in developing the system after they were informed from the site of the goals and progress of AneSoft.

2.3.4 EVALUATION FORM

To reduce the amount of time spent in evaluating the current build of AneSoft, I used CGI technology and developed an evaluation form which comes with AneSoft. Users, including domain experts, could fill out the form and send it to the author directly via the Internet. When the experts ran AneSoft and thought it necessary to modify or add certain rules, they could easily launch this evaluation form in HTML format and write down their suggestions in a text box. After they hit the Submit button, the suggestions were automatically collected by the CGI program. An email message containing the

suggestions made by the experts would be sent to me. This Internet evaluation form provided another valuable tool for knowledge acquisition on the AneSoft project.

2.4 Process of Knowledge Acquisition

Knowledge acquisition occurred throughout the whole AneSoft project and was intertwined with the design, implementation, and evaluation phases. At the outset, a simple decision making diagram was created to represent the knowledge for anesthetic selection. The domain experts then examined it. After doing that, they generally gained a better understanding as to how the system would reason. They could then assist in perfecting the decision making diagram. After that the system was designed based on the revised diagram.

During this stage, if questions arose with regard to the problem-solving process or the knowledge obtained, the domain experts would be consulted for solutions. Newly acquired knowledge would then be integrated into the existing knowledge base. The program would then be revised. After the revisions were implemented, the program would be executed to make sure it behaved correctly. In this stage, the necessary information was added constantly and the design might undergo frequent changes. After a successful implementation, the domain experts were asked to run the system and evaluate it. System corrections were made according to their inputs. Then the design and implementation were repeated. New information or features were also added at this time.

The main system development involved just one expert; therefore there was no question of conflicts between experts. However, later during system validation when

other experts were reviewing AneSoft, there were disagreements concerning the rules in the knowledge base. These were generally resolved by discussions between the domain experts. I also played a role in resolving the differences by actually amending the existing system based on one expert's feedback and showing the results of the amendment to all experts. This turned out to be a useful way of resolving the different opinions among the experts as it demonstrated to them the different results of their different approaches, some valid and some not.

After going through this cycle many times, the decision tree was expanded, a subdomain added, the interface refined, and more information about anesthetics was incorporated. As can be seen from above, though knowledge acquisition was, to a large extent, a distinctive phase in the whole development cycle of the system, it interacted with other development activities and took place throughout the entire development process.

Judging by the experts' reaction to the whole elicitation process, I can safely say that they all enjoyed it and were stimulated by it.

CHAPTER 3

ANESOFT SYSTEM DESIGN AND IMPLEMENTATION

After knowledge was acquired, the next step would be representing the knowledge as Prolog rules or facts. I would design and implement the whole expert system in order to present the knowledge to users in a way that they could easily understand. AneSoft, which consists of a knowledge base, an inference engine, and a user interface, is an expert system implemented in LPA WIN-Prolog. Prolog was chosen as the primary development tool due to its built-in inference engine, which can search the knowledge base in a depth-first fashion automatically. In addition, we decided to use a Windows-based version of Prolog—LPA Win-Prolog—considering the greater ease it affords in graphical user interface (GUI) design.

This chapter will first delineate the evolution of AneSoft, the control structure for AneSoft, the knowledge base and inference engine in AneSoft, and the user interface for AneSoft.

3.1 EVOLUTION OF ANESOFT

As a computer program, AneSoft went through the life cycle typical of software development. The steps include software design, prototyping, implementation, testing,

evaluation, enhancement and maintenance. There are several models we can follow. What I adopted for AneSoft was the iterative model, which emphasizes that these processes are interwoven, as illustrated in the following diagram (Figure 3.1):

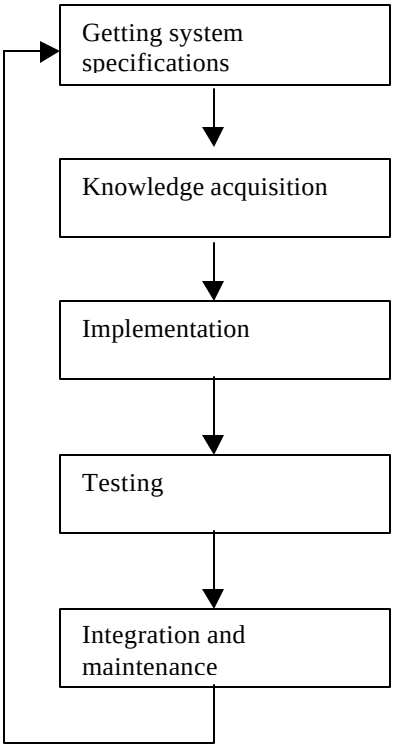

Figure 3.1 Iterative Model Used for AneSoft

There were three stages of development for AneSoft. In the first stage, I concentrated on the development of the knowledge base. The system at that time had no graphical user interface other than a simple text input and output mechanism run on the console. This text-based console allowed an easy understanding of how an expert system worked and also allowed me to build prototypes quickly for knowledge acquisition. In the second stage, after I felt that I had acquired most of the knowledge and tested it, I built a

Windows-based user interface in LPA WIN-Prolog. This graphical user interface made the program much easier to use. Last, a final version of AneSoft was developed by modifying the graphical user interface and adding more functionality to enhance the application. On-line help and consultation reports in HTML format were developed for the system. Using CGI technology, an evaluation form, initially used for knowledge acquisition, was refined and incorporated into AneSoft in this stage. The form allows the user to send comments to me directly via the Internet. It was also during this stage that the Supervised Learning module, the Unsupervised Learning module and the Case Generator module were added to AneSoft. Sharing the same knowledge base as the standard expert system, these three modules add much to the expert system. By the time of these additions, AneSoft went far beyond being just a standard expert system;--it became a program with many useful capabilities for various purposes, especially for education. Worth mentioning here is the Case Generator module, which can not only assist professors in instruction and students in learning but also, as pointed out in the previous chapter, proved quite useful in knowledge acquisition.

3.2 THE KNOWLEDGE BASE

3.2.1 THE KNOWLEDGE BASE FOR STANDARD EXPERT SYSTEM MODE

The knowledge base for AneSoft standard expert system module (anes_kb.pl) includes rules for choosing individual anesthetic drugs, for selecting appropriate drugs for each category of anesthesia, and for constructing complete anesthetic plans. Specifically,

the knowledge base for AneSoft consists of the following components: 1. Initial clauses to unload any knowledge base rules that are already in memory before a new consultation session begins. 2. A brief introduction indicating what the system is and how to use it. 3. Clauses to configure the default settings for AneSoft. The default settings will, for example, allow AneSoft to recommend anesthetic plans in standard expert system mode and display the results in simple text format . While AneSoft is running, the user can change these setting at any time by accessing the Settings menu. 4. A set of recommendation rules (indication, contraindication, and incompatibility rules). 5. Clauses from which recommendations and explanations can be constructed (clauses for rec/5, rec/2, xkb_text/2, update_expl/1 and construct_expl/0).

The following is a typical rule for selecting individual drugs:

```
contraindicated(drug,atropine):-
      (known(problems,`Pancreatitis`)     % gastrointestinal
      ;     % or
      known(temperature,T), number_string(Num,T),Num > 103.5,
      ;     % or
      known(reason,`Parotid duct transplant`)
      ).
```

This rule advises against using atropine, assuming that any of these three conditions are true, that is, if the dog has pancreatitis, if the body temperature of the dog is greater than 103.5 degrees Fahrenheit, or if the reason for anesthesia is to perform a

parotid duct transplant. These conditions might be met or they might not, depending on information obtained from the user.

Rules in the AneSoft knowledge base are hierarchically structured so as to reflect the nature of human reasoning in the domain area. For example, consider the following rules:

```
contraindicated(preanesthetic,meditomidine):-
      (geriatric
      ;
      sick
      ;
      belongs_to(brachycephalic)
      ).


geriatric :-
      known(year,A),number_string(Age,A),
      (giant_breed,Age>6
      ;
      Age>9
      ).


giant_breed :-
      known(breed,Brd),member(Brd,[`Mostiff`,`Gt. Dane`,
      `Rottweiler`,`Poberman`,`Wolfhound`,`Irish wolfhound`]).
```

In order to check for the first rule the program will first examine if the second rule is satisfied, which in its turn will check if the third rule is satisfied by invoking the clause

giant_breed/0 and looking for information about the breed of the dog. After that, it is possible to determine, through backward chaining, if the first rule is satisfied.

Typical rules for making anesthetic plans for each category of anesthesia are:

```
rec(Category,DrugList):-
      get_pfd_list(Category,PfdList),
      get_avail_list(Category,AvailList),

      one_each(PfdList,AvailList,TypeDrugList),
      get_drugs(Category,TypeDrugList,DrugList).

get_pfd_list(Category,PfdList):-
      findall(Pfd,
            indicated(Category,Pfd),
            PfdList).

get_avail_list(Category,AvailList):-
      findall((Type,D),
            (Category(Type,D),      % common protocol
             \+ contraindicated(type,Type),
             \+ contraindicated(drug,D),
             \+ contraindicated(Category,D)
            ),
              AvailList).
```

The above rules state that in order to make an anesthetic plan for a certain category of anesthesia, the program will first check if there are preferred drugs for any

type within that category, for example anticholinergics in preanesthesia. At the same time the program retrieves all drugs that are not contraindicated. If the preferred drugs are also found among those not contraindicated, they will be selected even if they are not included in the common protocol for dogs.

The explanations for AneSoft include the common anesthetic protocol for dogs and how the system refines it based on the information the user provides. AneSoft explains this plan refinement process by specifying the following: (1) Which drugs are not in the common protocol but are selected and why (what indication rules apply); (2) Which drugs are in the common protocol but are removed from our plan and why (what indication rules apply so that other drugs replace the drugs in the common protocol, or the drugs in the common protocol are contraindicated); (3) Preferred replacement drugs, which are indicated; (4) Other replacement drugs; (5) What drugs are contraindicated and can never be used for a particular patient.

3.2.2 Rules for Other Modes

Besides the Standard Expert System mode that recommends an anesthetic plan based on information provided by the user, AneSoft runs in three other modes, namely the Case Generator mode, the Supervised Learning mode, and the Unsupervised Learning mode. Although the three modes share the knowledge base for standard expert system mode, each of these modes has its special rules.

(1) Case Generator:

As elaborated in Chapter 2, this mode generates specified number of cases with or without system recommended plans.

The predicate *cases* takes two arguments, number of cases and file name, from the user. It first generates a working database randomly and writes to the file. After that, it will generate the forms to display the anesthetic plans based on the information in the working database or to evaluate anesthetic plans the user will provide later on. Depending on the settings, the anesthetic plans may or may not be written to the file.

(2) Unsupervised Learning

Unsupervised learning mode allows the user to select anesthetic drugs without any immediate feedback until the user finishes making his or her anesthetic plan.

The rules for unsupervised learning include the predicates for displaying the drugs contraindicated but selected by the user, or the drugs indicated but not selected by the user. They also include predicates to construct the user's anesthetic plan, predicates to evaluate the plan, and predicates that explain the criteria used for the evaluation.

(3) Supervised Learning

Supervised learning mode provides immediate feedback while the user is selecting anesthetic drugs.

The Supervised Learning module utilizes all the rules in the Unsupervised Learning module. In addition, it uses rules not found in the other module. Rules unique to the Supervised Learning module generally fall into one of the following three categories.

(a) predicates for checking individual drugs.

The rule *check_individual_drugs* will check and see if the next drug the user selects is appropriate by itself.

Upon invocation of this rule, the program will read in the name of the selected drug and then check for the constraints for this drug in three steps.

First, the drug should be appropriate for dogs. If the opposite is found to be true, AneSoft will display a warning message, deselect the item, and prompt the user to make another selection.

Once the suitability of the drug for dogs is confirmed, AneSoft proceeds to verify that it can be used for the intended stage, such as preanesthetic, induction or maintenance:

In this step, the system first retrieves a list of drugs that can be used for the needed anesthesia and then searches for the selected drug in the list. If not found in the list, the drug in question will be eliminated from the pool of candidate drugs. A warning message will be displayed and the user will be prompted for the next selection. Otherwise, the program continues to check whether the drug is contraindicated.

After passing all the three checks, the selected drug will be determined as appropriate in itself.

(b) predicates for checking combinations of drugs within each category of anesthesia.

The rule *check_combination/1* examines the drugs in a specific category to see if they can be used in combination with one another to produce the intended anesthesia. For instance, *atropine* and *glycopyrrolate* cannot both be selected for preanesthesia. If the user chooses both, a warning message will be issued.

All the constraints needed for performing this check are defined in the *check_combination_aux/5* predicate.

(c) predicates for checking combinations of drugs in the whole anesthetic plan.

With this kind of predicate, the system will perform a "global" check for the entire anesthetic plan to see if it is appropriate. This kind of checking can ensure, for example, that no more than two inhalant drugs are selected for each plan.

The knowledge base, as we know, is the heart of an expert system. Assuring the quality of a knowledge base is the most important concern during the knowledge engineering process because the quality of the knowledge base determines the overall usefulness and correctness of an expert system. The knowledge for AneSoft was derived from domain experts and existing studies on the management of anesthetic drugs, which had been evaluated for correctness.

3.3 THE INFERENCE ENGINE AND INFERENCE PROCESS

AneSoft employs a backward-chaining inference approach (Stefik 1995), which is built into LPA WIN-Prolog. With backward chaining, the system is given a specific goal and the inference engine searches the knowledge base for a rule with the same goal. Once the engine finds a matching rule, it sequentially searches through the static facts and rules in the knowledge base and the dynamic information in the current working database to determine if the premises of the rule are satisfied. If they are not satisfied, the inference engine will start looking for another matching rule. The process continues until either the premises of the matching rule are satisfied or the inference engine runs out of

rules. Suppose the chosen goal is to determine whether *isoflurane* is suitable for induction. The inference engine will search through the knowledge base and find the following rule with the matchimg goal :

```
indicated(induction,isoflurane):-
      known(month,Month),number_string(NumOfMonth,Month),
NumOfMonth =<2.5.
```

This rule states that *isoflurane* is a preferred drug if the dog is two and a half months or younger. If the condition is not met, the inference engine will backtrack until it finds a solution. If the rules are hierarchically structured, as discussed in 3.3, the Prolog inference engine will search through the entire knowledge base from the “top” rules down to the “bottom” in a depth-first fashion.

AneSoft uses plan refinement schemes to make anesthetic decisions. Starting with a common anesthetic protocol for dogs, the program checks to see if there are preferred (indicated) drugs for the intended anesthesia or if there are any drugs contraindicated for the dog in question. By adding new drugs into and removing existing drugs from the protocol, a new anesthetic plan is constructed. All of these operations are enforced by the inference process in AneSoft.

For example, the common protocol specifies three drugs for canine preanesthesia, namely atropine for anticholinergic, acepromazine for sedative, and morphine for opioid . In case of a dog with heart base tumor, atropine is contraindicated as stipulated in the following rule :

```
contraindicated(preanesthetic,atropine):-
      known(problems,`Heart base tumor`).
```

Since atropine is eliminated, the other drug in the anticholinergics group, glycopyrrolate, is selected in its place. In the resulting plan glycopyrrolate, acepromazine, and morphine are recommended for the dog.

3.4 THE USER INTERFACE

The user interface is a necessary component of an expert system as it handles all the communications between the users and the inference engine. In Anesoft, the interface consists of a set of forms where the user can input the raw data needed for a consultation. It also includes screens where users can select their preferred drugs for each category of anesthesia and for a specific scenario where anesthesia is needed. Once the user completes a form, the data are translated into a format understandable by the inference engine. When the inference engine reaches a conclusion, the user interface will pass this information back to the user in natural languages so it can be understood by the latter. To clear up any confusion the user might have regarding a specific conclusion, the user interface supplies additional explanations as to how the inference engine reached the conclusion.

Building a sophisticated user interface is crucial to the success of an expert system. However, interface design can be a rather time-consuming process, placing significant demands on developers of expert systems who are already busy enough with knowledge acquisition and representation to ensure all the knowledge in the system is

correct. To solve this problem, a tool is needed that can allow knowledge engineers to build a sophisticated user interface without consuming too much of their time. One of the reasons that LPA WIN-Prolog was chosen as the development tool was because it allowed rapid development of the user interface for AneSoft, especially with the aid of the dialog editor.

LPA Win-Prolog (Prolog hereafter) provides an environment for the development of graphical user interface (GUI) through the use of a set of predicates designed specifically for building user interface. All that the knowledge engineer needs to do is to create the windows and controls using the *wdcreate* and *wccreate* predicates and to write handlers for them.

AneSoft utilizes what is called the SDI (single document interface) type of layout. The main window consists of four menus providing access to lists of choices. The screenshots below illustrate the user interface for AneSoft. The user requests a consultation session by clicking the Start button. After a consultation, the caption of the same button changes to “New Consultation”, prompting the user for another session of consultation if he or she so desires.

The interface will be elaborated in section 3.5.2.

3.4.1 CONTROL STRUCTURE OF ANESOFT

As a computer program, AneSoft needs a mechanism to control the order of executions of the clauses. This mechanism will determine which module to run at a given time, depending on the existing conditions. For example, if the user selects the

unsupervised learning mode in system configuration, the control mechanism will notify the Unsupervised Learning module to run after the user finishes filling out the form.

At the beginning of a consultation session, the user chooses system settings. The default setting is for Anesoft to run in a standard expert system mode and display the result in the main window in simple text format. If the user chooses to run the Case Generator module, he or she will be prompted to enter the number of cases to be created and the name of the file in which to save the case data. Otherwise, the user will need to provide information about the patient by filling out some forms.

After the user completes the forms, the flow of control branches out into two directions depending on system settings. In the Standard Expert System mode, a recommended plan will be immediately presented in simple text or HTML format. In either of the two learning modes, the user will need to select drugs from the list of drugs for each category of anesthesia. The only difference between the supervised and the unsupervised learning modes is that in the former, Anesoft provides just-in-time critique of each of the drugs or group of drugs selected by the user while in the later, no such feedback is offered.

3.4.2 THE FIVE PRIMARY WINDOWS

The main window of AneSoft appears when the user launches the program (Figure 3.2).

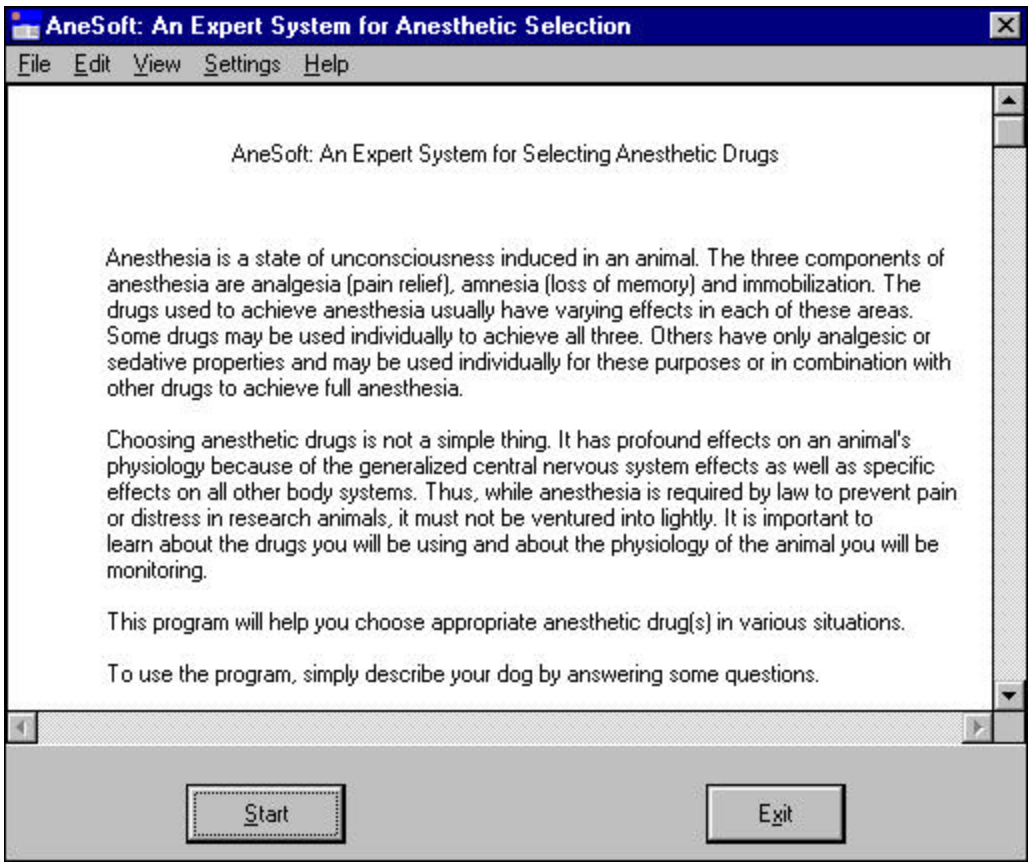

Figure 3.2 Main Window of AneSoft

The initial screen introduces the user to the system and provides directions for the user to follow. When the user finishes filling out the forms and/or selecting drugs, consultation results will also be displayed in this window including the system recommended anesthetic plan, a detailed explanation, and/or comments on the user's anesthetic plan. If the Case Generator mode is selected, once the cases are generated a message will be displayed in the main window to that effect.

The menus at the top of the main window provide the user with system-level controls. The File menu includes routines to save the consultation report (as a text file), to print the report, and to terminate the AneSoft program.

The Edit menu provides facilities for selecting and copying text.

The View menu allows the user to view the current information in the working database or the result of the current consultation in plain text. In addition, the user can view both the current information and the result of the consultation using the system's default Web browser by selecting the View Current Consultation in HTML option. All the options on the View menu are action sub-menus. They remain disabled until a consultation is made.

The Settings menu contains two groups of options. The first group allows the user to choose a mode in which AneSoft will be running. This includes five choices: (1) Standard Expert System Mode, (2) Supervised Learning Mode, (3) Unsupervised Learning Mode, (4) Case Generator Mode with System Recommended Anesthetic Plan, (5) Case Generator Mode without System Recommended Anesthetic Plan. The second group of options determines the format in which the consultation result will be displayed. The user can choose to view the result later on in either text format in the main window or in HTML format from a Web browser.

The Help menu provides access to an HTML help file that includes an anesthetic glossary, instructions on how to use the system, and version and credits information about the system.

In all modes except for the Case Generator mode, clicking the Start button on the initial screen will bring up the Form Window (Figure 3.3). The main purpose of the Form window is to obtain needed information from the user and then feed that information into the inference engine.

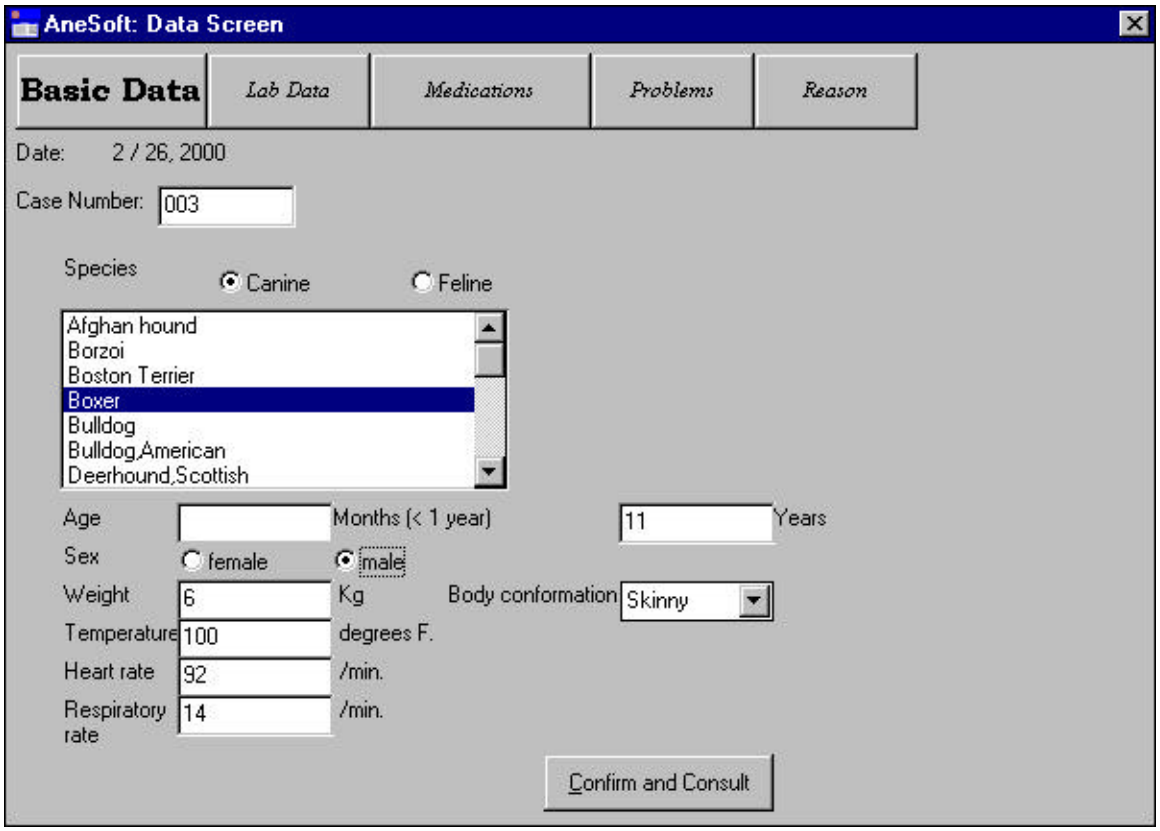

Figure 3.3 Form Window of AneSoft

In the Form Window four kinds of controls are used for getting the user input. With the radio buttons one choice is selected to the exclusion of all the others, With the

check boxes, users can select multiple options or select none. With list boxes, one or more options can be selected from a list of items. With text boxes, the user can enter values directly in the blank provided.

The data about the patient are organized into five “folders”, each of which focuses on a specific aspect of the patient. These are the data a human expert normally seeks at a face-to-face consultation.

The Basic Data screen is intended to retrieve the basic data about the dog, for example, its breed, age, sex, heart rate, respiratory rate, and weight. Most of these are required and dictate what kind of plans will be generated in the end.

The Lab Data screen includes the lab data about the dog, for example, its BUN and K.

The Medications screen allows the user to check or select all the previous medications administered to the dog, for example, fentanyl patch.

The Problems screen allows the user to check or select all medical problems for the dog, current and anticipated, such as head trauma.

The Reasons screen lets the user specify the reasons for anesthesia. The user can choose from three groups of possible reasons: (1) Diagnostic imaging, for example, radiography; (2) Medical reasons, for example, bone marrow aspirate; (3) Surgical reasons, for example, castration.

On each tab, hitting the Confirm and Consult button at the bottom of the screen will trigger the system to validate the data entered by the user. If the system finds certain required data missing or invalid data entered, an error message will display. Once user input is validated, the system will proceed on one of the following two paths.

As pointed out earlier on, if the system was set to the Standard Expert System mode at the beginning of the consultation session, a consultation result will show up right away in the Main Window or in the target machine's default Web browser, depending on the view option chosen by the user. In contrast, if Anesoft currently runs in either of the two learning modes, the system will take users to the Selection Windows (Figure 3.4), where they will have the opportunity to practice selecting drugs for induction, maintenance, preanesthetic, intraoperative, and recovery respectively. On each of the Selection windows, the user is to select drugs for a specific category of anesthesia from a list of drugs provided.

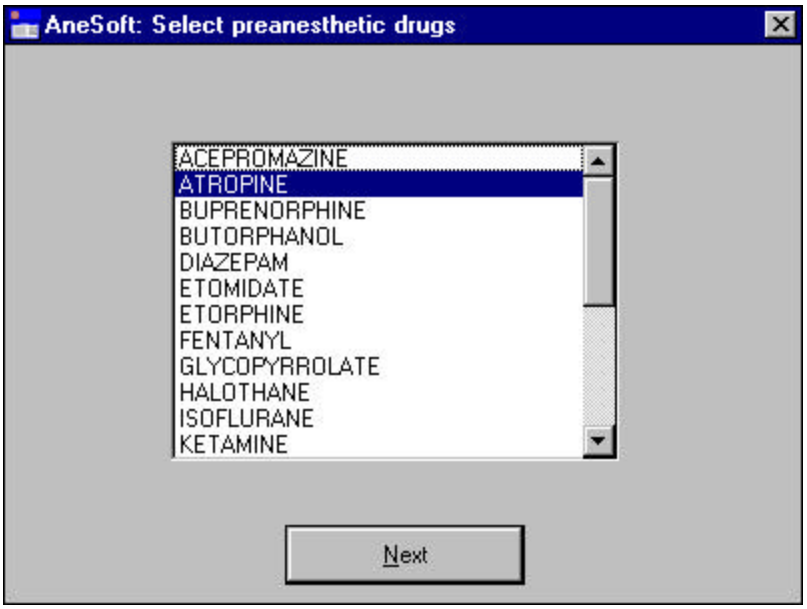

Figure 3.4 Selection Window of AneSoft

In the Supervised mode, each time the user chooses a drug from the list the system will check to see if it is contraindicated or compatible with other drugs already selected; if either condition is found to be true, a message box will pop up informing the

user why he or she can not choose that drug. The user can click the OK button in the box to return to the list. By then, the drug in question will have become deselected.

At the clicking of the Next button, AneSoft assumes that the user has completed selecting drugs for the needed anesthesia. At that point, it will check and see if there are any invalid combinations of drugs selected. If all combinations are found valid, the user is allowed to navigate to the next Selection Window.

The Critique Window (Figure 3.5) appears when the user completes selecting drugs for all of the five categories. The Critique Window lists all the drugs that have been selected but are contraindicated as well as those that have not been selected but are indicated based on the information the user provided in the Form Window. When the user clicks one of the drugs listed, a message will be displayed in the text box on the bottom, explaining why the drug is indicated or contraindicated. By clicking the Continue Critique button, the user can view the result of the current consultation, including the user's anesthetic plan, system recommended plan and the comments on the user's plan.

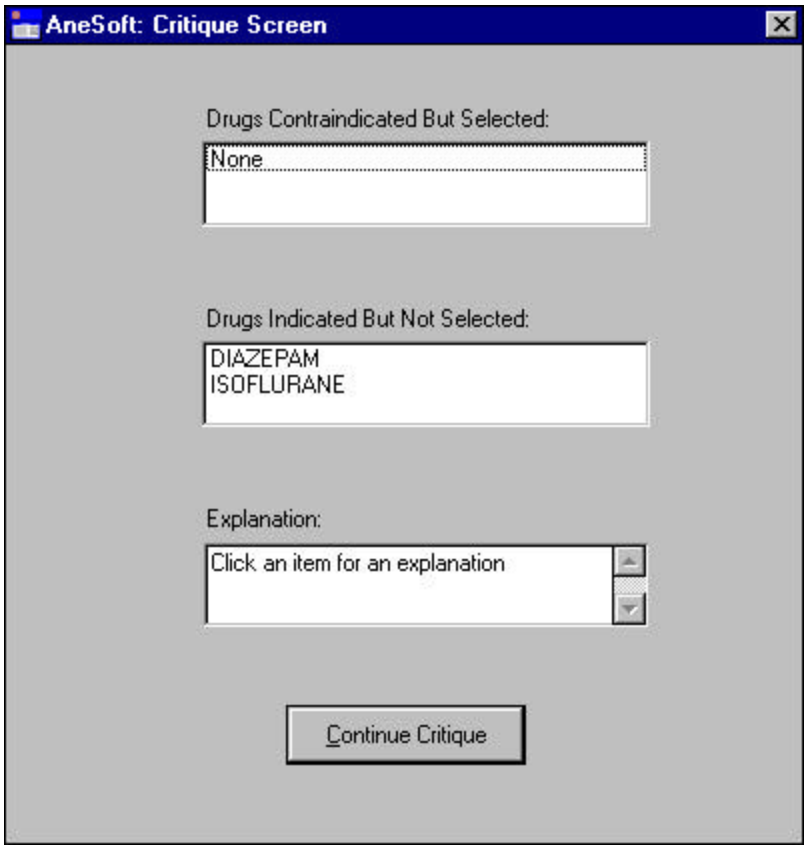

Figure 3.5 Critique Window of AneSoft

CHAPTER 4

CONCLUSION

AneSoft, an anesthetic selection expert system, has been programmed directly from human expertise and domain documents. The application is not large but fairly complex and a useful high performance system has been produced with a friendly user interface. In the development of the system, we took full advantage of what LPA WIN-Prolog has to offer, including Prolog's efficient inference engine and declarative knowledge base, and the friendly user interface, potential for rapid application development, and procedural control of the Windows-based version.

Our primary concern in Anesoft's development was ensuring the accuracy of all of its recommendations, for difficult cases as well as simple ones. This was eventually achieved by a methodical approach with attention to detail in two main areas: obtaining a good understanding of the domain, and a precise representation of this understanding in a form that can be maintained. Instead of resorting to complex artificial intelligence techniques, we employed straightforward (but detailed) logic programming, which proved to be sufficient for our purposes. Such an approach was flexible enough to allow the implementation of the system to adapt easily to the on-going modifications to the initial designs.

4.1 EVALUATION

Three criteria were established to evaluate AneSoft: Correctness, completeness, and ease of use. Correctness means AneSoft can recommend correct anesthetic plans as human experts do. Completeness refers to how thoroughly AneSoft covers the subject of making ansthetic plans. Ease of use addresses the questions of whether the interface is intuitive, easy to follow, and provides effective communications between the user and the system.

While the doman experts, other experts, and non-expert users all provided valuable feedback regarding Anesoft, Dr. Trim, Dr. Moore, and Dr. Egger played a major role in evaluating the program throughout the development cycle. With each build, they carefully tested the system and inspected each solution to make sure that AneSoft behaved as intended. Based on their suggestions, corrections, additions, and refinements were made after each evaluation session. In the final evaluation, I tested the system and submitted 20 representative results to the experts for review. In addition, I went over all these results one by one with Dr. Egger. These experts all agreed that in terms of completeness, the knowledge base of AneSoft covered a broad range of cases. They estimated that approximately 90% of the results generated by AneSoft were correct and that AneSoft is accurate enough to make anesthetic plans. All acknowledged the potential effectiveness of AneSofte in assisting students in anesthesiology learning. Evaluations by other experts and non-experts have been conducted primarily through the evaluation form that comes with AneSoft.

At the time of the last evaluation session, both domain experts and non-experts regarded AneSoft as a well-built system which runs as intended and is capable of making appropriate recommendations. They all found the program easy to learn and use, having had no trouble navigating through the program. However, they did detect some problems with certain rules and suggest several improvements for AneSoft. One expert found, for instance, that a couple of questions, recommendations and warning messages were poorly worded and suggested they be rephrased.

As far as the rules were concerned, the AneSoft built at that time permitted the user to choose a maximum of two inhalant drugs for each anesthetic plan. The experts recommended that for each plan only one inhalant be allowed, which better reflects the practice of the experts.

The above changes were all incorporated into the most recent build of Anesoft. With these improvements, I feel that the original goals set forth for the development of the system have been met.

4.2 Lessons Learned

Three major lessons were learned from the project. The first is that high performance expert systems can only be based on a good understanding of the domain. Secondly, the implementation must allow for thorough debugging and frequent revisions of the whole system. The knowledge engineer can hardly expect to obtain a sufficiently good understanding of the domain at the outset of the project. Major changes to the overall plan of the system are likely to take place as his/her understanding evolves and

matures in the course of the project. When the effects of other factors are equal, ease in maintenance seems to be the way to go. In other words, when we have more than one way to represent the domain knowledge, it is more practical to opt for the one that is easiest to maintain, provided that both represent the domain equally well.

Finally, there is no one absolutely correct method of knowledge acquisition. The exact methods used by individual knowledge engineers for knowledge acquisition and for system implementation probably do not matter that much as long as the key requirements—namely accuracy, completeness, ease in maintenance—are all met. Current theories on building experts systems often do not take into account the significant degree of personal variation present in the learning process and in building maintainable systems. Certain schemes or methods, as a matter of fact, may work better for some knowledge engineers and for some experts than they do for others. Before knowledge acquisition and system implementation, I tried several methods which are recommended in several books on techniques for expert systems. At last I used the one that best suited my needs and proved to be effective in building AneSoft.

4.3 Future Enhancement of AneSoft

First, the current knowledge base can certainly be expanded to allow more information to be included in the final consultation results, such as information on how to administer the drugs. Anesoft is more than an expert system; it is meant to be a teaching tool as well. In future builds of AneSoft, more multimedia contents can be introduced

into the system to render the software even more user-friendly and realistic and the learning process more engaging. In addition, it would be interesting to explore the possibilities of developing the current help file into a full-fledged hypertext reference component which is a mini-expert system in itself.

BIBLIOGRAPHY

Benumof, J.L. (1991) Management of the difficult adult airway. *Anesthesiology* 75: 1087-1110.

Byrne A. J., Hilton P. J., and Lunn J. N. (1994) Basic simulations for anaesthetists. A pilot study of the ACCESS system. *Anaesthesia*. 49: 376-81.

Covington, Nute, and Vellino (1997) *Prolog Programming in Depth.* Second edition. Englewood Cliffs, NJ: Prentice Hall.

Global Anesthesiology Server Network. (1999) http://www.gasnet.org

Hall, L. W. et al (1991) *Veterinary Anaesthesia*. Ninth edition. London, England: Balliere Tindall.

Jackson, P. (1998) *Introduction to Expert Systems.* Third edition. Reading, Massachusetts: Addison-Wesley Publishing Company.

Klein, M.; and L.B. Methlie (1990) *Expert systems: A decision support approach*.

Reading, Massachusetts: Addison-Wesley.

Logic Programming Associates (1997) Win-Prolog. Logic Programming Associates, Ltd., London, England.

Lumb, William V.; and Jones, E. W. (1973) *Veterinary Anesthesia*. Hagerstown, MD: Harper and Row.

Nute, Donald; and M. Rauscher (1995) A Toolkit Approach to Developing Forest Management Advisory Systems in Prolog. *AI Applications* 9(3): 39-58.

Paddleford, Robert R. ed. (1999) *Manual of Small Animal Anesthesia*. Hagerstown, MD: Harper and Row.

Plant, R.E. and Stone, N.D., 1991. *Knowledge-based Systems in Agriculture.* New York: McGraw-Hill Inc..

Short, C. E. (1974) *Clinical Veterinary Anesthesia: A Guide for the Practitioner*. Hagerstown, MD: Harper and Row.

Stefik, M. (1995) *Introduction to Knowledge Systems.* Los Altos, CA: Morgan Kaufmann.

Trim, Cynthia M., (1999) AnestheticEmergencies and Complications. In *Manual of Small Animal Anesthesia*, 2nd edition, pp.147-195. Philadelphia, Pennysvania: W.B. Saunders Company.

Trim, Cynthia M., (1999) AnestheticEmergencies and Complications. In *Manual of Small Animal Anesthesia*, 2nd edition, pp.196-226. Philadelphia, Pennysvania: W.B. Saunders Company.