VIRTUAL VETERINARY EMERGENCY ROOM: A SOFTWARE SYSTEM THAT PRESENTS DYNAMIC, INTERACTIVE MEDICAL SCENARIOS FOR TEACHING VETERINARY MEDICINE

by

JASON GEORGE SCHLACHTER

(Under the Direction of Dr. Donald Nute)

ABSTRACT

We are developing an artificial intelligence system that enables veterinary students, clinicians, and scientists to simulate clinical scenarios in veterinary medicine. The design incorporates a physiological simulator, an expert system to model the emergency room experience, an agent to analyze user performance, and a user-tested graphical interface. The program is designed to present highly accurate medical information in the same way it would be available in a true emergency room situation. The program also models many diagnostic and treatment procedures that may be used to diagnose and stabilize the patient.

This is a teaching tool that provides a bridge between textbook material and clinical practice where students can apply their medical knowledge without endangering the lives of real animals.

INDEX WORDS: Artificial Intelligence, Expert Systems, Simulation, Computer Software, Educational Software, Veterinary Education, Prolog

VIRTUAL VETERINARY EMERGENCY ROOM: A SOFTWARE SYSTEM THAT PRESENTS DYNAMIC, INTERACTIVE MEDICAL SCENARIOS FOR TEACHING VETERINARY MEDICINE

by

JASON GEORGE SCHLACHTER

B.S., The University of Georgia, 2000

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

2004

VIRTUAL VETERINARY EMERGENCY ROOM: A SOFTWARE SYSTEM THAT PRESENTS DYNAMIC, INTERACTIVE MEDICAL SCENARIOS FOR TEACHING VETERINARY MEDICINE

by

JASON GEORGE SCHLACHTER

ACKNOWLEDGEMENTS

Thesis writing is no simple task; it requires long-term planning, self discipline, and support from friends, family, and faculty. I would like to thank the following people for their contributions towards the success of my thesis research and my graduate career:

My major professor Dr. Donald Nute for providing me with technical and personal support throughout my graduate career and my thesis research.

Dr. Scott Brown for spending countless hours discussing the direction of my research and providing the expert data that was essential to the creation of VVER. This project would not have been possible without his time and effort.

My family, Mom, Dad, Mitch, Teri, Dara, David, Doug, Thomas, Daniel, Cari, Grandma Mona, Grandpa Ed, Grandma Ann. Each of you helped me stay focused and always reminded me that I am loved.

My close friends and my peers in the AI program. All of you, more than anything else, have made these last few years an incredible experience, one I will never forget.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

VVER: A TEACHING SYSTEM

The VVER Experience

A hysterical man and his son carrying a dog on a board burst through the door of the clinic screaming “Help him, help him!” The attending physician, a first year student from the University of Georgia’s College of Veterinary Medicine, glances back at me for reassurance. This dog is to be her first patient.

From speaking with the owner, she learns the patient is a Great Dane weighing sixty-five kilograms that has been hit by a car nearly twenty minutes ago. Blood begins to pool on the floor, reinforcing the urgency of the situation. The student immediately performs a physical exam and learns the patient is hemorrhaging from the right medial thigh. The injury is substantial. A quick check of her patient’s blood pressure confirms a significant amount of blood has been lost.

The student pauses to reflect on the hundreds of charts and figures from her veterinary textbooks outlining the proper care for trauma patients. Then with a bit of guidance, she inserts a catheter, begins a glucose drip, and prepares the patient for surgery to clamp the hemorrhage at the right medial thigh. In only two hours and fifteen minutes, at a cost of $720.00, the student has successfully stopped the bleeding. She then takes the dog’s blood pressure and administers a whole blood transfusion. An hour later, when the transfusion is complete, she takes the blood pressure again and decides to transfer the dog to intensive Care for monitoring. As the Great Dane recovers, she receives a briefing on her performance.

How does a first year veterinary student get to make critical decisions described above, relating to the animal's survival? The answer: she was working within a highly accurate software program designed to teach veterinary students the skills they will need to effectively diagnose and treat real animals. The software provides an interface through which students can query the physiological status of a virtual patient by ordering diagnostic procedures such as patient history, physical exam, arterial blood gases, or thoracic radiographs. Similarly, they can order treatment procedures in the form of various drug, intravenous, surgical, and other case-specific modalities to correct the disturbance. Using the software, simulated patients can be programmed into the system for observation, diagnosis and treatment. What separates this from current systems is that the virtual patients *respond dynamically* to the students' actions. Their responses are not simply scripted.

The program is called Virtual Veterinary Emergency Room, or VVER, and was developed by building upon the work of Yunxiu Xu, a former graduate student of the Artificial Intelligence Center at the University of Georgia who developed K9ER for her master's thesis in collaboration with Dr. Scott Brown from the College of Veterinary Medicine at the University of Georgia and Dr. Donald Nute of the Artificial Intelligence Center (Yunxiu, 2002).

Yunxiu's work focused on integrating an industry standard physiological simulator, QCP3, with an expert system and user interface to model emergency room medical situations. In collaboration with Dr. Brown and Dr. Nute, I extended her work by validating the medical accuracy of the program, expanding the functionality, improving the interface through user testing, refining the knowledge base of scenarios for teaching, and creating a virtual critic capable of analyzing a student's performance and providing feedback. To reflect these changes and the fact that the program can simulate several different species, the software program was renamed Virtual Veterinary Emergency Room (VVER).

The remainder of this chapter describes current methods of instruction, establishes the need for better techniques, and explains how VVER fills this need.

Current Teaching Tools in Veterinary Medicine and VVER

Today, veterinary students apply their knowledge of animal physiology to clinical scenarios through instructor led role-playing, interactive software scenarios, and live animal laboratories. However, faculty at the Veterinary College at the University of Georgia believe there is a need for improved teaching methodologies to help veterinary students master the basic scientific principles relating to the practice of veterinary medicine (Brown, 2003). Our goal in developing VVER was to combine the advantages of existing methods into a single model while eliminating their disadvantages.

The first of these, instructor-led role playing, occurs in the classroom when a professor leads his or her class through a hypothetical medical scenario. It begins with the instructor describing an initial set of conditions and continues as the class works through the process of diagnosing and treating the patient. At each point in the process, the professor and the students determine what actions should be taken and the effect they will have on the patient. Training is dynamic because both students and professor determine the flow of the scenario as it unfolds. This is important because real life medical traumas rarely follow scripts, and the closer a training exercise is to its real life counterpart, the more effective it will be as an educational tool. One disadvantage is that professors must develop and present the scenario, so its use is limited to class time. Furthermore, the whole class is responsible for the outcome of the scenario, not the individual student.

Another technique utilizes interactive teaching software to model clinical scenarios and engage the student in the decision making process of emergency medicine. This method is advantageous because a single case created by faculty can be used an indefinite number of times by any number of students for no additional cost to the developer. Many such scenarios can be found on the internet and used by anyone with access to a computer. Unfortunately, it requires more time and effort to develop them than to develop those used for instructor-led role playing. This is because software developers must map out every possible situation that might exist and every action that might be taken before they create software to model it. This process requires a lot of forethought and limits robustness because developers cannot anticipate every situation that might arise in real life clinical scenarios. An obvious disadvantage in using this approach is that the case flow is usually scripted and generally fails to fully simulate the dynamic environment of real life veterinary medicine.

Live animal laboratories are also used to simulate clinical scenarios for teaching. This approach is the most similar to actual clinical practice. Unfortunately, it contains several serious drawbacks. For instance, animal laboratories are sometimes designed so that the animal will be euthanized in the end. This is especially true for vivisections, and ethical concerns have reduced their presence in the veterinary curriculum. It is also costly and resource intensive. While live animal training has its place in veterinary education, the need for such a model can be reduced with appropriately sophisticated software.

All of these existing methods have unique advantages and disadvantages inherent in their design. The goal in developing VVER was to create a system that would combine the advantages of current methods while eliminating or minimizing their disadvantages. To this end, VVER combines a physiological simulator and expert system that allows for a dynamic case flow, something that was previously only available with instructor led role playing and live animal labs. The advantages of a software approach are that it may be distributed and used by many students with no or little additional cost to the developer.

To further enhance its use as an educational tool, this computer program will undergo field tests in existing courses at the College of Veterinary Medicine at the University of Georgia.

The idea of using computer simulation and expert system software to teach is not a novel one. Computers provide an interactive environment that can be used to expand an educational experience. They are able to present information to the user, allow the user to interact with that information and they give immediate feedback. The students who participated in the user testing for VVER were excited about its interactivity and felt the software forced them to think differently.

In addition, many commercial, military, research, and educational institutions use computer systems to teach or train personnel. For example, the University of Southern California has developed an extensive collection of training software to prepare military officers for combat (ICT Games Project, 2004). In the life sciences and medical field, computer-based models of the human body have been developed to supplement anatomy course work when laboratory resources are not readily available (Mackenzie, Baily, Nitsche, and Rashbass, 2003).

A search of existing veterinary databases and relevant academic journals reveals there are no software programs that address these specific needs.

With the development of Virtual Vet Emergency Room our aim was to change the status quo by using available technologies in a novel way. The software surpasses the functionality of current systems by providing a more dynamic environment designed to instruct as well as to evaluate, something invaluable to the field of veterinary medicine

CHAPTER 2

SYSTEM ARCHITECTURE

VVER consists of five major components; the VVER Control Module, User Interface, Interface to the Simulator, Simulator, and the Expert System. Figure 1 shows the paths of communication between each component of VVER. It also groups components together based on the computer language in which they are programmed. Note that the VVER Control Module, the Expert System, the Critic file, the Case file, and the User interface are all programmed in WIN-Prolog, a language developed by Logic Programming Associates (Logic Programming Associates, 2003). The Interface to the Simulator is programmed in C++ and the Simulator is programmed in C.

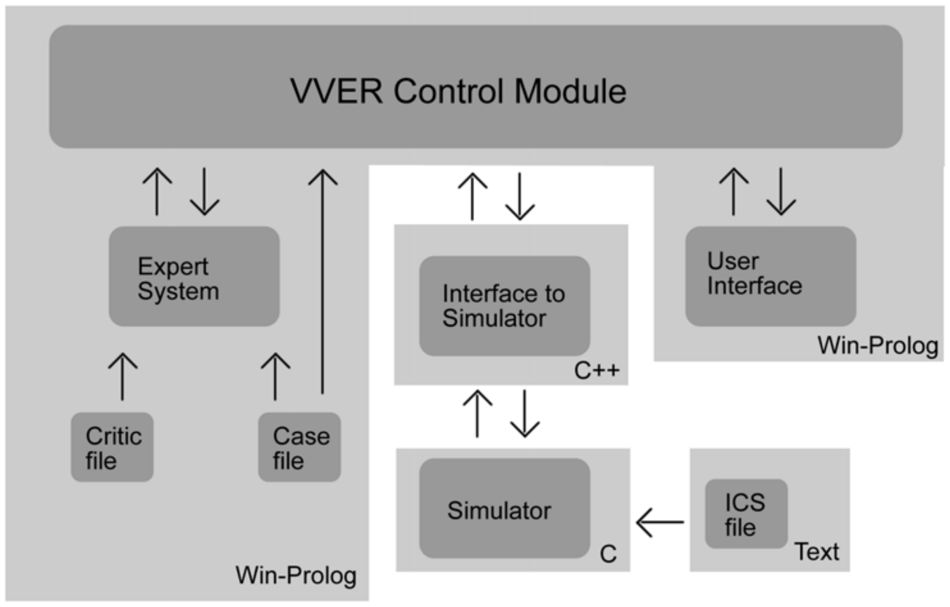
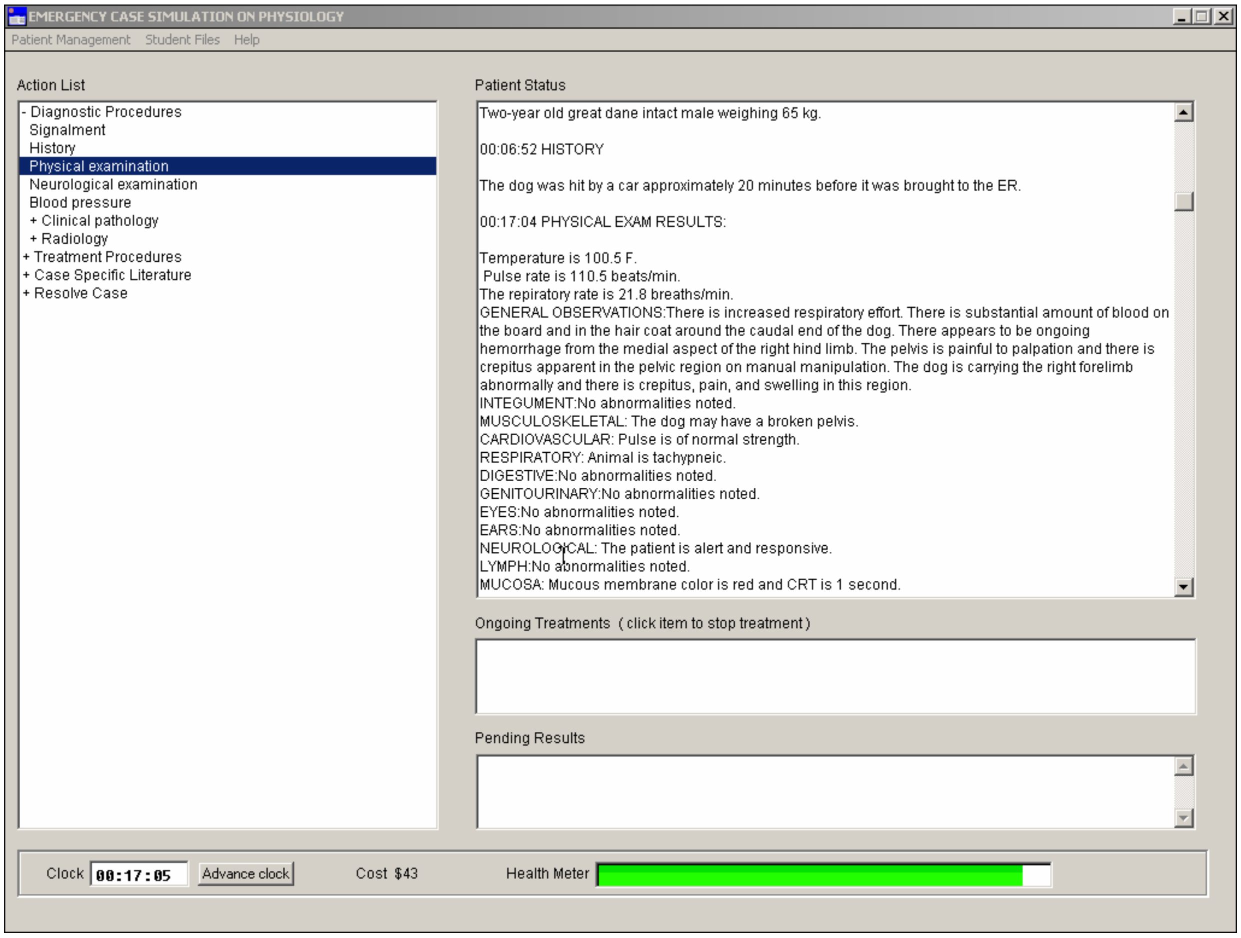


Figure 1. VVER System Architecture

The VVER Control Module

The VVER control module is written in WIN-Prolog and is responsible for communicating with the expert system, the interface to the simulator, and the user interface. It contains the methods and procedures that run VVER and allow communication between its various components. The code for this module is primarily contained in the file `emergency0702_interface.pl.`



```
Two-year old great dane intact male weighing 65 kg.

00:06:52 HISTORY

The dog was hit by a car approximately 20 minutes before it was brought to the ER.

00:17:04 PHYSICAL EXAM RESULTS:

Temperature is 100.5 F.
 Pulse rate is 110.5 beats/min.
The repiratory rate is 21.8 breaths/min.
GENERAL OBSERVATIONS:There is increased respiratory effort. There is substantial amount of blood on
the board and in the hair coat around the caudal end of the dog. There appears to be ongoing
hemorrhage from the medial aspect of the right hind limb. The pelvis is painful to palpation and there is
crepitus apparent in the pelvic region on manual manipulation. The dog is carrying the right forelimb
abnormally and there is crepitus, pain, and swelling in this region.
INTEGUMENT:No abnormalities noted.
MUSCULOSKELETAL: The dog may have a broken pelvis.
CARDIOVASCULAR: Pulse is of normal strength.
RESPIRATORY: Animal is tachypneic.
DIGESTIVE:No abnormalities noted.
GENITOURINARY:No abnormalities noted.
EYES:No abnormalities noted.
EARS:No abnormalities noted.
NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.
```

Figure 2. VVER Interface after Ordering Signalment, History, and Physical Exam

The User Interface

The user interface is also programmed in WIN-Prolog and is graphical in nature. It provides an environment from which the user can order diagnostic, treatment, or resolution procedures and can view the patient's response to the same. It also displays information relevant to the medical scenario such as: The time the patient has been in the emergency room; the general health of the patient; and the cost of treatment. The user interface communicates with the VVER control module, which in turn calls on the Expert System when necessary. The interface is depicted in Figure 2, and the code can be found in `emergency0702_interface.pl`.

The Interface to the Simulator

The interface to the simulator is coded in C++ and its sole purpose is to allow the control module, coded in WIN-Prolog, to communicate with the simulator, coded in C. To communicate with the interface to the simulator, the VVER control module writes commands to a text file and sets a system flag. The interface to the simulator reads the file when the flag is present and sends the commands on to the simulator. The commands are of two forms: Either they ask the simulator to generate additional physiological data for the virtual patient; or they ask the simulator to alter some physiological variable in the model and then run the simulator. Once the simulator receives and executes the commands, the interface to the simulator outputs the appropriate physiological data to a text file and also sets a system flag. The VVER control module reads from the file when the flag is present, adds this information to its knowledge base, and removes the flag.

The Simulator

VVER uses a physiological simulator called Quantitative Circulator Physiology, or QCP3, developed by Biological Simulators, Inc. for the simulation of human physiology (Coleman, 1999). The software maintains the relationship between more than 1200 physiological variables through differential equations. Since the simulator is designed to model the human body, input to the simulator and output from the simulator must be scaled appropriately. The expert system contains rules that specify how this data should be scaled for each species of animal that may be simulated in VVER.

To begin a simulation, QCP3 must load initial values for each of the more than 1200 physiological variables from an ICS file. Since there are several of these files that come with QCP3, the case file must specify which of them will be used to begin the simulation. For example, in the Scooby scenario, the file `scooby.case` tells the VVER control module to load the `mr_norm.ics` to begin the simulation.

The Expert System

Expert systems are “computer programs designed to perform tasks that require special knowledge or problem-solving skills” (Covington, Nute, and Vellino, 1988, pp265-267). They have three major components: An interface; a knowledge base; and an inference engine (Covington, Nute, and Vellino, 1988, pp265-267).

The expert system interface communicates with the VVER Control Module which, in turn, calls the Interface to the Simulator or the User Interface when necessary. While it could be said that the VVER user interface is a part of the expert system, I feel that the complexity of this interface combined with the fact that it interacts with components outside of the expert system make it a separate component.

The knowledge base consists of facts and rules programmed in WIN-Prolog. Some of these are built into VVER, such as those that model diagnostic and treatment procedures. Others are case-specific and are added to the knowledge base when a medical scenario is loaded. The inference engine is Prolog itself, since the language uses its own inference engine to execute code.

The expert system knowledge base models the following nine diagnostic procedures:

```
1.  `History`
2.  `Signalment`
3.  `Physical examination`
4.  `Neurological examination`
5.  `Arterial blood gases`
6.  `Hematology, Biochemical Profile, and Urinalysis`
7.  `Exercise Fitness Test`
8.  `Thoracic radiographs`
9.  `Blood pressure`
```

Each procedure is represented by one or more rules of the following form:

```
% diag_proc/6
%
% diag_proc(+Name,+Cost,+TimeToOnset,+TimeToPerform,+Report,+SimulationInstructions).
%
%       +Name           <string>       name of diagnostic procedure
%       +Cost           <number>       cost of procedure in dollars
%       +TimeToOnset    <number>       number of minutes b/f procedure is executed
%       +TimeToPerform  <number>       number of minutes required to perform the procedure
%       +Report         <number>       number of minutes b/f results will be back
%       +SimCommands    <list>         list of instructions to the program/simulator
%
```

Each rule contains six parameters that are used to model the procedure. The first parameter, `+Name`, contains a string that denotes the procedure's name as it appears in the action list window. The second parameter, `+Cost`, contains a number that denotes the cost of the procedure in dollars. The `+TimeToOnset`, `+TimeToPerform`, and `+Report` parameters denote the amount of time required to begin the procedure, the time required to perform the procedure, and the time before results will be available, respectively. The sixth parameter, `+SimCommands`, is a list that contains text to be written to the patient status window as well as commands to VVER or variables that will acquire values as the rule executes. The rule for checking Arterial Blood Gases is used below to illustrate this point.

```
diag_proc(`Arterial blood gases`,34,0,5,15,
        [`ARTERIAL BLOOD GASES`,`~M~J`,
         `~M~J`,`pH `,
         'Blood Acid, Venous pH',
         ` (normal range 7.30 - 7.50); Arterial PO2 = `,
         'Arterial O2, pO2',
         ` mmHG (normal range 90 - 100); Arterial PCO2 = `,
         'CO2 Systemic Arteries, pCO2',
         ` mmHg (normal range 35 - 45)`]).
```

We can see that this procedure will cost the pet's owner $34 (2nd parameter) and will begin immediately after it is ordered (3rd parameter). We also see that it will take five minutes to perform (4th parameter), but results will not be back from the lab for another fifteen minutes (5th parameter). The last parameter is a list that contains text enclosed in (`) that will be written to the patient status window and commands enclosed in (') that will instruct the control module to query the simulator for physiological data.

For example, `'Blood Acid, Venous pH'`, `'Arterial O2, pO2'`, and `'CO2 Systemic Arteries, pCO2'` denote physiological variables that instruct the control module to generate queries for the simulator. The results from these queries will be concatenated with the other procedure specific text and written to the patient status window. Below is an example of what might be written to the patient status window when this procedure is executed.

```
00:20:40 ARTERIAL BLOOD GASES

pH 7.3 (normal range 7.30 - 7.50); Arterial PO2 = 91.5 mmHG (normal range 90 - 100); Arterial
PCO2 = 37.5 mmHg (normal range 35 - 45)
```

The procedure illustrated above is a relatively simple one because it lacks variables that must be given values, or instantiated, through additional computation.

Let's examine a more complex rule that models the `` `Physical examination` `` procedure and contains variables that must be instantiated during execution. Variables are denoted by a string of alpha-numeric characters that begin with a capital letter and are not enclosed in quotation marks. In the following code, `Cardio_String`, `Resp_String`, `Nervous_String`, and `Mucosa_String` are variables that must be instantiated by the rule as it executes.

```
diag_proc(`Physical examination`,Cost,0,10,0,
               [`PHYSICAL EXAM RESULTS: `,`~M~J`,
               `~M~J`,`Temperature is `,'Core Heat, Temp (F)',` F.`,
               `~M~J`,` Pulse rate is `,'Heart Rate, Rate',` beats/min.`,
               `~M~J`,`The repiratory rate is `,
               'Resp Center, Respiration Rate',` breaths/min.`,
               `~M~J`,case(physical_exam,general),
               `~M~J`,case(physical_exam,integument),
               `~M~J`,case(physical_exam,musculoskeletal),
               `~M~J`,Cardio_String,
               `~M~J`,Respiratory_String,
               `~M~J`,case(physical_exam,digestive),
               `~M~J`,case(physical_exam,genitourinary),
               `~M~J`,case(physical_exam,eyes),
               `~M~J`,case(physical_exam,ears),
               `~M~J`,Nervous_String,
               `~M~J`,case(physical_exam,lymph),
               `~M~J`,Mucosa_String]):-
```

This list is the `+SimCommands` *parameter.*

```
        current_time(Time),
        Index is Time//60,
        f('Systemic Arteries, Systolic BP',Index,SystolicValue),
        f('Systemic Arteries, Diastolic BP',Index,DiastolicValue),
        Difference is SystolicValue-DiastolicValue,
        (
          % **if diff b/w systolic & diastolic BP is < 20 mmHg
          Difference  =< 20,
          Cardio_String = `CARDIOVASCULAR: Pulse is weak. `
        ;
          % **if diff b/w systolic & diastolic BP is b/w 20-50 mmHg
          Difference >= 20,
          Difference =< 50,
          Cardio_String = `CARDIOVASCULAR: Pulse is of normal strength. `
        ;
          % **if diff b/w systolic & diastolic BP is > 50 mmHg
          Difference >= 50,
          Cardio_String = `CARDIOVASCULAR: Pulse is pounding. `
```

Here the system compares values from the simulator to determine what text will be assigned to the `Cardio_String` *variable.*

The remaining variables in `+SimCommands` are instantiated in a similar manner but I have not included the code here. Consult the file, `vver_procedures.pl` for the rest of the code.

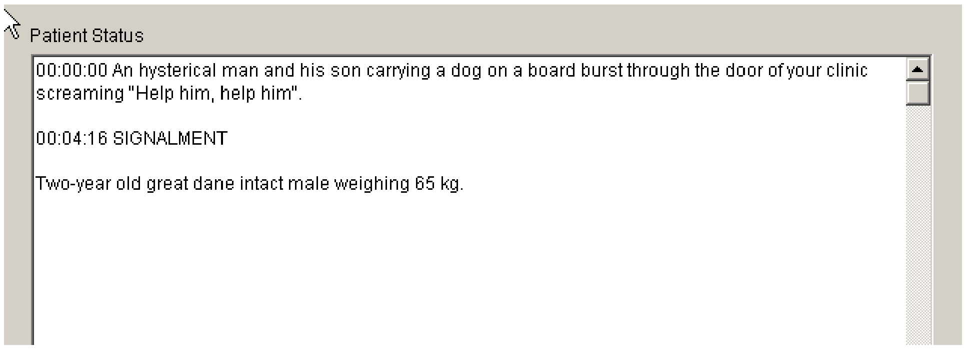
In addition to the foregoing, the expert system models the following twenty-four treatment procedures.

```
1.  `Insert catheter`
2.  `General Anesthesia`
3.  `IV lactated ringer's drip`
4.  `Whole blood transfusion`
5.  `Administer IV glucose drip`
6.  `Stop hemorrhage`
7.  `Administer CPR`
8.  `Place on Oxygen`
9.  `Place animal on ventilator`
10. `Pericardiocentesis and drain`
11. `Administer epinephrine`
12. `Give IV dolbutamine drip`
13. `Administer propranolol`
14. `Administer phenoxybenzamine`
15. `Administer atropine`
16. `Administer cardiac specific calcium channel blocker`
17. `Administer enalapril`
18. `Administer single dose furosemide`
19. `Administer chlorothiazide chronically`
20. `Take digoxin chronically`
21. `Insulin injection`
22. `Administer erythropoietin`
23. `General Anesthesia`
24. `Perform Surgery`
```

The rules that model these procedures are similar to those used to model the diagnostic procedures, with a few important exceptions. The format for the treatment procedure rules is as follows:

```
% treat_proc/6
%
% treat_proc(+Name,+Cost,+TimeToOnset,+TimeToPerform,+Report,+SimulationInstructions).
%
%      +Name          <string>       name of diagnostic procedure
%      +Cost          <number>       dollar cost of procedure
%      +TimeToOnset   <number>       number of minutes b/f procedure is executed
%      +TimeToPerform <number>       number of minutes required to perform the procedure
%      +Report        <list>         list of strings displayed in patient status window
%      +SimCommands   <list>         list of instructions to the program/simulator
%
% These predicates determine the treatment procedures that exist in the program
```

The `+Name`, `+Cost`, `+TimeToOnset`, and `+TimeToPerform` parameters are the same as for the diagnostic procedures, however the `+Report` parameter performs a different function in this context. This is because treatment procedures do not return results so there is no need to specify how long it will take before they are back. Instead, this parameter contains a list of text strings that are displayed to the patient status window, shown in Figure 3, when the procedure is performed.



```
00:00:00 An hysterical man and his son carrying a dog on a board burst through the door of your clinic
screaming "Help him, help him".

00:04:16 SIGNALMENT

Two-year old great dane intact male weighing 65 kg.
```

Figure 3. Patient Status Window

The last parameter, `+SimCommands,` is still a list as in the diagnostic procedures, but here it only contains commands that are used by the control module to modify physiological variables in the simulator. To illustrate, let's look at the `` `Administer epinephrine` `` rule.

```
treat_proc(`Administer epinephrine`,35,0,1,
               [`Epinephrine administered at the rate of 10,000.`],
               [(`"Epi Pump, Switch"`, 1),
                (`"Epi Pump, Setting"`, 10000)
               ]).
```

Notice how the first parameter, `+Name`, contains the procedure's name as it appears in the action list window of the user interface. The second parameter, `+Cost`, indicates that it costs $35 to perform. The procedure begins immediately when ordered, as indicated by the third parameter, `+TimeToOnset`, and is completed in one minute, as specified by the fourth parameter, `+TimeToPerform.` The second line of code contains the fifth parameter,

`+Report`, which specifies the text that will be written to the patient status window when the procedure is completed.

The third, fourth, and fifth lines of code contain the last parameter, `+SimCommands`, which is a list of commands to the simulator in the form of two element pairs. The first element denotes the simulator variable and the second specifies its new value.

CHAPTER 3

CREATING THE EXPERT SYSTEM

History

Knowledge engineering is a critical part of expert system development and often the most resource intensive. Yunxiu Xu did a significant amount of knowledge engineering for her software, K9ER. She met with veterinary experts over a two year period to model diagnostic and treatment procedures and to create medical scenarios. All this work formed the foundation for my software and provided a starting point for my knowledge engineering.

When I inherited the project from Yunxiu Xu, I examined her existing medical scenarios and used them to create new ones within the framework of her software. In doing this, it became clear that her software was an excellent proof of concept but it needed additional work before it could be used as course software. Many of the procedures, as well as the control structure, case file architecture, and user interface needed to be retooled to improve medical accuracy and usability before the application could be incorporated into veterinary school curriculums. This became the focus of my thesis work for the following year.

Developing Diagnostic and Treatment Procedures

Many of the diagnostic and treatment procedures available in VVER were developed by Yunxiu when she created K9ER. I expanded upon her work by developing procedures for conducting neurological examinations, exercise fitness tests, and for inserting a catheter and performing surgery. Additionally, some of the procedures created in K9ER underwent significant modification as they made their way into VVER based on data collected during user testing.

One such procedure is the physical examination. In K9ER it functioned by displaying case-specific text that it read in from the case file. This text was static, meaning it was always the same. This proved problematic because as the patient's condition changed in the simulator, the text did not. During initial user testing with VVER, students were frustrated and often confused as to whether the procedures they were ordering had any effect on the patient.

To address this concern, the physical examination procedure in VVER was modified to generate text dynamically based on the physiological state of the patient whenever possible. Since not all of the text necessary for the physical examination can be generated this way, the representation of case-specific text in the case file was altered to accommodate rules that dictate under what conditions the text should be displayed. In this way, text that can not be generated from the physiology of the patient can still be somewhat dynamic in nature.

K9ER also lacked a procedure for a neurological examination. In our user testing, many students felt that this procedure would aid them in diagnosing and treating the patient; so Dr. Brown and I met to develop such a procedure. Dr. Brown identified variables within QCP3 that reported on the nervous system and I developed procedures that generated text based on their values.

Through our user testing I was also able to identify several inconsistencies in our system that resulted from the way in which treatment procedures were processed. For example, many procedures, such as administering an IV lactated ringer's drip, could be done with out inserting a catheter, something necessary in actual clinical practice. This became problematic when students who had administered IV liquids without a catheter, then tried to perform surgery without anesthesia. They were surprised to find that the program interceded with an error message and warned them to anesthetize the patient first. They felt the assumptions made by the software were inconsistent. In response to this feedback, Dr. Brown and I decided to revamp many of the procedures to ensure that the assumptions made in VVER were consistent across all procedures. This process required a significant addition to our knowledge base in the form of logical relations between one or more procedures. Since the vast majority of the procedures in K9ER were used in VVER, development of this aspect of the knowledge base became an iterative process of evaluating the model and redesigning features to increase the accuracy with which VVER modeled the clinical experience.

Developing Medical Scenarios

To create new scenarios I met with Dr. Brown to decide what species of animal and what medical condition(s) to model. Then we identified the physiological parameters that had to be changed from the simulator's default configuration to model the patient's condition. He provided case-specific text for the diagnostic procedures and the conditions under which the text would be displayed. At our next meeting Dr. Brown gave me this information and we outlined the best course of action to diagnose and stabilize the virtual patient. In discussing the case, we determined how to represent this knowledge in the case file and identified potential technical hurtles that might arise in doing so.

To facilitate a standard method for creating scenarios, I developed a template file to form the basis for each case. The template contains default implementations of all case-specific rules and facts that are needed by VVER. Procedures may be left in their default state or they may be replaced with appropriate code. For instance, the canned text for 'General Observations' of the 'Physical Examination' procedure is set to display 'No abnormalities noted.' However, this could also be changed to display ‘The patient appears tired and is having difficulty standing up’ with a simple modification.

Armed with the expert data and case template, I created case files that contain all case-specific changes to the simulator, case-specific text, and other supporting code to model the scenarios. The process of case creation is described in more depth in APPENDIX A: HOW TO CREATE A NEW SCENARIO.

When the case was complete, I loaded it into VVER to be sure it functioned properly. When I was certain that the code had no syntactical and logical errors, Dr. Brown and I put the scenario through a trial run. This step was crucial because it allowed us to compare the actual responses of the virtual patient to the responses we expected. If our treatment regimen was correct, and the patient’s condition was modeled correctly, then our assumption was that we should have been able to diagnose and stabilize the patient without difficulty.

However, if our treatment did not produce the expected results, we needed to re-evaluate our model. This may have been because some procedure was not modeled exactly as we had assumed or perhaps our treatment plan was improper. In either case, the expert data and the implementation of the scenario were reexamined to improve the medical accuracy of the model and another trial run was performed. Verifying the scenario through extensive testing was a crucial step in understanding the model and improving its medical accuracy. Thus the process of scenario creation became an iterative process of design, implementation, and verification.

Since the model is dynamic, it also had to react predictably to any combination of procedures, not just those allowing for ideal solutions. Even after determining that our ideal treatment regimen worked as expected, many other combinations of procedures had to be tested in an attempt to ‘break’ the model. If the model ‘broke’ during these tests, then it returned to the design phase of development and the process continued. After several iterations of tests, if the model failed to 'break', then we assumed it was robust and medically accurate. Of course the case underwent further testing during user trials but this came later in the development cycle. Figure 4 shows this process in flow chart form.

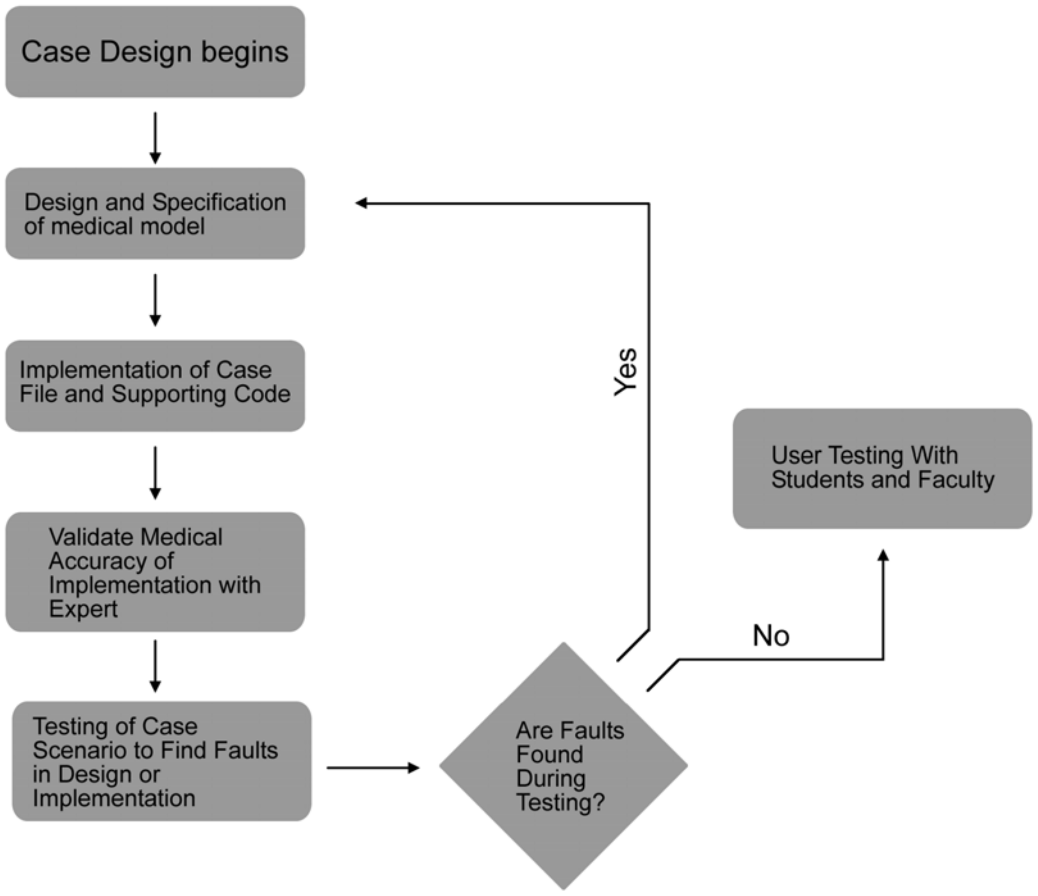

Case Design begins
Design and Specification of medical model
Implementation of Case File and Supporting Code
Validate Medical Accuracy of Implementation with Expert
Testing of Case Scenario to Find Faults in Design or Implementation
Are Faults Found During Testing?
Yes
No
User Testing With Students and Faculty

Figure 4. Flow Chart of Scenario Development

Developing the Critic

VVER was designed so that the expert system could provide feedback to the student on their performance at the conclusion of the medical scenario. This required the development of critic rules and a critic agent to review the procedures ordered by the user in relation to the patient's physiological state.

To generate the rules, Dr. Brown and I leveraged the scenarios we had previously validated by working through them and considering how he would evaluate the potential actions the user might take. He was able to present me with a list of procedures and identify those that were indicated, contra-indicated, or neutral for a specific patient. Additionally, I would played the role of a student and asked, "Should I order surgery before I perform a neurological examination?" or "How soon after admitting a patient should I have performed a physical examination?" to elicit his feedback. In this way we generated upwards of 30 to 40 rules for a given case.

Three types of critic rules were created: rules that label a procedure as indicated, contra-indicated, or neutral; rules that specify the relative order in which procedures should be executed; and rules which specify a time threshold by which a certain procedure should be ordered.

Rules were added to the knowledge base by going through the critic rules developed for existing medical scenarios and extracting those that were global in nature. Those that were global in nature, meaning they apply to all medical scenarios, were incorporated into the expert system knowledge base while those that were case-specific are specified in a critic file. This file has the same name as the case file but with the `.critic` extension. For example, the case file `scooby.case` also has a corresponding file, `scooby.critic`, which contains rules that are added to the knowledge base when the case is started.

Rules that label a procedure as indicated, contra-indicated, or neutral are generally case specific. For example, in the case of Scooby, the hemorrhaging dog, performing surgery to clip the hemorrhage is an indicated procedure. Likewise, all supporting procedures such as ordering a physical examination, taking the blood pressure, inserting a catheter, administering general anesthesia, and performing a whole blood transfusion are also indicated procedures. Procedures that would have a negative effect on the patient's condition, such as administering epinephrine or performing an exercise fitness test are considered to be contra-indicated. Those that have little effect on the diagnosis or treatment of the patient but nonetheless incur additional cost and time are neutral. For example, conducting a neurological examination might be coded as a neutral procedure since it is does not worsen the patient's condition but it is unlikely to aid in the diagnosis of the patient.

Critic rules based on relative order are often general enough to include in the program's knowledge base. For example, the knowledge base contains a rule that models the fact that a physical examination should be performed before a patient is placed under general anesthesia or before surgery is performed. Another rule models the fact that epinephrine should be administered before CPR is performed on the patient.

Time based rules may be either global in nature or case-specific and are used to specify an absolute time by which a specific procedure should be ordered. For instance, there are global time based rules that indicate the history and signalment should be ordered within the first five minutes and the physical exam should be conducted within the first fifteen. Other time based rules are case-specific such as those in the `scooby.case` file that indicate general anesthesia and surgery should be performed within three hours and the patient should be stabilized and in the ICU within seven.

In addition to the three types of rules discussed above, the system is also capable of processing health based rules that identify indicated and contra-indicated procedures based on the physiological state of the patient. However, there are no rules of this sort currently implemented in VVER or in the critic files. Future work might focus on specifying these types of rules since the critic agent can handle them and the format for such rules has been documented in the source code

CHAPTER 4

USER TESTING

User testing was performed early in the development of VVER and the data collected was used in project planning. The study was not intended to be a rigorous scientific study with statistically significant results. Instead, the goal was to observe students using the software and to perform a qualitative analysis on their feedback to identify design features that caused confusion or prevented them from completing the assigned tasks. In her book, “GUI Design for Dummies,” Laura Arlov describes a good user test as “one where you learn something new – in other words, one where you get to observe problems” (Arlov, 1997). This was the aim of my user testing study.

The number of participants used was determined from the guidelines outlined by Jakob Nielson in his book, “Usability Engineering.” He recommends testing three to five participants to obtain the best benefits-to-cost ratio (Nielsen, 1994). This is consistent with the numbers of participants I have seen used in industry. Therefore, I scheduled fifteen students for testing with the expectation that there would be some level of attrition and that it might be useful to have a greater amount of user feedback, even if it resulted in a less than optimal benefits to cost ratio. Permission to use human subjects was obtained from the IRB.

Methods

Participants were selected from Dr. Scott Brown's first-year physiology class at the College of Veterinary Medicine in the spring of 2003. This is the same demographic that will use the software when it's deployed next fall at the veterinary college.

To recruit participants, I asked Dr. Brown's students if they would volunteer for the study. I then distributed a sign-up sheet to each student with a brief description of the software and asked them to provide their contact information. The students were not given compensation or grade incentives, nor did I inform them that their professor, Brown, was associated with the project in any manner.

From a class of approximately seventy-five students, sixty-nine indicated they would participate. Of these students, I randomly selected fifteen to test the software in an isolated area at their college's library. Ten of them followed through with their commitment to complete the testing. The five that did not participate failed to arrive at the testing site during their scheduled day and time. When I contacted them later to follow up on their absence, they cited forgetfulness and/or last-minute scheduling conflicts.

Participants were tested individually in sessions lasting approximately one hour. This included an introduction to the software, a demonstration of a sample scenario, a scenario for the user to test and a user survey. In each session, the testing was presented in a standard format to minimize variation across participants. This format is outlined in APPENDIX E: USER TESTING: TESTING PROTOCOL.

When students arrived at the testing site, they were given a consent form and asked to sign it. The form can be found in APPENDIX D: USER TESTING: CONSENT FORM. After signing, they received a five-minute orientation in which I explained who the intended audience for the software was; the major functions of the software; and the corresponding controls on the user interface. They then were presented with a sample scenario in which I diagnosed, treated and stabilized a virtual patient in VVER. This second phase generally lasted approximately ten minutes.

In the third phase, students were asked to load a scenario on their own. They were instructed to diagnose and treat the patient safely, quickly, and without incurring excessive costs. Throughout the process, they were asked to verbalize their thoughts as they used the software, a method known as “thinking-out-loud.” I documented their thoughts, but was careful not to prompt them for comments or lead them in their thinking. This method of obtaining feedback during user testing is widely used and accepted in the industry for gathering qualitative data (Barnum, 2002). Generally, most students completed this phase of testing within thirty minutes.

At the conclusion, participants were given a survey that contained twenty-eight statements and were asked, using a five-point likert scale, to indicate the extent to which they agree or disagree with each statement (Saccuzzo and Kaplan, 1996). A likert scale is “a rating scale measuring the strength of agreement towards a set of clear statements” that is often used in the field of psychology and generally “administered in the form of a questionnaire used to gauge attitudes or reactions” (Usability by Design Ltd, 2004).

The survey also contained the following open-ended questions: “What are the program’s strengths?”, “What are the program’s weaknesses?”, “What could be changed, added or removed to make the program easier to use?” and “Please add any other thoughts or impressions about your experience with K9ER.” Participants were asked to write a sentence or more for each question. The complete survey can be found in APPENDIX F: USER TESTING: SURVEY FORM.

When students completed the survey, I debriefed them on the study by answering their questions and thanked them for their participation.

Results

The average user response for each of the twenty-eight survey items was calculated and standard deviations were found. Averages that deviated significantly from what I had expected, or from what I wanted, were used to identify design features that required refinement.

Some of the survey statements were positive in nature and responses like, “I somewhat agree” or “I agree” (numerically “four” or “five”) were desired. For statements that were negative in nature, responses like, “I disagree” and “I somewhat disagree” (numerically “one” or “two”) were desired. In order to compare responses across all survey items, the responses from negative statements were inverted so that higher scores always corresponded to desired responses. In this way a score of one on a negative item would be come a five and a score of two would become a four. I was then able to define desirable responses as those at or above 4.0 and undesirable responses as those below this cutoff.

Figure 5 below shows the averages and standard deviations for each survey item. The averages are represented by the grey bars and the standard deviations by the black lines that extend from the top of each bar. The averages for items 3, 5, 12, 19, 20, 21, and 24 are below the 4.0 cutoff, indicating that the responses for these statements were undesirable.

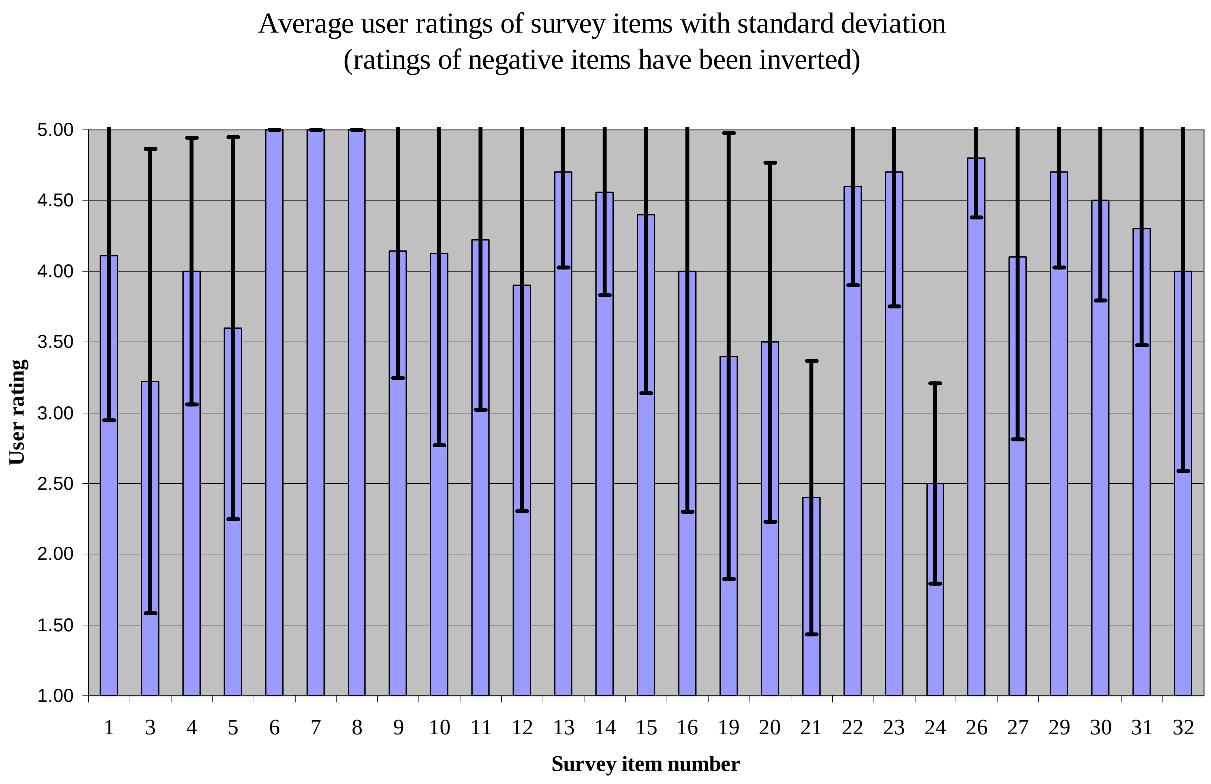


Figure 5. Survey Results from User Testing

The statements that generated undesirable responses are shown in Table 1 along with their averages. Responses from the open ended questions were also a critical part of the survey. In order to process them effectively, I created tables with the responses for each question and their frequency values, or the number of times a nearly identical response was given. This data is also displayed in Tables 2, 3, 4, and 5.

Table 1. Survey Statements That Generated Undesirable Feedback

Item #	Corresponding Statement	Avg.
3	It was not clear what type of information each of the white text boxes would display.	3.22
5	The long wait time required to simulate the patient's condition at the beginning of the case made me wonder if the program was working properly.	3.60
12	When viewing the available treatment procedures, it was clear that the plus signs to the left of each line indicated that the item could be expanded to reveal more specific treatment procedures.	3.90
19	After beginning the case, I felt confused about what I should do next.	3.40
20	After loading the case, the program would have been easier to use if it gave instructions on what options and tests were available to the user.	3.50
21	After each procedure, the program would have been easier to use if it presented a list of options available to the user.	2.40
24	After requesting a treatment procedure that might have stabilized the patient, I was confused about what to do next.	2.50

Table 2. Open Ended Survey Questions: The Program's Strengths

What are the program's strengths?	
Answers:	**Frequency:**
the software is a great idea	1
good that the data is dynamic from the simulator	2
easy to understand interface	1
it helped me to think about my cases and plan my course of action	2
it was fun	1
interesting and thought provoking	1
allows for application of clinical knowledge without endangering real patient	2
caused me to feel pressured to manage a case appropriately	1
the ability to offer the user information as in a real life situation and then the user's ability to respond to this information and get feedback on his/her actions	2
solo interaction is good vs. classroom Q&A	1
it integrates cost, time, & treatment	1

Table 3. Open Ended Survey Questions: The Program’s Weaknesses

What are the program's weaknesses?	
Answers:	**Frequency:**
not clear how animal responds to treatment	1
recommends that we work more closely with emergency vets to generate more realistic scenarios	1
time for blood chemistry seemed long for a private practice	1
recommends that costs should be more realistic	1
it would benefit from more diagnostic and treatment procedures	4
some areas, such as surgery, are too general	1
would like the text from the physical exam to change and not just the numerical data	1
information sequencing needs tweaking and better understanding	1
it should be clarified what is assumed and what is not and what procedures are available	1
just the software bugs	1
needs to be more visual to help it be more interactive (Images/photos/videos)	1
add a random function such that users can wait in ER for a case to open	1
possibly allow for more than one case to open at one to force user to triage cases	1
not clear when an animal was stable and should be transferred elsewhere	2
no normal values for lab work	1

Table 4. Open Ended Survey Questions: Recommended Changes

What could be changed, added, or removed to make the program easier to use?	
Answers:	**Frequency:**
better feedback	1
clear indication that animal has been stabilized or gotten worse	1
reference ranges for blood work or lab tests	3
more diagnostic options such as neurological tests and other radiographs	1
more options on treatment	2
more information available about patients health, mm color, hydration, neurological exams, etc	1
I would like an "associate consult" button to ask for advice on management	1
switch signalment & history on the list due to tendency to ask for signalment first	1
expand physical exam to help user make decisions about next steps	1
find out better time parameters for certain procedures, i.e. Allowing time for catheter placing, blood drawing, etc.	1
a training demo for new users	1
a help screen that describes the diagnostic procedures available, perhaps one that can be dynamically tailored to user's experience level	1

Table 5. Open Ended Survey Questions: Thoughts and Impressions

Please add any other thoughts or impressions about your experience with K9ER.	
Answers:	**Frequency:**
very nice start	1
great idea, helpful in and out of classroom	2
it may have a few problems, but it gets people thinking about what to do in an emergency situation	1
very good tool so far, great tool for future vet students to use	2
hopes to see it implemented soon, possibly as a PDA application	1
great idea, the model tested, while not perfect, provides a great practical exercise in situation management to ingrain the priority and sequence of events necessary to manage medical emergencies	2
there is greater potential available with more testing and input	1
it should be more focused on a specific vet school year like freshman, sophomore, junior, etc.	1
very user friendly	1
I wouldn't have known how to start a case without your instructions	1

The thinking aloud method also provided a very rich data set from which I learned a great deal about the strengths and weakness of VVER. Since our goal in user testing was to learn something new and observe where the user gets stuck, listening to them work through a sample scenario was perhaps the best way to accomplish this. Tables 6, 7, 8, and 9 reflect the feedback given during this process. They are organized in terms of Critical Feedback, Important Feedback, Long Term Considerations, and Considerations if Time Allows.

Table 6. Action Plan from User Testing: Critical Feedback

Issue:	**Status:**
physical exam should include more info, gum coloration, capillary refill rate, palpation, and other tests	done
surgery and its canned text is only applicable to Scooby case	done
reference ranges should be provided for lab test results	done
thoracic radiographs may not be possible offsite	done
present a window for user to confirm procedure after selecting it, display time for procedure, cost, and time to results in this window	done
display units for all dosages, amounts	done
for IV drips, allow setup time before drip begins	done
signalment b/f history b/c it is more intuitive	done
Remove pneumothorax from cases where it is not appropriate	done
add "insert catheter" to procedures, require this to be done b/f fluids are administered	done
Canned text of physical exam must change in response to changes in animal's condition, i.e. After surgery is performed	done
Replace "administer anesthesia" with "begin anesthesia" and "end anesthesia"	open
fix bug that is preventing many treatments from working	done
place animal on ventilator does not work	done
Canned text from hematology assumes that surgery has not been performed, need to make this dynamic or remove it	done
cost of admittance and procedures should vary according to species	done
adv clock feature should tell user how long they must adv the clock to get back specific pending results	done
results can be split when two procedures try to output results at the same time	Open
change simulator running window so user isn't wondering what QCP3 is doing	done
clarify pending treatment times, users were confused as to whether these were times of day, time from now, or time from start of case	done
fix bug with pending treatments so that that they update properly and the are removed from list after results are returned	done
when test was ordered more than once, results from first never showed in status window, results from second did show	open
re-implement evaluate function…it is not currently part of my interface	done

Table 7. Action Plan from User Testing: Important Feedback

Issue:	Status:
neurological testing should be available or part of physical exam	done
allow for palpitation of abdomen, etc.	done
add surgical procedures for treating pneumothorax of lung, such as thoracocentesis	open
physical exam might be a sub-list heading, allowing user to customize exam	done
add additional treatment and diagnostic procedures	done
remove assumptions or keep them constant in the program, ie what the user must explicitly ask for and what K9ER will automatically do	done

Table 8. Action Plan from User Testing: Long Term Considerations

Issue:	Status:
text size is too small, not bold enough	done
picture of animal, diff for each case, at top left of window	discussed
perhaps offer a drop down box in the adv clock window for users to select a pending result and the clock will advance to that time, rather than the user typing in the time	discussed
randomized case loading for more realistic ER experience	-
need for better feedback about patient's condition to user	done
survey data suggests need for clearer instructions about what to do after loading a case or after requesting a procedure	done
make it clearer that a long wait time with QCP3 is normal	done

Table 9. Action Plan from User Testing: Considerations if Time Allows

Issue:	Status:
add chest tap to treatment list	open
program should force user to administer fluids b/f radiographs	done
robust model of dog might be replaced with frail model of dog	-
surgical prep should be separated from surgery	done
add other radiographs such as pelvic radiographs in Scooby case	open
provide more explicit feedback of patient stabilization	done
implement post operative checkup, feedback?	done
add the cost of each procedure in the status window so that users can see how their current cost came to be	done
add other types of anesthesia, such as local for treating Scooby's hemorrhage	open
user would like to have fecal culture procedure	Discussed
user would like to know how the urine is gotten	open
history should include more information such as: eating, bathroom, inside/outside cat, distractions in the house, etc	open
can blood work results can be back in 20 minutes in private practice	done
there is currently no way to treat morris.case, no surgery available for eye	open
implement glucose test?	done
implement a help screen that lists all diagnostic procedures available, perhaps dynamic to the user's experience level	open

Discussion

The average user response to survey statement three indicates they were uncertain, to some degree, what type of information each of the white text boxes would display. This might be remedied with context sensitive help boxes that appear when the mouse is placed over a window and explain its function. This feature is not currently implemented but it should be explored if future work is done.

Average responses to statement five indicate that users may have wondered if the program was functioning properly while waiting for the simulator to generate initial physiological data for the patient. To help alleviate confusion, I changed the message displayed when the simulator runs to indicate the process may take several minutes to complete. Perhaps this will cease to be an issue in the near future as computers become faster.

The rating for statement twelve indicates that users sometimes failed to recognize that a plus sign in the `Action List` window indicates the presence of a sub-menu. The average response to this statement was 3.9 which is only slightly below the 4.0 threshold. Given the many other concerns that were raised during open ended questions and talking aloud sessions, I decided to address this concern at a later time of development. Additionally, this issue might also be resolved with the implementation of the contextual help boxes mentioned earlier.

Users' response to statement nineteen indicates that they were not confident in knowing what they needed to do after loading a case file. In talking with Dr. Brown, we decided that this feeling was desired since students would likely have a similar feeling when admitting their first patient to the emergency room.

The next three statements conveyed a similar sentiment. For example, user responses to statement twenty indicate that the program would have been easier to use if it gave instructions on what options and tests were available to the user immediately after the case was loaded. Again, Dr. Brown and I decided that this feedback was not a problem, as it appeared to model the clinical scenario. Users also felt a list of available options should have been presented after they ordered a procedure, as evidenced by their average response of 2.4 to statement twenty-one. Again, we decided this feeling was a natural response to the clinical situation and did not require a remedy.

Responses to statement twenty-four indicate that users were confused about what to do after ordering a procedure that might have stabilized the patient. In looking at the data collected from the talking aloud process, it appears that some of this confusion resulted from text in the physical examination that did not change with the physiological state of the patient. The problem was exacerbated by diagnostic procedures that did not always include normal ranges. Since testing, the physical exam has been redesigned to be more dynamic and normal ranges were added for each diagnostic test result.

Results from the open ended survey questions and the talking aloud method were most useful in evaluating the current state of the software and determining what steps were necessary to improve its functionality, usability, and medical accuracy. Feedback from these methods was compiled in a spreadsheet and organized based on how necessary it was to address a specific item and how quickly it should be done. These tables served as a schedule for development of the software for the next few months as Dr. Brown and I worked to resolve many of the problems discovered through user testing (Tables 6, 7, 8, & 9).

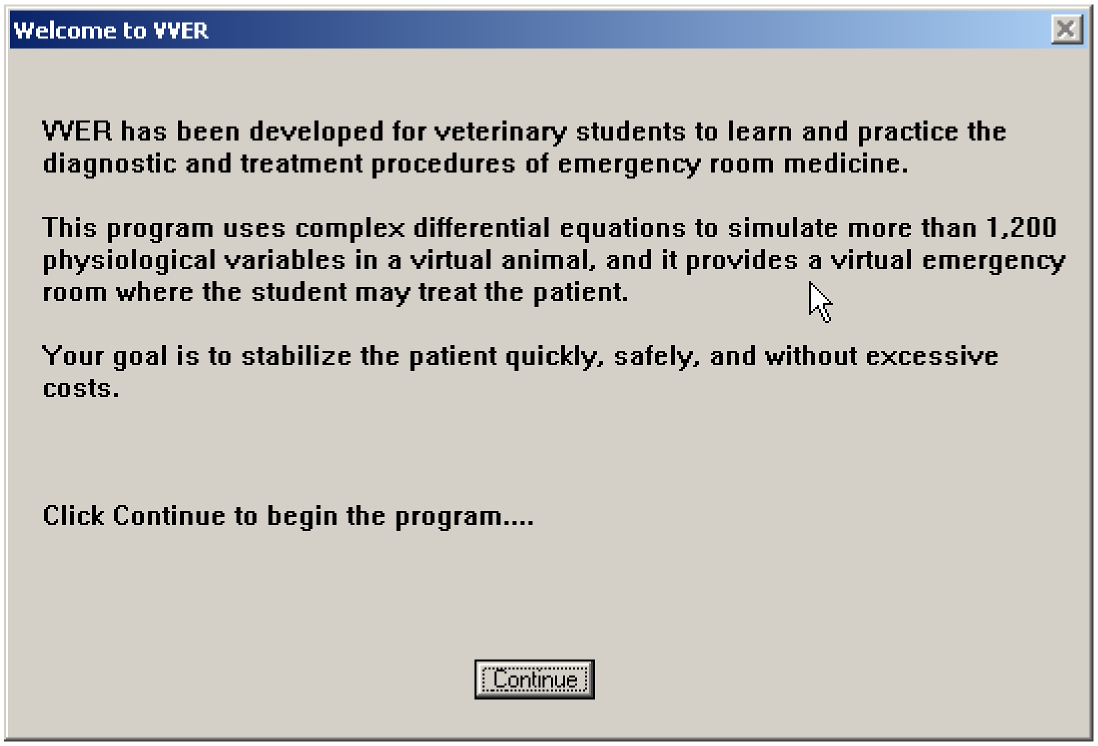
CHAPTER 5

A SAMPLE SCENARIO IN VVER

This chapter recreates a typical user experience in Virtual Vet Emergency Room.

Loading the Program

To start the program, double click on the VVER.exe file. A dialog window will appear (Figure 6), providing an overview of the software. To continue on to the main screen, click the `Continue` button at the bottom of the window.



VVER has been developed for veterinary students to learn and practice the diagnostic and treatment procedures of emergency room medicine.

This program uses complex differential equations to simulate more than 1,200 physiological variables in a virtual animal, and it provides a virtual emergency room where the student may treat the patient.

Your goal is to stabilize the patient quickly, safely, and without excessive costs.

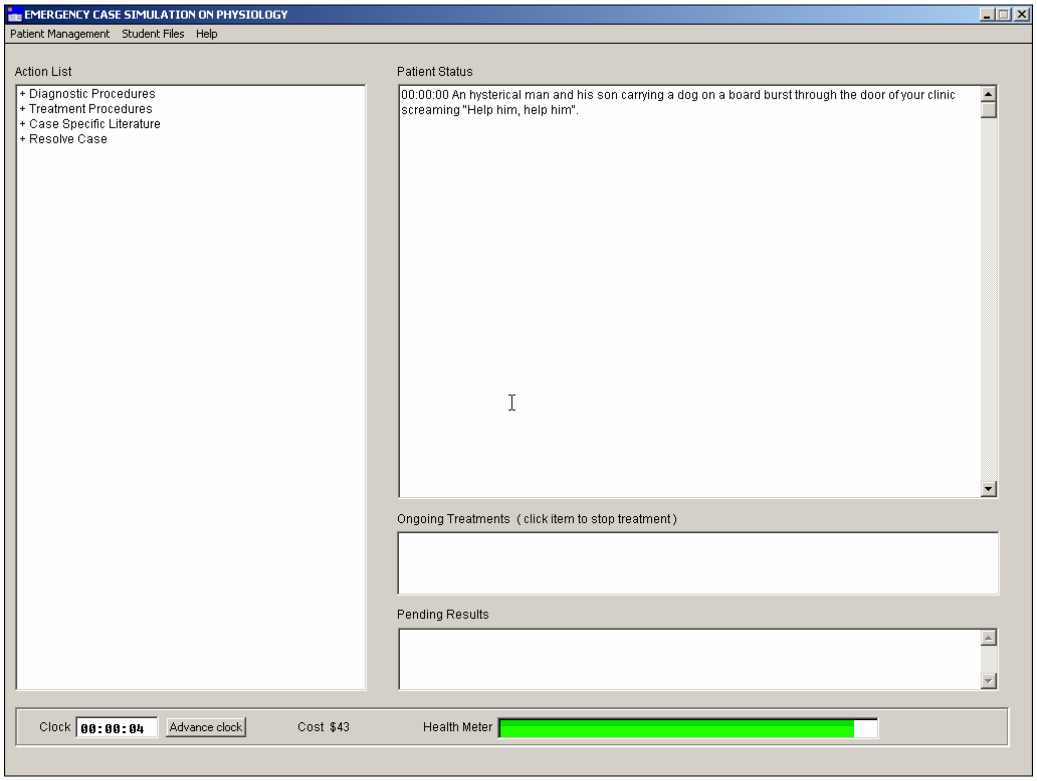
Click Continue to begin the program....

Figure 6. Welcome to VVER Splash Screen

Next, the user interface loads but not the scenario. To load a scenario, click on `Patient Management` in the menu bar and select `Load Case` to choose a case from list of those that appear. For the purposes of this example, the `scooby.case` file has been chosen.

Examining the user interface:

The case begins by calling the simulator to generate the patient's physiological state; this will take a minute or two. When this completes, the user interface screen will be slightly changed. Now the `Patient Status` window contains initial information about the case, the `Health Meter` shows the patient's health, the `Cost` displays `$43.00`, the `Clock` has started counting from `00:00`, and procedures are visible in the `Action List` window as shown in Figure 7.

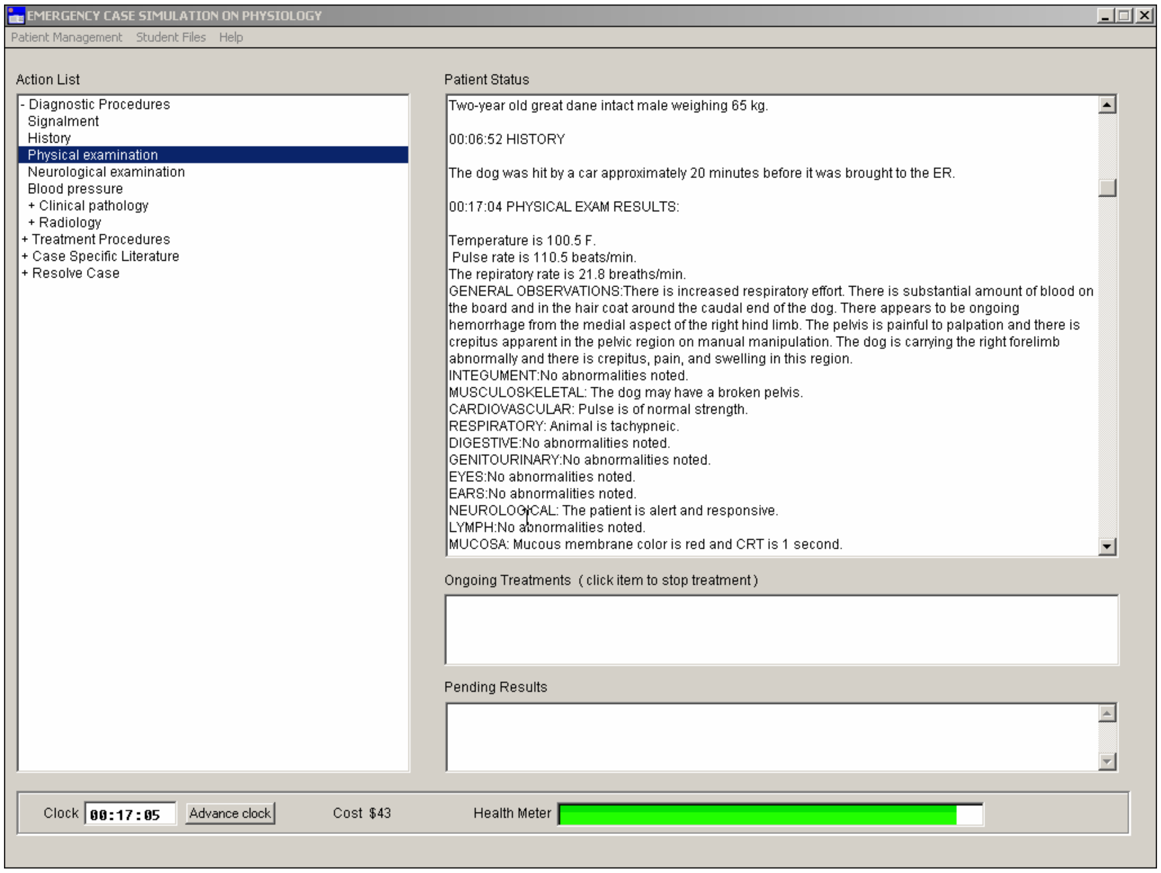


00:00:00 An hysterical man and his son carrying a dog on a board burst through the door of your clinic screaming "Help him, help him".

Figure 7. VVER Interface after Case Has Been Loaded

Diagnosing and Treating the Patient: An Example

Let's find out what's wrong with Scooby by ordering the `Signalment` (the weight and breed of the patient), `History`, and `Physical exam` procedures from the `Action List` window. To do this we click on `Signalment`, and then click on `Ok` in the confirmation window that appears. This window presents the cost and time required to perform the procedure and requires the user to click `Ok` to proceed or `Cancel` to abort. Next we do the same for the `History` and the `Physical exam` procedures. Figure 8 shows results from these procedures displayed in the `Patient Status` window.



Two-year old great dane intact male weighing 65 kg.

00:06:52 HISTORY

The dog was hit by a car approximately 20 minutes before it was brought to the ER.

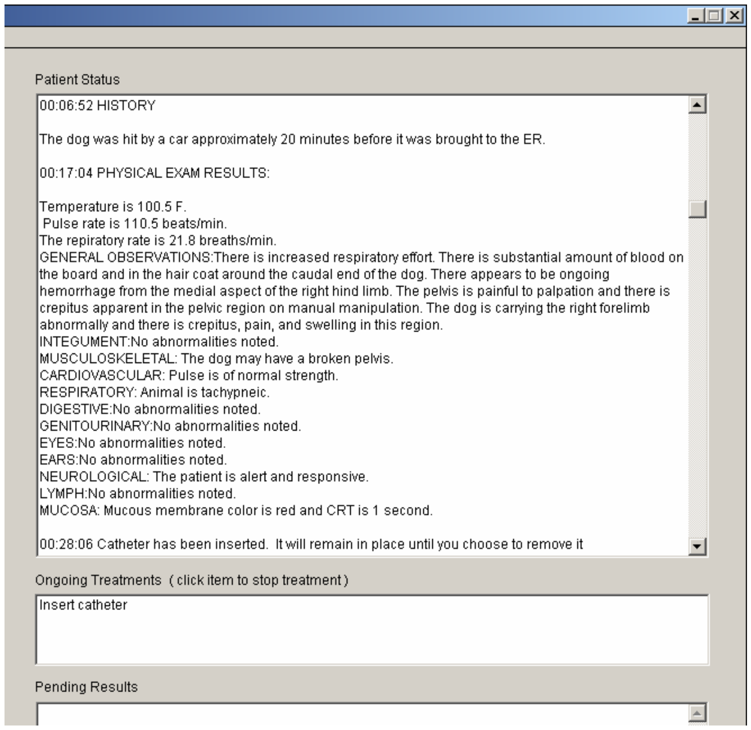
00:17:04 PHYSICAL EXAM RESULTS:

Temperature is 100.5 F.
Pulse rate is 110.5 beats/min.
The repiratory rate is 21.8 breaths/min.
GENERAL OBSERVATIONS:There is increased respiratory effort. There is substantial amount of blood on the board and in the hair coat around the caudal end of the dog. There appears to be ongoing hemorrhage from the medial aspect of the right hind limb. The pelvis is painful to palpation and there is crepitus apparent in the pelvic region on manual manipulation. The dog is carrying the right forelimb abnormally and there is crepitus, pain, and swelling in this region.
INTEGUMENT:No abnormalities noted.
MUSCULOSKELETAL: The dog may have a broken pelvis.
CARDIOVASCULAR: Pulse is of normal strength.
RESPIRATORY: Animal is tachypneic.
DIGESTIVE:No abnormalities noted.
GENITOURINARY:No abnormalities noted.
EYES:No abnormalities noted.
EARS:No abnormalities noted.
NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.

Figure 8. VVER Interface after Ordering Signalment, History, and Physical Exam

Obviously, the history and the physical exam indicate the patient is in poor health. The hemorrhaging is the most immediate health problem, so let's place a catheter, begin a glucose drip, administer anesthesia, and clip the hemorrhage. To place the catheter we must click on the item, `Intravenous Therapies` to expand its submenu, then click on, `Insert Catheter` and confirm our choice to begin the treatment.

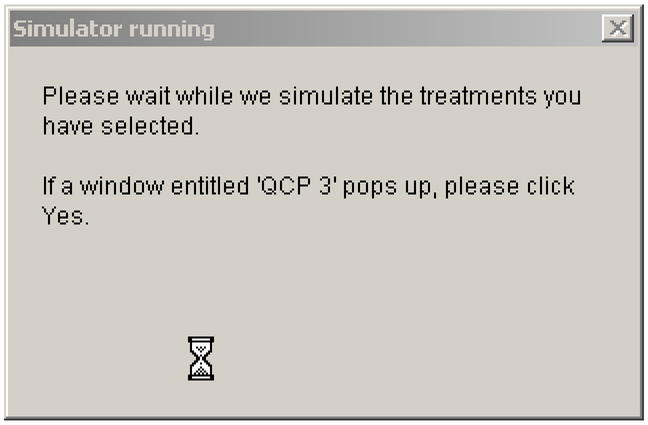
In Figure 9, notice that `Insert Catheter` is now displayed in the `Ongoing Treatment` window. Procedures that do not end immediately are displayed in this window to remind us that they are ongoing. Furthermore, when we choose to remove the catheter, we will do so by clicking on the `Insert Catheter` item in the `Ongoing Treatment` window and selecting `Ok` in the confirmation dialog window that appears.



```
00:06:52 HISTORY

The dog was hit by a car approximately 20 minutes before it was brought to the ER.

00:17:04 PHYSICAL EXAM RESULTS:

Temperature is 100.5 F.
 Pulse rate is 110.5 beats/min.
The repiratory rate is 21.8 breaths/min.
GENERAL OBSERVATIONS:There is increased respiratory effort. There is substantial amount of blood on
the board and in the hair coat around the caudal end of the dog. There appears to be ongoing
hemorrhage from the medial aspect of the right hind limb. The pelvis is painful to palpation and there is
crepitus apparent in the pelvic region on manual manipulation. The dog is carrying the right forelimb
abnormally and there is crepitus, pain, and swelling in this region.
INTEGUMENT:No abnormalities noted.
MUSCULOSKELETAL: The dog may have a broken pelvis.
CARDIOVASCULAR: Pulse is of normal strength.
RESPIRATORY: Animal is tachypneic.
DIGESTIVE:No abnormalities noted.
GENITOURINARY:No abnormalities noted.
EYES:No abnormalities noted.
EARS:No abnormalities noted.
NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.

00:28:06 Catheter has been inserted.  It will remain in place until you choose to remove it
```

Figure 9. Ongoing Treatments Window with Catheter Inserted

With the catheter in place, it is possible to start an intravenous glucose drip. We do this by selecting the item `Administer IV Glucose Drip` from the `Intravenous Therapies` submenu. As usual, we confirm this procedure by choosing `Ok` in the confirmation dialog window that appears. This treatment will affect the physiological state of the patient, so the simulator must run to recalculate the patient's data. While this is happening you will see the window in Figure 10.

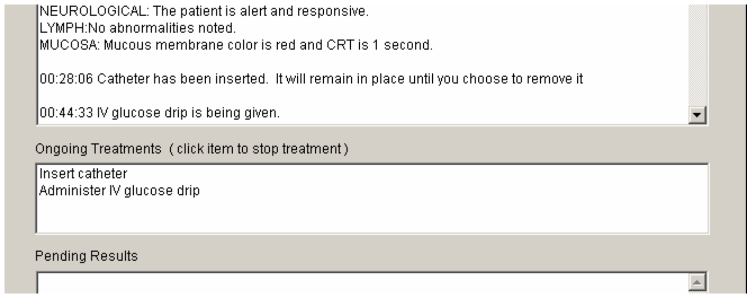

Simulator running

Please wait while we simulate the treatments you have selected.

If a window entitled 'QCP 3' pops up, please click Yes.

Figure 10. Simulator Running Dialog Window

In Figure 11, notice how the `Administer IV Glucose Drip` has been added to the `Ongoing Treatments` window. We could stop it at any time by clicking the item in that window.


NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.

00:28:06 Catheter has been inserted. It will remain in place until you choose to remove it

00:44:33 IV glucose drip is being given.

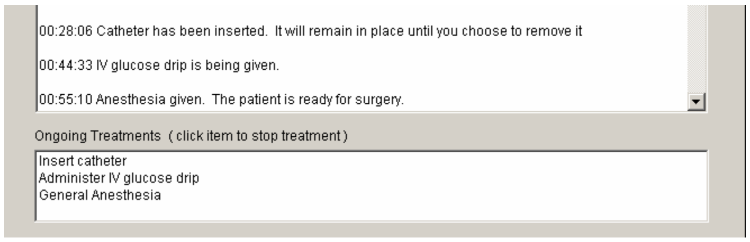
Ongoing Treatments (click item to stop treatment)

Insert catheter
Administer IV glucose drip

Pending Results

Figure 11. Ongoing Treatments Window with IV Administered

Now let's anesthetize the patient and begin surgery. First we select the item, `Drug Therapies`, then its sub-item, `Anesthesia Agents`, followed by the procedure, `General Anesthesia`. Finally we confirm our choice in the confirmation window to proceed. Figure 12 shows that `General Anesthesia` has also been added to the `Ongoing Treatments` window.



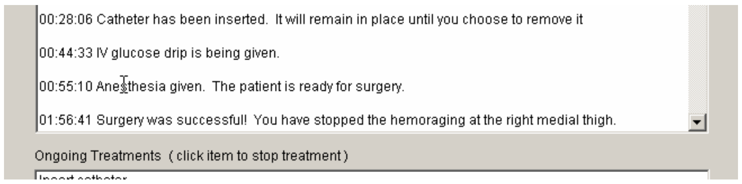
```
00:28:06 Catheter has been inserted.  It will remain in place until you choose to remove it
00:44:33 IV glucose drip is being given.
00:55:10 Anesthesia given.  The patient is ready for surgery.
```

Ongoing Treatments (click item to stop treatment)

```
Insert catheter
Administer IV glucose drip
General Anesthesia
```

Figure 12. Ongoing Treatments Window with Anesthesia Administered

Now the patient is anesthetized and ready for surgery. Let's begin surgery by choosing the item, `Surgical Therapies`, followed by its sub-item, `Perform Surgery`. After selecting this item a dialog window will appear and require that we specify the location and type of surgery to perform. In our case we define the type as `Stop hemorrhage` and the location as `Right medial thigh`, then select `Ok`. Once again we must confirm our selection. A new message in the Patient Status Window (Figure 13) shows that the surgery was a success! We've stopped the hemorrhaging.

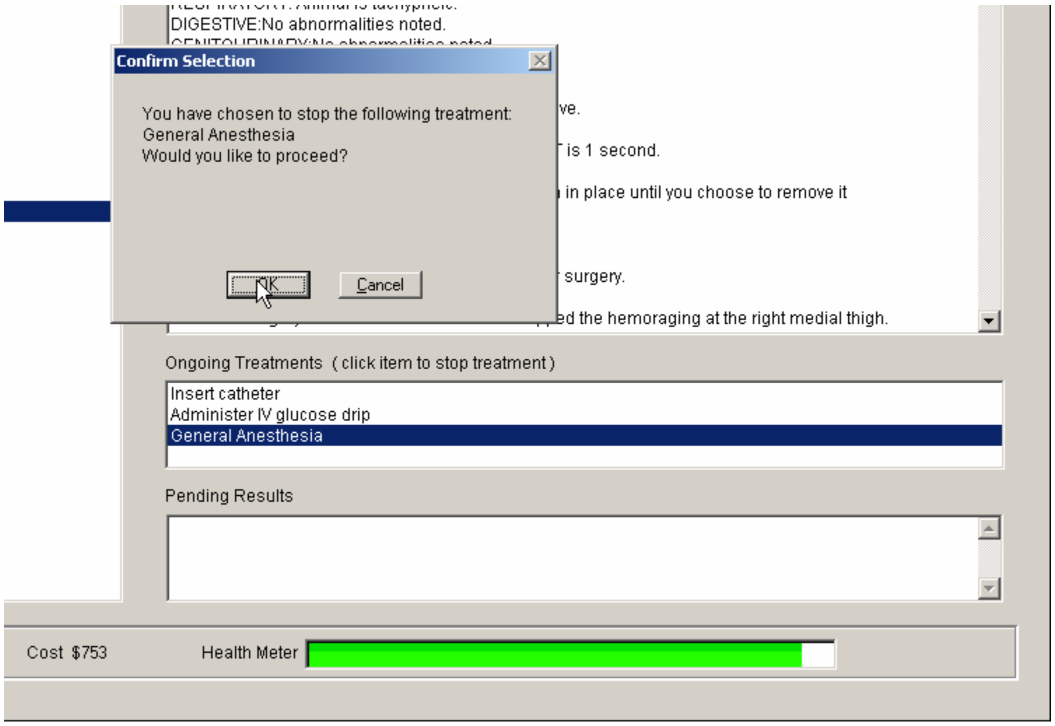


```
00:28:06 Catheter has been inserted.  It will remain in place until you choose to remove it
00:44:33 IV glucose drip is being given.
00:55:10 Anesthesia given.  The patient is ready for surgery.
01:56:41 Surgery was successful!  You have stopped the hemoraging at the right medial thigh.
```

Ongoing Treatments (click item to stop treatment)

Figure 13. Patient Status Window after Performing Surgery

The surgery is complete, but the patient is still under anesthesia. To bring the patient out of anesthesia, we must click on the `General Anesthesia` item in the `Ongoing Treatments` window and confirm our decision by selecting `Ok` (Figure 14).


DIGESTIVE:No abnormalities noted.

Confirm Selection

You have chosen to stop the following treatment:
General Anesthesia
Would you like to proceed?

ve.

is 1 second.

in place until you choose to remove it

surgery.

ed the hemoraging at the right medial thigh.

Ongoing Treatments (click item to stop treatment)

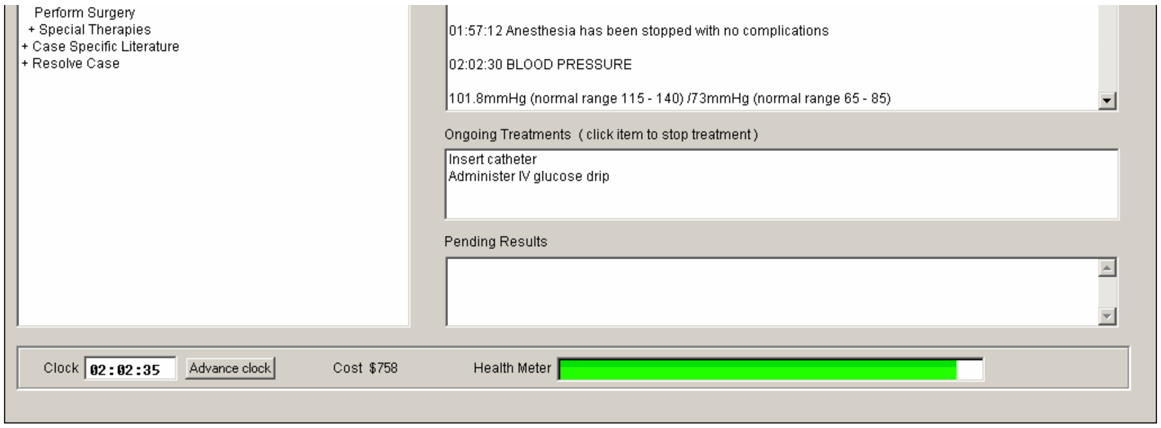
Insert catheter
Administer IV glucose drip
General Anesthesia

Pending Results

Cost $753

Figure 14. Ending General Anesthesia

Hopefully we've stabilized the patient by stopping the hemorrhage. Let's find out by checking the patient's blood pressure. Figure 15 shows the results of this procedure. Notice the cost and time that we have incurred.



```
01:57:12 Anesthesia has been stopped with no complications

02:02:30 BLOOD PRESSURE

101.8mmHg (normal range 115 - 140) /73mmHg (normal range 65 - 85)
```

Ongoing Treatments (click item to stop treatment)

Insert catheter
Administer IV glucose drip

Pending Results

Clock 02:02:35 Advance clock Cost $758 Health Meter

Figure 15. Patient Status Window after Checking Blood Pressure

The blood pressure is low, so let's order a whole blood transfusion to increase blood volume. To do this, we select `Intravenous Therapies` to expand its submenu, and then select `Whole blood transfusion` to begin the procedure as illustrated in the Figure 16.

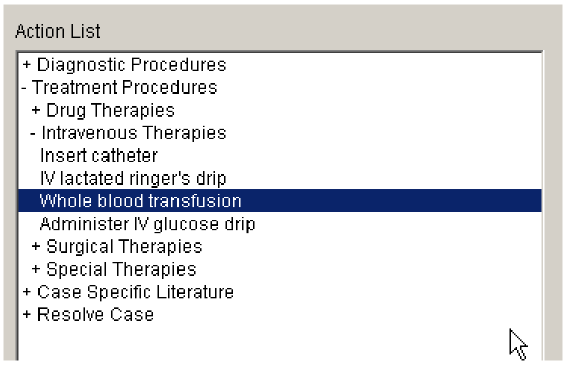
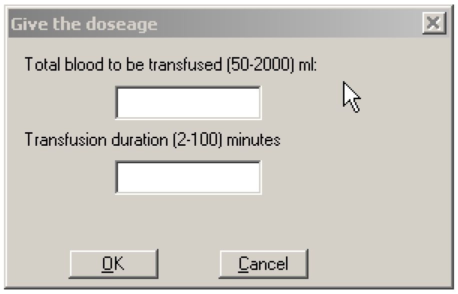


Figure 16. Selecting the Whole Blood Transfusion

After clicking on the item, a dialog window (Figure 17) will ask us for the amount of blood to transfuse and the duration of the procedure.

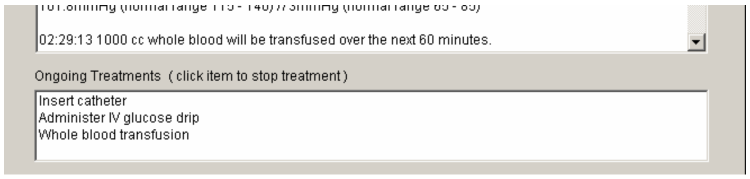

Give the doseage

Total blood to be transfused (50-2000) ml:

Transfusion duration (2-100) minutes

Figure 17. Whole Blood Transfusion Dialog Window

Choose to administer 1,000 ml of blood over 60 minutes. Once this is done and the `Ok` button is selected, the Patient Status window will appear as in Figure 18.


101.8mmHg (normal range 115 - 140)/73mmHg (normal range 65 - 85)

02:29:13 1000 cc whole blood will be transfused over the next 60 minutes.

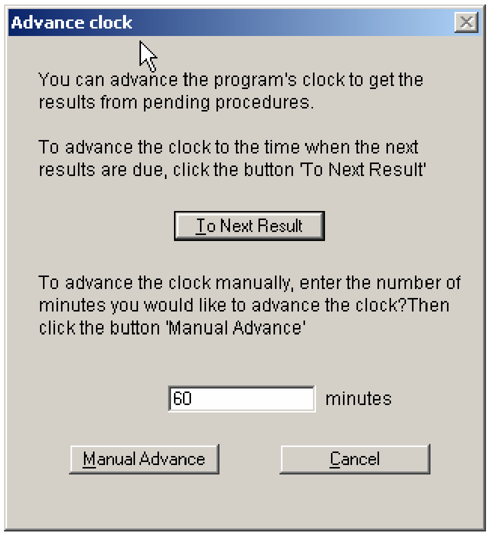
Ongoing Treatments (click item to stop treatment)

Insert catheter
Administer IV glucose drip
Whole blood transfusion

Figure 18. Ongoing Treatments Window with Whole Blood Transfusion

Instead of waiting an hour for the transfusion to finish, let's advance the program's clock to the time we desire. To do this, we click on the button, `Advance Clock`, then enter 60 (minutes) and choose `Manual Advance` in the `Advance Clock` dialog window (Figure 19).

Now, the hemorrhage has been stopped for more than an hour and the whole blood transfusion is complete. We take the blood pressure again to see if it has improved. In Figure 20 we can see that the blood pressure is within the normal ranges.



Advance clock

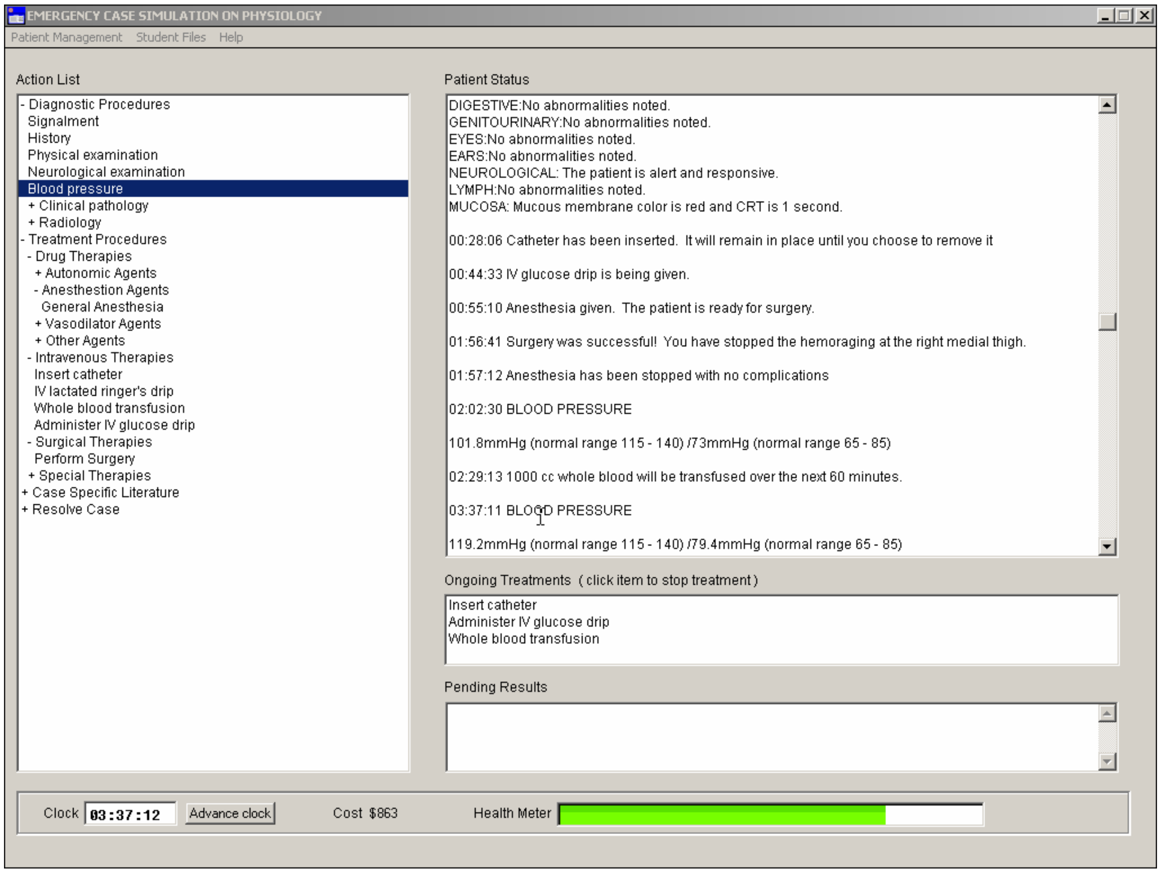
You can advance the program's clock to get the results from pending procedures.

To advance the clock to the time when the next results are due, click the button 'To Next Result'

To advance the clock manually, enter the number of minutes you would like to advance the clock?Then click the button 'Manual Advance'

60 minutes

Figure 19. Advance Clock Dialog Window



- Diagnostic Procedures
  Signalment
  History
  Physical examination
  Neurological examination
  Blood pressure
  + Clinical pathology
  + Radiology
- Treatment Procedures
  - Drug Therapies
    + Autonomic Agents
    - Anesthestion Agents
      General Anesthesia
    + Vasodilator Agents
    + Other Agents
  - Intravenous Therapies
    Insert catheter
    IV lactated ringer's drip
    Whole blood transfusion
    Administer IV glucose drip
  - Surgical Therapies
    Perform Surgery
  + Special Therapies
+ Case Specific Literature
+ Resolve Case

Patient Status

```
DIGESTIVE:No abnormalities noted.
GENITOURINARY:No abnormalities noted.
EYES:No abnormalities noted.
EARS:No abnormalities noted.
NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.

00:28:06 Catheter has been inserted.  It will remain in place until you choose to remove it

00:44:33 IV glucose drip is being given.

00:55:10 Anesthesia given.  The patient is ready for surgery.

01:56:41 Surgery was successful!  You have stopped the hemoraging at the right medial thigh.

01:57:12 Anesthesia has been stopped with no complications

02:02:30 BLOOD PRESSURE

101.8mmHg (normal range 115 - 140) /73mmHg (normal range 65 - 85)

02:29:13 1000 cc whole blood will be transfused over the next 60 minutes.

03:37:11 BLOOD PRESSURE

119.2mmHg (normal range 115 - 140) /79.4mmHg (normal range 65 - 85)
```

Ongoing Treatments (click item to stop treatment)

Insert catheter
Administer IV glucose drip
Whole blood transfusion

Pending Results

Clock 03:37:12 Advance clock Cost $863 Health Meter

Figure 20. VVER Interface after Stabilizing Patient

Resolving the Case

The patient's vital signs are stabilizing so it is a good time to transfer the dog to the ICU for recovery. To do this, we select the item `Resolve Case` from the `Action List` to expand its sub-items, then click on `Transfer to the ICU` and confirm our selection by clicking `Ok` in the confirmation window. The image in Figure 21 illustrates this process.

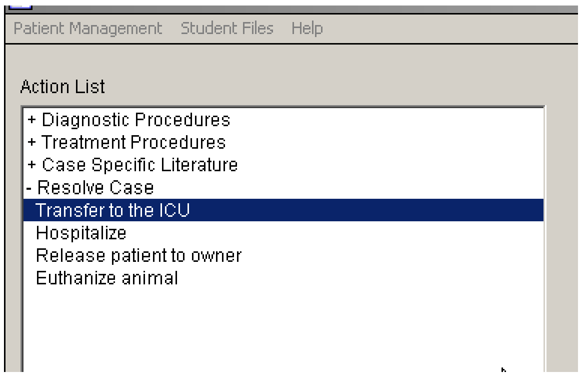
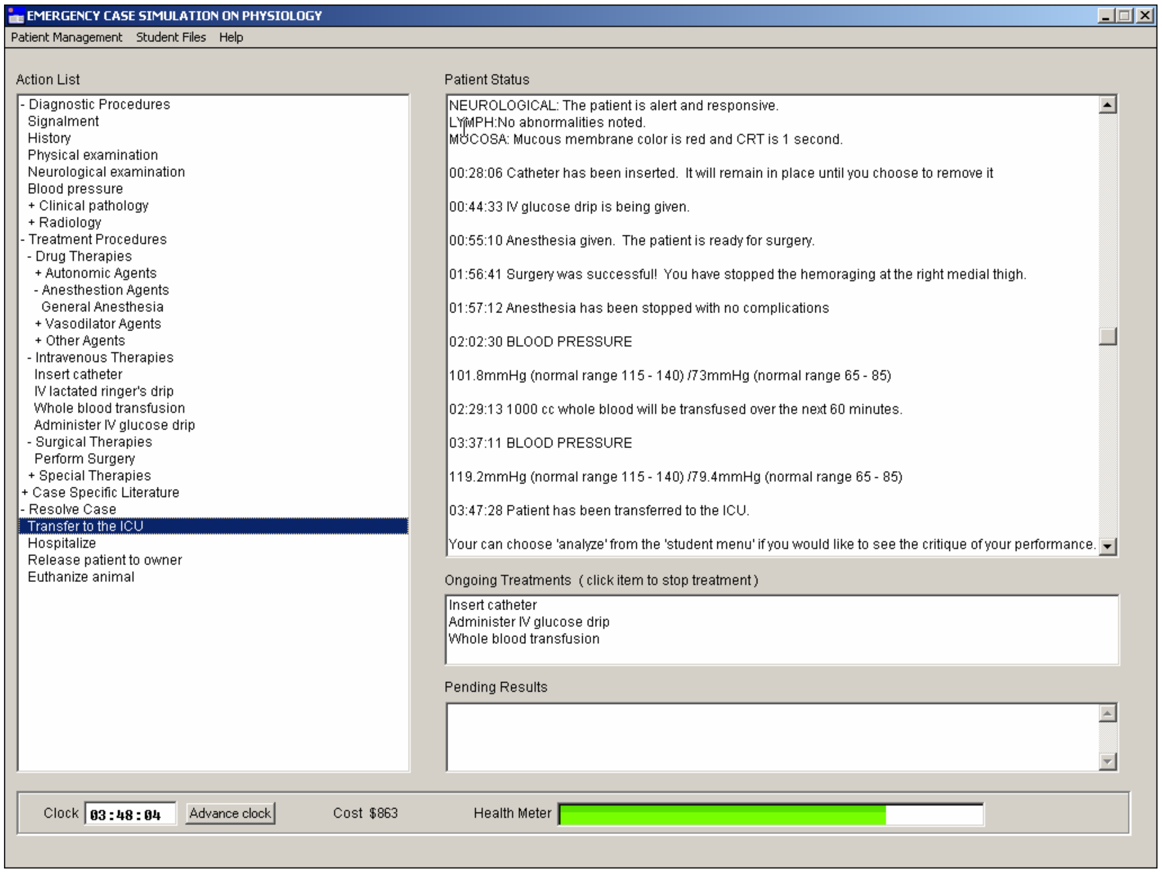


Figure 21. Selecting Transfer to the ICU

After selecting this procedure, we will be prompted with a dialog window that asks us to save our work in a report file. Click `Ok`, then enter a file name and click `Save`. This will place the text from the `Patient Status` window into a student report file.

As in Figure 21, the `Patient Status` window now indicates the patient has been transferred, and all procedures in the `Action List` window are inoperable. At this point the case has been resolved.



- Diagnostic Procedures
Signalment
History
Physical examination
Neurological examination
Blood pressure
+ Clinical pathology
+ Radiology
- Treatment Procedures
- Drug Therapies
+ Autonomic Agents
- Anesthestion Agents
General Anesthesia
+ Vasodilator Agents
+ Other Agents
- Intravenous Therapies
Insert catheter
IV lactated ringer's drip
Whole blood transfusion
Administer IV glucose drip
- Surgical Therapies
Perform Surgery
+ Special Therapies
+ Case Specific Literature
- Resolve Case
Transfer to the ICU
Hospitalize
Release patient to owner
Euthanize animal

```
NEUROLOGICAL: The patient is alert and responsive.
LYMPH:No abnormalities noted.
MUCOSA: Mucous membrane color is red and CRT is 1 second.

00:28:06 Catheter has been inserted.  It will remain in place until you choose to remove it

00:44:33 IV glucose drip is being given.

00:55:10 Anesthesia given.  The patient is ready for surgery.

01:56:41 Surgery was successful!  You have stopped the hemoraging at the right medial thigh.

01:57:12 Anesthesia has been stopped with no complications

02:02:30 BLOOD PRESSURE

101.8mmHg (normal range 115 - 140) /73mmHg (normal range 65 - 85)

02:29:13 1000 cc whole blood will be transfused over the next 60 minutes.

03:37:11 BLOOD PRESSURE

119.2mmHg (normal range 115 - 140) /79.4mmHg (normal range 65 - 85)

03:47:28 Patient has been transferred to the ICU.

Your can choose 'analyze' from the 'student menu' if you would like to see the critique of your performance.
```

Insert catheter
Administer IV glucose drip
Whole blood transfusion

Figure 22. VVER Interface after Resolving Case

You can also save the text from the `Patient Status` window at anytime by selecting the `Student Files` item from the menu bar at the top of the screen and selecting `Save Report`. When prompted, enter a name for your report and choose `Save`.

You may start another case, restart the same case, or exit Virtual Vet Emergency Room by selecting the appropriate item in the `Patient Management` submenu.

CHAPTER 6

FUTURE WORK

VVER is a work in progress. Additional features and further testing can only serve to improve the program.

Testing the Software

A program of alpha and beta testing with first through fourth year veterinary students should be conducted and faculty members should be asked to evaluate the software. The evaluation should initially center on the usability of the program and the accuracy of the data presented. Then it should expand to evaluate the degree to which the program enhances the educational experience of veterinary students, since this is after all the true goal of the project. Surveys should be developed and used to gather feedback in these areas.

It is my understanding that this testing will begin during the fall of 2004 in the Veterinary College at the University of Georgia.

Expanding and Refining the Knowledge Base

Additional medical scenarios should be created for VVER, since there are only a few scenarios currently available. Professors should work with software developers to design and implement additional scenarios since this would greatly increase the usefulness of the software. I envision a centralized database of medical scenarios from which professors at any university might download and incorporate them into their curricula.

In developing these scenarios, it may be necessary to have diagnostic and treatment procedures that are not currently supported by VVER. Therefore, future work should also focus on expanding the knowledge base to incorporate a broader range of diagnostic and treatment procedures. Similarly, the development of additional medical scenarios may reveal gaps in the critic's performance that are not noticeable in current scenarios. If this occurs, then future work should also focus on refining and expanding the critic rules to provide consistent performance across all scenarios.

Accessing the Software

VVER is coded in WIN-Prolog and is a stand alone program. This means that it must be installed and run on a client computer using the Windows Operating System. When updates to VVER are available, they will need to be downloaded and installed on every computer running the software. Similarly, as new scenarios become available, they will need to be downloaded and placed in the appropriate folder. Student reports will also be limited in their usefulness because they are stored on a single computer or a networked drive. If the professor or student is not at that computer, on the network, or with remote access to those drives, they will not be able to view the report. I believe these are barriers to the rapid and widespread adoption of the software in veterinary colleges.

An internet based version of VVER, coded in Java, could remove all of these barriers to adoption. Such an implementation would always present the user with the newest version of the software, eliminating the need to check for, download, and install updates. Likewise, a central database would always provide the user with the most current medical scenarios, ensuring consistency of experience across all users. Student reports could also be stored on a central server and accessible from any location via a secure web page.

A caveat to this approach is that the simulator, QCP3, is computationally expensive and it might prove difficult to run many instances of this program on an internet server. A remedy would be to have enough computational resources so that QCP3 can run without problem. This would be a straight forward solution, but probably an expensive one that may not prove scalable. Another option may be to alter QCP3 or use a different physiological simulator such that it would be able to run on the client's computer, perhaps in the form of a java applet. This would be difficult to implement, but it would be a more complete solution.

Redesigning VVER to function over the internet is no simple task; however, I believe it crucial for the widespread adoption of the software at veterinary colleges.

BIBLIOGRAPHY

Arlov, Laura (1997) *GUI Design for dummies*, IDG Books Worldwide.

Barnum, Carol M. (2002) *Usability Testing and Research*, New York: Longman.

Brown, Scott (2003) Funding proposal: Application For The International Foundation for Ethical Research Graduate Fellowship in Alternatives in Science Research. unpublished.

Coleman T., (1999) *QCP DESolver: Solver Control Manual.* Biological Simulators, Inc..

Covington, Michael A.; Nute, Donald; Vellino, André. (1988) *Prolog programming in depth*, Glenview, Ill.

Gordon, James A.; Oriol, Nancy E.; Cooper, Jeffrey B. (2004) Bringing Good Teaching Cases "To Life": A Simulator-Based Medical Education Service, *Academic Medicine*, 79.1:23-27.

ICT Games Project Home Page. Institute for Creative Technologies, University of Southern California. Retrieved July 12, 2004 from http://www.ict.usc.edu/disp.php?bd=proj_games

Logic Programming Associates (2003) WinProlog. Logic Programming Associates, Ltd., London, England.

Mackenzie, Jonathan.; Baily, Gavin; Nitsche, Michael; and Rashbass, Jem. 'Gaming Technologies for Anatomy Education', (unpublished conference presentation) 7th International Conference on Information Visualisation IV'03 16-18 July 2003, London.

Medical Technology Working Group. Summary and recommendations of the medical technology breakout session. In: Summary Report of the November 29, 1999 Dean’s Educational Workshop: Medical Education in the New Millennium. Boston: Harvard Medical School, 2000.

Nielsen, Jakob (1994) *Usability Engineering*, Morgan Kaufmann.

Saccuzzo, Dennis P., Kaplan, Robert M. (1996) *Psychological Testing*, Brooks/Cole Publishing Company.

Swartout, W., Van Lent, M. (2003) Making a Game of System Design. *Communications of the ACM* (July 2003), 32-39.

Usability by Design Ltd. (2004) www.usability.uk.com Definition: *likert*. Retrieved July 21, 2004 from www.usability.uk.com/glossary/likert-scale.htm.

Xu, Yunxiu. (2002) K9er (canine emergency room): a veterinary practice simulator based on the integration of an expert system and a physiological simulation. *Master’s Thesis, The University of Georgia*, Athens, GA.

APPENDIX A

HOW TO CREATE A NEW SCENARIO

This document explains how to create scenarios for Virtual Veterinary Emergency Room.

Creating a Case File

The case file is where the details for each scenario are stored. Several case files are included with the software, but new cases can be created by modifying a template file.

To create a new case file, open the template file, `template.case`, in any text editor and save it under a different name with the .case extension. For example, you might chose to save the file as `myFirstCase.case`.

The template file contains documentation and code for each parameter that must be specified in the case file. The sections in this document explain how to modify that code to meet your needs. For many of the parameters, the existing code will be sufficient but there will be a few cases where you will need to define additional rules. This will be covered more in later sections.

The first few lines in the template file are comments that specify the name of the case file, the author, the date it was created, and the last time it was changed. Comments are lines of text that contain a percent sign in front of them. They are not processed by VVER; they are notes to the programmer or other humans who use the file.

In the template, these comments appear as follows:

```
% template.case

%
% AUTHOR(S): John/Jane Doe
% DATE CREATED: --/--/--
% LAST UPDATED: --/--/--
%
% DESCRIPTION OF SCENARIO:
% This is where a short description of the scenario might go...
%
```

Change them in your file to reflect your case by replacing the generic information with your own.

Specifying the Species

The next parameter in the case file specifies the species of the patient. In the template file you should see three lines of commented text like this:

```
% species(dog).
% species(cat).
% species(horse).
```

The above are called facts. In Prolog we refer to this group of facts as `species/1` because the name, or functor, of the facts is 'species' and it contains `1` argument. This notation will be used for the remainder of this document.

To specify the species of the patient in your scenario, uncomment the appropriate fact by removing the percent sign that is in front of it. For example, if our scenario involved a dog, then we would specify the species as follows:

```
species(dog).
% species(cat).
% species(horse).
```

Specifying the ICS file

The simulator used in VVER is called QCP3 and it maintains over 1200 physiological variables that are related through complex differential equations. When the simulator is loaded, it must have initial values for each of these variables.

These are provided by ICS files which are specified in the case file. There are several ICS files that come with QCP3 and each represents a different physiological state from which the simulation begins. Choose the one that is most similar to your medical scenario. It is acceptable if the ICS file does not exactly model the medical scenario you envision because there is other code in the case file that will allow you to alter parameters when the simulator begins.

Below are the names of two default ICS files and the description of the physiological state that will result from using this file with the simulator in VVER:

```
MR_HOBBS.ICS
"Mr. Hobbs was in a marine accident and was lost at sea for 3
days with no food and water."

MR_DIXON.ICS
"Mr. Dixon is diagnosed as primary aldosteronism."
```

After you've chosen this file, set the parameter for `simulation_data_file/1` to the file's name. For example, if we are going to use `mr_norm.ics` in our case, the case file will contain the following:

```
simulation_data_file(`mr_norm.ics`).
```

Specifying the Granularity of Simulation Data

The parameter for `simulation_duration/1` specifies the block of time for which the simulator will generate data. This is necessary because the simulator is not running in real time during the scenario. Instead, it generates physiology data for a specified amount of time and outputs it to a file (in this case it generates 20 minutes worth of physiology data). When a treatment procedure is ordered or when data runs out, the simulator will again generate the specified amount of data and write it to file. Increasing this value significantly will cause the simulator to run longer when generating simulation data in VVER.

The parameter for `intervalG/1` specifies the granularity of the simulation data. In the template, the value is `1.0` so the physiology data is captured once each minute. A value of `0.5` would capture data twice a minute and a value of `2.0` would capture data every two minutes. The model has only been tested with a value of `1.0` so it should not be changed this without good reason.

The following lines of code, found in the template file, corresponds to the features discussed above.

```
simulation_duration(20.0).  % in minutes
intervalG(1.0).              % in minutes
```

Specify Changes to the Simulator

The next option is `parameter_change/2`. This specifies how the physiology of the patient should differ from the initial parameters set by the ICS file. It is necessary to specify parameter changes for the case to model physiological disturbances other than those included in the default ICS files. There are two types of parameter changes: Those which are facts, and those which are rules. The first type, those which are facts, are of the following general form:

```
parameter_change(+Param,+Value)
```

`+Param` is the name of a QCP3 parameter and `+Value` is the value that the parameter should have. Here's an example fact that was taken from scooby.case:

```
parameter_change(`"Arterial Hemorrhage, Switch"`,1).
```

In this example, ``` `"Arterial Hemorrhage, Switch"` ``` is the parameter name and `1` is the parameter's value. This line instructs QCP3 to create an Arterial hemorrhage (i.e. it turns on the hemorrhage switch) immediately after loading the ICS file. All `parameter_change/2` facts work this way, they modify the QCP3 parameter values immediately after loading the file.

In contrast to `parameter_change/2` facts, there are also `parameter_change/2` rules. Rules only change the QCP3 parameters when certain preconditions are met. Therefore a rule based parameter change consists of the QCP3 parameter name, the new parameter value, and preconditions that must be met before the parameter should be changed. These preconditions are conjunctions of one or more logical statements and are of the following general form:

```
parameter_change(+Param,+Value):-
    some_precondition(+SomeValue),
    another_precondition(+AnotherValue).
```

The following is an example of a rule based parameter change from `scooby.case`:

```
parameter_change(`"Arterial Hemorrhage, Duration"`,Duration) :-
        (
        \+ hem_stop(_)
        ;
        hem_stop(0)
        ;
        hem_stop(Time),
        simulation_duration(Dur),
        Time + Dur >= 1000
        ),
        Duration = 1000.
```

The first line contains the QCP3 parameter, `` `"Arterial Hemorrhage, Final Volume"` ``, and its value, represented as the variable `Volume` (Volume is later given the value of `10000` in the last line). The remaining lines are the preconditions that must be met before `Volume` will be given a value and the command will affect the simulator.

Creating these preconditions (rules) requires a working knowledge of WIN-Prolog (specifically the LPA 4300 implementation) and some understanding of how VVER works internally. If you do not know Prolog, you should read a couple of chapters from any introductory Prolog book. I would recommend the book that I used to learn Prolog, “Prolog Programming in Depth” (Covington, Nute, and Vellino, 1988). To understand the internal workings of VVER, you should read the comments in the program’s code.

It's important to test rule based parameter changes to be sure they behave as expected. Often in computer programming, several iterations of a program are needed to produce the desired behavior.

Specifying the Data Returned by the Simulator

The case file must specify which parameter values should be returned by the simulator. These parameters are specified in `simulation_outputs/1`. The parameters in the default list are necessary for the normal functioning of VVER; however it is possible to add QCP3 parameter names to this list. It may be necessary to output additional parameter values if you have written code that relies on this information.

Configuring the Health Meter Display

The `show_health_meter/0` fact toggles the health meter display on/off. If you want the user to see a health meter during the program, leave this as is. If you do not want them to see a health meter, comment this out by placing a percent sign in front of the text. For example:

```
% Here the health meter is OFF
% show_health_meter.

% Here the health meter is ON
show_health_meter
```

If the health meter display is enabled, then the `health_skew/1` option may be used. It allows for a biasing of the health meter display so that the patient will appear either more or less healthy than he/she actually is. The default value of `0.0` for `health_skew/1` indicates no bias, positive numbers create a positive bias, and negative numbers create a negative bias. How does this work? The health of the patient is represented internally as a value between `0.0` and `1.0` and the bias works by adding a specified value to the patient's health. For example, if the health meter measured the patient's health as `0.5` and there was a `0.1` health skew, the health would be displayed on the screen as though the patient's health was `0.6`.

Specify Which Procedures are Available to the User

Next in the template file, you will find a list of `on_procedure/1` facts. If you want the user to have access to a specific procedure, leave the line that corresponds to that procedure as it appears in the template. If you want to block access, comment out the line by placing a percent sign in front of the text.

For example, the fact below tells Virtual Vet ER to allow the user to query the patient's history.

```
on_procedure(`History`).
```

If the user should not be able to ask for the patient's history, the statement should look like this:

```
% on_procedure(`History`).
```

Defining the Case Specific Text

VVER can not generate text to describe the patient's signalment. It must be read into the system from the case file for each medical scenario. The same is true for the initial message displayed to the user when the case is loaded, the patient's history, and the results of thoracic radiographs.

This text is specified in the case file by `case/2` facts. These are facts that associate specific text messages with qualitative features of the case in the following format:

```
case(+Feature,+CannedText)
```

The first parameter, `+Feature`, specifies what qualitative feature is being described. The second parameter, `+CannedText`, is the text that describes the feature.

Below is the default implementation for `case/2` facts, you must replace the following text with your case specific text.

```
case(start,
    [`This is where the text goes that displays `,
    `to the status window immediately after the case `,
    `is loaded in to Virtual Vet ER. You can add as `,
    `many of these lines of text as you would like, `,
    `just be sure to follow this format of the text `,
    `being contained within backquotes and a comma `,
    `following each line except the last.`]).

case(signalment,
    [`This is where you place the text for age, `,
    `breed, and weight of the dog.`]).

case(history,
    [`Write what you want the user to see when `,
    `they request the patient's history,`,
    `for example, "The dog was hit by a car `,
    `approximately 20 minutes ago!".`]).

case(thoracic,
    [`Thoracic radiographs appear normal.`]).
```

For example, `scooby.case` has the following facts:

```
case(start,
    [`An hysterical man and his son carrying `,
    `a dog on a board burst through the door`,
    `of your clinic screaming "Help him, help him".`]).

case(signalment,
    [`Two-year old great Dane intact male weighing 65 kg.`]).

case(history,
    [`The dog was hit by a car approximately 20 `,
    `minutes before it was brought to the ER.`]).
```

```
case(thoracic,
    [`Thoracic radiographs appear normal.`]).
```

The physical and neurological exam also contains qualitative features that require canned text. These are specified by `case/3` in the following format:

```
case(+ExamType,+Feature,-CannedText)
```

The first parameter is the type of exam (either `physical_exam` or `neuro_exam`), the second parameter is the exam feature (i.e. `general`, `integument`, `musculoskeletal`, `digestive`, etc.), and the third parameter is text that describes the feature.

In the template case file, there are templates for the `neuro_exam` and `physical_exam` features. The default text for every feature is `"No abnormalities noted"`. To change the text for these features, just replace this with the appropriate text, surrounded by back-quotes (the symbol to the left of the 1 key). For example, the following code specifies the text for the musculoskeletal aspect of the physical exam in the case, `scooby.case`:

```
case(physical_exam,musculoskeletal,[`MUSCULOSKELETAL: `,
    `The dog may have a broken pelvis.`]).
```

So far we have specified facts, or statements that provide data to the program. It is also possible, and perhaps necessary, to specify rules for some features. Once again, rules are statements that are only true if certain preconditions are met. In the Scooby case, once the user has treated the hemorrhage by giving the dog general anesthesia and clipping the source of the hemorrhage, the text returned for `GENERAL OBSERVATIONS` must change. The simulator can not generate this new text automatically, so the text must be contained in the case file. This is done by devising two rules, one that is true when the hemorrhage is ongoing and one that is true when the hemorrhage has been stopped. Therefore, the first rule has the precondition that the hemorrhage must be ongoing, while the second has the precondition that the hemorrhage must be stopped.

When the `scooby.case` file is initially loaded, the dog is hemorrhaging and the first rule will be satisfied. The following code shows this rule and the text that will be written to the Patient Status window when the rule executes.

```
% if hemorrhage has not been stopped
case(physical_exam,general,[`GENERAL OBSERVATIONS:`,
          `There is increased respiratory effort. `,
          `There is substantial amount of blood on `,
          `the board and in the hair coat around the `,
          `caudal end of the dog. There appears to be `,
          `ongoing hemorrhage from the medial aspect `,
          `of the right hind limb. The pelvis is painful `,
          `to palpation and there is crepitus apparent `,
          `in the pelvic region on manual manipulation. `,
          `The dog is carrying the right forelimb `,
          `abnormally and there is crepitus, pain, and `,
          `swelling in this region.`]):-
    \+ hem_stop(_).
```

```
"There is increased respiratory effort. There is substantial
amount of blood on the board and in the hair coat around the
caudal end of the dog. There appears to be ongoing hemorrhage
from the medial aspect of the right hind limb. The pelvis is
painful to palpation and there is crepitus apparent in the
pelvic region on manual manipulation. The dog is carrying the
right forelimb abnormally and there is crepitus, pain, and
swelling in this region."
```

After the user stops the hemorrhage, the second rule will be satisfied, but not the first. The second rule and the text from this rule are displayed:

```
% if hemorrhage has been stopped
case(physical_exam,general,[`GENERAL OBSERVATIONS:`,
           `There is blood on the board but the bleeding `,
           `has stopped. The pelvis is painful `,
           `to palpation and there is crepitus apparent `,
           `in the pelvic region on manual manipulation. `,
           `The dog is carrying the right forelimb `,
           `abnormally and there is crepitus, pain, and `,
           `swelling in this region.`]).
```

```
"There is blood on the board but the bleeding has stopped. The
pelvis is painful to palpation and there is crepitus apparent in
the pelvic region on manual manipulation. The dog is carrying
the right forelimb abnormally and there is crepitus, pain, and
swelling in this region."
```

Any feature of the neurological or physical exam must be specified in the case file (those that are listed in the template file) and may be specified as a single fact or as two or more rules. Remember that rules are tried in the order they appear, so if multiple rules are satisfied for a given feature, then only the first one (from top of the file to the bottom) will be satisfied. Look through the other included case files to see more examples of rules. Each rule will have a comment that explains when it is true (i.e. used by the software).

Specifying the Surgical Problem

If surgery is needed to stabilize the patient, then it must be specified in the case file. In the template file there is a fact called `surgical_problem/1` which specifies the type of surgery and the location at which it must be performed to improve the patient's condition.

`surgical_problem/1` is of the following form:

```
surgical_problem(+Type,+Location)
```

`+Type` is a string that describes the type of surgery needed and `+Location` is a string that describes the location of the surgery. Included in the template file is the following example:

```
% surgical_problem(`Stop hemorrhage`,`Right medial thigh`).
```

If a surgical procedure is needed in your scenario, replace the sample text with your own (these may be any short descriptive strings) and uncomment the line by removing the percent sign (%) from the front.

When the user chooses to perform surgery, he or she will be presented with a drop down box for each option and will have to choose the type and location for the surgery. If they choose the correct values for both, then the surgery is performed, the patient's physiology is altered accordingly, and text is returned to the Patient Status window.

NOTE: The system will know how to change the physiology of the patient after the surgery because this will be specified in the parameter change rules. There should be at least one or more parameter change rules for each type of surgical procedure that is required in the scenario.

If they choose the wrong type or location for the surgery, then the system will not perform the surgery and the user will be notified that they selected an improper procedure. In this case, their actions will be written to the report file even though the procedure will not execute.

Specifying the Text Message to Follow Surgery

If you have not defined a surgical problem for your case, you do not need to read this section. As mentioned above, the system returns text to the Patient Status window after the surgery is performed successfully. This text is specified in the case file by `surgical_message/3` which takes the following form:

```
surgical_problem(+Type,+Location,+Message)
```

The first two parameters will be the same as those specified in `surgical_problem/2`. This allows the system to match this fact with the appropriate surgical problem. The third parameter is the text string that will be displayed in the Patient Status window. The following example is taken from the template file.

```
% surgical_problem(`Stop hemorrhage`,`Right medial thigh`,
%           [`Surgery was successful!`]).
```

Once again, replace the sample text with your own and uncomment the fact by removing the percent sign from the front.

APPENDIX B

CASE FILE TEMPLATE

```
% template.case
%
% AUTHOR(S): John/Jane Doe
% DATE CREATED: --/--/--
% LAST UPDATED: --/--/--
%
% DESCRIPTION OF SCENARIO:
% this is where a short description of the scenario might go...
%


:- multifile[continue_time/1,current_cost/1].
:- dynamic[continue_time/1,hem_stop/1,parameter_change/2,
surgical_problem/2].


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_data_file/1
%
% simulation_data_file(+File)
%
%     +File       <string>    name of QCP3 ICS file to start simulation
%
% simulator start parameters
%

simulation_data_file(`mr_norm.ics`).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% species/1
%
% species(+Type)
%
%     +Type   <atom>    species of patient (ie. dog, cat, horse, cow)
%

species(dog).
% species(cat).
% species(horse).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_duration/1
```

```
%
% simulation_duration(+Time)
%
%     +Time       <number>    # of minutes for which simulator will run
%
% 20.0 minutes in default.  Higher time
% intervals will lead to longer simulation time
%


simulation_duration(20.0).
intervalG(1.0).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parameter_change/2
%
% parameter_change(+Param,+Value)
%
%     +Param            <string>    name of parameter to be altered
%     +Value            <number>    new value of parameter
%
% Changes from the initial parameter values that are loaded into the program
from the ICS file
%

parameter_change(`"Arterial Hemorrhage, Switch"`,1).

parameter_change(`"Arterial Hemorrhage, Final Volume"`,Volume) :-
            (
            \+ hem_stop(_)
            ;
            hem_stop(0)
            ;
            hem_stop(Time),
            simulation_duration(Dur),
            Time + Dur >= 1000
            ),
            Volume = 10000.

parameter_change(`"Arterial Hemorrhage, Final Volume"`,NewVol) :-
            hem_stop(Time),
            simulation_duration(Dur),
              NewDur is Time + Dur,
            NewVol is 10000*NewDur//1000.


parameter_change(`"Arterial Hemorrhage, Duration"`,Duration) :-
            (
            \+ hem_stop(_)
            ;
            hem_stop(0)
            ;
            hem_stop(Time),
            simulation_duration(Dur),
            Time + Dur >= 1000
            ),
            Duration = 1000.

parameter_change(`"Arterial Hemorrhage, Duration"`,NewDur) :-
      hem_stop(Time),
      simulation_duration(Dur),
      NewDur is Time + Dur.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_outputs/1
%
% simulation_outputs(+List)
%
%     +List       <list>      list of params to be output by simulator
%
%

simulation_outputs([
                  `"Systemic Arteries, Pressure"`,
                  `"Systemic Arteries, Systolic BP"`,
                  `"Systemic Arteries, Diastolic BP"`,
                  `"Heart Rate, Rate"`,
                  `"Resp Center, Respiration Rate"`,
                  `"Urine Glucose, Conc (mMol/L)"`,
                  `"Core Heat, Temp (F)"`,
                  `"Blood Acid, Venous pH"`,
                  `"Urine Protein, Conc"`,
                  `"Arterial O2, pO2"`,
                  `"CO2 Systemic Arteries, pCO2"`,
                  `"Urine Osmolarity, Osmolarity"`,
                  `"Red Cells, Volume (mL)"`,
                  `"Blood, Volume"`,
                  `"Plasma Protein, [Conc] G/dL"`,
                  `"Sodium ECFV, [Na+] mEq/L"`,
                  `"K+ ECFV, Conc (mEq/L)"`,
                  `"Chloride ECFV, Conc (mEq/L)"`,
                  `"Urine KA-, Conc (mMol/L)"`,
                  `"Urea, [Conc] mG/dL"`,
                  `"Glucose ECFV, Conc (mG/dL)"`,
                  `"Brain Function, Confused"`,
                  `"Brain Function, Confused"`,
                  `"Brain Function, Impaired"`,
                  `"Brain Function, Comatose"`,
                  `"Brain Function, Convulsing"`,
                  `"Brain Function, Not breathing"`,
                  `"Brain Function, May be Dead"`,
                  `"Brain Function, Really Dead"`
               ]).
```

```
% system code, specifies how long the system can expect data from
% the simulator before it should request additional data

continue_time :-
      retractall(continue_time(_)),
      assert(continue_time(60)).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% show_health_meter/0
%
% remove this fact to toggle health meter off.
%

show_health_meter.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% health_skew/1
%
% health_skew(+Skew)
%
%     +Skew       <number>          degree of skewedness
%
% change this fact to skew health graphic from actual health meter
% value, it must be float...the value of its argument will be added to
% the value of the health meter, which is 0.0 - 1.0.  This would be
% useful if you wanted to emphasize a problem that doesn't drastically
% effect the health of the animal.
%

health_skew(0.0).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% on_procedure/1
%
% on_procedure(+Procedure)
%
%     +Procedure  <string>    name of procedure to be allowed in case
%
% comment the procedures that you do not available for this case
% **They will still display in the action list but they will not
% execute when clicked on.
%

% diagnostic procedures
on_procedure(`History`).
on_procedure(`Signalment`).
on_procedure(`Physical examination`).
on_procedure(`Neurological examination`).
on_procedure(`Arterial blood gases`).
on_procedure(`Hematology, Biochemical Profile, and Urinalysis`).
on_procedure(`Exercise Fitness Test`).
on_procedure(`Thoracic radiographs`).
on_procedure(`Blood pressure`).
on_procedure(`Consultant`).
% treatment procedures
on_procedure(`Insert catheter`).
on_procedure(`IV lactated ringer's drip`).
on_procedure(`Whole blood transfusion`).
on_procedure(`Administer IV glucose drip`).
on_procedure(`General Anesthesia`).
on_procedure(`Perform Surgery`).
on_procedure(`Stop hemorrhage`).
on_procedure(`Administer CPR`).
on_procedure(`Place on Oxygen`).
on_procedure(`Place animal on ventilator`).
on_procedure(`Pericardiocentesis and drain`).
on_procedure(`Administer epinephrine`).
on_procedure(`Give IV dolbutamine drip`).
on_procedure(`Administer propranolol`).
on_procedure(`Administer phenoxybenzamine`).
on_procedure(`Administer atropine`).
on_procedure(`Administer cardiac specific calcium channel blocker`).
on_procedure(`Administer enalapril`).
on_procedure(`Administer single dose furosemide`).
on_procedure(`Administer chlorothiazide chronically`).
on_procedure(`Take digoxin chronically`).
on_procedure(`Insulin injection`).
on_procedure(`Administer erythropoietin`).
% resolve procedures
on_procedure(`Transfer to the ICU`).
on_procedure(`Hospitalize`).
on_procedure(`Release patient to owner`).
on_procedure(`Euthanize animal`).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% case/2
%
% case(+Feature,?CannedText)
%
%     +Feature          <atom>      name of message
%     ?CannedText       <string>    message to be displayed to GUI
%
% according to diagnosis, give the message on the screen.
%

case(start,
    [`This is where the text goes that displays `,
    `to the status window immediately after the case `,
    `is loaded in to Virtual Vet ER. You can add as `,
    `many of these lines of text as you would like, `,
    `just be sure to follow this format of the text `,
    `being contained within back quotes and a comma `,
    `following each line except the last.`]).
```

```
case(signalment,
    [`This is where you place the text for age, `,
    `breed, and weight of the dog.`]).

case(history,
    [`Write what you want the user to see when `,
    `they request the patient's history,`,
    `for example, "The dog was hit by a car `,
    `approximately 20 minutes ago!".`]).

case(thoracic,
      [`For example, "Thoracic radiographs appear normal."`]).


case(digoxin, [0.5,0.5,4]).         % scaling factor for digoxin medicine.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% case/3
%
% case(+ExamType,+Feature,?CannedText)
%
%     +ExamType         <atom>
%     +Feature          <atom>      subset of phys exam from list
%                             [general,integument,musculoskeletal,
%                             digestive,genitourinary,eyes,ears,lymph]
%     ?CannedText       <list>      text displayed to status window
%
% Physical exam format.  Facts or Rules can be added as was
% done with some cases below. If no changes are made to the facts
% below, the default text is 'No abnormalities noted'.  When a case
% specific fact or rule is needed, replace those below. There should
% not be more than one fact for a given feature but there may be more
% than one rule.
%
% specifications for canned text of physical exams.
% change text of facts as desired or create rules

case(physical_exam,general,[`GENERAL OBSERVATIONS:`,
      `No Abnormalities noted.`]).
case(physical_exam,integument,[`INTEGUMENT:`,`No Abnormalities noted.`]).
case(physical_exam,musculoskeletal,[`MUSCULOSKELETAL:`,
      `No Abnormalities noted.`]).
case(physical_exam,digestive,[`DIGESTIVE:`,`No Abnormalities noted.`]).
case(physical_exam,genitourinary,[`GENITOURINARY:`,
      `No Abnormalities noted.`]).
case(physical_exam,eyes,[`EYES:`,`No Abnormalities noted.`]).
case(physical_exam,ears,[`EARS:`,`No Abnormalities noted.`]).
case(physical_exam,lymph,[`LYMPH:`,`No Abnormalities noted.`]).
```

```
% specifications for canned text of neuro exams.
% change text of facts as desired or create rules

case(neuro_exam,general,[`GENERAL:`,`No Abnormalities noted.`]).
case(neuro_exam,status,[`STATUS:`,`No Abnormalities noted.`]).
case(neuro_exam,palpation,[`PALPATION:`,`No Abnormalities noted.`]).
case(neuro_exam,postural,[`POSTURAL:`,`No Abnormalities noted.`]).
case(neuro_exam,spinal,[`SPINAL:`,`No Abnormalities noted.`]).
case(neuro_exam,cranial,[`CRANIAL:`,`No Abnormalities noted.`]).
case(neuro_exam,sensation,[`SENSATION:`,`No Abnormalities noted.`]).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% surgical_problem/2
%
% surgical_problem(?Type,?Location)
%
%     -Type       <string>    name of surgery to be performed
%     -Location   <string>    location where surgery will be performed
%
% Case specific surgery defined when surgery is needed for the case.
% If it is not defined no surgery is needed.  An example is below.
% Uncomment it and change parameters if there is a surgical problem in
% your case.
%

% surgical_problem(`Stop hemorrhage`,`Right medial thigh`).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% surgical_message/3
%
% surgical_message(+Type,+Location,?Message)
%
%     +Type       <string>    name of surgery that was performed
%     +Location   <string>    location at which surgery was performed
%     -Message    <string>    message that is returned to the user when
%                             the surgery is performed
%
% Define this to have a message go to the user when the proper surgery
% is performed. Example is below.  Uncomment and change parameters if
% there is a surgical problem in your case.

% surgical_message(`Stop hemorrhage`,`Right medial thigh`,
%                  [`Surgery was successful!`]).
```

APPENDIX C

SAMPLE CASE FILE

```
% scooby.case
%
% Authors: Jason Schlachter, Donald Nute, Scott Brown, Whit Zweifel
% Last Updated: 08/23/2003
%


% embedded queries updated 6/23/2003
:- multifile[continue_time/1,current_cost/1].
:- dynamic[continue_time/1,hem_stop/1,parameter_change/2,
      surgical_problem/3].



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% species/1
%
% species(+Type)
%
%     +Type       <atom>      species of patient (i.e. dog,cat,horse)
%

species(dog).
% species(cat).
% species(horse).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_data_file/1
%
% simulation_data_file(+File)
%
%     +File       <string>    name of QCP3 ICS file to start simulation
%
% simulator start parameters
%

simulation_data_file(`mr_norm.ics`).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_duration/1
%
% simulation_duration(+Time)
%
%     +Time       <number>    # of minutes for which simulator will run
%
% 20.0 minutes in default.  Higher time intervals will lead to longer
% simulation time
%

simulation_duration(60.0).
% simulation_duration(20.0).
intervalG(1.0).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parameter_change/2
%
% parameter_change(+Param,+Value)
%
%     +Param      <string>          name of parameter to be altered
%     +Value      <number>          new value of parameter
%
% Changes from the initial parameter values that are loaded into the
% program from the ICS file
%

parameter_change(`"Arterial Hemorrhage, Switch"`,1).

parameter_change(`"Arterial Hemorrhage, Final Volume"`,Volume) :-
            (
            \+ hem_stop(_)
            ;
            hem_stop(0)
            ;
            hem_stop(Time),
            simulation_duration(Dur),
            Time + Dur >= 1000
            ),
            Volume = 10000.

parameter_change(`"Arterial Hemorrhage, Final Volume"`,NewVol) :-
            hem_stop(Time),
            simulation_duration(Dur),
              NewDur is Time + Dur,
            NewVol is 10000*NewDur//1000.


parameter_change(`"Arterial Hemorrhage, Duration"`,Duration) :-
            (
            \+ hem_stop(_)
            ;
            hem_stop(0)
            ;
            hem_stop(Time),
            simulation_duration(Dur),
            Time + Dur >= 1000
            ),
            Duration = 1000.

parameter_change(`"Arterial Hemorrhage, Duration"`,NewDur) :-
      hem_stop(Time),
      simulation_duration(Dur),
      NewDur is Time + Dur.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation_outputs/1
%
% simulation_outputs(+List)
%
%     +List       <list>      params that will be output by simulator
%
%

simulation_outputs([
                  `"Systemic Arteries, Pressure"`,
                  `"Systemic Arteries, Systolic BP"`,
                  `"Systemic Arteries, Diastolic BP"`,
                  `"Heart Rate, Rate"`,
                  `"Resp Center, Respiration Rate"`,
                  `"Urine Glucose, Conc (mMol/L)"`,
                  `"Core Heat, Temp (F)"`,
                  `"Blood Acid, Venous pH"`,
                  `"Urine Protein, Conc"`,
                  `"Arterial O2, pO2"`,
                  `"CO2 Systemic Arteries, pCO2"`,
                  `"Urine Osmolarity, Osmolarity"`,
                  `"Red Cells, Volume (mL)"`,
                  `"Blood, Volume"`,
                  `"Plasma Protein, [Conc] G/dL"`,
                  `"Sodium ECFV, [Na+] mEq/L"`,
                  `"K+ ECFV, Conc (mEq/L)"`,
                  `"Chloride ECFV, Conc (mEq/L)"`,
                  `"Urine KA-, Conc (mMol/L)"`,
                  `"Urea, [Conc] mG/dL"`,
                  `"Glucose ECFV, Conc (mG/dL)"`,
                  `"Brain Function, Confused"`,
                  `"Brain Function, Confused"`,
                  `"Brain Function, Impaired"`,
                  `"Brain Function, Comatose"`,
                  `"Brain Function, Convulsing"`,
                  `"Brain Function, Not breathing"`,
                  `"Brain Function, May be Dead"`,
                  `"Brain Function, Really Dead"`
               ]).
```

```
% specifies how much data is available before the simulator
% will need to be called again

continue_time :-
      retractall(continue_time(_)),
      assert(continue_time(60)).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% show_health_meter/0
%
% remove this fact to toggle health meter off.
%

show_health_meter.



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% health_skew/1
%
% health_skew(+Skew)
%
%     +Skew       <number>          degree of skewedness
%
% change this fact to skew health graphic from actual health meter
% value. it must be float...the value of its argument will be added to
% the value of the health meter, which is 0.0 - 1.0.  This would be
% useful if you wanted to emphasize a problem that doesn't drastically
% effect the health of the animal.
%

health_skew(0.1).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% on_procedure/1
%
% on_procedure(+Procedure)
%
%     +Procedure  <string>    name of procedure to be allowed in case
%
% comment the procedures that you do not available for this case
% **They will still display in the action list but they will not
% execute when clicked on.
%

% diagnostic procedures
on_procedure(`History`).
on_procedure(`Signalment`).
on_procedure(`Physical examination`).
on_procedure(`Neurological examination`).
on_procedure(`Arterial blood gases`).
on_procedure(`Hematology, Biochemical Profile, and Urinalysis`).
on_procedure(`Exercise Fitness Test`).
on_procedure(`Thoracic radiographs`).
on_procedure(`Blood pressure`).
on_procedure(`Consultant`).
% treatment procedures
on_procedure(`Insert catheter`).
on_procedure(`IV lactated ringer's drip`).
on_procedure(`Whole blood transfusion`).
on_procedure(`Administer IV glucose drip`).
on_procedure(`General Anesthesia`).
on_procedure(`Perform Surgery`).
on_procedure(`Stop hemorrhage`).
on_procedure(`Administer CPR`).
on_procedure(`Place on Oxygen`).
on_procedure(`Place animal on ventilator`).
on_procedure(`Pericardiocentesis and drain`).
on_procedure(`Administer epinephrine`).
on_procedure(`Give IV dolbutamine drip`).
on_procedure(`Administer propranolol`).
on_procedure(`Administer phenoxybenzamine`).
on_procedure(`Administer atropine`).
on_procedure(`Administer cardiac specific calcium channel blocker`).
on_procedure(`Administer enalapril`).
on_procedure(`Administer single dose furosemide`).
on_procedure(`Administer chlorothiazide chronically`).
on_procedure(`Take digoxin chronically`).
on_procedure(`Insulin injection`).
on_procedure(`Administer erythropoietin`).
% resolve procedures
on_procedure(`Transfer to the ICU`).
on_procedure(`Hospitalize`).
on_procedure(`Release patient to owner`).
on_procedure(`Euthanize animal`).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% case/2
%
% case(+Canned,-Message)
%
%     +Canned     <atom>            name of message
%     -Message    <string>          message to be displayed to GUI
%
% according to diagnosis, give the message on the screen.
%

case(start,[`An hysterical man and his son carrying `,
            `a dog on a board burst through the door `,
            `of your clinic screaming "Help him, help him".`]).

case(signalment,[`Two-year old great Dane intact male weighing 65 kg.`]).

case(history,[`The dog was hit by a car approximately 20 minutes before it
was brought to the ER.`]).
```

```
case(thoracic,[`Thoracic radiographs appear normal.`]).

case(digoxin, [0.5,0.5,4]).         % scaling factor for digoxin medicine.




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% case/3
%
% case(physical_exam,+Type,+Results)
%
%     physical_exam     <atom>
%     +Type             <atom>      subset of physical exam from the
%                                   following list
%                             [general,integument,musculoskeletal,
%                             digestive,genitourinary,eyes,
%                             ears,lymph]
%     +Results          <list>      [list of strings or QCP3 variables
%                                   to be displayed to status window]
%
% Physical exam format.  Rules can be added as was done with some cases
% below, otherwise the program will tell the user 'No abnormalities
% noted'.  When a case specific fact is asserted, the default is
% removed.
%
% uncomment the ones that are needed, all others will be default.
%


case(physical_exam,integument,[`INTEGUMENT:`,`No abnormalities noted.`]).

case(physical_exam,digestive,[`DIGESTIVE:`,`No abnormalities noted.`]).

case(physical_exam,genitourinary,[`GENITOURINARY:`,
      `No abnormalities noted.`]).

case(physical_exam,eyes,[`EYES:`,`No abnormalities noted.`]).

case(physical_exam,ears,[`EARS:`,`No abnormalities noted.`]).

case(physical_exam,lymph,[`LYMPH:`,`No abnormalities noted.`]).
```

```
% if hemorrhage has not been stopped
case(physical_exam,general,[`GENERAL OBSERVATIONS:`,
                  `There is increased respiratory effort. `,
                  `There is substantial amount of blood on `,
                  `the board and in the hair coat around the `,
                  `caudal end of the dog. There appears to be `,
                  `ongoing hemorrhage from the medial aspect `,
                  `of the right hind limb. The pelvis is painful `,
                  `to palpation and there is crepitus apparent `,
                  `in the pelvic region on manual manipulation. `,
                  `The dog is carrying the right forelimb `,
                  `abnormally and there is crepitus, pain, and `,
                  `swelling in this region.`]):-
      \+ hem_stop(_).


case(physical_exam,general,[`GENERAL OBSERVATIONS:`,
                  `There is blood on the board but the bleeding `,
                  `has stopped. The pelvis is painful `,
                  `to palpation and there is crepitus apparent `,
                  `in the pelvic region on manual manipulation. `,
                  `The dog is carrying the right forelimb `,
                  `abnormally and there is crepitus, pain, and `,
                  `swelling in this region.`]).


case(physical_exam,musculoskeletal,[`MUSCULOSKELETAL: `,
                  `The dog may have a broken pelvis. `]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% case/3
%
% case(neuro_exam,+Type,+Results)
%
%     neuro_exam  <atom>
%     +Type       <atom>      subset of neuro exam from the list:
%                             [general,status,palpation,
%                             postural,spinal,cranial,sensation]
%     +Results    <list>      [list of strings or QCP3 variables to be
%                             displayed to status window]
%
% neuro_exam case specific fact format.  Works the same way as the
% physical exam.
%


case(neuro_exam,general,[`GENERAL:`,`No abnormalities noted.`]).

case(neuro_exam,status,[`STATUS:`,`No abnormalities noted.`]).

case(neuro_exam,palpation,[`PALPATION:`,`No abnormalities noted.`]).

case(neuro_exam,postural,[`POSTURAL:`,`No abnormalities noted.`]).

case(neuro_exam,spinal,[`SPINAL:`,`No abnormalities noted.`]).
case(neuro_exam,cranial,[`CRANIAL:`,`No abnormalities noted.`]).

case(neuro_exam,sensation,[`SENSATION:`,`No abnormalities noted.`]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% surgical_problem/2
%
% surgical_problem(-Type,-Location)
%
%     -Type       <string>    name of surgery to be performed
%     -Location   <string>    location where surgery will be performed
%
% Case specific surgery defined when surgery is needed for the case.
% If it is not defined no surgery is needed.
%

surgical_problem(`Stop hemorrhage`,`Right medial thigh`,
                  [`Surgery was successful!  You have stopped the
                  hemorrhaging at the right medial thigh.`]).
```

APPENDIX D

USER TESTING: CONSENT FORM

Note that this form was used to test the software at the beginning of my research, before it was renamed VVER, so it refers to the software as K9ER.

K9ER Usability Testing Consent Form

I, ________________________________, agree to participate in a research study titled "K9ER Usability Testing" conducted by Jason Schlachter from the Artificial Intelligence Center at the University of Georgia (542-0358) under the direction of Dr. Donald Nute, of the Artificial Intelligence Center, University of Georgia (542-0358). I understand that my participation is voluntary. I can stop taking part without giving any reason, and without penalty. I can ask to have all of the information about me returned to me, removed from the research records, or destroyed.

The purpose of this study is to test the user interface for K9ER, a software program designed to train veterinary students to diagnose and treat emergency room patients such as dogs, cats, and/or horses. The duration of this study is expected to be approximately one hour.

If I volunteer to take part in this study, I will be asked to do the following things:
1) Complete a 10 minute training program on how to use the K9ER software.
2) Diagnose and recommend treatment procedures for simulated patients on K9ER for a period of 30 minutes. During this time I will be videotaped and asked to say aloud my thoughts (as they relate to K9ER).
3) Complete a 10 minute survey about my experience with K9ER.

The goal of this study is to make the software, K9ER, easier and more intuitive to use. This work may one day benefit veterinary students by allowing them to learn Veterinary medicine in a more interactive, engaging environment. I, the participant, may benefit from this study by learning about new methods of instruction for Veterinary students and about the process of software usability testing.

No risk is expected in this study. I will not experience any discomfort during this study beyond that which may arise from normal computer use.

No information about me, or provided by me during the research, will be shared with others without my written permission, except if it is necessary to protect my welfare (for example, if I were injured and need physician care) or if required by law. Participation is confidential. I will be assigned an identifying number and this number will be used on all of the questionnaires I fill out. There will not be a master list that will connect me to this number; the number is only used to group data from the same experiment. The videotapes from this study will be destroyed following the completion of the user interface research for K9ER. We expect this will be during the spring of 2004 at the latest.

The investigator will answer any further questions about the research, now or during the course of the project (542-0358).

I understand that I am agreeing by my signature on this form to take part in this research project and understand that I will receive a signed copy of this consent form for my records.

___________________________	_________________________	___________
Name of Researcher	**Signature**	**Date**

Telephone: ________________
Email: ______________________________

___________________________	_________________________	___________
Name of Participant	**Signature**	**Date**

Please sign both copies, keep one and return one to the researcher.

Additional questions or problems regarding your rights as a research participant should be addressed to Chris A. Joseph, Ph.D. Human Subjects Office, University of Georgia, 606A Boyd Graduate Studies Research Center, Athens, Georgia 30602-7411; Telephone (706) 542-3199; E-Mail Address IRB@uga.edu

APPENDIX E

USER TESTING: TESTING PROTOCOL

Note that this form refers to the software as K9ER because it was used to test the software before it was renamed VVER. Also, some of the program's features mentioned in this protocol were removed after analyzing feedback from the user testing and others were added. Therefore this protocol will not match up perfectly with VVER in its current state.

User Testing Protocol

The purpose of this material is to teach veterinary students how to use the major features of K9ER. It assumes that the student has the medical background necessary to treat an emergency room patient (dog, cat, or horse). The outline below provides guidelines about what the training should entail. These should be followed to provide consistency in the training across experiments.

AN OVERVIEW OF K9ER SOFTWARE

1) What is this software?
    a) It is proof of concept for a teaching/training tool for veterinary students.
    b) It is a simulator (of animal physiology) and an expert system (a program that consists of rules and facts about the domain of expertise) that work together. Similar systems are used to train medical students in anesthesiology.
2) How does it work?

a) Veterinary experts create an emergency room scenario and store it as a case file. The student loads a case file to begin the program.

b) When a user loads this file, the file sets all the initial parameters for the physiological simulator (1230 of them) and starts the simulation. The expert system uses the information from the simulator in addition to its facts and rules to create a virtual emergency room in which the user tries to treat the patient.

LET'S BEGIN WITH THE SOFTWARE NOW

1) The researcher starts the program.
2) The researcher shows the user the program's GUI
   a. Explain the menu system at the top of the K9ER window.
   b. Explain the use of each text window
      i. Tell user what type of information will appear in each
   c. Explain the function of each button in the GUI.
3) The researcher shows the user how to load a case file using aforementioned menu.
   a. Load the case file called `scooby.case`.
   b. Show user the text that appears in right text window.
4) Explain the student's goal in using K9ER. The objective of the student is to stabilize the patient as quickly as possible while not incurring any unnecessary costs to the animal's owner.
   a. Show clock at bottom or screen. Explain.
   b. Show cost at bottom of screen. Explain.
   c. Show 'Evaluate' button at top right of GUI. Explain.
5) The researcher orders a diagnostic procedure on Scooby.
   a. Click 'Diagnose' button
   b. Click 'History', click 'Apply' button, click 'OK' popup button.
   c. Show user resulting text in right text window.
   d. Show user how an asterisk to the right of 'History' appears to show that this diagnostic procedure has been ordered before.
6) The researcher orders a treatment procedure on Scooby.

   a. Click ‘Treatment’ button.
   b. Click ‘Drug Therapies’, click ‘Autonomic Agents’, click ‘Administer epinephrine’, click ‘Apply’ button, click ‘OK’ button.
   c. Show user resulting text in right text window.
   d. Show user how an asterisk to the right of ‘Administer epinephrine’ appears to show that this treatment procedure has been ordered before.

7) The researcher orders a different type of treatment procedure on Scooby.
   a. Click on ‘Intravenous Therapies’, click on ‘IV lactated ringer’s drip’, click on apply button.
   b. Enter needed information within the ranges specified. Click ‘Ok’ button. When popup window asks for confirmation, click ‘Ok’ button.
   c. Show users clock.
   d. Explain the advance clock function. Use the advance clock function.
8) The researcher uses the ‘Restart’ button to restart the case.
9) The researcher closes the case file and concludes the tutorial.

NOTE TO RESEARCHER

It is perfectly reasonable to answer the participant’s questions as you give this tutorial, however you should not get too off track because we want the user to rely a bit on the GUI to convey information. This tutorial should last approximately 15 minutes.

APPENDIX F

USER TESTING: SURVEY FORM

Note that this survey, as well as the other user testing tools were designed to test the software before it was renamed VVER. Therefore this survey may refer to the software as K9ER and it may reference features that are no longer present in the final version of VVER.

User Testing

K9ER Survey

Part One
Instructions:

The following statements are opinions about the software program (K9ER).
To the right of each statement are numbers that have the following meaning:
***1** = I disagree, **2** = I somewhat disagree, **3** = neutral, **4** = I somewhat agree, **5** = I agree*
Your job is to circle the number that best captures your feelings about each statement.

1. The initial welcome screen was informative and helped me to understand the software.	1	2	3	4	5
2. The process of loading a case, such as snoopy, was intuitive.	1	2	3	4	5
3. It was not clear what type of information each of the white text boxes on the screen would display.	1	2	3	4	5
4. The long wait time required to simulate the patient's condition at the beginning of the case made me wonder if the program was working properly.	1	2	3	4	5
5. The long wait time after selecting a treatment procedure made me wonder if the program was working properly.	1	2	3	4	5
6. It was simple and intuitive to request the patient's history.	1	2	3	4	5
7. It was simple and intuitive to request a physical exam or neurological exam.	1	2	3	4	5
8. I understood that the clock at the bottom left of the window displays the total time in which the patient has been in the ER.	1	2	3	4	5
9. The costs associated with each procedure seemed appropriate.	1	2	3	4	5

10. I understood that some of the diagnostic procedures would not return results immediately.	1 2 3 4 5
11. It was clear that some treatment procedures were ongoing and did not stop automatically.	1 2 3 4 5
12. When a treatment procedure was ongoing, it was clear to me that I could click on the item in the ‘Ongoing Treatments’ window in order to stop it.	1 2 3 4 5
13. When choosing to apply a diagnostic procedure such as ‘Arterial Blood Gases’, I was not sure **when** I would see the results from the diagnosis.	1 2 3 4 5
14. When choosing to apply a diagnostic procedure such as ‘Arterial Blood Gases’, it was not clear **where** the results would appear.	1 2 3 4 5
15. When viewing the available treatment procedures, it was clear that the plus signs to the left of each line indicated that the item could be expanded to reveal more specific treatment procedures.	1 2 3 4 5
16. It was intuitive to use the mouse to expand a heading and reveal its submenu.	1 2 3 4 5
17. It was confusing that I could not apply treatment procedures that had pluses or minuses to the left of them.	1 2 3 4 5
18. I understood that I could advance the clock to avoid waiting for results from a diagnostic procedure.	1 2 3 4 5
19. The steps required to advance the clock were clear.	1 2 3 4 5
20. After **beginning** the case, I felt confused about what I should do next.	1 2 3 4 5
21. After **beginning** the case, the program would have been better if it gave instructions on what options and tests were available to the user.	1 2 3 4 5
22. Upon **beginning** the case, I knew that I could use the diagnostic procedures to learn more about the patient’s health.	1 2 3 4 5

23. After requesting the first **diagnostic procedure**, I knew that I could either request another diagnostic procedure or I could request a treatment procedure.	1	2	3	4	5
24. After requesting a **treatment procedure** that might have stabilized the patient, I was confused about how to determine if the patient was stabilized.	1	2	3	4	5
25. The health meter at the bottom of the screen was helpful in providing a rough estimate of the patient’s condition.	1	2	3	4	5
26. There was enough feedback from the diagnostic procedures and the health meter for me to determine when the patient had been stabilized.	1	2	3	4	5
27. I understood that the case would end when either the patient died or when I stabilized the patient and chose to resolve the case in the correct manner.	1	2	3	4	5
28. To the best of my knowledge, the program did a good job of acurately modeling an emergency room experience. *(given the constraints of a computer program)*	1	2	3	4	5
29. I would have enjoyed having this type of interactive software to supplement my text book readings and classroom lectures.	1	2	3	4	5
30. The instruction that I received at the beginning of the study was helpful.	1	2	3	4	5
31. The instruction that I received at the beginning of the study helped me to understand the **menu system at the top of the window**.	1	2	3	4	5
32. The instruction that I received at the beginning of the study helped me to understand how I could use the **diagnostic procedures** to learn about the patient’s condition.	1	2	3	4	5
33. The instruction that I received at the beginning of the study helped me to understand how I could use the **treatment procedures** to stabilize the patient.	1	2	3	4	5

33. The instruction that I received at the beginning of the study helped me to understand how I could use the **treatment procedures** to stabilize the patient.	1 2 3 4 5
34. I felt comfortable exploring the software and trying to stabilize the patient after going over the sample case.	1 2 3 4 5
35. The instruction at the beginning of this session would have been more helpful if it had included more than one case.	1 2 3 4 5
36. If a program similar to K9ER was used to supplement lectures and textbook material in a veterinary medicine course, it would be best to spend more than one course period to teach students how to use the program.	1 2 3 4 5

Part Two
Instructions:

The following questions are open-ended questions that give you a chance to elaborate on your experience with K9ER. Your answers will be helpful in identifying the strengths and weaknesses of K9ER in its current form so that it may be improved in its next version.

1. What are the program's strengths?

2. What are the program's weaknesses?

3. What could be changed, added, or removed to make the program easier to use?

4. Please add any other thoughts or impressions about your experience with K9ER.

THANK YOU FOR YOUR TIME AND ATTENTION. PLEASE RETURN THIS TO THE RESEARCHER TO CONCLUDE THE STUDY.