THE SURROGATE STUDENT EXPERT SYSTEM FOR TACTICAL TEAM TRAINERS

Adil K. Soofi
Sanders, A Lockheed Company
Information Systems Division
95 Canal Street
Nashua, NH 03061

ABSTRACT

The quality of large tactical team trainers, ranging from embedded trainers to classroom trainers, will be enhanced by the inclusion of expert system tools that ease the burden on the instructors. Such tools will increase both the quality and efficiency of training exercises by freeing up the instructor's time previously spent role playing for the missing student(s). We present an architecture for an expert system-based tool, the Surrogate Student, which replaces missing teams or team members. A prototype version of the Surrogate Student has been developed to model a generic passive Towed Array Sonar (TAS) operator in a Naval Anti-Submarine Warfare (ASW) training scenario. The system analyzes raw sensor data and emulates the action and thinking processes of a human sonar operator. The prototype has been interfaced to a complex distributed training/simulation environment by simply adding another node to the existing Ethernet network.

Introduction

The growing complexity of modern team trainers has established the need for intelligent software tools to assist instructors in providing quality training. Such tools will assist the instructors by easing the burden of both coordinating the exercise as well as substituting for missing students. Expert system-based tools, which provide missing team member substitution and intelligently controlled platforms, can enhance the quality of tactical training devices. Such tools present students with a dynamic, responsive training environment while relieving instructors from unnecessary duties, thereby allowing them to perform their primary function: teaching. These tools can also be applied to embedded training applications by substituting for entire missing operator stations. Sanders has developed such an expert system based prototype, the Surrogate Student, under an IR&D project.

The objectives of a generic Surrogate Student are to provide, through Artificial Intelligence (AI) software methods, substitutes for missing team members in a tactical team training environment, and missing operator stations for embedded training applications. As Figure 1 shows, an expert system-based Surrogate Student must be capable of adequately performing the functions of the missing operator so that his (or her) absence does not affect the exercise, nor require an instructor to play the part of the missing student. These objectives must

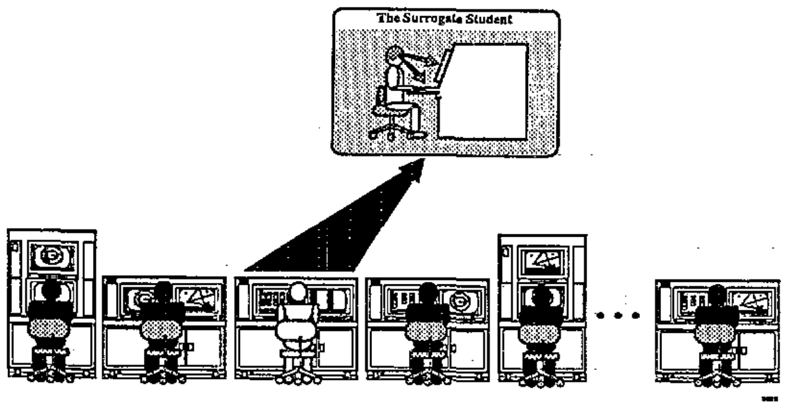

Figure 1. Team Member Replacement

be attained while adhering to another broad set of requirements. The completed system must be as generic as possible to fulfill the role of any team member in a variety of training domains (e.g. naval, battlefield, or air tactical). The initial results of a Surrogate Student should be capable of being incorporated into a larger context, that of replacing an entire trainee team, thus supplying the functionality of a simulated station. Finally, the basic architecture must be easily extensible to allow modular development of an initial prototype.

Our goal in developing our first prototype was to fulfill the requirements of the generic Surrogate Student. Our first candidate for substitution was a passive towed array sonar operator onboard a Naval Anti-Submarine Warfare (ASW) platform. The knowledge applicable to this particular domain was acquired through a process of interviewing experts in the field of passive sonar operation. This knowledge was then translated into rules in an expert system shell. The Surrogate Student architecture was based on a layered rule-base with each layer attributable to a certain function, or level of abstraction. Further, the Surrogate Student has been designed as a state machine, with each state representing a mode of a trainee's operation. Such an approach, in addition to being easy to implement, permits modifications and enhancements to be carried out very efficiently. It allows complex abstractions and ideas to be mapped into the problem space in a structured way.

This paper provides a historical perspective of the evolution of the Surrogate Student. It discusses the architecture and implementation of our first prototype system. The requirements and goals of a Surrogate Student are analyzed both from a generalized point of view and from the detailed perspective of our specific problem. We address the issues involved in implementing our first prototype and discuss the steps we took to integrate the expert system-based Surrogate Student into a tactical training device. The results of the first prototype are discussed and future enhancements for the Surrogate Student are suggested.

History

The foundations of Surrogate Student concepts are evident in the areas of autonomous control, real-time AI/expert systems, and operator modelling, as well as some work on intelligent student substitution. Autonomous control addresses issues of creating independent agents that can exist in uncertain and continually changing environments with little or no supervision. Real-time AI deals with the problems in interfacing AI based software with real-time software. Operator modelling is the process of understanding the behavioral aspects of operator functionality and attempting to replicate it. Surrogate Student is an integration of these ideas resulting in a highly-effective training tool.

In an earlier autonomous control project Katcher[1] developed an autonomous simulated hostile submarine that modelled the decision making process of an enemy submarine commander. This expert system was designed with a hierarchical architecture. Evers, et.al.[2] address some of the practical aspects of implementing expert systems into real-time environments. The computation required to manage a real-time control problem is primarily concerned with routine calculations which must be executed in real-time. Typical expert systems cannot be forced to operate under these same time constraints. The expert system and the real-time environment must be separated in order for the interface between the two to work efficiently. Although this is highly application-dependent, Evers defines general principles that provide an efficient functional separation. Madni, et.al.[3] discuss the importance of incorporating different opponent behaviors for challenging modern tactical decision-makers in a simulation environment. They point out several drawbacks to current software implementations of tactical threats/targets, such as not being able to represent decision-making behavior of enemy tacticians, and not taking into account the training objectives of trainees. They describe how they developed knowledge-based techniques for modelling and controlling the opponent's behavior. Zivovic[4] developed a generic model to simulate the decision-making process of a Tactical Action Officer (TAO) using an expert system. He demonstrates how a knowledge-based approach can be used to train student TAOs. Recent work in student substitution, such as that of Pasewark[5] focuses on using expert system technology to ease the workload of role players in a simulation environment. He shows how expert systems can be made to automate some low level decisions made by a role player in a simulation.

Analysis

The goal of our Surrogate Student project is to develop, using expert system technology, a tool that can be used to substitute for missing human operators in a team training device. The surrogate operator must be capable of adequately performing the functions of the missing operator, without allowing the student's absence to be felt by the rest of the trainee team. In order to provide a complete emulation of a human operator, the Surrogate Student must possess a skill level commensurate with the entire team, and be able to communicate with other team members in a manner similar to a real operator. The Surrogate Student should provide a generic template upon which an entire class of substitute operators can be created. To achieve these goals, the Surrogate Student design allows the high level functionality to be logically separated from the details of the implementation, permitting a broad spectrum of problems to be solved using the the same higher level design. The higher levels represent generic functions that are common to a class of problems, while the lower levels embody the details of the particular problem. This way, the design neatly decouples the problem-dependent and the problem-independent parts. Such an architecture can be extended to various operators in different domains by simply modifying the lower level functionality. Furthermore, a collection of Surrogate Students can be made to replace entire teams in embedded training applications.

For our particular implementation of the Surrogate Student, we chose to simulate the actions and decision-making processes of a passive sonar operator onboard a surface ASW platform. We chose this particular operator because the passive sonar operator does not have an excessive amount of interaction with other team members, thus limiting the voice communication interface and permitting operation in a stand-alone mode. Note that for the first prototype, real voice communications using voice recognition and synthesis was avoided in favor of concentrating on the core expert system. In addition, the simulation software for this operator station was readily available from the Sanders Device 14A12 Surface ASW Trainer program. This software offers a very high fidelity data simulation of a generic passive sonar. For our first prototype, we restricted the scope of our expert system-based operator to be able to handle just one target, or threat platform. In this way, we were able to thoroughly study the problem at hand in a limited context, before proceeding to more ambitious endeavors involving multiple platforms.

The expert system-based Surrogate Student goes through the setup, search, detect, classify, and track phases of a passive sonar operator's duties, while maintaining communication with the sonar supervisor and other team members. At first, the operator must **setup** the passive sonar variable parameters and then **search** for, or 'listen' to, the simulated acoustic energy impinging on the towed array. After the **detection** of initial acoustic patterns, (i.e., frequency lines on a lofar display format), the operator tries to **classify** the source of this acoustic energy based on frequencies detected, harmonic relationships, mechanical coupling ratios, and relative amplitudes. After initial classification, the operator continually tries to justify the initial classification until directed to **track** the target by the supervisor. During the tracking phase, the operator supports Target Motion Analysis (TMA) on the target in an attempt to estimate its range, bearing, speed, and depth. This continues until the contact is lost or the sonar supervisor directs the operator to stop. During the course of the exercise the operator has to interact with the console's peripherals, in our case, a trackball with switches, a keypad, and a pair of touchscreens. The operator uses the trackball to position the frequency and bearing cursors over desired locations on various sonar display formats. The keypad is used to enter numbers in response to queries made by the station software. The touchscreens are used by the operator, in conjunction with the passive sonar software, to set up parameters, choose sonar display formats, and select display options. The passive sonar operator also interacts with other team members and the sonar supervisor, by reporting observations and receiving instructions on an ongoing basis.

The expert system-based Surrogate Student must emulate the functionality of a passive sonar operator as described in the previous paragraph. The Surrogate Student accomplishes this goal through a process of interacting with the simulation software's data structures. The Surrogate Student must also interact with the operator's peripheral devices, manipulating them in such a way as to provide the functionality of a live operator at the missing student's console.

Architecture

As shown in Figure 2, the Surrogate Student architecture is logically divided into three main parts: the simulation module, the Decision Making Module (DMM), and the interface between the two, the communication module. The DMM, consisting of the expert system and Data Acquisition Routines (DARs), is the cognitive part of Surrogate Student. The expert system's database contains knowledge specific to the problem in the form of: 'if: <conditions> then: <actions>' type rules. The expert system gathers information from the simulation module via the communication module, and places that data into its memory elements, referred to as its frames. A part of the expert system, the inference engine, continually attempts to satisfy these rules based on data contained in its frames. The DARs, upon command from the expert system, examine sensor data present in the simulation module and pass back filtered data to the expert system's frames. Thus, the DMM can be thought of as being distributed, with the expert system working in conjunction with the DARs to provide an emulation of how an operator extracts and processes sensor data.

Surrogate Student uses data structures to store and transmit massive amounts of simulation data. The entire system depends upon information stored in these data structures on both the expert system side and the simulation side, as illustrated in Figure 3. The simulation data structures contain information on the state of the operator, what data he (or she) is currently 'looking' at, as well as information that is derived by the operator. The expert system-based sonar operator has to sift through a lot of unprocessed sonar data. Therefore, these data structures were designed to be easily accessed by DARs and to properly represent the state of the operator's mind. In addition, 'stuffing' these structures for compaction

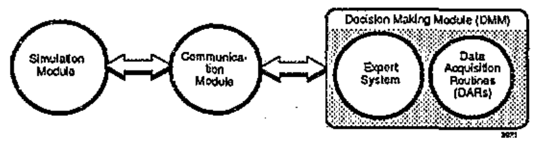
Simulation Module
Communication Module
Decision Making Module (DMM)
Expert System
Data Acquisition Routines (DARs)

Figure 2. Surrogate Student High Level Software Design

and transmittal corresponds to a human operator reducing and manipulating this data. The algorithms used to do this cannot be too mechanistic, or 'computer-like', which would make the process seem artificial. This filtering mechanism is an extension of the operator's level of expertise in the sense that different operators extract patterns and perceive raw sonar data in different ways.

Surrogate Student, at a high level of abstraction, can be thought of as a state machine. Each state represents a different mode of operation of the student, as shown in Figure 4a. There are a set of rules that control state transitions, which represent how and under what conditions the operator may move to another state. Superimposed on the state machine representation is the Surrogate Student's hierarchical framework, as illustrated in Figure 4b. There are three levels of abstraction, with each lower level representing a higher degree of detail about the particular implementation. The following describes the functionality within each hierarchical level:

<u>LEVEL 1</u> describes the high level states, and a valid set of transitions between them. It defines the preconditions to move from state to state. This can be thought of as a high level control layer that directs the interaction between different states of Surrogate Student.

<u>LEVEL 2</u> describes each modelled state in detail. Each state is described in terms of a flowchart, or map, representing the actions and decisions an operator goes

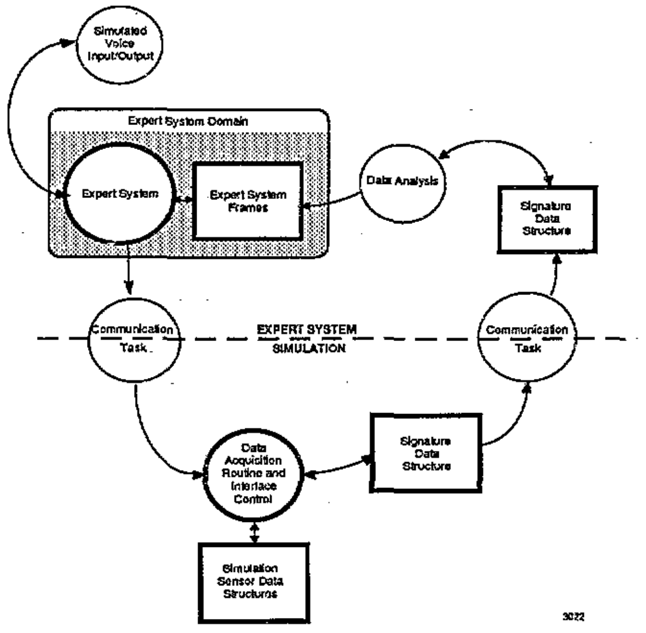

Figure 3. Surrogate Student Data Structures

through in that state. Some aspects of communication are also embodied in this level of abstraction.

LEVEL 3 can best be described as an "implementation" layer. This is where all the interaction between the decision-making rules and expert system frames occurs. All the calls to the data acquisition and manipulation routines are conducted at this level.

Each state, at every level, was designed in two steps: first, in terms of what its function is, and then in terms of how it interacts with other states. Such structuring was important not only from the point of view of ease of development and debugging, but also for expansibility. Using this design methodology, one can place additional states into the implementation with ease. Additional

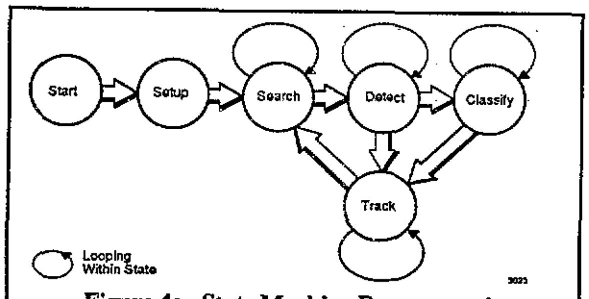

Figure 4a. State Machine Representation

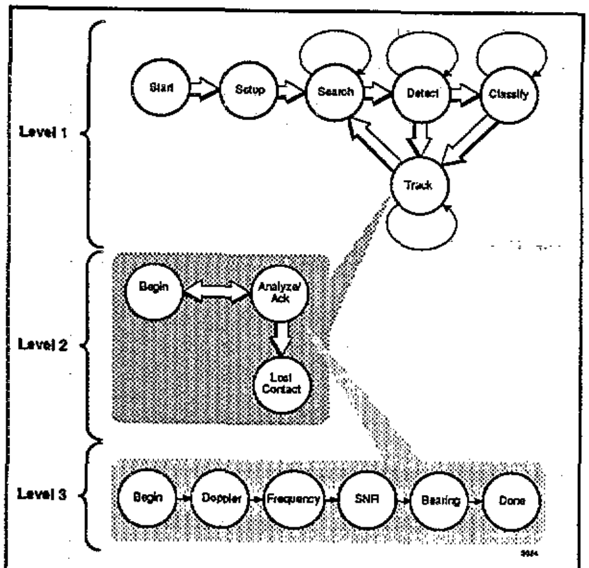

Figure 4b. State/Machine Hierarchial Representation

states would represent new modes of operation for the Surrogate Student. Hierarchical design permits us to conceal the low level details of the design from higher level abstractions. The higher levels represent generic functions that are common to a 'class' of problems, while the lower levels embody the details of the particular problem. In this way, the design neatly decouples the problem-independent part from the problem-dependent part. Also, additional layers can be placed atop the existing ones to realize higher levels of abstractions, for example, a control layer that could handle multiple contacts.

Implementation

The Surrogate Student prototype was implemented across a distributed MicroVAX-II network as shown in Figure 5. The MicroVAXes, running the MicroVMS operating system, communicate via an Ethernet Local Area Network (LAN). The present system contains two consoles and a total of four computers. Each console consists of a computer, two high-resolution color displays, and a set of peripherals (touchscreen, trackball with switches, keypad). The instructor console runs the software to coordinate the entire exercise, along with platform validation software to verify the behavior of each platform. From this console a user can control the state of each platform and its dynamics through a Tactical Situation Display (TSD), and a set of interactive menus. The TSD graphically presents information about all the platforms in the exercise, and permits the user to display the detailed state of any of the platforms. The sonar display processor (DP) console runs the software used to display and manipulate sonar information in a number of different formats. The auxiliary processor (AP) is responsible for the acoustic modelling of each platform given its current operating parameters (e.g. speed, depth, range). The expert system computer runs Nexpert-Object, our expert system, along with its entire knowledge base that comprises the cognitive part of Surrogate Student. The expert system computer communicates with the sonar DP computer via utility routines. The expert system passes control information to either emulate the use of one of its peripheral devices (i.e. trackball, trackball switches, key pad, or touchscreens) or request sonar acoustic information. The sonar DP returns data to the expert system via utility routines that deposit data into its frames.

The expert system, through a process of matching the contents of its continuously updated working memory with its rules, decides the course of action for the student to take. The expert system can either initiate an action or request a particular kind of data. Actions correspond to the manipulation of sonar operator controls such as touchscreens, trackball, trackball switches, and keypad. Requests for data correspond to acquisition of sensor information via the DARs. The following describes scenarios of both cases. The expert system may, for example, decide to change one of the sonar display formats. It does this by issuing the appropriate command through a communication task, along with the appropriate parameters. The software on the DP computer receives the command and manipulates the sonar data structures in such a way as to simulate the activation of a touchscreen area which executes the corresponding function. Another example would be that of the expert system deciding that it wants to move the trackball to position the frequency cursor over a particular detection, or enter a number on the sonar console's keypad in response to a query. These actions are also executed in the manner described above. Another class of transaction is that of data transmittal from the sonar environment to the expert system. The Surrogate Student requests data in the same manner as described above. Such a request for data triggers the appropriate DAR on the sonar side. The selected DAR extracts and filters the data, then sends information back to the expert system. This data gets examined periodically, manipulated, and then deposited into the expert system's working memory.

The knowledge base for the Surrogate Student was developed through a knowledge engineering effort, which can be defined as the process of gathering information about the problem from a domain expert, and capturing that knowledge in the form of expert system rules. For our first prototype, Sonalysts, Inc., a consulting firm specializing in sonar operations, served as our domain expert and helped us to devise a comprehensive model for the passive towed array sonar operator. The operator model included various aspects of communication, behavior, and decision making. The communication model covered the interaction between the passive sonar operator, other operators, and the sonar supervisor, for the purpose of relaying instructions and reporting observations. The operator's behavior model was constructed via analysis of operator guidelines for a typical passive towed array sonar. This model defines the actions and decisions to be taken by the operator under various conditions and phases of the mission. The decision-making model contains all the domain-specific information about the sonar operator. This information is highly dependent on the skill level of the operator being modelled. Factors affecting student performance in this model are: experience, intelligence, ambition, and attitude (confidence).

After the operator model was compiled, we translated this knowledge into 'if-then' type rules in the generic syntax of a typical expert system. By not translating these rules into the specific syntax of a particular expert shell, we were able to maintain English-like 'pseudo-rules', easily transformable into the context of a specific expert shell. Once we selected our expert system shell, Nexpert-Object, we were easily able to enter these rules into its knowledge base. The initial knowledge engineering effort yielded about 120 rules. This expert shell has a very powerful user interface, which permits different components of the knowledge-base to be

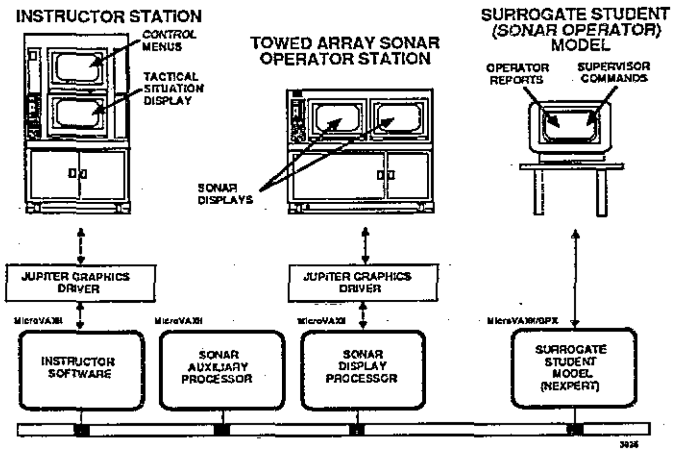

Figure 5. Surrogate Student System Configuration

seen and manipulated graphically. Nexpert-Object is an object-based expert shell which provides a rich set of primitives with which to represent the problem space. It provides integrated backward and forward chaining inference strategies which can be changed from the rules themselves. It also has a very powerful interface to external routines, which allows interfacing the expert system to user-developed application code. Furthermore, the expert system can be totally controlled from external routines, providing a framework for the creation of powerful applications. Figure 6 shows an example of part of the Surrogate Student rule base displayed through the Nexpert-Object graphic rule network interface.

The Surrogate Student and Device 14A12 modules are logically divided into distinct parts. The Device 14A12 subsystem can run in a stand-alone mode with the Surrogate Student tasks 'switched off'. The GPX workstation is dedicated to the Decision Making Module (DMM) of Surrogate Student. Only one other computer on the LAN has the Surrogate Student tasks and data structures resident on it: the sonar display processor (DP). These tasks, which are still logically separate from the Device 14A12 system, are responsible solely for communications to and from the expert system. This clean division of software was important in creating an architecture that is not dependent on any particular implementation, but instead provides a generic template for a Surrogate Student.

Evaluation

This section discusses the evaluation of the first Surrogate Student prototype. As a typical training exercise is run one can observe the Surrogate Student as it progresses through its different states and carries out the corresponding functions in every state. The supervisor first gives instructions to the Surrogate Student to setup the sonar console, at which point it initializes sonar parameters (i.e. cable scope length, time constant). The Surrogate Student then enters the search state in which it monitors the Narrow Band and Broad Band display formats for contacts. The Surrogate Student first observes the OwnShip signature in order to discriminate from contact acoustics. Once a possible contact pattern is detected, the Surrogate Student enters the detect state in which it tries to ascertain, based on the limited data available to it so far, whether the signature is being generated by a friendly or threat platform. For this it relies on apriori heuristic knowledge embedded in the expert system rules. At this point the Surrogate Student makes the appropritate screen selections and format changes to further analyze data before proceeding to the classification state. In the classification state the Surrogate Student tries to put together all the sensor information it has gathered so far and classify the contact based on its knowledge of paltform acoustic signatures. Once an initial classification has been made, the Surrogate Student keeps trying to refine it based on newly aquired sensor data. Once the Surrogate Student reaches a certain level of confidence about the classification it proceeds to the tracking state in which it monitors changing sensor parameters. It continuously tries to correlate these changes to platform maneuvers (e.g. speed, depth, and range changes). If it loses track of the contact it reinitiates the entire process, otherwise it continuously tries to get a better picture of contact maneuvers based on new data.

The actions and decisions of the Surrogate Student are communicated to the supervisor through a graphic interface, providing him with information about the contact. The Surrogate Student continuously explains its behavior including its decision making process, its reason for taking a certain action, and why it fired a given rule. On looking at the sonar screens, one can see the Surrogate Student going back and forth between different display formats, moving cursors to position them over suspected contacts, and keying in numbers in response to queries.

The expert system's inference engine processes new data and fires rules at an extremely rapid rate approaching real time (i.e. the expert system is aware of and reacts to changes in the simulation data immedeately). In fact, we had to insert time delays to slow the process down and give the impression of a real human operator using the sonar console. Repetitive and algorithmic functions that were initially implemented as rules were migrated to external routines in order to make the entire process more efficient and streamlined. This, in addition to NEXPERT's efficient pattern-matching vocabulary, reduced the total number of rules from about 120 to about 90. The overall quality of a knowledge-based system cannot be judged alone by the number of rules present in its knowledge base. It is the combination of a powerful set of rules and external routines that together contribute to the effectiveness of an expert system.

The Surrogate Student is now ready to be formally evaluated by passive sonar experts to assess its effectieness in modelling a human operator from a high level, behavioral point of view. The Surrogate Student will be tested with a variety of different scenarios that will include threat platform and OwnShip maneuvers (e.g. speed, course, range, and depth changes). Such maneuvers can be done interactively by using the Instructor station menu-driven software. The acoustic patterns of some threat platforms will be modelled so that certain components appear and disappear at different times during the exercise. The Surrogate Student's ability to handle lost contacts, and subsequently regain them will also be formally tested. Again, because of Surrogate Student's graphic interface, this process will be simplified.

Conclusions

The Sanders Surrogate Student provides an innovative approach to solving the problem of missing team member substitution. This approach promises to reduce demand on the instructor's time while providing a high quality replacement of a trainee operator. The Surrogate Student was designed so that enhancements and modifications are very simple. The hierarchical approach to knowledge base design permits the complete design and testing of one layer, before moving on to the next. In this way, the lower level rules that embody the implementation details can be hidden from higher level abstractions in the upper layers. The state machine representation within each layer permits functionality to be added to a given layer in a modular way. One first defines a new state and its inner functionality, and then designs the interface to other states. The final step is defining how the system will transition between this state and others. Using this general methodology, complex ideas and representations can be easily mapped into the problem space in a structured manner.

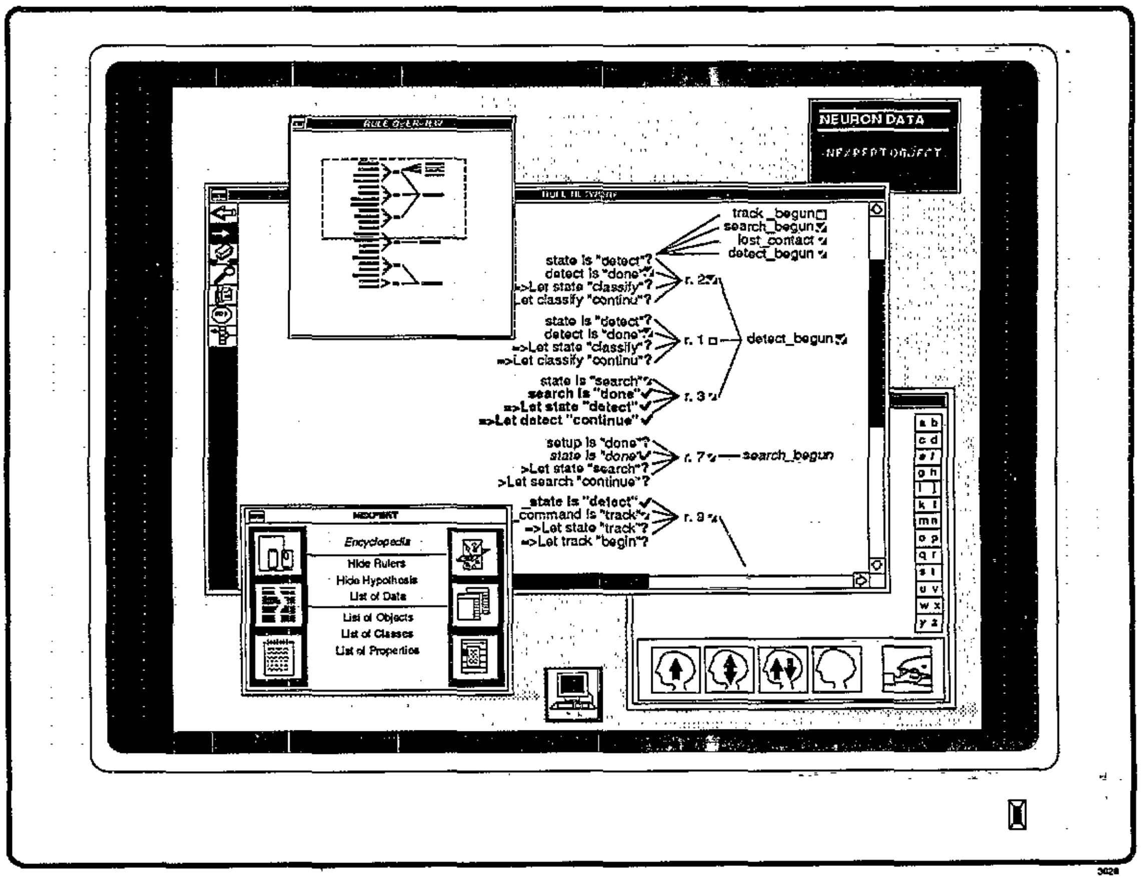

Figure 6. The Expert System Graphic Interface

The Surrogate Student is not confined to the hardware used in its first prototype. Given the hierarchical approach, the lower level routines that communicate with the hardware can be replaced/modified to match the specifications of the targeted platform. The expert system knowledge base, present in compiled 'C', can be easily ported to another hardware platform/operating system. All the routines that communicate directly with the expert system make use of Nexpert AI-kernel calls, which are entirely hardware/operating system independent. There is a very clean, distinct division between the Surrogate Student code and the simulation software. The simulation software can run without the Surrogate Student modules, which can be thought of as intelligent add-ons to the distributed simulation environment.

LIST OF ACRONYMS

ASW	Anti-Submarine Warfare
DAR	Data Acquisition Routine
DMM	Decision Making Module
DP	Display Processor
ES	Expert System
LAN	Local Area Network
SS	Surrogate Student
TAO	Tactical Action Officer
TMA	Target Motion Analysis
TSD	Tactical Situation Display

REFERENCES

[1] Katcher, D.I. "A Flexible Expert System Architecture for Tactical Trainers," Proceedings I/ITSC, 1988.

[2] Evers, D.C, Smith, D.M., Staros, C.J. "Interfacing an Intelligent Decision-Maker to a Real-Time Control System," SPIE Vol. 485 Applications of Artificial Intelligence, 1984.

[3] Madni, A.M., Ahlers, R., Chu, Y. "Knowledge-Based Simulation: An Approach to Intelligent Opponent Modelling for Training Tactical Decisionmaking," Proceedings I/ITSC, 1987.

[4] Zivovic, S. "Can Expert Systems Help Train Tactical Action Officers: Some Experiences From An Early Prototype," Thesis: Naval Postgraduate School, 1986.

[5] Pasewark, G.J. "Application of Expert System Technology to Aid Controller/Role Players in a High Realism Training Environment," Proceedings I/ITSC, 1987

ABOUT THE AUTHOR

Adil Soofi is an Electrical Engineer for Sanders Associates, Inc., A Lockheed Company. He received his MS in Electrical Engineering from Purdue University, where he did work in Artificial Intelligence and Expert System development, and his BS in Electrical Engineering from the University of South Florida. Mr. Soofi is currently enrolled in an MS-Computer Science program at Boston University. He has been involved with IR&D work at Sanders including Surrogate Student development and prototyping a low cost graphics console. Mr. Soofi is a member of the TauBeta Pi and Eta Kappa Nu honor societies, and a member of AAAI.

*** MicroVAX , MicroVMS , and GPX are registered trademarks of Digital Equipment Corporation**

**** Nexpert-Object is a trademark of Neuron Data, Inc.**