1989-06

An intelligent computer-aided instruction system for Naval ship recognition

Bernier, Denise R.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INTELLIGENT COMPUTER-AIDED
INSTRUCTION SYSTEM
FOR NAVAL SHIP RECOGNITION

by

Denise R. Bernier

June 1989

Thesis Advisor: Neil C. Rowe

Unclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: UNCLASSIFIED	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report: Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization: Naval Postgraduate School; 6b Office Symbol (*If Applicable*): 52	7a Name of Monitoring Organization: Naval Postgraduate School
6c Address (*city, state, and ZIP code*): Monterey, CA 93943-5000	7b Address (*city, state, and ZIP code*): Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization; 8b Office Symbol (*If Applicable*)	9 Procurement Instrument Identification Number
8c Address (*city, state, and ZIP code*)	10 Source of Funding Numbers: Program Element Number; Project No; Task No; Work Unit Accession No

11 Title (*Include Security Classification*)
AN INTELLIGENT COMPUTER-AIDED INSTRUCTION SYSTEM FOR NAVAL SHIP RECOGNITION

12 Personal Author(s)
Bernier, Denise R.

13a Type of Report	13b Time Covered	14 Date of Report (*year, month,day*)	15 Page Count
Master's Thesis	From To	June 1989	97

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (*continue on reverse if necessary and identify by block number*)
Field	Group	Subgroup	Artificial Intelligence, Intelligent Computer-Aided Instruction, Ship Recognition, Prolog

19 Abstract (*continue on reverse if necessary and identify by block number*)

This thesis discusses the design and implementation of an intelligent computer-aided instruction system for Naval ship recognition. The system uses artificial-intelligence techniques to provide an interactive tutoring environment. The student's abilities for ship recognition are tested using randomly selected side-view photos. The student's response is compared to the correct ship in an expert module. If the response is incorrect the features of the correct ship are compared with those of the incorrect ship to formulate a hypothesis concerning the student's misconceptions. Tutoring strategies are chosen based on this comparison. The system provides a recognition test, a summary review, and an individual photo review. A review of recognition features for each ship is supplied during the recognition test. A final summary is generated at the end of testing.

20 Distribution/Availability of Abstract	21 Abstract Security Classification
[X] unclassified/unlimited [ ] same as report [ ] DTIC users	UNCLASSIFIED
22a Name of Responsible Individual: Prof. Neil C. Rowe	22b Telephone (*Include Area code*): (408) 646-2462; 22c Office Symbol: Code 52Rp

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted — security classification of this page

All other editions are obsolete — Unclassified

Approved for public release; distribution is unlimited.

An Intelligent Computer-Aided Instruction System for Naval Ship Recognition

by

Denise R. Bernier
Lieutenant, United States Navy
B.S., Medical College of Virginia, 1981

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

ABSTRACT

This thesis discusses the design and implementation of an intelligent computer-aided instruction system for Naval ship recognition. The system uses artificial-intelligence techniques to provide an interactive tutoring environment. The student's abilities for ship recognition are tested using randomly selected side-view photos. The student's response is compared to the correct ship in an expert module. If the response is incorrect the features of the correct ship are compared with those of the incorrect ship to formulate a hypothesis concerning the student's misconceptions. Tutoring strategies are chosen based on this comparison. The system provides a recognition test, a summary review and an individual photo review. A review of recognition features for each ship is supplied during the recognition test. A final summary is generated at the end of testing.

TABLE OF CONTENTS

I. INTRODUCTION

Computer-assisted instruction systems in use today can provide a sophisticated learning experience for students, in some cases rivaling the instruction provided by human instructors. One advantage of computer-based training is the one-on-one teaching available for students. In the majority of today's classrooms there is little opportunity for students to receive the direct attention that is possible with computer-based instruction.

Although computerized instruction has been available to one degree or another for many years, it has not been utilized as extensively as it could have been. One reason for this is the cost of computer hardware and software. Systems in use in the past used expensive machines to provide tutoring services that could be provided through books and other types of less expensive media. Computer tutoring systems needed to develop further in order for the benefits to justify the cost.

In recent years a great deal of research has been done in achieving "thinking" systems. These new "thinking" systems provide a mechanism for computer-based instruction that is much closer to the teaching possible from human instructors. This thesis will investigate the possible use

of a "thinking" computer system for training military personnel in ship recognition.

A. SHIP RECOGNITION

Ship recognition is the ability to differentiate between different classes of ships. Weapons systems, which are easily recognizable, can be used to visually determine the general mission of an observed ship from a side view. Once a ship is identified by class, intelligence information can provide further data concerning the armament and capabilities of the contact. Positive identification of unfriendly ships, before they come within attack range of our own ships, provide a strong incentive for up-to-date intelligence [Ref. 1].

Currently ship recognition is taught using drill methods. Intelligence officers with knowledge of each ship class drill officers and crewmen using slides of both ship photos, line drawings and silhouettes. There are also flash cards and books available for individual study. Recently, computer programs for use on microcomputers have been introduced into military commands. These programs provide another method of drilling the student on ship features, while not requiring an instructor on hand.

B. METHODOLOGY

We have programmed a computer-based ship recognition tutor from information provided by military recognition

guides and other recognition books [Refs. 2,3,4]. A ship feature description and a menu of ship names are provided to the user. The user is required to select the name of the correct ship or ask for help. Help is provided in the form of the correct answer along with a list of the identifying features of the ship in question. If the user selects a ship name the ship recognition tutor will compare the user's answer with that of an expert ship-identification module. Based on this comparison the user receives immediate results in the form of a positive response, with a review of key features, or a negative response, with a feature-by-feature review of the differences between the correct response and the user's response.

Artificial intelligence (AI) techniques implemented in the Prolog programming language were utilized in the design of the ship recognition tutor. This type of computer program is referred to as an intelligent computer-aided instruction (ICAI) system or an intelligent tutoring system (ITS).

C. PREVIEW

Chapter II provides an overview of the use of computers in instruction, with a review of computer-aided instruction (CAI) and intelligent computer-aided instruction (ICAI). Chapter III provides an overview of ship recognition and current training techniques. Chapter IV discusses the

design and implementation of our ship recognition tutor. Chapter V discusses the performance of the ship recognition tutor. Chapter VI is the conclusion and discusses the benefits and limitations of the ship recognition tutor. Appendix A provides a representative user session. Appendix B contains a User manual for the program, with information for use and modifications. Appendix C contains the Prolog source code for the ship recognition tutor.

II. OVERVIEW OF COMPUTER-AIDED INSTRUCTION

A. BACKGROUND

Computers were first used for teaching in the late 1950's. Computers have gone on to be used in all levels of education. Medical students use computers to practice diagnosis and prescription on patients simulated by computer programs. Engineering students use computers to assist in problem solving. Computers are also used for educational purposes outside schools. Children learn spelling and arithmetic and high school students practice for college entrance examinations using educational software available to the general public [Ref. 5].

Computer-aided instruction (CAI) is the name given the use of computers in education. Intelligent computer-aided instruction (ICAI) refers to instructional systems utilizing artificial intelligence technology. The acronym CAI will be used to refer to the more traditional approaches to computerized instruction.

B. COMPUTER-AIDED INSTRUCTION

Traditional CAI programs tend to be statically organized structures that contain both the domain and pedagogical knowledge of the expert. This is similar to the idea that books contain the knowledge of their authors.

There is no expectation that a book can dynamically access the knowledge it contains to answer unexpected questions, or that the book can add new knowledge to its current contents [Ref. 6]. CAI programs translate the teacher's decisions into a program, all the possible circumstances that might require a decision are considered, and code to deal with each of these possibilities is included in the program.

Early research into CAI tried to build systems which contained course material in sequential lessons. These early programs were either electronic "page-turners", which printed prepared text, or drill-and-practice monitors, which asked students to answer questions and then responded using prestored answers and comments. Educational uses of computers have expanded to include free-style use of the machine, where the student learns problem solving by programming, as well as the use of games and simulations as instructional tools.

C. ICAI SYSTEMS

The idea of research into instructional systems utilizing artificial intelligence is to capture the knowledge that experts use to compose an instructional interaction. The goal is to produce a system that is capable of autonomous reasoning on the basis of only primitive principles. Reasoning from the expertise of experts rather than looking

up their decisions, creates the possibility of new decisions that may not have been anticipated by the experts [Ref. 6].

1. Structure

ICAI systems consist of three main components: the problem-solving expertise, which is the knowledge to be imparted; the tutoring strategies, which indicate how the system will teach the student; and the student model, which indicates what the student knows or does not know [Ref. 7]. These components are respectively referred to as the expert module, the tutor module and the student model. An interface module is also needed to provide a smooth, clear presentation to the user. Figure 2.1 shows the components of an ICAI system.

a. Expert Module

The expert module serves two functions. It acts as a source of the knowledge to be tested and a means of evaluating the student's responses [Ref. 6]. The expert module for tutors of technical skills should be drawn from the knowledge of several experts in the field to be taught, in order to prevent blind spots in the knowledge base [Ref. 8].

b. Tutor Module

The tutor module selects problems to be solved, monitors performance and provides assistance. This module

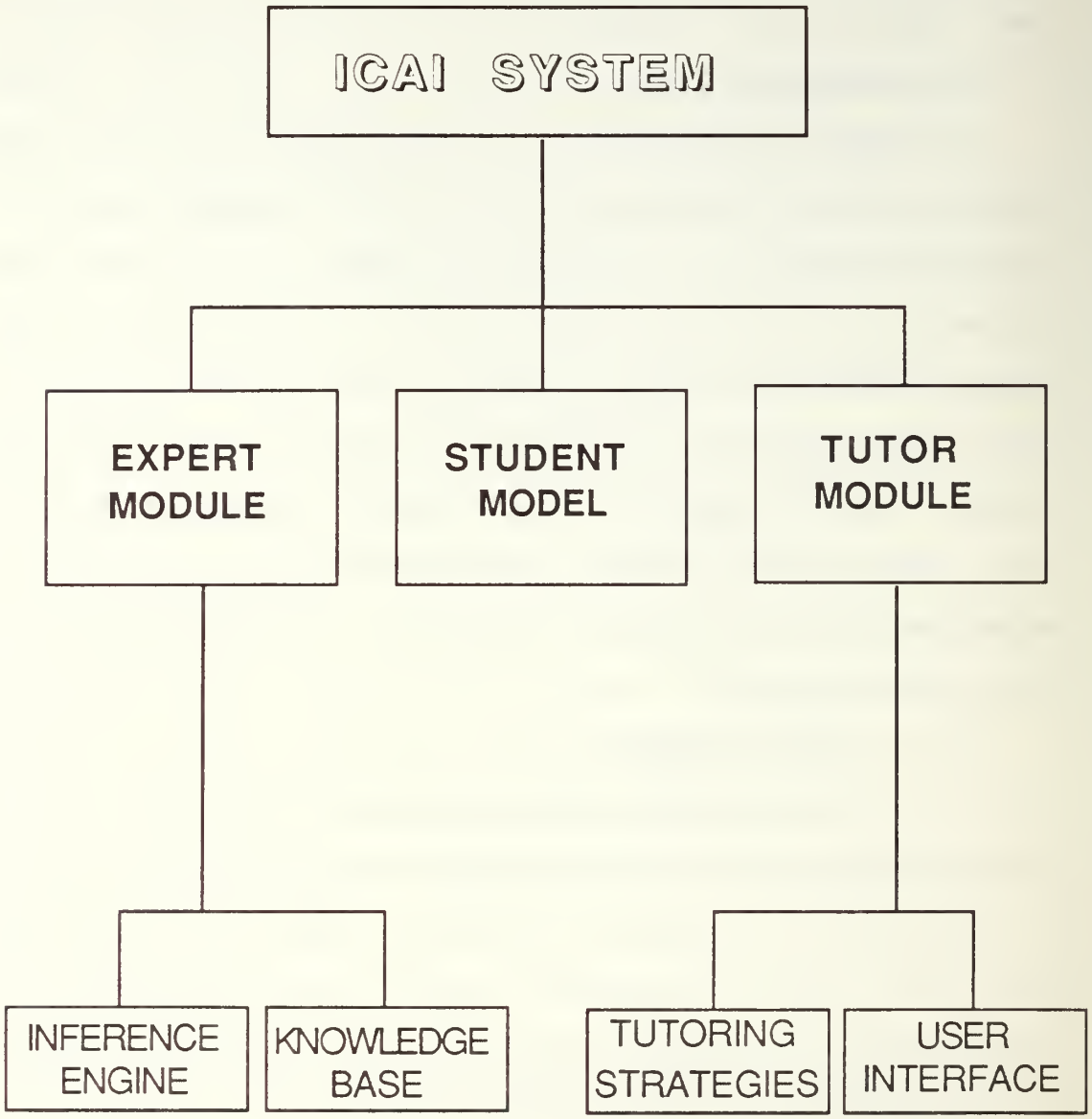

Figure 2.1 ICAI System Components

requires knowledge in addition to knowledge of the subject being taught: knowledge of teaching skills.

Most ICAI research into tutor modules has explored diagnostic modeling. The program evaluates the student's understanding of the topic by interpreting his response. Feedback is then provided, to allow the student to learn skills he has used incorrectly or not at all. Recently there has been an attempt to decide just the right thing to say that will allow the student to realize his error and to use this knowledge to switch to a better method. This classical method, the Socratic method, requires repetitive questions that guide the student along a specific line of reasoning [Ref. 7].

A second teaching strategy is called *coaching*. This method does not follow a strict lesson plan, but encourages learning through activities such as computer games. The primary aim of the student is enjoyment of the game, with learning as a by-product. The computer "coach" observes the progress of the game and offers suggestions or new information as required. The coaching method requires little interruption and allows the student to proceed relatively on his own. [Ref. 7]

c. Student Model

The student model provides a representation of what the student does and does not know. This information is generally obtained by comparing the student's response

to that of the expert. Once a comparison has been made, the information is used to determine possible misconceptions and to decide the best tutoring strategy to correct the misconception.

2. Applications

SCHOLAR is a mixed-initiative computer-based tutoring system that tutors about facts concerning South American geography. Both the student and the system can initiate conversations by asking questions. This program was a pioneering effort in the development of programs that could handle unanticipated student questions and react appropriately [Ref. 7].

WHY, which tutors students in the causes of rainfall, was developed in response to a need for systems that could deal with subject matter that is not purely factual. Student errors could involve not only forgotten facts but also misunderstandings as to why processes work the way they do [Ref. 7].

The SOPHIE tutoring system allows the student to learn by trying out his ideas rather than by direct instruction. A model of problem-solving techniques is contained in the system domain along with heuristics for answering student questions and handling misconceptions. These features allow a one-to-one interaction between the student and a computer "expert", who helps the student come

up with his own ideas. The problem-solving skills taught in SOPHIE are taught in the context of an electronics laboratory [Ref. 7].

WEST was the first ICAI system to utilize a coaching strategy. This method involves a computer game that the student plays, while a "coach" observes in the background, occasionally offering suggestions. A coaching strategy does not interrupt the student so often that it becomes intrusive. This would destroy the fun of playing the game. The idea of the WEST system is to provide drill-and-practice in arithmetic to elementary school children. WUMPUS is another coaching system which requires knowledge of logic, probability, decision theory and geometry to track down and destroy the Wumpus while avoiding various traps [Ref. 7].

GUIDON is a system that teaches diagnostic problem-solving in the medical field. The system utilizes the MYCIN consultation system as the expert domain. GUIDON's teaching knowledge is totally separate from the MYCIN subject domain. MYCIN provides infectious-disease rules that constitute the topic to be discussed and a basis for evaluating student responses. The system provides a mixed-initiative dialogue that goes beyond merely responding to the student's last response or repetitive questioning [Ref. 7].

GUIDON's case method of presentation closely resembles the presentation method we used in the ship recognition tutor. The student is required to concentrate on specific cases or, in our system, specific ships. GUIDON allows the student to ask questions to gather information, whereas our ship recognition tutor provides information through a menu-based dialogue. In both systems the tutorial system only intervenes when the student asks for help or when his responses are suboptimal [Ref. 6].

STEAMER is a system used to train engineers who will operate large ships. The training is designed to help the student form a mental model of the steam propulsion plant and understand related engineering principles. STEAMER is important in the research into ICAI systems since it stimulated interest in the development of object-oriented graphic simulations in training [Ref. 7].

III. OVERVIEW OF SHIP RECOGNITION

Numerous changes have occurred in naval weaponry in the past decade, resulting in warships that are hard to distinguish. Knowledge of identifiable features and weapon systems is required to differentiate between different classes of warships and to determine a ship's general mission. Positive identification of unfriendly ships or aircraft before they reach attack range can mean the difference between life and death in a wartime situation.

A. PURPOSE

Training of a ship's crewmen, intelligence personnel and pilots in the identification of ship classes and weapons systems is required to provide up-to-date intelligence information. Air reconnaissance, both visual and photographic, plays an important role in the intelligence-gathering mission. Aircraft personnel must not only be able to recognize unfriendly ships, but must also be able to identify weapons systems on board. Information acquired through air reconnaissance is rapidly passed to analysts on the ground or on board ship. Once the ship is identified by class, the operational intelligence personnel can brief the commander on the capabilities of the contact. [Ref. 1,10]

B. PROCEDURES

Naval ship recognition, as taught to Naval personnel, follows two basic ideas. A fast, overall impression of the ship provides operational identification. Then there is a feature-by-feature analysis, requiring prolonged observation. This observation can be either direct or through study of reconnaissance photos. Information that can help in the identification of ships includes the geographical location of the ship, knowledge of which ships are known to cruise in the region and intelligence information gathered prior to the sighting. Features of interest to intelligence personnel are: (1) armament such as missile launchers, guns and ASW weapons; (2) armor on sides decks, turrets, towers and superstructures; (3) electronic equipment; and (4) navigational equipment. [Ref 1,10]

C. TRAINING PROGRAMS

There are several aids for training military personnel in ship recognition. Generally ship recognition is included in training given by intelligence personnel. Sessions, utilizing slides of ship photos along with either line drawings or silhouettes, are provided to allow personnel to become familiar with various ships. During these sessions the instructor points out key identifying features and suggests memory aids to help the student remember the ship along with its features. The intelligence officer will

also provide classes to review ships that may be encountered in specific areas of deployment.

Supplemental resources available for learning ship recognition include flash cards, with line drawings or silhouettes, and books such as Jane's Fighting Ships. There are CAI programs as well. The programs provide a silhouette of the ship, allowing the student to answer a multiple-choice question regarding the identity of the ship. The student is provided with a positive or negative response and a review of the features of the ship, similar to the flash cards mentioned earlier.

Classroom training in ship recognition provides a knowledgeable instructor to supply the student with comparison feedback for incorrect answers. For example the instructor can point out features that are present on the student's answer and show that they are not present on the ship in question. He can also show slides of an incorrect ship the student selected for an immediate visual comparison. But this is not often done due to the limited number of instructors available and the time required for individual instruction. An ICAI program can provide this direct comparison capability and provide one-on-one instruction for everyone.

D. ASSUMPTIONS

In writing the ship recognition tutor we assume that the student has some knowledge of ship features and weapon systems. The system requires that photos, line drawings or silhouettes be provided for the training syllabus. Appendix B provides information that will allow modifications to the system to include additional ships. The ships currently included in the system are listed along with the source code in Appendix C.

IV. SHIP RECOGNITION: AN ICAI APPROACH

A. SYSTEM ORGANIZATION

The ship recognition tutor was designed and implemented on an IBM-compatible microcomputer using the Arity Prolog programming language. It requires 512K of RAM memory. The following files are needed: *tdriver.ari, tmenu.ari, comparer.ari*[1], *ships.ari, tdescrip.ari, utilities.ari, summary.exe* and *getstime.exe*. Figure 4.1 illustrates the relationships between the program files. When the tutor is operated on a hard-drive system, the time to identify each ship is approximately one minute. The operating time is greatly dependent on the expertise of the student, since more time is required when incorrect responses are given. The operating time may increase slightly when using a floppy-drive system.

---

1. This program was started as a class project in CS4311 at the Naval Postgraduate School, Monterey, California.

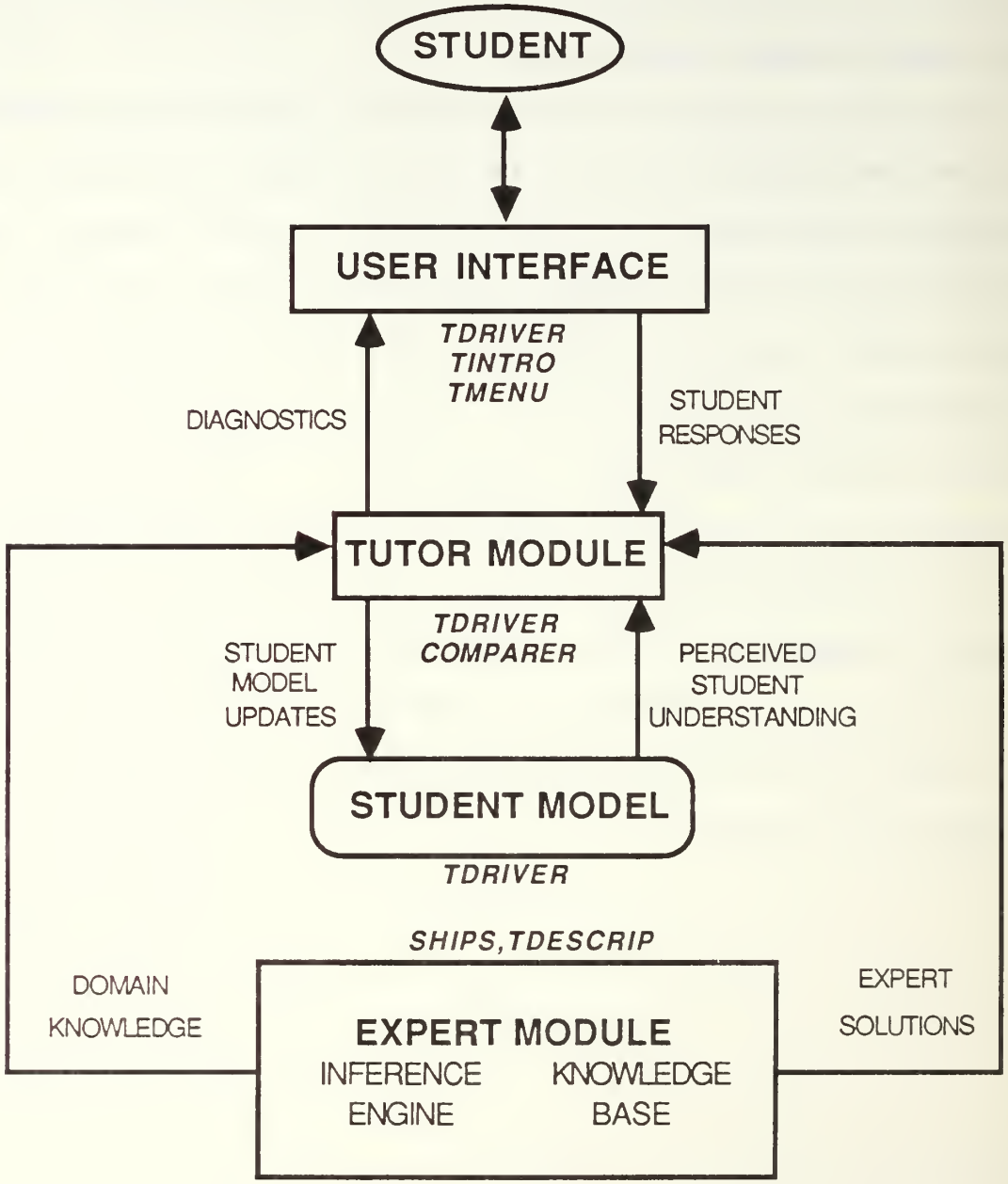

Figure 4.1 Ship Recognition Tutor Structural Model

B. EXPERT MODULE

1. Knowledge Base

The rules which identify ships are contained in the file *ships1.ari*. The information is contained in a series of predicates, that include a list of the identifying features of each ship. The format of this predicate is:

rule(ship_id(<'ship-name'>),[<feature-fact-list>])

The items in **<feature-fact-list>** have arguments that are brief descriptions of the features. This descriptive information is included to differentiate two features in the same general category with different overall appearances. Some examples of ship identification rules are displayed in Figure 4.2.

```
rule(ship_id('Kresta II CG'),
    [mast_2('two masts'),
    foremast('large obelisk type foremast'),
    aftermast('smaller pyramid aftermast'),
    radar('radar atop mack'),
    top_sail('TOP SAIL radar'),
    helo('raised helo pad aft'),
    ssm('canted launchers below bridge wings')]).
rule(ship_id('Kiev CVHG'),
    [fight_deck('angled flight deck'),
    superstructure('tiered superstructure'),
    other_features('looks like cruiser in profile')]).
rule(ship_id('Kirov CGN'),
    [mast_1('large mack amidship'),
    radar('atop mack'),
    top_sail(TOP SAIL radar'),
    superstructure('superstructure amidship with
     large mack'),
    forecastle('long, slightly stepped, sharply
     raked bow')]).
```

Figure 4.2 Sample Ship Identification Rules

2. Inference Engine

The ship recognition system uses comparison techniques as its inference engine. The student's response is compared with the known correct response obtained using the following predicate:

ship_photo_comb(<photo-number>,<ship-name>)

If the student's response is correct, the student is notified appropriately; if not, the system goes on to compare the correct response rule with the incorrect response rule.

This comparison is done in the file *comparer.ari* utilizing the predicate **compare**. The format for the comparison is as follows:

```
compare(<correct_answer>,<student_answer>,
  <Features_not_present>,<Features_missed>,
  <Features_details_diff>).ls2
```

<Features_not_present> is bound to a list of features that were present in the incorrect answer, but not in the correct answer. **<Features_missed>**, is bound to a list of the features contained in the correct ship's description but not in the description of the ship that the student chose. Finally, the predicate **<Features_details_diff>** contains a list of the features in the student's answer that differ only in the detailed descriptions (arguments) of the recognition features.

C. KNOWLEDGE ACQUISITION

The domain expertise for the ship recognition tutor was obtained from Department of Defense Recognition guides and from other recognition books [Refs. 2,3,4]. Although there was no direct consultation with experts, the knowledge obtained from published sources was deemed adequate for the development of this prototype tutor. An operational ship recognition tutor should be modified as applicable by the

instructors involved in the teaching syllabus. Information concerning modifications to this system is contained in Appendix B.

In reference guides for ship recognition, the features are listed along with a photo and a silhouette or a line drawing of the ship in question. In this tutoring system however, the features are listed for the student only in a final review of the ship. For incorrect answers the tutor addresses ship features individually and requires acknowledgement from the student.

D. TUTOR MODULE

Our tutor module is implemented in *tdriver.ari*. Tutor modules differentiate ICAI systems from CAI systems in that ICAI systems separate the tutoring strategies from the expert module. The tutor module is responsible for overseeing the student's progress and selecting the appropriate information to best instruct the student. The tutoring rules used in the ship recognition tutor are handled initially by the predicate **check_answer**. The student is asked to choose the correct ship from a menu of choices. The possible responses are:

- help. The tutor will give the correct answer and a review of the ship's features.
- quit. The tutor will quit, giving a summary of the session.

- correct response. The tutor will give an affirmative response along with a review of the ship's name and features.
- incorrect response. The tutor will give a negative response and will begin tutoring the student.

A major issue in deciding tutor strategies for this sort of system lies in the attempt to determine where the student went wrong. The ship recognition system tries to determine the student's misconceptions using the **compare** predicate to compare the student's ship to the correct ship. Once the features that appear to have caused the student difficulty are found, the next issue is which category of misconceptions is most important, or more likely to have caused the incorrect response.

We decided that the student was more likely to have problems with features in the same general category, features that were common to both the correct response and the incorrect response, but possessed different descriptive arguments. An example is shown in Figures 4.3 and 4.4. The ships are the Kresta I CG and the Kresta II CG, which are very similar. For example, both ships have SSM launchers under the bridge wings, but on the Kresta I the launchers are horizontal and on the Kresta II the launchers are canted. The **compare** predicate would return this information in the **Feature_details_diff** argument. Since a misconception of this type is very likely to occur, the student is first directed to these features.

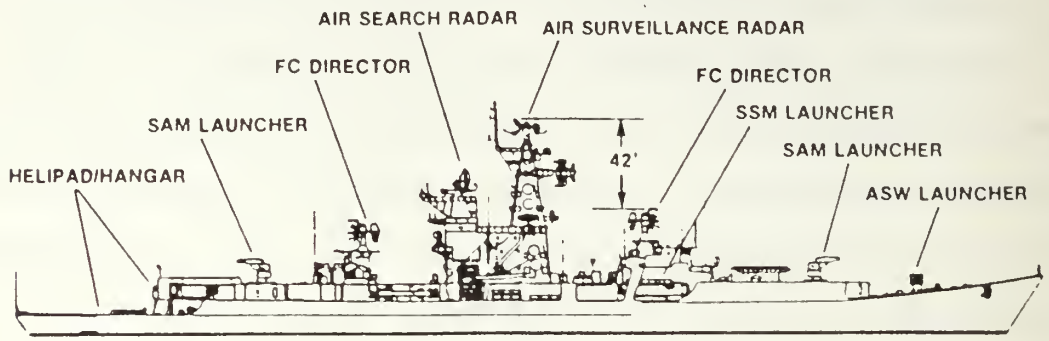

Figure 4.3 Kresta I CG

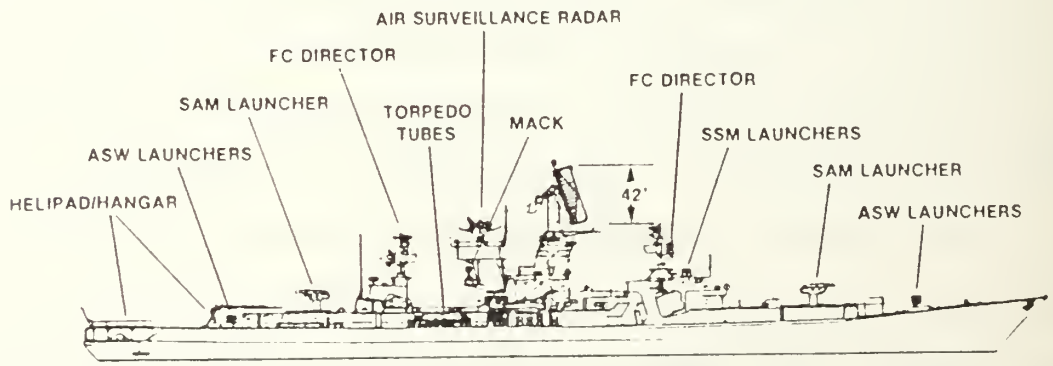

Figure 4.4 Kresta II CG

After any such student errors have been tutored, the student is next instructed on any features that are present in the correct ship, but not on the ship that he chose as the answer. These features are contained in **Features-missed**. Finally, the student is tutored on any **Features_not_present**, features that are present in the ship he chose but not in the correct ship.

The student is given three chances to choose the correct ship, his initial choice and two more. The second chance occurs following the entire tutoring sequence described above, and the third chance after another pass through the sequence. Each time he makes an incorrect choice the tutoring strategy involves the latest choice. After three chances, the system tells the student the correct answer and provides him with the identifying features of the correct ship.

When the student responds with an incorrect answer, the program first checks the length of time it took him to respond. The tutor utilizes the information the student entered concerning his experience level to determine a minimum length of time the student should view a photo prior to making a ship name selection. This is only relevant when the student's response is incorrect. It was arbitrarily determined that a Beginner should review an unfamiliar ship at least 30 seconds before attempting to make an identification, an Intermediate user ten seconds,

and no minimum time for an Expert user. Incorrect responses given prior to these times were considered "guesses".

Another important issue is that student misconceptions, or errors, fall into two major categories, perceptual and conceptual. When a student appears to see features that are not on the ship or miss features which are there, the error is perceptual; conceptual errors are knowledge misconceptions. When the student misidentifies a ship, our system first checks for perceptual errors by asking the student if the features identified by the comparison between the two ships are present or not. If the student still does not see a feature that is present on the ship, the system will ask him to look more closely and tell him that the feature is there. If the student thinks a feature is present that is not actually there, the system will tell the student that the feature is present on the incorrect ship but not on the correct ship.

E. STUDENT MODEL

The student model represents the student's knowledge of the problem, as perceived by the tutoring system. This representation is based on the student's responses to the questions asked by the system. The student model is only used at the conclusion of each user session, when the

system provides the student with a summary of his responses (Figure 4.5). This summary includes:

- names and number of ships identified on the first look
- names and number of ships identified on the second look
- names and number of ships identified on the third look
- names of ships that the student did not identify
- number of ships when the student appeared to guess the answer

There is no "grade" given; the information is provided to encourage the student to review areas of difficulty.

```
You identified 8 ship(s) on the first try.
The ships(s) were:
Jiang Hu I FF          Jiangdong FF             Kresta I CG
Kresta II CG           Grisha I/II/III FFL      Kaman PTG
Kara CG                Kiev CVHG

You identified 0 ship(s) on the second try.
The ships(s) were:

You identified 0 ship(s) on the third try.
The ships(s) were:

You did not identify the following ship(s):
Jiang Hu II FF

You appeared to guess on the first look at 0 ship(s).
```

Figure 4.5 Sample Summary Screen

F. USER INTERFACE

1. Input

A menu-based approach was used for user input to the tutor. The information contained in the menus is contained primarily in the file *tdescrip.ari*. The menu formats are contained in the file *tmenu.ari*.[2] Determination of the choice of menus to be used is made as part of the tutoring strategy in *tdriver.ari*.

The ship recognition tutor initially displays information concerning the tutoring system, the options available and reminders specific to the Prolog programming language. The user is next prompted for his experience level. The choices are Beginner, Intermediate and Expert; the system will use this information to determine if the user has responded to a query for the ship name with a guess. Following this information, the first menu appears (Figure 4.6) allowing the user to make one of four choices.

---

2. Menus are modifications of menu formats used in Ref. 12.

Ship Recognition Main Menu

1.Ship Recognition Test
2.Review a specific photo
3.View a summary
4.Quit

Enter 1.,2.,3. or 4.

Figure 4.6 Main Menu

If the user chooses the first option the test begins. The user is presented with a menu of ships from which to choose (Figure 4.7) and is provided with a book of photos or line drawings to identify. The system asks the student to look at randomly chosen photos. Following each incorrect answer the user is presented with a review of the significant features. The questions are in either a yes/no format (Figure 4.8) or a list of feature descriptions (Figure 4.9).

Your ship choices are:

1.Kresta I CG	2.Kresta II CG
3.Jiangdong FF	4.Jiang Hu I FF
5.Jiang Hu II FF	6.Godavari FF
7.Grisha I/II/III FFL	8.Kaman PTG
9.Kara CG	10.Kiev CVHG
11.Kirov CGN	12.Krivak I FFG
13.Krivak II FFG	14.Krivak III WPGF

Select the name that matches the photo or "q" to quit.

Figure 4.7 Sample List of Ships

Is the foremast with the following description:
large obelisk type foremast
present on photo number 3.

Enter "y." for yes or "n." for no.

Figure 4.8 Sample Yes/No Question

Select the best description of the onboard helo structures.
1.hangar and pad aft
2.helo pad aft
3.raised helo pad aft

Make your selection.

Figure 4.9 Sample List of Feature Descriptions

If the user selects the second system option, in the tutor main menu, to review specific photos, he is asked

the number of the photo he wishes to review. The user then sees the key features for the ship he selected.

The third system option shows the user a summary of his latest ship recognition test. Section E of this chapter describes the user summary.

2. Error Checking

The system will check for input that does not conform to acceptable standards. For example, if there are fifteen ship choices in the menu, the system will not allow the user to select "16" as the answer. The user will be asked to select an answer within the correct range of choices.

V. Results

Our ship recognition tutor was tested on a database of 20 Soviet ships, with the capability to add more ships as required. Test runs were completed by five different intermediate and beginning students, for a total of 25 runs. The system requires 50K of disk space and 368K of RAM memory not counting the Prolog interpreter. The system response time to incorrect student answers is one to three seconds. Tutoring time ranges from five seconds for correct answers to four minutes for incorrect responses. Tutoring times vary based on the individual user.

Demonstrations of the ship recognition tutor are in Appendix A. Instructions as to how to add more ship data to the knowledge base is in Appendix B.

VI. CONCLUSION

A. ACHIEVEMENTS

We have shown that an ICAI system for tutoring ship recognition is possible. This fully computerized system provides many of the benefits of an instructor. It is a portable program that will run on any IBM-compatible microcomputer. The source code is easily adaptable to teach most ships. Portability is the main weakness of the GUIDON tutor discussed in Chapter II. An ICAI system allows the instructor more flexibility than flash card methods in providing the student a useful learning environment. The combination of these two features makes it a worthwhile addition to any fleet training program.

B. LIMITATIONS

The ship-recognition tutor uses visual features such as weaponry and structural characteristics referenced in several ship recognition texts. These features, while important to the overall identification of ships, may not be the features required for quick recognition. Most experts in this field recognize ships based on one or two main features. The inclusion of "extra" features may only add to the new student's confusion.

A second possible limitation concerns the program's level of instruction. This level may be most applicable to intermediate students. More experienced students may find the method of tutoring too basic and therefore irritating. Beginning students will not know how to proceed when shown a ship. Nonetheless, the recognition test can still provide a good idea of any student's level of knowledge.

C. RECOMMENDED SYSTEM ENHANCEMENTS

There are several areas that can be enhanced to improve the ship-recognition tutor. The first is to provide more detail on the features used to identify the ships, as well as memory mnemonics to help the student remember key features. A means of specifying the locations of these features, perhaps using screen graphics, would also be helpful.

Background information is another area for improvement. The ease of identifying ships in the real world can be greatly improved when information is included concerning (1) geographical location of the sighting, (2) previous ship sightings in the area and (3) intelligence data obtained concerning political activities at the time of sighting.

A third addition to the system might be to provide the ship silhouettes as graphics on the computer screen. A drawback to this addition would be to make the overall

system less portable. Not all microcomputers are capable of producing graphics that are clear enough to allow the user to discern fine details.

Another addition would be to increase the knowledge base. This system was developed as a prototype, and therefore contains a limited number of ships. The knowledge base is limited to Soviet warships. Additional ships from other countries along with merchant ships would provide a better balance for the student.

There are some improvements that can be made to the instructional strategies. The menus could be presented in a hidden multiple choice method, where the student is given a multiple choice menu one item at a time. Another improvement would be to consider previous sessions when preparing the tutoring session. This would provide the student with more practice on the ships which cause him the most difficulty.

Some interface considerations might be to provide more flexible entry methods. The use of a mouse would be easier than keyboard entry for menu selections. Creating natural language input capability would also be helpful.

APPENDIX A - DEMONSTRATION

```
********** Next Screen **********

  SSSSS    HH     HH  II   PPPPPPP
SS     SS  HH     HH  II  PP     PP
SS         HH     HH  II  PP     PP
SS         HH     HH  II  PP     PP
  SSSSS    HHHHHHHHH  II  PPPPPPP
      SS   HH     HH  II  PP
      SS   HH     HH  II  PP
SS    SS   HH     HH  II  PP
  SSSSS    HH     HH  II  PP

RRRRRRR    EEEEEEEEE   CCCCCC    CCCCCC    OOOOOO
RR     RR  EE         CC    CC  CC    CC  OO    OO
RR     RR  EE         CC    CC  CC    CC  OO    OO
RR     RR  EE         CC        CC        OO    OO
RRRRRRR    EEEEEEE    CC        CC        OO    OO
RR RR      EE         CC        CC        OO    OO
RR  RR     EE         CC    CC  CC    CC  OO    OO
RR   RR    EE         CC    CC  CC    CC  OO    OO
RR     RR  EEEEEEEEE   CCCCCC    CCCCCC    OOOOOO

Strike a key when ready . . .
```

```
********** Next Screen **********

Ship recognition training is required for many Naval personnel.  It is taught by Intelligence personnel using slides, flash cards and other drill methods.


This ship recognition tutor is designed to help you improve your proficiency in ship recognition.  You will be provided with a list of ships from which to identify the photo specified.  The method of tutoring is based on your level of experience.  The system also provides you with a means of viewing a summary of your last session.  You may also review specific photos.

PROLOG REMINDER:

ALL ENTRIES MUST BE FOLLOWED BY A PERIOD (".")

Strike a key when ready . . .
********** Next Screen **********

                        Level of experience
                        -------------------
                           1. Beginner
                           2. Intermediate
                           3. Expert


Enter 1.,2., or 3.: 2.

********** Next Screen **********

                  Ship Recognition Main Menu

                    1. Ship recognition test
                    2. Review a specific photo
                    3. View a summary
                    4. Quit


Enter 1.,2.,3. or 4.1.
```

```
************************ Next Screen **********************

Please look at photo number 11
Your ship choices are:
    1.Kresta I CG                2.Kresta II CG
    3.Jiangdong FF               4.Jiang Hu I FF
    5.Jiang Hu II FF             6.Godavari FF
    7.Grisha I/II/III FFL        8.Kaman PTG
    9.Kara CG                   10.Kiev CVHG
   11.Kirov CGN                 12.Krivak I FFG
   13.Krivak II FFG             14.Krivak III WPGF
   15.Slava                     16.Kashin DDG
   17.Sovremennyy DDG           18.Yurka MSF
   19.Kynda CG                  20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. 13.

That answer is incorrect.
The Krivak II FFG is not the correct ship.

************************ Next Screen **********************

Select the best description of the gun mounts.
    1.single mount
    2.forward of superstructure, aft of deckhouse
    3.gunmounts flat and square
    4.gunmounts rounded with fence between
    5.enclosed gunmount on bow
    6.two enclosed gun mounts, fore and aft
    7.two enclosed gun mounts forward
    8.spheroid enclosed gun mount forward and aft

Make your selection: 5.

The gun description you chose:
enclosed gunmount on bow
is not present on a Krivak II FFG,
but it is present on the correct ship.

Type any letter to continue. c.

************************ Next Screen **********************

Is the helo with the following description:
hangar and pad aft
present on photo number 11.

Enter "y." for yes or "n." for no. y.
```

```
************************* Next Screen **********************

Is the ssm with the following description:
4 SSM launchers on bow
present on photo number 11.

Enter "y." for yes or "n." for no. n.

The ssm is present on the Krivak II FFG, but not on the
correct ship.

Type any letter to continue. c.

************************* Next Screen **********************

You have identified the following features:

fact(gun('enclosed gunmount on bow')).
fact(helo('hangar and pad aft')).

Try again to identify photo number 11
Remember, it is not the Krivak II FFG
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG

Select the name that matches the photo,"q" to quit or "h"
for help. 14.

************************* Next Screen **********************

Very good! You have chosen the correct ship.

Photo number 11 is the Krivak III WPGF
Here is a reminder of the key identifying features.

mast_2(two masts)
masts(latticed)
superstructure(superstructure with masts)
stack(short, wide with lip on trailing edge)
gun(enclosed gunmount on bow)
helo(hangar and pad aft)
```

```
Type any letter to continue. c.
************************* Next Screen **********************

Please look at photo number 20
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. 15.

************************* Next Screen **********************

Very good! You have chosen the correct ship.

Photo number 20 is the Slava

Here is a reminder of the key identifying features.

mast_2(two masts)
foremast(large pyramid mast structure)
aftermast(smaller radar mast)
stack(twin stacks amidship)
radar(Top Dome on after deckhouse)
helo(hangar and pad aft)
missiles(16 SS-N-12 Sandbox missiles paired forward)

Type any letter to continue. c.
```

```
************************* Next Screen **********************

Please look at photo number 9
Your ship choices are:
    1.Kresta I CG                2.Kresta II CG
    3.Jiangdong FF               4.Jiang Hu I FF
    5.Jiang Hu II FF             6.Godavari FF
    7.Grisha I/II/III FFL        8.Kaman PTG
    9.Kara CG                   10.Kiev CVHG
   11.Kirov CGN                 12.Krivak I FFG
   13.Krivak II FFG             14.Krivak III WPGF
   15.Slava                     16.Kashin DDG
   17.Sovremennyy DDG           18.Yurka MSF
   19.Kynda CG                  20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. 7.

************************* Next Screen **********************

Very good! You have chosen the correct ship.

Photo number 9 is the Grisha I/II/III FFL

Here is a reminder of the key identifying features.

mast_1(heavy pylon mast with latticed extensions)
mast_1_loc(on top of the bridge)
stack(located aft)
rocket_launcher(RBU launcher on a raised mid section)

Type any letter to continue. c.

************************* Next Screen **********************

Please look at photo number 6
Your ship choices are:
    1.Kresta I CG                2.Kresta II CG
    3.Jiangdong FF               4.Jiang Hu I FF
    5.Jiang Hu II FF             6.Godavari FF
    7.Grisha I/II/III FFL        8.Kaman PTG
    9.Kara CG                   10.Kiev CVHG
   11.Kirov CGN                 12.Krivak I FFG
   13.Krivak II FFG             14.Krivak III WPGF
   15.Slava                     16.Kashin DDG
   17.Sovremennyy DDG           18.Yurka MSF
   19.Kynda CG                  20.Udaloy DDG
```

```
Select the name that matches the photo,"q" to quit or "h"
for help. 4.


That answer is incorrect.
The Jiang Hu I FF is not the correct ship.

************************ Next Screen **********************

Is the mast_1 with the following description:
latticed tripod mast
present on photo number 6.

Enter "y." for yes or "n." for no. y.


************************ Next Screen **********************

Is the stack with the following description:
large, square
present on photo number 6.

Enter "y." for yes or "n." for no. y.

************************ Next Screen **********************

Is the ssm with the following description:
twin launchers forward and aft of stack
present on photo number 6.

Enter "y." for yes or "n." for no. y.

************************ Next Screen **********************

Is the ssm_2 with the following description:
a SSM 1 launcher forward
present on photo number 6.

Enter "y." for yes or "n." for no. y.

************************ Next Screen **********************

Is the bridge with the following description:
forward placed
present on photo number 6.

Enter "y." for yes or "n." for no. y.
```

```
************************* Next Screen **********************

Is the helo with the following description:
helo pad aft
present on photo number 6.

Enter "y." for yes or "n." for no. y.

************************* Next Screen **********************

You have identified the following features:

fact(mast_1('latticed tripod mast')).
fact(stack('large, square')).
fact(ssm('twin launchers forward and aft of stack')).
fact(ssm_2('a SSM 1 launcher forward')).
fact(bridge('forward placed')).
fact(helo('helo pad aft')).

Try again to identify photo number 6
Remember, it is not the Jiang Hu I FF
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG

Select the name that matches the photo,"q" to quit or "h"
for help. 5.

************************* Next Screen **********************

Very good! You have chosen the correct ship.

Photo number 6 is the Jiang Hu II FF
Here is a reminder of the key identifying features.

mast_1(latticed tripod mast)
stack(large, square)
ssm(twin launchers forward and aft of stack)
ssm_2(a SSM 1 launcher forward)
bridge(forward placed)
helo(helo pad aft)

Type any letter to continue. c.
```

```
************************* Next Screen **********************

Please look at photo number 7
Your ship choices are:
    1.Kresta I CG                2.Kresta II CG
    3.Jiangdong FF               4.Jiang Hu I FF
    5.Jiang Hu II FF             6.Godavari FF
    7.Grisha I/II/III FFL        8.Kaman PTG
    9.Kara CG                   10.Kiev CVHG
   11.Kirov CGN                 12.Krivak I FFG
   13.Krivak II FFG             14.Krivak III WPGF
   15.Slava                     16.Kashin DDG
   17.Sovremennyy DDG           18.Yurka MSF
   19.Kynda CG                  20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. 9.



That answer is incorrect.
The Kara CG is not the correct ship.

************************* Next Screen **********************

Select the best description of the mast.
    1.heavy pylon mast with latticed extensions
    2.latticed tripod mast
    3.large radome atop mast
    4.large pyramid mast, supports TOP SAIL radar
    5.large mack amidship with TOP SAIL radar
    6.single oil derrick mast
    7.latticed

Make your selection: 3.

The mast_1 description you chose:
large radome atop mast
is not present on a Kara CG,
but it is present on the correct ship.

Type any letter to continue. c.
```

```
************************ Next Screen **********************

Select the best description of the SSM launchers(s).
    1.4 SSM launchers forward of the superstructure
    2.twin launchers forward and aft of stack
    3.located between superstructure and deckhouse
    4.4 SSM launchers located on bow
    5.canted launchers below bridge wings
    6.horizontal launchers under bridge wings
    7.two (HOT DOG PACKS), 1 forward, 1 aft

Make your selection: 3.

The ssm description you chose:
located between superstructure and deckhouse
is not present on a Kara CG,
but it is present on the correct ship.

Type any letter to continue. c.

************************ Next Screen **********************

Is the gun with the following description:
forward of superstructure
present on photo number 7.

Enter "y." for yes or "n." for no. n.


The gun is present in the photo.  Look carefully to identify
it.  This feature is not present on a Kara CG

Type any letter to continue. c.

************************ Next Screen **********************

Is the gun_2 with the following description:
aft of deckhouse
present on photo number 7.

Enter "y." for yes or "n." for no. y.

************************ Next Screen **********************

Is the superstructure with the following description:
enclosed,streamlined with separated deckhouse
present on photo number 7.

Enter "y." for yes or "n." for no. y.
```

```
************************* Next Screen **********************

Is the stack with the following description:
large, square
present on photo number 7.

Enter "y." for yes or "n." for no. n.

The stack is present on the Kara CG, but not on the correct
ship.

Type any letter to continue. c.

************************* Next Screen **********************

Is the helo with the following description:
helo pad aft
present on photo number 7.

Enter "y." for yes or "n." for no. n.

The helo is present on the Kara CG, but not on the correct
ship.
Type any letter to continue. c.

************************* Next Screen **********************

You have identified the following features:

fact(mast_1('large radome atop mast')).
fact(ssm('located between superstructure and deckhouse')).
fact(gun('forward of superstructure')).
fact(gun_2('aft of deckhouse')).
fact(superstructure('enclosed,streamlined with separated
deckhouse')).

Try again to identify photo number 7
Remember, it is not the Kara CG
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG
```

```
Select the name that matches the photo,"q" to quit or "h"
for help. 8.
************************* Next Screen **********************

Very good! You have chosen the correct ship.

Photo number 7 is the Kaman PTG
Here is a reminder of the key identifying features.

mast_1(large radome atop mast)
gun(forward of superstructure)
gun_2(aft of deckhouse)
ssm(located between superstructure and deckhouse)
superstructure(enclosed,streamlined with separated deck-
house)

Type any letter to continue. c.

************************* Next Screen **********************

Please look at photo number 8
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. h.

************************* Next Screen **********************

The name of the correct ship is Kara CG

This is a list of the descriptive features of a Kara CG:

mast_1(large pyramid mast, supports TOP SAIL radar)
stack(large, square)
helo(helo pad aft)
ssm(canted launchers below bridge wings)
```

```
Identify each feature in the photo.

Remember the name of the ship is Kara CG

Type any letter key to continue. c.
************************* Next Screen **********************

Please look at photo number 1
Your ship choices are:
    1.Kresta I CG              2.Kresta II CG
    3.Jiangdong FF             4.Jiang Hu I FF
    5.Jiang Hu II FF           6.Godavari FF
    7.Grisha I/II/III FFL      8.Kaman PTG
    9.Kara CG                 10.Kiev CVHG
   11.Kirov CGN               12.Krivak I FFG
   13.Krivak II FFG           14.Krivak III WPGF
   15.Slava                   16.Kashin DDG
   17.Sovremennyy DDG         18.Yurka MSF
   19.Kynda CG                20.Udaloy DDG


Select the name that matches the photo,"q" to quit or "h"
for help. q.

************************* Next Screen **********************

You identified 2 ship(s) on the first try.  The ship(s)
were:
Slava                      Grisha I/II/III FFL

You identified 3 ship(s) on the second try.  The ship(s)
were:
Krivak III WPGF            Jiang Hu II FF            Kaman PTG

You identified 0 ship(s) on the third try.  The ship(s)
were:

You did not identify the following ship(s):
Kara CG

You appeared to guess on the first look at 1 ship(s).

Type any letter key to continue. c.

************************* Next Screen **********************

Please enter an 8 character filename for your summary.
The filename must begin with a letter and contain only
letters and numbers.  Remember this filename to
retrieve your summary at your next user session.  mysum.
```

Your filename is mysum

Your summary has been saved to the file mysum.
Remember this filename to review this summary.

Type any letter key to continue. c.
************************* Next Screen **********************

Ship Recognition Main Menu

1. Ship recognition test
2. Review a specific photo
3. View a summary
4. Quit

Enter 1.,2.,3. or 4.2.

************************* Next Screen **********************

Enter the number of the photo you wish to review followed by a period. 15.

************************* Next Screen **********************

The name of the ship in photo number 15 is Sovremennyy DDG

This is a list of the descriptive features of a Sovremennyy DDG:

bridge(radome atop bridge)
gun(spheroid enclosed gun mount foward and aft)
ssm(canted launchers under bridge wings)
stack(single stack amidship)
helo(pad and telescoping hangar aft of stack)
radar(air surveillance radar atop mack)

Identify each feature in the photo.

Remember the name of the ship is Sovremennyy DDG

Type any letter key to continue. c.

```
************************* Next Screen ***********************

                  Ship Recognition Main Menu

                    1. Ship recognition test
                    2. Review a specific photo
                    3. View a summary
                    4. Quit


Enter 1.,2.,3. or 4.3.

What is the name of your summary file?
mysum.
************************* Next Screen ***********************

You identified 2 ship(s) on the first try.  The ship(s)
were:
Slava                     Grisha I/II/III FFL

You identified 3 ship(s) on the second try.  The ship(s)
were:
Krivak III WPGF           Jiang Hu II FF            Kaman PTG

You identified 0 ship(s) on the third try.  The ship(s)
were:

You did not identify the following ship(s):
Kara CG

You appeared to guess on the first look at 1 ship(s).


Type any letter key to continue. c.

************************* Next Screen ***********************

                  Ship Recognition Main Menu

                    1. Ship recognition test
                    2. Review a specific photo
                    3. View a summary
                    4. Quit


Enter 1.,2.,3. or 4.4.
```

APPENDIX B - USER'S MANUAL

A. USING THE PROGRAM

The ship recognition tutor is implemented in the Arity Prolog programming language. It requires approximately 50K of disk space and 512K of RAM memory. Once you have loaded Arity Prolog, and started the interpreter, you will be presented with a prompt:

```
?-
```

Type "[loadfile]." to begin. **REMEMBER: PROLOG REQUIRES A PERIOD (".") FOLLOWING ALL INPUT.**

```
?-[loadfile].
```

The tutoring system will take several minutes to load, please be patient. When the system responds with:

```
yes
?-
```

type "gotutor."

```
?-gotutor.
```

You will be presented with menus from this point. To quit the system you may type "q." at the ship choice menu or "4." at the main menu.

1. Prolog Reminders

1. All input must be followed by a period (".").

2. Letter entries, such as "q" or "h", must be made in lower case.

3. Typing extra characters after the period and before the carriage return will cause the system to fail. No summary will be made and the program will halt. You may type "gotutor." at the ?- prompt to restart the program.

B. MODIFICATIONS AND ADDITIONS

The expert module of the ship recognition tutor is easily modifiable, in order to add or delete ships, or to change identifying features of the ships already in the system. When modifying the system several things need to be kept in mind:

1. Before making any changes be sure to make backup copies of all files.

2. Whenever a change is made to the *ships.ari* file, changes must also be made to the *tdescrip.ari* file and vice versa.

3. Descriptive features, ship names, and predicate names must match exactly whenever they are used.

The following is an example of the changes that need to be made when adding a ship to the system:

1. Add a *rule* predicate:

```
rule(ship_id('Grisha I/II/III FFL'),
  [mast_1('heavy pylon mast with latticed
     extensions'),
  mast_1_loc('on top of the bridge'),
  stack('located aft'),
  rocket_launcher('RBU launcher on a raised mid
      section')]).
```

Use predicate names such as *stack*, which are already defined when possible. The same goes for feature descriptions such as, *'located aft'*. Remember they should be character-for-character the same as they appear anywhere else in the program.

2. Add a *ship_photo_comb* predicate:

```
ship_photo_comb(9,'Grisha I/II/III FFL').
```

Be sure the photo number you assign is not already assigned and that the ship name is typed exactly as it appears in the *rule* predicate and the *ship_names* predicate. The numbers must be sequential.

3. Add the name of the ship to the *ship_names* predicate. The ship name should be character-for-character the same as it appears elsewhere.

```
ship_names('Your ship choices are: ',
  ['Kresta I CG','Kresta II CG','Jiangdong FF',
  'Jiang Hu I FF','Jiang Hu II FF','Godavari FF',
  'Grisha I/II/III FFL','Kaman PTG','Kara CG',
  'Kiev CVHG','Kirov CGN','Krivak I FFG',
  'Krivak II FFG','Krivak III WPGF','Slava',
  'Kashin DDG','Sovremennyy DDG','Yurka MSF',
  'Kynda CG','Udaloy DDG']).
```

The order of the ship names, as they appear in this predicate, is the order of ship names in the choices menu.

4. Increase the number of total photos in the predicate *total_photos*.

```
total_photos(20).
```

5. Modify or add a *descriptors* predicate for every feature which is added in the *rule* predicate.

```
descriptors(stack,
  'Select the best description of the stack
       structure.',
  ['located aft',
  'large, square',
  'short, wide with lip on trailing edge',
  '4 in symmetry to the mast, canted out',
  '4 stacks in pairs amidships',
  'oval shaped stack (YURKED TO THE SIDE)',
  'twin stacks amidship',
  'single stack amidship']).
```

For example, the predicate *stack* was added in the *rule* predicate above and the description was *'located aft'*. The exact same descriptive phrase is added to the *descriptors* predicate for stack. If there is already a *descriptors* predicate which contains the correct phrase, no change needs to be made.

Steps 1-4 concern changes made to the file *ships.ari*, step 5 concerns changes made to the file *tdescrip.ari*.

APPENDIX C - SOURCE CODE

```
/* tdriver.ari */

This file contains the tutoring portion of the ship recog-
nition tutor.  It uses the files tmenu.ari, tdescrip.ari,
ships.ari, utilities.ari and intro.ari.  The file is loaded
by loadfile.ari.  Two executable files, summary.exe and
getstime.exe, are also needed.  They were created using
Turbo Pascal.

No modifications to this file are necessary, in order to
update or add to the database.

/* Initializing variables */

first_try(0,[]).
second_try(0,[]).
third_try(0,[]).
misses(0,[]).
num_guesses(0).
number_seen(0).


gotutor :-
        abolish(stop_marker,0),
        abolish(quit_system,0),
        intro_screen,
        shell(cls),
        experience_level,
        choose_option.

choose_option :- quit_system, !.
choose_option :- main_menu, start_tutor, choose_option.

/* Direct the system to the user chosen option. */

start_tutor :- system_option(test),
        abolish(viewed_photo,1),
        assert(viewed_photo(0)),
        initialize_seed,
        start_test, abolish(stop_marker,0), !.
```

```
start_tutor :- system_option(photo_review),
        shell(cls), blines(10),
        write('Enter the number of the photo you wish to
            review '),
        write('followed by a period. '),
        read(Number),
        ship_photo_comb(Number,Correct_answer),
        rule(ship_id(Correct_answer),Correct_answer_list),
        shell(cls), blines(4),
        nl, write('The name of the ship in photo number '),
        write(Number), write(' is '),
        write(Correct_answer), nl, nl,
        write('This is a list of the descriptive features
            of a '),
        write(Correct_answer), write(':'), nl, nl,
        prettyprint1(Correct_answer_list), nl, nl,
        write('Identify each feature in the photo.'), nl,
        nl,write('Remember the name of the ship is '),
        write(Correct_answer), nl, nl,
        write('Type any letter key to continue. '),
        read(Anykey), nl, !.
start_tutor :- system_option(view_summary),
        shell(summary), nl, nl, nl,
        write('Type any letter key to continue. '),
        read(Anykey), nl, !.
start_tutor :- system_option(quit),
        assert(quit_system).


/* Begin the ship recognition test. */

start_test :- stop_marker, conclusion, !.

start_test :- number_seen(Number),
        total_photos(X),
        Number >= X, conclusion, !.

start_test :-
        clear_vars,
        shell(cls), blines(5),
        get_photo_num,
        photo_num(Number),
        write('Please look at photo number '),
        write(Number), nl,
        ship_names(Header,Names),
        write(Header), nl,
        shipmenu(Names),nl, nl,
        write('Select the name that matches the photo,'),
        write('"q" to quit or "h" for help. '),
        shell(getsthetime), consult(timefile), time(A,B),
        retract(time(A,B)),
```

```
        ship_ask_which(Names), nl,
        shell(getsthetime), consult(timefile), time(C,D),
        retract(time(C,D)),
        timecomp(A,B,C,D,Total),
        assert(time_to_look(Total)),
        ship_fact(Student_answer),
        abolish(ship_fact,1),
        check_answer(Student_answer,Number),
        retract(number_seen(N)),
        N1 is N + 1,
        assert(number_seen(N1)),
        start_test.


/* Randomly select the photos to be presented. */

initialize_seed :- shell(getsthetime),
        consult(timefile), retract(time(A,B)),
        assert(seed(B)).

get_photo_num :-
        total_photos(R),
        check_number(R,N), !.

check_number(R,N) :-
        not(viewed_photo(N)),
        assert(viewed_photo(N)),
        assert(photo_num(N)), !.

check_number(R,N) :-
        random(R,N1),
        check_number(R,N1), !.

random(R,N) :-
        retract(seed(S)),
        N is (S mod R) + 1,
        NewSeed is (25 * S + 1) mod 1096,
        asserta(seed(NewSeed)), !.

/* Check the student's first answer and respond appropri-
ately. */

check_answer('quit',Photo_num) :- assert(stop_marker), !.

check_answer('help',Photo_num) :-
        ship_photo_comb(Photo_num,Correct_answer),
        rule(ship_id(Correct_answer),Correct_answer_list),
        shell(cls), blines(4),
        nl, write('The name of the correct ship is '),
        write(Correct_answer), nl, nl,
```

```
        write('This is a list of the descriptive features
            of a '),
        write(Correct_answer), write(':'), nl, nl,
        prettyprint1(Correct_answer_list), nl, nl,
        write('Identify each feature in the photo.'), nl,
        nl,write('Remember the name of the ship is '),
        write(Correct_answer), nl, nl,
        maintain_score(4,Correct_answer),
        write('Type any letter key to continue. '),
        read(Anykey), nl, !.

check_answer(Student_answer,Photo_num) :-
        ship_photo_comb(Photo_num,Student_answer),
        shell(cls), blines(4),
        write('Very good! You have chosen the correct
            ship.'), nl, nl,
        write('Photo number '), write(Photo_num),
        write(' is the '),
        write(Student_answer), nl, nl,
        write('Here is a reminder of the key identifying
            features.'), nl,
        nl, rule(ship_id(Student_answer),R),
        prettyprint1(R), nl,
        numtries(X),
        maintain_score(X,Student_answer),
        write('Type any letter to continue. '),
        read(Anykey), nl, nl, !.

/* Beginner guess */

check_answer(Student_answer,Photo_num) :-
        user_level(beginner), time_to_look(X),
        X < 30,
        ship_photo_comb(Photo_num,Correct_answer),
        maintain_guesses,
        write('That answer is incorrect.'), nl,
        write('The '), write(Student_answer),
        write(' is not the correct ship.'), nl, nl,
        rule(ship_id(Correct_answer),Correct_answer_list),
        compare(Correct_answer_list,[],Features_not_present,
                Features_missed,Feature_details_diff), nl,
        nl,
        review_features(Correct_answer,Correct_answer_list,
           Student_answer,Feature_details_diff,
           Features_missed,Features_not_present), !.
```

```
/* Intermediate user guess */

check_answer(Student_answer,Photo_num) :-
        user_level(intermediate), time_to_look(X),
        X < 10,
        ship_photo_comb(Photo_num,Correct_answer),
        maintain_guesses,
        write('That answer is incorrect.'), nl,
        write('The '), write(Student_answer),
        write(' is not the correct ship.'), nl, nl,
        rule(ship_id(Correct_answer),Correct_answer_list),
        compare(Correct_answer_list,[],Features_not_present,
                Features_missed,Feature_details_diff), nl,
        nl,
        review_features(Correct_answer,Correct_answer_list,
           Student_answer,Feature_details_diff,
           Features_missed,Features_not_present), !.
/* Non-guess */

check_answer(Student_answer,Photo_num) :-
        ship_photo_comb(Photo_num,Correct_answer),
        write('That answer is incorrect.'), nl,
        write('The '), write(Student_answer),
        write(' is not the correct ship.'), nl, nl,
        rule(ship_id(Student_answer),Student_answer_list),
        rule(ship_id(Correct_answer),Correct_answer_list),
        compare(Correct_answer_list,Student_answer_list,
                Features_not_present,Features_missed,
                Feature_details_diff),
        nl,nl,
        review_features(Correct_answer,Correct_answer_list,
           Student_answer,Feature_details_diff,
           Features_missed,Features_not_present), !.

/* Begin review of features after incorrect response */

review_features(Correct_answer,Correct_answer_list,
    Student_answer,Feature_details_diff,Features_missed,
    Features_not_present) :- stop_review, !.

review_features(Correct_answer,Correct_answer_list,
    Student_answer,Feature_details_diff,Features_missed,
    Features_not_present) :- stop_marker, !.

review_features(Correct_answer,Correct_answer_list,
    Student_answer,Feature_details_diff,Features_missed,
    Features_not_present) :-
        numtries(X), X<3,
        review_diffs(Feature_details_diff,Student_answer,
                Correct_answer_list), nl,
        review_misses(Features_missed,Student_answer), nl,
```

```
        review_not_present(Features_not_present,
            Student_answer),
        ask_ship(Correct_answer,Student_answer,New_answer),
        rule(ship_id(New_answer),New_answer_list),
        compare(Correct_answer_list,New_answer_list,
            Features_not_present2,Features_missed2,
            Feature_details_diff2),
        review_features(Correct_answer,Correct_answer_list,
            New_answer,Feature_details_diff2,
            Features_missed2,Features_not_present2), !.
review_features(Correct_answer,Correct_answer_list,
    Student_answer,Feature_details_diff,Features_missed,
    Features_not_present) :-
        still_no_answer(Correct_answer,Correct_answer_list),
        nl, !.

review_diffs([],Student_answer,Correct_answer_list) :-  !.

review_diffs([Detail|Features],Student_answer,
    Correct_answer_list) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        fact(Check_detail),
        review_diffs(Features,Student_answer,
            Correct_answer_list), !.

review_diffs([Detail|Features],Student_answer,
    Correct_answer_list) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        incorrect_fact(Check_detail),
        review_diffs(Features,Student_answer,
            Correct_answer_list), !.

review_diffs([Detail|Features],Student_answer,
    Correct_answer_list) :-
        Detail =.. [Detail_feature,Detail_descrip],
        descriptors(Detail_feature,Listheader,Descrip_list),
        shell(cls), blines(5),
        write(Listheader),nl,
        writemenu(Descrip_list),nl,
        write('Make your selection: '),
        descrip_ask_which(Descrip_list,Item),
        Selected_item =.. [Detail_feature,Item],
        member(Selected_item,Correct_answer_list),
        assertz(fact(Selected_item)),
        nl, write('The '), write(Detail_feature),
        write(' description you chose: '), nl,
        write(Item), nl, write('is not present on a '),
        write(Student_answer), write(', '), nl,
        write('but it is present on the correct ship.'),
```

```
        nl,
        nl, write('Type any letter to continue. '),
        read(Anyletter),
        review_diffs(Features,Student_answer,
            Correct_answer_list), nl, !.

review_diffs([Detail|Features],Student_answer,
    Correct_answer_list) :-
        nl, write('That is not the best description'), nl,
        review_diffs([Detail|Features],Student_answer,
                Correct_answer_list), nl, !.
review_misses([],Student_answer) :-  !.

review_misses([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        fact(Check_detail),
        review_misses(Features,Student_answer), !.

review_misses([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        incorrect_fact(Check_detail),
        review_misses(Features,Student_answer), !.

review_misses([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        photo_num(Number), shell(cls), blines(5),
        write('Is the '), write(Detail_feature),
        write(' with the following description:'), nl,
        write(Detail_descrip), nl,
        write('present on photo number '), write(Number),
        write('.'), nl,
        nl, write('Enter "y." for yes or "n." for no. '),
        read(Answer),
        interp_miss_answer(Answer,Detail_feature,
           Student_answer),
        assertz(fact(Detail)),
        review_misses(Features,Student_answer), nl, !.


interp_miss_answer(Answer,Detail_feature,Student_answer) :-
        name(Answer,[121]), nl, !.

interp_miss_answer(Answer,Detail_feature,Student_answer) :-
        name(Answer,[110]),
        nl, nl, write('The '), write(Detail_feature),
        write(' is present in the photo.  Look carefully to
            identify it.'),
        nl, write('This feature is not present on a '),
        write(Student_answer),
```

```
        nl, nl, write('Type any letter to continue. '),
        read(Anyletter),
        nl, !.

interp_miss_answer(Answer,Detail_feature,Student_answer) :-
        nl, write('Your input is not a "y" or a "n".
               Please reenter: '),
        read(New_entry),
        interp_miss_answer(New_entry,Detail_feature,
            Student_answer), !.


review_not_present([],Student_answer) :-  !.

review_not_present([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        fact(Check_detail),
        review_not_present(Features,Student_answer), !.

review_not_present([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        Check_detail =.. [Detail_feature,Other_descrip],
        incorrect_fact(Check_detail),
        review_not_present(Features,Student_answer), !.

review_not_present([Detail|Features],Student_answer) :-
        Detail =.. [Detail_feature,Detail_descrip],
        photo_num(Number), shell(cls), blines(5),
        write('Is the '), write(Detail_feature),
        write(' with the following description:'), nl,
        write(Detail_descrip), nl,
        write('present on photo number '), write(Number),
        write('.'),
        nl, nl,
        write('Enter "y." for yes or "n." for no. '),
        read(Answer),
        interp_not_present_answer(Answer,Detail_feature,
            Student_answer),
        assertz(incorrect_fact(Detail)),
        review_not_present(Features,Student_answer), nl, !.

interp_not_present_answer(Answer,Detail_feature,
    Student_answer) :-
        name(Answer,[121]),
        nl, write('The '), write(Detail_feature),
        write(' is not present in the photo.
            Please look more carefully.'),
        nl, nl, write('It is present on the '),
        write(Student_answer),
        write('.'), nl, nl,
```

```
        write('Type any letter to continue. '),
        read(Anyletter), nl, !.

interp_not_present_answer(Answer,Detail_feature,
    Student_answer) :-
        name(Answer,[110]),
        nl, write('The '), write(Detail_feature),
        write(' is present on the '),
        write(Student_answer),
        write(', but not on the correct ship.'),
        nl, nl, write('Type any letter to continue. '),
        read(Anyletter), nl, !.

interp_not_present_answer(Answer,Detail_feature,
    Student_answer) :-
        nl, write('Your input is not a "y" or a "n".
                Please reenter: '),
        read(New_entry),
        interp_not_present_answer(New_entry,Detail_feature,
            Student_answer), !.



/* Request user to try again to identify the correct
   ship. */

ask_ship(Correct_answer,Student_answer,New_answer)  :-
        shell(cls), blines(5),
        write('You have identified the following
            features:'), nl, nl,
        listing(fact), nl,
        photo_num(Number),
        write('Try again to identify photo number '),
        write(Number), nl,
        write('Remember, it is not the '),
        write(Student_answer), nl,
        ship_names(Header,Names),
        write(Header), nl,
        shipmenu(Names),nl,
        write('Select the name that matches the photo,'),
        write('"q" to quit or "h" for help. '),
        ship_ask_which(Names), nl,
        ship_fact(New_answer),
        abolish(ship_fact,1),
        numtries(X),
        abolish(numtries,1),
        X1 is X + 1,
        assert(numtries(X1)),
        check_answer2(New_answer,Number), !.
```

```
/* Check users response for second and third tries at iden
tification. */

check_answer2('quit',Photo_num) :- assert(stop_marker), !.

check_answer2('help',Photo_num) :-
        ship_photo_comb(Photo_num,Correct_answer),
        rule(ship_id(Correct_answer),Correct_answer_list),
        shell(cls), blines(5),
        nl, write('The name of the correct ship is '),
        write(Correct_answer), nl, nl,
        write('This is a list of the descriptive features
            of a '),
        write(Correct_answer), write(':'), nl, nl,
        prettyprint1(Correct_answer_list), nl, nl,
        write('Identify each feature in the photo.'), nl,
        nl,write('Remember the name of the ship is '),
        write(Correct_answer), nl, nl,
        maintain_score(4,Correct_answer),
        write('Type any letter key to continue. '),
        read(Anykey),
        assert(stop_review), nl, !.


check_answer2(New_answer,Photo_num) :-
        ship_photo_comb(Photo_num,New_answer),
        shell(cls), blines(5),
        write('Very good! You have chosen the correct
            ship.'), nl, nl,
        write('Photo number '), write(Photo_num),
        write(' is the '),
        write(New_answer), nl,
        write('Here is a reminder of the key identifying
            features.'), nl,
        nl, rule(ship_id(New_answer),R), prettyprint1(R),
        nl, numtries(X),
        maintain_score(X,New_answer),
        write('Type any letter to continue. '),
        read(Anykey), nl, nl,
        assert(stop_review), !.

check_answer2(New_answer,Photo_num) :-
        ship_photo_comb(Photo_num,Correct_answer),
        write('That answer is incorrect.'),
        write('  The '), write(New_answer),
        write(' is not the correct ship.'), nl, !.
```

```
/* After third incorrect response. */

still_no_answer(Correct_answer,Correct_answer_list) :-
        stop_marker, !.

still_no_answer(Correct_answer,Correct_answer_list) :-
        stop_review, !.

still_no_answer(Correct_answer,Correct_answer_list) :-
        shell(cls), blines(5),
        write('The name of the correct ship is '),
        write(Correct_answer), nl, nl,
        write('This is a list of the descriptive features
            of a '),
        write(Correct_answer), write(':'), nl, nl,
        prettyprint1(Correct_answer_list), nl, nl,
        write('Identify each feature in the photo.'), nl,
        nl, write('Remember the name of the ship is the '),
        write(Correct_answer), nl, nl,
        maintain_score(4,Correct_answer),
        write('Type "c." to continue. '),
        read(Anykey), nl, !.


/* Keep summary of student's responses. */

maintain_score(X,Student_answer) :-
        X == 1,
        first_try(N,Tries),
        N1 is N + 1,
        abolish(first_try,2),
        append(Tries,[Student_answer],Tries2),
        assert(first_try(N1,Tries2)), !.

maintain_score(X,Student_answer) :-
        X == 2,
        second_try(N,Tries),
        N1 is N + 1,
        abolish(second_try,2),
        append(Tries,[Student_answer],Tries2),
        assert(second_try(N1,Tries2)), !.

maintain_score(X,Student_answer) :-
        X == 3,
        third_try(N,Tries),
        N1 is N + 1,
        abolish(third_try,2),
        append(Tries,[Student_answer],Tries2),
        assert(third_try(N1,Tries2)), !.
```

```
maintain_score(X,Student_answer) :-
        misses(N,Tries),
        N1 is N + 1,
        abolish(misses,2),
        append(Tries,[Student_answer],Tries2),
        assert(misses(N1,Tries2)), !.

maintain_guesses :- num_guesses(X),
        X1 is X + 1,
        abolish(num_guesses,1),
        assert(num_guesses(X1)), !.

maintain_guesses :- !.


/* Final summary of student's responses. */

conclusion :- shell(cls), blines(4),
        first_try(A,First_tries), nl,
        write('You identified '), write(A),
        write(' ship(s) on the first try.'),
        write('  The ship(s) were:'), nl,
        print_3(First_tries), nl,
        second_try(B,Second_tries),
        write('You identified '), write(B),
        write(' ship(s) on the second try.'),
        write('  The ship(s) were:'), nl,
        print_3(Second_tries), nl,
        third_try(C,Third_tries),
        write('You identified '), write(C),
        write(' ship(s) on the third try.'),
        write('  The ship(s) were:'), nl,
        print_3(Third_tries), nl,
        misses(D,Missed_ships),
        write('You did not identify the following
            ship(s): '), nl,
        print_3(Missed_ships), nl,
        num_guesses(G),
        write('You appeared to guess on the first look
            at '),
        write(G), write(' ship(s).'), nl, nl,
        write('Type any letter key to continue. '),
        read(Anykey), nl, nl,
        write_to_file, !.

/* Write student's summary to file. */

write_to_file :-
        shell(cls), blines(10),
        write('Please enter an 8 character filename
            for your summary.  '), nl,
```

```
        write('The filename must begin with a letter
            and contain only '), nl,
        write('letters and numbers.  Remember this
            filename to '), nl,
        write('retrieve your summary at your next user
            session.  '),
        read(Filename),
        nl, nl, write('Your filename is '),
        write(Filename),
        tell(Filename),
        first_try(A,First_tries), nl,
        write('You identified '), write(A),
        write(' ship(s) on the first try.'),
        write('  The ship(s) were:'), nl,
        print_3(First_tries), nl,
        second_try(B,Second_tries),
        write('You identified '), write(B),
        write(' ship(s) on the second try.'),
        write('  The ship(s) were:'), nl,
        print_3(Second_tries), nl,
        third_try(C,Third_tries),
        write('You identified '), write(C),
        write(' ship(s) on the third try.'),
        write('  The ship(s) were:'), nl,
        print_3(Third_tries), nl,
        misses(D,Missed_ships),
        write('You did not identify the following
            ship(s): '), nl,
        print_3(Missed_ships), nl,
        num_guesses(G),
        write('You appeared to guess on the first
            look at '),
        write(G), write(' ship(s).'), nl, nl, nl,
        told, nl, nl, nl,
        write('Your summary has been saved to the file '),
        write(Filename), write('.'), nl,
        write('Remember this filename to review this
            summary.'), nl, nl,
        write('Type any letter key to continue. '),
        read(Anykey), nl, nl, !.


/* Clear memory for the next photo. */
```

```
clear_vars :-
        abolish(photo_num,1),
        abolish(ship_fact,1),
        abolish(fact,1),
        abolish(incorrect_fact,1),
        abolish(time_to_look,1),
        abolish(stop_review,0),
        abolish(fact,1),
        abolish(incorrect_fact,1),
        abolish(numtries,1),
        assert(numtries(1)), !.
```

```
/* Comparer.ari */

/* This program does the comparison between the two lists.
Three lists are generated consisting of features in the
first list which are not in the second, features in the
second list which are not in the first, and features whose
predicates are the same in both lists, but whose arguments
are different. */


/*Any duplicate items in the two lists are deleted from
both prior to further comparisons.*/

compare(Answer,Student,Features_not_present,Features_missed,
       Feature_details_diff) :-
            delete_duplicates(Answer,Student,Answer2,
               Student2),
            compare2(Answer2,Student2,Features_not_present,
               Features_missed,Feature_details_diff).


/* Delete_duplicates deletes duplicate terms from Answer
and Student.     */

delete_duplicates([],Student,[],Student).

delete_duplicates([A|Answer],Student,Answer3,Student3) :-
            member(A,Student),
            delete(A,Student,Student2),
            delete_duplicates(Answer,Student2,Answer3,
               Student3),!.

delete_duplicates([A|Answer],Student,[A|Answer3],
   Student3) :-
            delete_duplicates(Answer,Student,Answer3,
               Student3).



/* Base condition */

compare2([],[],[],[],[]).


/* Answer has more terms than the Student model. */

compare2([A|Answer],[],Features_not_present,
   [A|Features_missed],Feature_details_diff) :-
            compare2(Answer,[],Features_not_present,
               Features_missed,Feature_details_diff),!.
```

```
/* Student model has more terms than the Answer. */

compare2([],[S|Student],[S|Features_not_present],
   Features_missed,Feature_details_diff) :-
            compare2([],Student,Features_not_present,
               Features_missed,Feature_details_diff),!.


/* The Answer has terms with different arguments than the
Student model.  */

compare2(Answer,Student,Features_not_present,
      Features_missed,[S|Feature_details_diff]) :-
            match_terms(Answer,Student,A,S),
            delete(A,Answer,Answer2),
            delete(S,Student,Student2),
            compare2(Answer2,Student2,Features_not_present,
               Features_missed,Feature_details_diff),!.


/* The Answer has terms not in the Student model, but the
Student model is not empty. */

compare2(Answer,Student,Features_not_present,
   [Term|Features_missed],Feature_details_diff) :-
            find_a_term(Answer,Student,Term),
            delete(Term,Answer,Answer2),
            compare2(Answer2,Student,Features_not_present,
               Features_missed,Feature_details_diff),!.

/* The Student model has terms not in the Answer, but the
Answer is not empty. */

compare2(Answer,Student,[Term|Features_not_present],
   Features_missed,Feature_details_diff) :-
            find_s_term(Answer,Student,Term),
            delete(Term,Student,Student2),
            compare2(Answer,Student2,Features_not_present,
               Features_missed,Feature_details_diff),!.


/* Match_terms finds the terms in The Answer which match
the terms in the Student model except that the arguments
are different.  It returns the terms that match. */

match_terms([A|Answer],Student,A,S) :-
            A =.. [Pred,TA],
            S =.. [Pred,TS],
            member(S,Student).
```

```
/* Find_a_term finds terms in Answer which are not in the
Student model.  Used when Student model is not empty.  Will
not find terms with mismatched arguments. */

/* Example: a predicate with no arguments */

find_a_term([Term|Answer],Student,Term) :-
            Term =.. [Pred],
            S =.. [Pred],
            not(member(S,Student)).

/* Example: any one argument predicate */

find_a_term([Term|Answer],Student,Term) :-
            Term =.. [Pred,TA],
            S =.. [Pred,TS],
            not(member(S,Student)).

/* Find_s_term finds terms in the Student model which are
not in the Answer.  Used when the Answer is not empty.
Will not find terms with mismatched arguments. */

find_s_term(Answer,[Term|Student],Term) :-
            Term =.. [Pred],
            A =.. [Pred],
            not(member(A,Answer)).

find_s_term(Answer,[Term|Student],Term) :-
            Term =.. [Pred,TS],
            A =.. [Pred,TA],
            not(member(A,Answer)).
```

```
/* tmenu.ari */

/* This file contains the menus for the tutor program */


writemenu([]) :- !.
writemenu(Descrip_list) :- length(Descrip_list,N),
        writemenu2(Descrip_list,N),!.

writemenu2([],N) :- !.
writemenu2([I|Descrip_list2],N) :-
        length(Descrip_list2,N2),
        N3 is N - N2,
        tab(3), bspace(N3),
        write(N3), write('.'),
        write(I), nl,
        writemenu2(Descrip_list2,N).



shipmenu([]) :- !.
shipmenu(Descrip_list) :- length(Descrip_list,N),
        shipmenu2(Descrip_list,N), !.

shipmenu2([],N) :- !.
shipmenu2([I1|[]],N) :-
        length(Descrip_list2,N2),
        N3 is N - N2,
        tab(3), bspace(N3),
        write(N3), write('.'),
        write(I1), nl,
        shipmenu2(Descrip_list2,N), !.
shipmenu2([I1,I2|Descrip_list2],N) :-
        length(Descrip_list2,N2),
        N4 is N - N2,
        N3 is N4 - 1,
        tab(3), bspace(N3),
        write(N3), write('.'),
        write(I1),
        name(I1,L), length(L,L1),
        Spaces is 25 - L1,
        tab(Spaces), bspace(N4),
        write(N4), write('.'),
        write(I2), nl,
        shipmenu2(Descrip_list2,N).


ship_ask_which(Names) :-
        read(Shipname), create_ship_fact(Shipname,Names),
        nl,nl.
```

```
create_ship_fact(Shipname,Names) :-
        name(Shipname,[113]), assert(ship_fact('quit')), !.

create_ship_fact(Shipname,Names) :-
        name(Shipname,[104]), assert(ship_fact('help')), !.

create_ship_fact(Shipname,Names) :-
        total_photos(X),
        integer(Shipname),
        Shipname > 0,
        Shipname =< X,
        item(Shipname,Names,I),
        assertz(ship_fact(I)), !.


create_ship_fact(Shipname,Names) :- nl, nl,
        total_photos(X),
        write('Your input is not within the range of
            1 to '),
        write(X), write('.'), write('  Please reenter: '),
        read(New_entry),
        create_ship_fact(New_entry,Names), !.



descrip_ask_which(Descrip_list,I) :-
        read(Description),
find_description(Description,Descrip_list,I).

find_description(Description,Descrip_list,I) :-
        length(Descrip_list,N),
        integer(Description),
        Description > 0,
        Description =< N,
        item(Description,Descrip_list,I), !.

find_description(Description,Descrip_list,I) :- nl, nl,
        length(Descrip_list,N),
        write('Your input is not within the range of
            1 to '),
        write(N), write('.'), write('  Please reenter: '),
        read(New_entry), find_description(New_entry,De-
scrip_list,I), !.

item(1,[X|L],X).
item(N,[X|L],I) :- N > 1, N2 is N - 1, item(N2,L,I).
```

```
main_menu :- abolish(system_option,1),
        shell(cls),
        blines(10), tab(17),
        write('Ship Recognition Main Menu'), nl, nl,
        tab(17), write(' 1. Ship recognition test'), nl,
        tab(17), write(' 2. Review a specific photo'), nl,
        tab(17), write(' 3. View a summary'), nl,
        tab(17), write(' 4. Quit'), blines(3),
        write('Enter 1.,2.,3. or 4.'),
        read(Choice), check_entry(Choice,4,main_menu),
        assert_option(Choice), blines(3).

assert_option(1) :- assert(system_option(test)).
assert_option(2) :- assert(system_option(photo_review)).
assert_option(3) :- assert(system_option(view_summary)).
assert_option(4) :- assert(system_option(quit)).
assert_option(_).

experience_level :- abolish(user_level,1),
        shell(cls),
        blines(10), tab(25),
        write('Level of experience'), nl, tab(25),
        write('-----------------'),
        nl, tab(25), write('   1. Beginner'), nl,
        tab(25), write('   2. Intermediate'), nl,
        tab(25), write('   3. Expert'), blines(3),
        write('Enter 1.,2., or 3.: '),
        read(Choice),
check_entry(Choice,3,experience_level),
        assert_level(Choice), blines(3).

assert_level(1) :- assert(user_level(beginner)).
assert_level(2) :- assert(user_level(intermediate)).
assert_level(3) :- assert(user_level(expert)).
assert_level(_).

check_entry(Choice,Num_entries,Pred) :-
        Choice > 0,
        Choice =< Num_entries, !.

check_entry(Choice,Num_entries,Pred) :- Pred, !.
```

```
/* ships.ari */

This file contains the database for the ship recognition
tutor.  All ships in the database are contained in the rule
predicates.  The ship_names predicate is used by when
creating the ship choices menu.

When adding ships to the database tdescrip.ari must also be
updated.  Be sure to update the total_photos, ship_names
and
ship_photo_comb predicates also.


/* Number of photos in the data base. */

total_photos(20).

/* Information for tmenu.ari, should exactly match the
names in the predicate rule and ship_phot_comb. */

ship_names('Your ship choices are: ',
        ['Kresta I CG','Kresta II CG','Jiangdong FF',
        'Jiang Hu I FF','Jiang Hu II FF','Godavari FF',
        'Grisha I/II/III FFL','Kaman PTG','Kara CG',
        'Kiev CVHG','Kirov CGN','Krivak I FFG',
        'Krivak II FFG','Krivak III WPGF','Slava',
        'Kashin DDG','Sovremennyy DDG','Yurka MSF',
        'Kynda CG','Udaloy DDG']).

/* Assigns photo numbers to ship names.  Be sure to number
photos correctly. */

ship_photo_comb(1,'Jiang Hu I FF').
ship_photo_comb(2,'Jiangdong FF').
ship_photo_comb(3,'Kresta I CG').
ship_photo_comb(4,'Kresta II CG').
ship_photo_comb(6,'Jiang Hu II FF').
ship_photo_comb(5,'Godavari FF').
ship_photo_comb(9,'Grisha I/II/III FFL').
ship_photo_comb(7,'Kaman PTG').
ship_photo_comb(8,'Kara CG').
ship_photo_comb(14,'Kiev CVHG').
ship_photo_comb(12,'Kirov CGN').
ship_photo_comb(13,'Krivak I FFG').
ship_photo_comb(10,'Krivak II FFG').
ship_photo_comb(11,'Krivak III WPGF').
ship_photo_comb(15,'Sovremennyy DDG').
ship_photo_comb(16,'Udaloy DDG').
ship_photo_comb(17,'Yurka MSF').
ship_photo_comb(18,'Kynda CG').
```

```
ship_photo_comb(19,'Kashin DDG').
ship_photo_comb(20,'Slava').
/* ships expert system */

rule(ship_id('Godavari FF'),
        [mast_2('two masts'),
        foremast('solid tower foremast'),
        mainmast('smaller pylon mainmast'),
        ssm('4 SSM launchers forward of the
            superstructure'),
        sam('single launcher'),
        helo('hanger and pad aft')]).
rule(ship_id('Grisha I/II/III FFL'),
        [mast_1('heavy pylon mast with latticed
            extensions'),
        mast_1_loc('on top of the bridge'),
        stack('located aft'),
        rocket_launcher('RBU launcher on a raised mid
            section')]).
rule(ship_id('Jiangdong FF'),
        [mast_2('two masts'),
        foremast('latticed tripod foremast'),
        foremast_loc('abut the superstructure'),
        aftermast('quadruped mast'),
        aftermast_loc('atop the aft deckhouse'),
        sam('launcher forward of bridge'),
        sam_2('an identical pair of SAM launchers aft'),
        gun('single mount'),
        bridge('globular fire control on top')]).
rule(ship_id('Jiang Hu I FF'),
        [mast_1('latticed tripod mast'),
        stack('large, square'),
        ssm('twin launchers forward and aft of stack'),
        bridge('forward placed')]).
rule(ship_id('Jiang Hu II FF'),
        [mast_1('latticed tripod mast'),
        stack('large, square'),
        ssm('twin launchers forward and aft of stack'),
        ssm_2('a SSM 1 launcher forward'),
        bridge('forward placed'),
        helo('helo pad aft')]).
rule(ship_id('Kaman PTG'),
        [mast_1('large radome atop mast'),
        gun('forward of superstructure'),
        gun_2('aft of deckhouse'),
        ssm('located between superstructure and
            deckhouse'),
        superstructure('enclosed,streamlined with separated
            deckhouse')]).
```

```
rule(ship_id('Kara CG'),
        [mast_1('large pyramid mast, supports TOP SAIL
            radar'),
        stack('large, square'),
        helo('helo pad aft'),
        ssm('canted launchers below bridge wings')]).
rule(ship_id('Kresta I CG'),
        [mast_2('two masts'),
        foremast('large obelisk type foremast'),
        aftermast('smaller pyramid aftermast'),
        radar('air surveillance radar atop mack'),
        helo('hangar and pad aft'),
        ssm('horizontal launchers under bridge wings')]).
rule(ship_id('Kresta II CG'),
        [mast_2('two masts'),
        foremast('large obelisk type foremast'),
        aftermast('smaller pyramid aftermast'),
        radar('air surveillance radar atop mack'),
        top_sail('TOP SAIL radar'),
        helo('raised helo pad aft'),
        ssm('canted launchers below bridge wings')]).
rule(ship_id('Kiev CVHG'),
        [fight_deck('angled flight deck'),
        superstructure('tiered superstructure'),
        other_features('looks like cruiser in profile')]).
rule(ship_id('Kirov CGN'),
        [mast_1('large mack amidship'),
        radar('atop mack'),
        top_sail('TOP SAIL radar'),
        superstructure('superstructure amidship with large
            mack'),
        forecastle('long, slightly stepped, sharply raked
            bow')]).
rule(ship_id('Krivak I FFG'),
        [mast_2('two masts'),
        masts('latticed'),
        superstructure('superstructure with masts'),
        ssm('4 SSM launchers on bow'),
        stack('short, wide with lip on trailing edge'),
        gun('gunmounts flat and square')]).
rule(ship_id('Krivak II FFG'),
        [mast_2('two masts'),
        masts('latticed'),
        superstructure('superstructure with masts'),
        ssm('4 SSM launchers on bow'),
        stack('short, wide with lip on trailing edge'),
        gun('gunmounts rounded with fence between')]).
rule(ship_id('Krivak III WPGF'),
        [mast_2('two masts'),
        masts('latticed'),
        superstructure('superstructure with masts'),
```

```
        stack('short, wide with lip on trailing edge'),
        gun('enclosed gunmount on bow'),
        helo('hangar and pad aft')]).
rule(ship_id('Kashin DDG'),
        [mast_1('single oil derrick mast'),
        stack('4 in symmetry to the mast, canted out'),
        sam('2 SAM launchers, fore and aft'),
        gun('2 enclosed gun mounts, fore and aft')]).
rule(ship_id('Kynda CG'),
        [mast_2('two masts'),
        masts('pyramid masts'),
        ssm('two (HOT DOG PACKS), 1 forward, 1 aft'),
        other_features('mast-stack-mast-stack')]).
rule(ship_id('Udaloy DDG'),
        [mast_2('two masts'),
        masts('tripod mast preceeds each set of stacks'),
        stack('four stacks in pairs amidship'),
        helo('twin helo hangar and raised helo pad aft'),
        ssm('canted launchers below bridge wings'),
        gun('2 enclosed gun mounts forward')]).
rule(ship_id('Yurka MSF'),
        [stack('oval shaped stack (YURKED TO THE SIDE)'),
        mast_1('latticed'),
        radar('fire control radar atop mack'),
        gun('gun mount forward and aft')]).
rule(ship_id('Slava'),
        [mast_2('two masts'),
        foremast('large pyramid mast structure'),
        aftermast('smaller radar mast'),
        stack('twin stacks amidship'),
        radar('Top Dome on after deckhouse'),
        helo('hangar and pad aft'),
        missiles('16 SS-N-12 Sandbox missiles paired
            forward')]).
rule(ship_id('Sovremennyy DDG'),
        [bridge('radome atop bridge'),
        gun('spheroid enclosed gun mount foward and aft'),
        ssm('canted launchers under bridge wings'),
        stack('single stack amidship'),
        helo('pad and telescoping hangar aft of stack'),
        radar('air surveillance radar atop mack')]).
```

```
/* tdescrip.ari */

/* This file contains the descriptive information for each
feature.  The information exactly matches the features
descriptions used in the rule(ship_id) predicates.  This
file is used by to create the feature description menus.*/

/* When adding ships to the database be sure to update this
file also.   */


descriptors(mast_2,
        'Does the ship have: ',
        ['two masts']).

descriptors(masts,
        'Select the best description of the masts. ',
        ['latticed',
        'pyramid masts',
        'tripod mast preceeds each set of stacks']).

descriptors(foremast,
        'Select the best description of the foremast.',
        ['solid tower foremast',
        'latticed tripod foremast abuts superstructure',
        'large obelisk type foremast',
        'large pyramid mast structure']).

descriptors(foremast_loc,
        'Does the foremast: ',
        ['abut the superstructure?']).

descriptors(aftermast,
        'Select the best description of the aftermast.',
        ['quadruped mast',
        'smaller pyramid aftermast',
        'smaller radar aftermast']).

descriptors(aftermast_loc,
        'Is the aftermast located: ',
        ['atop the aft deckhouse']).

descriptors(mainmast,
        'Does the ship have a: ',
        ['smaller pylon mainmast']).
```

```
descriptors(mast_1,
        'Select the best description of the mast.',
        ['heavy pylon mast with latticed extensions',
        'latticed tripod mast',
        'large radome atop mast',
        'large pyramid mast, supports TOP SAIL radar',
        'large mack amidship with TOP SAIL radar',
        'single oil derrick mast',
        'latticed']).

descriptors(mast_1_loc,
        'Is the mast: ',
        ['on top of the bridge']).

descriptors(ssm,
        'Select the best description of the SSM
             launchers(s).',
        ['4 SSM launchers forward of the superstructure',
        'twin launchers forward and aft of stack',
        'located between superstructure and deckhouse',
        '4 SSM launchers located on bow',
        'canted launchers below bridge wings',
        'horizontal launchers under bridge wings',
        'two (HOT DOG PACKS), 1 forward, 1 aft']).

descriptors(ssm_2,
        'Is there: ',
        ['a SSM launcher forward']).

descriptors(sam,
        'Select the best description of the SAM
             launcher(s).',
        ['single launcher',
        'launcher forward of bridge, identical pair aft',
        'two SAM launchers, fore and aft']).

descriptors(sam_2,
        'Is there an: ',
        ['identical pair of SAM launchers aft']).

descriptors(rocket_launcher,
        'Does the ship have a:',
        ['RBU launcher on a raised mid section']).

descriptors(missiles,
        'Does the ship have:',
        ['16 SS-N-12 Sandbox missiles paired forward']).
```

```
descriptors(gun,
        'Select the best description of the gun mounts.',
        ['single mount',
        'forward of superstructure, aft of deckhouse',
        'gunmounts flat and square',
        'gunmounts rounded with fence between',
        'enclosed gunmount on bow',
        'two enclosed gun mounts, fore and aft',
        'two enclosed gun mounts forward',
        'spheroid enclosed gun mount forward and aft']).

descriptors(gun_2,
        'Is there a second gun :',
        ['aft of deckhouse']).

descriptors(radar,
        'Select the best description of the radar
            features. ',
        ['air surveillance radar atop mack',
        'Top Dome on after deckhouse',
        'fire control radar atop mack']).

descriptors(top_sail,
        'Does the ship have: ',
        ['TOP SAIL radar']).

descriptors(stack,
        'Select the best description of the stack
            structure.',
        ['located aft',
        'large, square',
        'short, wide with lip on trailing edge',
        '4 in symmetry to the mast, canted out',
        '4 stacks in pairs amidships',
        'oval shaped stack (YURKED TO THE SIDE)',
        'twin stacks amidship',
        'single stack amidship']).

descriptors(bridge,
        'Select the best description of the bridge
            structure.',
        ['globular fire control on top',
        'forward placed',
        'radome atop bridge']).
```

```
descriptors(superstructure,
        'Select the best description of the
             superstructure.',
        ['enclosed, streamlined with separated deckhouse',
        'tiered superstructure',
        'superstructure amidship with large mack',
        'superstructure with masts']).

descriptors(flight_deck,
        'Does the ship have a:',
        ['angeled flight deck']).

descriptors(helo,
        'Select the best description of the onboard helo
             structures.',
        ['hangar and pad aft',
        'helo pad aft',
        'raised helo pad aft',
        'twin helo hangar and raised helo pad aft',
        'pad and telescoping hangar aft of stack']).

descriptors(forecastle,
        'Does the ship have a:',
        ['long, slightly stepped, sharply raked bow']).

descriptors(other_features,
        'Some other features are:',
        ['looks like a cruiser in profile',
        'mast-stack-mast-stack']).
```

```
/* tintro.ari */

intro_screen :- shell(cls), blines(2), tab(17),
        write('      SSSSS    HH     HH  II  PPPPPPP'), nl,
tab(17),
        write('    SS     SS  HH     HH  II  PP     PP'),
nl, tab(17),
        write('    SS         HH     HH  II  PP     PP'),
nl, tab(17),
        write('    SS         HH     HH  II  PP     PP'),
nl, tab(17),
        write('      SSSSS    HHHHHHHHH  II  PPPPPPP'), nl,
tab(17),
        write('           SS  HH     HH  II  PP'), nl,
tab(17),
        write('           SS  HH     HH  II  PP'), nl,
tab(17),
        write('    SS     SS  HH     HH  II  PP'), nl,
tab(17),
        write('      SSSSS    HH     HH  II  PP'), nl,
        blines(2), tab(12),
        write('RRRRRRR    EEEEEEEEE   CCCCCC    CCCCCC
OOOOOO'), nl, tab(12),
        write('RR     RR  EE         CC    CC  CC    CC  OO
OO'), nl, tab(12),
        write('RR     RR  EE         CC    CC  CC    CC  OO
OO'), nl, tab(12),
        write('RR     RR  EE         CC        CC        OO
OO'), nl, tab(12),
        write('RRRRRRR    EEEEEEE    CC        CC        OO
OO'), nl, tab(12),
        write('RR RR      EE         CC        CC        OO
OO'), nl, tab(12),
        write('RR  RR     EE         CC    CC  CC    CC  OO
OO'), nl, tab(12),
        write('RR   RR    EE         CC    CC  CC    CC  OO
OO'), nl, tab(12),
        write('RR     RR  EEEEEEEEE   CCCCCC    CCCCCC
OOOOOO'), nl,
        blines(2), shell(pause),
        shell(cls), blines(5),
        write('Ship recognition training is required for
            many Naval '),
        write('personnel.  '), nl,
        write('It is taught by Intelligence personnel using
            slides, '),
        write('flash cards '), nl,
        write('and other drill methods.'),
        blines(3),
```

```
        write('This ship recognition tutor is designed to
            help you '),
        write('improve your '), nl,
        write('proficiency in ship recognition.  You will
            be provided '),
        write('with a '), nl,
        write('list of ships from which to identify the
            photo specified.'),
        write('  The method'), nl,
        write('of tutoring is based on your level of
            experience.  The '),
        write('system also '), nl,
        write('provides you with a means of viewing a
            summary of your '),
        write('last session.  '), nl,
        write('You may also review specific photos.'),
        blines(3),
        write('PROLOG REMINDER: '), nl, nl,
        write('ALL ENTRIES MUST BE FOLLOWED BY A
            PERIOD (".")'),
        blines(2), shell(pause).



/* loadfile.ari */

:- write('Please wait.'), nl.
:- nl.
:- [tdriver].
:- write('Still loading').
:- nl.
:- [tintro,tmenu].
:- write('Still loading').
:- nl.
:- [ships,tdescrip].
:- write('Still loading').
:- [utilities,comparer].
```

```
/* Arity utility predicates. */


abolish(F,A) :- functor(T,F,A), retractall(T).

retractall(X) :- retract(X), fail.
retractall(X) :- retract((X :- Y)), fail.
retractall(_).

length([],0).
length([X],1).
length([X|L],N) :- length(L,N1), N is N1 + 1.

append([],L,L).
append([X|L],L2,[X|L3]) :- append(L,L2,L3).

concatenate(S1,S2,S) :- name(S1,AS1), name(S2,AS2),
        append(AS1,AS2,AS), name(S,AS).

blines(0) :- !.
blines(N) :- nl, N1 is N - 1, blines(N1).

bspace(X) :- X < 10, write(' ').
bspace(X).

timecomp(A,B,C,D,Total) :-
       Min is C-A,
       Sec is D-B,
       Total is Min * 60 + Sec.



member(X,[X|L]).
member(X,[Y|L]) :- member(X,L).

delete(X,[],[]).
delete(X,[X|L],M) :- !, delete(X,L,M).
delete(X,[Y|L],[Y|M]) :- not X=Y, delete(X,L,M).


prettyprint([]).
prettyprint([Y|L]) :- prettyprint1(Y), prettyprint(L).

prettyprint1([]).
prettyprint1([X|L]) :- write(X), nl, prettyprint1(L).
```

```
print_3([]).
print_3([X,Y,Z|L]) :-
        write(X), name(X,X1), length(X1,A),
        A1 is 25 - A, tab(A1),
        write(Y), name(Y,Y1), length(Y1,B),
        B1 is 25 - B, tab(B1),
        write(Z), nl, print_3(L).
print_3([X,Y|[]]) :-
        write(X), name(X,X1), length(X1,A),
        A1 is 25 - A, tab(A1),
        write(Y), nl, print_3(L).
print_3([X|[]]) :-
        write(X), nl, print_3(L).
```

```
/* Pascal executable files */

Program Summary(input,output);

(* This program reads in the user session summary and
prints it to the screen. *)

type
   String80 = string[80];
   String25 = string[25];
var
   Line : String80;
   Filename : String25;
   Data : text;
begin
     Writeln('What is the name of your summary file?');
     Readln(Filename);
     Assign(Data,Filename);
     Reset(Data);
     While not EOF(Data) do
           begin
                Readln(Data,Line);
                Writeln(Line);
           end;
     Close(Data);
end.


program getstime(input,output);

uses dos;

var
   Hour,Min,Sec,Hun:word;
   data:text;
begin
     GetTime(Hour,Min,Sec,Hun);
     assign(data,'timefile.ari');
     reset(data);
     write(data,'time(');
     write(data,Min);
     write(data,',');
     write(data,Sec);
     write(data,')');
     close(data);
end.
```

LIST OF REFERENCES

1. *Fundamentals of Naval Intelligence*, pp.196-197, Naval Education and Training Command, 1975.

2. *COMPATWINGSPAC Recco Guide*, Commander Patrol Wings Pacific, 1988.

3. *Jane's Fighting Ships*, Janes's Publishing Company, Limited, 1988.

4. Polmar, Norman, *Guide to the Soviet Navy*, 4th ed., Naval Institute Press, 1986.

5. Zinn, K. L., "Computer-Assisted Learning and Teaching" in *Encyclopedia of Computer Science and Engineering*, 2nd ed., A. Ralston, pp.294-302, Van Nostrand Reinhold Co, 1983.

6. Wenger, E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, 1987.

7. Barr, A. and Feigenbaum, E. A., *The Handbook of Artificial Intelligence*, v.2, William Kaufmann, 1982.

8. Woolf, B. and Cunningham, P. A., "Multiple Knowledge Sources in Intelligent Teaching Systems," *IEEE Expert*, pp. 41-54, Summer 1987.

9. Kearsley, G. P., *Artificial Intelligence & Instruction: Applications and Methods*, Addison-Wesley, 1987.

10. Kennedy, William V., and others, *Intelligence Warfare*, Crescent Books, 1983.

11. Campbell, D. F., *An Intelligent Computer-Assisted Instruction System For Cardiopulmonary Resuscitation*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, DC 20350-2000	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	2
5.	Curriculum Officer, Code 37 Computer Technology Naval Postgraduate School Monterey, California 93943-5000	1
6.	Associate Professor Neil C. Rowe Code 52Rp Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	1
7.	Professor Timothy J. Shimeall, Code 52Sm Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	1
8.	Commanding Officer Naval Research Laboratory Washington, DC 20375	1
9.	Chief of Naval Education and Training Naval Air Station Pensacola Pensacola, Florida 32508	1

10. Dr. Hank Smith — 1
Education Coordinator
Patrol Squadron THIRTY-ONE
Naval Air Station
Moffett Field, California 94035

11. Tactical Training Team — 1
Patrol Squadron THIRTY-ONE
Naval Air Station
Moffett Field, California 94035

12. CPT Michael J. Bizer — 1
1204 Catskill Circle
Huntsville, Alabama 35802

13. LT Denise R. Bernier — 2
1908 Azalea Street
Denton, Texas 76205

Thesis
B4529 Bernier
c.1 An intelligent computer-aided instruction system for Naval ship recognition.