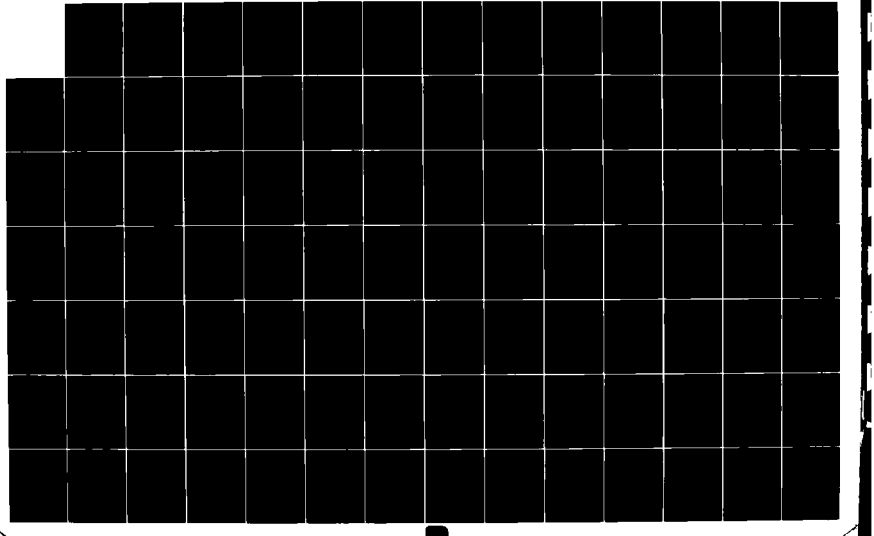
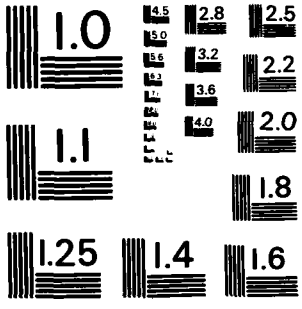
AD-A127 702 AN OVERVIEW OF ARTIFICIAL INTELLIGENCE-BASED TEACHING AIDS(U) MITRE CORP MCLEAN VA P K GROVESTON DEC 82 F19628-83-C-0001 1/2

UNCLASSIFIED F/G 9/2 NL





The MITRE Corporation
MITRE C³ Division
Washington C³ Operations
1820 Dolley Madison Boulevard
McLean, Virginia 22102

WORKING PAPER

Subject: An Overview of Artificial Intelligence-Based Teaching Aids

WP- 82W00646 (No / Vol / Series / Rev / Supp / Corr)

To: Zitta Z. Friedlander

From: P. Kathie Groveston

Dept.: W-74

Date: December 1982

Contract No.: F19628-83-C-0001

Sponsor: U.S. Army Intelligence Center and School

Project No.: 86060

Approved for MITRE Distribution: B Friedlander

ABSTRACT:

This paper discusses some of the ways that artificial intelligence research has contributed to the development of teaching systems, specifically in the areas of object-oriented programming languages and expert systems. The research was conducted in support of the Directorate of Training Developments at the U.S. Army Intelligence Center and School.

MITRE

11 March 1983
W74-030

Commander
U. S. Army Intelligence Center and
School
ATTN: ATSI-TD-EW
Ft. Huachuca, AZ 85613

Subject: Transmittal of WP82W00646, "An Overview of Artificial Intelligence-Based Teaching Aids," dated December 1982

Dear Captain Velicki:

Enclosed is a copy of MITRE Working Paper WP82W00646, "An Overview of Artificial Intelligence-Based Teaching Aids." The document discusses some of the ways that artificial intelligence research has contributed to the development of teaching systems, and shows how some non-teaching programs have been altered for use in teaching. The areas of interest here are object-oriented programming languages and expert systems.

If you have any questions or comments, please contact me at (703) 827-7088.

Sincerely,

Z Friedlander

Dr. Zitta Z. Friedlander
Project Leader
Systems Analysis

ZZF/PKG:mb

Enclosure

The MITRE Corporation
MITRE C^3 Division
Washington C^3 Operations
1820 Dolley Madison Boulevard, McLean, Virginia 22102
Telephone (703) 827-6000 • Telex 248-923

AN OVERVIEW OF ARTIFICIAL INTELLIGENCE-BASED TEACHING AIDS

P. K. Groveston

November 1982

ABSTRACT

This paper discusses some of the ways that artificial intelligence research has contributed to the development of teaching systems, specifically the use of object-oriented programming languages and expert systems. The research was conducted in support of the Directorate of Training Developments, for the U.S. Army Intelligence Center and School.

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 INTRODUCTION

Artificial intelligence is a branch of computer science which studies methods of making computers behave in ways we think of as "intelligent". Since the concept of intelligence and how it works is not yet fully understood, it is not surprising that this definition of artificial intelligence is vague; however, intelligent behavior is generally assumed to include reasoning, inference, and judgment.

Early uses of computer aided instruction (CAI), beginning in the 1960s, involved systems that presented text to the student (electronic "page turners") or posed problems to the student and responded to his answers with standard comments (drill and practice monitors). In the 1970s a new type of teaching system began to appear, called intelligent computer aided instruction (ICAI). An important distinction between CAI and ICAI is that the ICAI programs, instead of giving a preset response to a student's answer, attempt to diagnose the individual student's strength and weaknesses as well as his learning style by separating the subject area knowledge from the teaching strategy.

This paper discusses some of the ways that artificial intelligence research has contributed to the development of teaching systems. Two aspects of artificial intelligence will be covered: object oriented programming languages (languages which perform symbolic as well as numerical processing by sending messages to objects rather than calling procedures), and expert systems (computer programs which solve problems by using knowledge and procedures obtained from human experts).

2.0 OBJECT-ORIENTED PROGRAMMING LANGUAGES

Object-oriented programming languages differ from traditional languages in their structure, which is based on objects and messages rather than data and procedures. Instead of calling a procedure to act on data, the user sends a message to an object, e.g., the message "factorial" to the object "6". This structure does not preclude the use of procedures, however; a series of messages may be stored as a procedure or routine, then sent as a message to an object.(1)

These languages are included here because, although they may not technically be considered artificial intelligence, they lend themselves to teaching and to expert system applications. In addition, they encourage concepts of artificial intelligence such as hierarchical decomposition, or the art of breaking a problem down into manageable portions.(2)

Three object-oriented programming languages will be discussed: Logo, Smalltalk, and the Rule-Oriented Simulation System (ROSS).

I Object-Oriented Programming Languages

Definition: One that uses objects and messages instead of data and procedures. User sends a message to an object, instead of calling a procedure to act on data.

Examples:

- Logo
- Smalltalk
- Rule oriented Simulation System (ROSS)

2.1 Logo

Logo was designed to fill the need for a simple yet powerful language which would be of value to users at all levels of programming skill. It has become known primarily as a vehicle for teaching programming concepts, mathematics and general problem-solving techniques to students of pre-school through college levels. One of Logo's advantages in an educational application is its ability to allow the student to discover principles for himself.

Logo was developed in the late 1960's by the Logo Group at the Massachusetts Institute of Technology. The language is essentially a dialect of the LISP language the primary research language used in artificial intelligence. As such, Logo shares LISP's interactive and list-processing properties. Since there are versions of Logo for Apple computers, this language is practical for school use.

Logo - Overview

Purpose:

- Designed as simple but powerful language for users of all levels
- Best known as language for teaching programming concepts, mathematics, and problem solving *(pre-school through college)*
- Designed to allow the student to discover principles for himself

Background:

- Developed in late 1960's by Logo Group at MIT
- Dialect of LISP
  - Interactive
  - List Processing
- Available for Apple computers

2.1.1 Logo Example: Turtle Geometry

One application of Logo which has been used in teaching is turtle geometry. The turtle is a trian device which appears a display screen and can respond to commands or "messages". By sending the turtle a series of messages, the student can cause it to move about the screen; the message "pen down" results in a visible trace of the turtle's path.

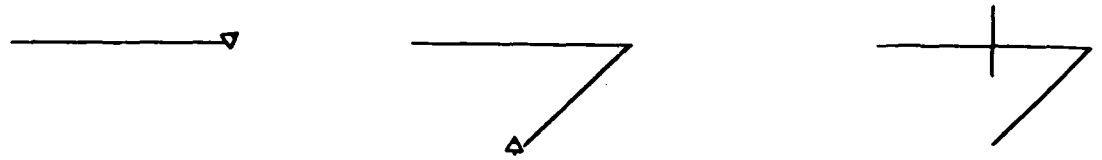
A series of commands, or messages, such as those at right which can trace a square, can be stored as a procedure and reproduced later by sending the procedure name as a message to the turtle. Such a procedure may in turn be included as a portion of another procedure. In the facing illustration, the "square" procedure is included as a portion of the "panes" procedure.

Logo - Example

Turtle Geometry

Turtle: Triangular device on screen

- Responds to commands
- "Carries" a pen to create graphic displays



FORWARD 150
LEFT 45

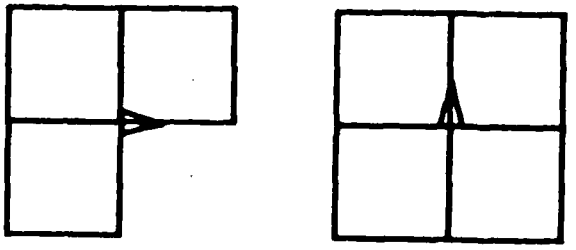
BACK 100
LEFT 45

PENUP
FORWARD 50
PENDOWN
FORWARD 50
HIDETURTLE

Moving the turtle with a simple sequence of Logo commands. **FORWARD** *moves the turtle in the direction it is facing.* **RIGHT** *and* **LEFT** *rotate the turtle.* **PENUP** *and* **PENDOWN** *raise and lower the pen—the turtle leaves a trace when it moves with the pen down.*

Forward 50	Right 90	Forward 50	Right 90
Forward 50	Right 90	Forward 50	Right 90

Building a square. With the turtle at the starting point, this sequence of commands will produce a square. These commands can also be turned into a Logo procedure



SQUARE **SQUARE**

PANES *as evolved from* **SQUARE**. *The figure is made by repeating the* **SQUARE** *procedure four times.*

2.1.2 The Teaching Utility of Logo

Logo is useful as a teaching aid in two ways. First, it teaches specific skills such as programming and mathematics. Turtle geometry in particular gives the student an understanding of geometrical concepts such as straight lines, angles, degrees, and geometrical figures. The student can experiment with various line lengths and angle sizes to discover for himself geometrical concepts and relationships.

Second, Logo teaches confidence, whether in programming, mathematics, or in more general applications. A mistake is not considered to be a "failure" on the part of the student; instead, it is a "bug" to be worked out. This debugging process can often lead to new ideas and a deeper understanding of the original problem. The student can also gain confidence by acting both as learner and teacher; in creating procedures, he is in effect "teaching the turtle a new word".

When Logo has been used to teach mathematics in elementary schools, there has been no dramatic improvement in students' scores on standardized mathematics tests. However, teachers report that the students have shown marked improvement in areas not measurable by standardized tests; they demonstrate greater self-confidence, group cooperation, and creativity and are better able to discuss and analyze the problems they are having in mathematics.(3)

Logo - Teaching Utility

Encourages:

- Understanding of geometrical concepts (lines, angles, degrees, figures)
- Confidence
  - Mistakes are not "failures" but bugs to be worked out
  - Unexpected results generate new ideas
  - Student becomes both learner and teacher (procedures = "teaching the turtle a new word")

Results:

- No dramatic improvement in math test scores
- Improvement in
  - Confidence
  - Group cooperation
  - Ability to articulate math difficulties
  - Creativity

2.2 Smalltalk

Smalltalk was created as the software to be used with the Dynabook, a hand-held, high-performance computer which is being developed by the Learning Research Group of the XEROX Palo Alto Research Center (PARC) in Palo Alto, California. Although Smalltalk is not intended as a language specifically for children, its several versions have been tested in educational applications.

The most recent of these versions, Smalltalk-80, exhibits several user-friendly features which are suitable for teaching:

- Windows - boxed-in work areas on the screen, which may overlap like sheets of paper
- Doors - entrances to areas of the system, such as graphics or the editor
- Menus - lists of options to guide the student through a session
- Mouse - an oblong device with three buttons by which the user causes a cursor on the screen to move
- "Modelessness" - The user may switch from one to another of the above features without disturbing his work in progress.

Smalltalk - Overview

Purpose:

- Intended as the language for Dynabook, a hand-held, high-performance computer under development

Background

- Developed in several versions by the Learning Research Group, XEROX Palo Alto Research Center (PARC)
- Not designed for children, but applications have been tested using students
- User friendly features (Smalltalk-80)
  - Windows
  - Doors (to areas such as graphics, editor)
  - Menus
  - Mouse (moves cursor on screen)
  - "Modeless" (can debug a portion of code without starting over)

2.2.1 Smalltalk Example: Smalltalk-80 Dance Kit

The Dance Kit is an application of Smalltalk-80 which is being developed for teaching programming to elementary and junior high school students.

Using this kit, the student choreographs a dance for a stick figure by sending messages, which specify dance steps, to objects, which correspond to portions of the figure. The figure is made up of five separate parts; the student can select successive replacements for the parts, giving the appearance of movement. The figure as a whole responds to messages such as "move" or "turn" which the student selects by means of the mouse. In this way, a student can create a dance without typing any commands. Bridges are used to indicate repetitions of an action; these repetitions may be stored as routines to be called later or to be used within other bridges.

Smalltalk - Example

Smalltalk-80 Dance Kit

Purpose: To teach programming concepts using Smalltalk

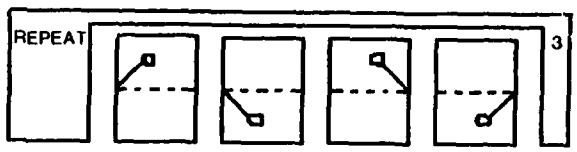
Method: "Programmer" choreographs a dance for a figure by sending messages (steps) to objects (portions of the figure)

- Creates replacement parts for objects so they seem to move
- Sends "move" or "turn" or other messages to figure by pointing to message on screen
- Uses bridges to repeat a sequence of movements

The facing figure illustrates the use of bridges and routines.

Smalltalk Example (Continued)

Creating a Dance

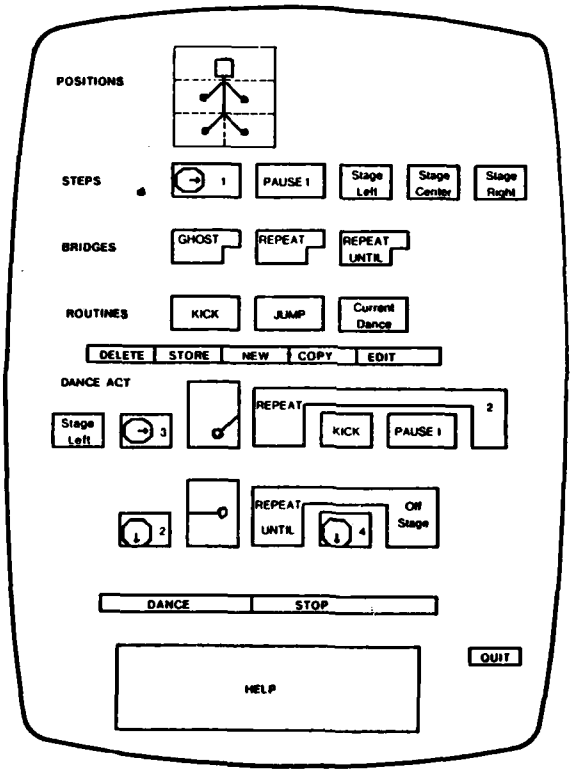


Specifying replacement parts produces an action, which is repeated by means of a bridge, then stored as the routine "WAVE".

WAVE is used within another bridge.

The facing figure illustrates the screen at the completion of the dance construction. It displays the five parts of the figure, the steps to be used, bridges (the "ghost" bridge specifies that previous positions of the figure are to remain on the screen as the figure "moves"), the previously stored routines, and the dance act itself.

Smalltalk Example (Continued)



2.2.2 The Teaching Utility of Smalltalk

In test applications with school children, the several versions of Smalltalk have been shown to successfully teach programming concepts. In addition, the programming tasks seemed to foster teamwork, with the students dividing themselves naturally into designer/programmer teams.(4)

Smalltalk is not now in routine use as a teaching aid, although educational applications are being developed, such as the Dance Kit and a Toolbox graphics kit.

Smalltalk - Teaching Utility

- Has been shown to teach programming concepts as well as teamwork (designer/programmer teams formed naturally)

- Educational applications are being designed (Dance Kit and other kits), although not in routine use now

2.3 ROSS (Rule-Oriented Simulation System)

ROSS is an object-oriented language which was designed for use in simulating complex real-world systems, specifically for the SWIRL (Simulating Warfare in the ROSS Language) air battle simulator.[5] The language was developed at the Rand Corporation and is being used in the SWIRL simulator on a DEC-20 mini-computer using the TOPS-20 operating system.

ROSS (Rule-Oriented Simulation System) - Overview

Purpose:

To be used in the SWIRL simulation to provide a simulated air battle for military planners and analysts

Background:

- Developed at RAND Corp. in 1981
- Runs on DEC-20 using TOPS-20 Operating System (OS)

In ROSS, the user defines a set of objects which send and receive messages. Objects are of two types: basic, or classes of objects such as "radar" or "aircraft"; and instances, or individual members of a class such as "radar 128". The objects have associated with them a set of characteristics and behaviors; these behaviors are rules which specify an object's response to various messages it may receive from other objects. The objects are positioned in a hierarchy consisting of superclasses and subclasses. Objects inherit characteristics and behaviors from objects above them in the hierarchy.

ROSS (Continued)

Includes:

- Objects and their characteristics
  - Basic (class of objects)
  - Instances (individual)
- Behaviors (rules)
- Messages invoke behaviors in objects

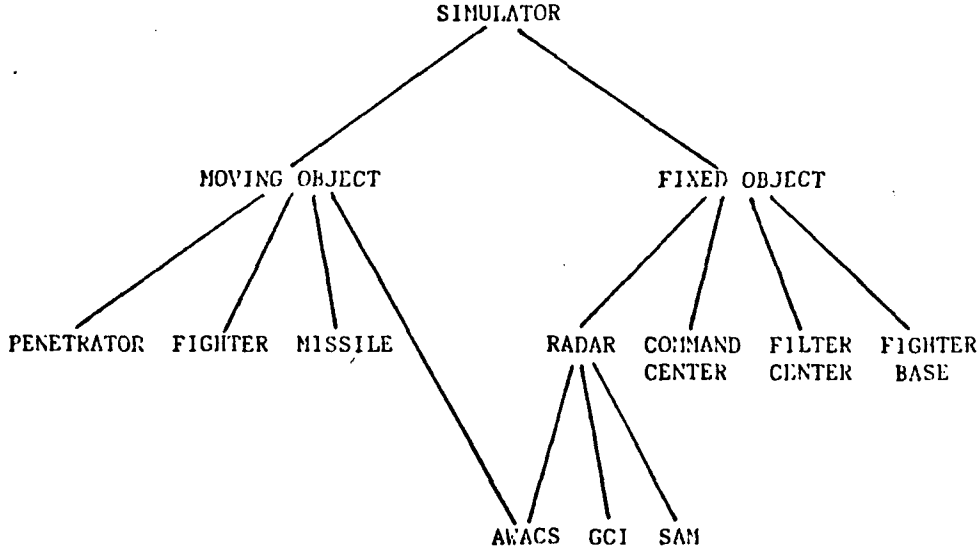
2.3.1 ROSS Example: SWIRL (Simulating Warfare in the ROSS Language)

One application of the ROSS language is the SWIRL air battle simulation. This simulation was intended as an aid to military strategists and planners, but also shows potential for use in training.

The diagram on the facing page shows the hierarchy of objects in SWIRL. Each object inherits characteristics from its parent object; thus every "radar" inherits the characteristics of a "fixed object". One object, the AWACS, inherits the characteristics of both "moving object" and "radar", since it is a moving radar.

ROSS - Example

Swirl Hierarchy of Basic Objects



The facing example illustrates the interactive quality of the ROSS language as used in SWIRL, as well as the English-like messages, both valuable characteristics for training systems.

To create a class of objects called "penetrator" the user sends a message to the basic object "moving object". In response to this message, "moving object" creates a subclass "penetrator" whose instances possess the characteristics "position", "maximum speed", "current speed", "current number of bombs", "status", and "flight plan".

ROSS - Example (Continued)

Simulating Warfare in the ROSS Language (SWIRL)

- User can create objects and give them characteristics:

```
(Ask moving-object create generic penetrator with

    position        'a position'
    max-speed       'a maximum speed'
    speed           'current speed'
    bombs           'current number of bombs'
    status          'a status'
    flight-plan     'a flight plan')
```

creates a generic "penetrator"

After creating an object, the user can define its behavior by specifying rules to govern the object's response to a message. The facing example defines a fighter's behavior when it receives a message ordering it to chase a penetrator which is guided by a GCI radar. First, the fighter cancels any plans to land, then resets its status to "scrambled", takes off if it is on the ground, and sends a message to its own radar requesting guidance to the penetrator.

ROSS - Example (Concluded)

- User can specify an object's behavior (behavior rules)

```
(Ask fighter when receiving (chase >penetrator guided
by >gci)
    (~you unplan all (land))
    (~you set your status to scrambled)
    (if (~you are on the ground) then (~you take off))
    (~requiring (~your guide-time) tell ~the gci guide
    ~yourself to ~the penetrator)).
```

defines the behavior of a fighter when it receives a "chase" message from its fighter base.

2.3.2 The Training Utility of ROSS

Although the ROSS language was not designed expressly as a training aid, several ROSS characteristics indicate its potential for use in training:

- The ROSS language allows for a high degree of interaction. The student can experiment with alternative force structures thus discovering for himself principles of conducting an air battle, or other military activities.
- The ROSS language is very close to natural English. This should shorten learning time for the student, and make the developing work self-documenting.
- The student can view the behavior rules associated with the objects, and can evaluate the consequences of modifying an object's behavior.

There are plans to expand both ROSS and SWIRL; ROSS developers hope to test the language in other than air-battle simulations, and SWIRL developers plan to upgrade its complexity and repertoire of objects. Both these improvements could result in greater flexibility for training applications.

ROSS - Training Utility

- Suitable for training use
  - Highly interactive
  - English-like (self-documenting)
  - Object behaviors easily viewed and modified
- Currently being expanded and upgraded

2.4 Conclusions

Although not all of these languages were intended as teaching aids, they have characteristics in common which would make them suitable for such use. They are interactive, giving the student immediate, visible results; they encourage the student to deduce principles for himself from his active participation; they do not require that the student have extensive experience with computers; Logo and ROSS use English-like commands, while the Dance Kit of Smalltalk-80 can be used with no typing at all. These languages also appear to foster group cooperation and individual self-confidence.

The application of these languages to education has primarily centered around programming and mathematics, but the procedures seem appropriate for teaching in other areas.

Object-Oriented Programming Languages

Conclusions

- These languages are useful for training purposes:
  - Give immediate feedback
  - Encourage self-discovery of principles
  - Use English-like commands (Logo, ROSS) or pointing devices (Smalltalk Dance Kit)
  - Encourage cooperation and confidence
- Originally used to teach programming and mathematics - apparently applicable to other areas

3.0 EXPERT SYSTEMS

3.1 General

Expert systems, also called "knowledge-based systems", are computer programs which solve problems by imitating the problem-solving techniques of human experts. These techniques include reasoning, judgment, and plausible inference. Expert systems are used when more symbolic processing rather than number processing is required.

Expert systems can generally be thought of as consisting of three components - the knowledge base, an inference mechanism, and the working memory.(6) The knowledge base contains facts about the problem area as well as heuristics, rules-of-thumb or *guidelines for good guessing. These heuristics* often take the form of IF-THEN rules and are derived by a "knowledge engineer" through interviews with a human expert in the problem area.(7) The inference mechanism, also known as the "inference engine" or "rule interpreter" contains the methodology for using the knowledge base. The working memory or data bases, is used to keep track of the progress of the problem solving and to maintain data about the problem, such as the status of variables supplied by the user.

II EXPERT Systems

Definition:

Computer programs which solve problems by imitating the problem solving techniques of human experts (also termed "knowledge-based" systems)

Consist of:

- Knowledge base
  - "Textbook" facts
  - Heuristics or "rules of thumb" (IF-THEN rules)
- Inference mechanism for using the knowledge ("inference engine", "rule interpreter")
- Working memory to keep track of progress and to contain the problem data

Not all problems require the use of an expert system. Expert systems are expensive and time-consuming to build, so relatively straightforward problems or those solved by rapid numerical calculation are better solved by traditional programs.

In the past, expert systems have worked well in problem areas with the following characteristics:

1. There existed at least one known expert in the problem area who is generally acknowledged as such.
2. The sources of his expertise were judgment and experience - knowledge that could be "taught" to a computer program.
3. The expert was able to explain his judgment and experience to a knowledge engineer.
4. The problem was well-bounded to prevent an unmanageably large search space.

Appropriate Tasks for Expert Systems

- At least one human expert
- Source of expertise must be judgment, experience
- Expert must be able to explain his knowledge and experience, and ways of applying them
- Well-bounded subject area

3.2 Expert Systems in Education

The facing chart lists the expert systems used in training which will be discussed in the remainder of this document. MYCIN/NEOMYCIN/TEIRESIAS, along with EMYCIN and GUIDON, are grouped together in section 3.2.1 MYCIN-Decimal Expert Systems because of their close relationships to MYCIN, while WEST and WUSOR-II, the two examples of computer coaches, are grouped together in section 3.2.7 Coaching Systems; the other systems are treated individually.

Expert Systems in Education

PROGRAM	DEVELOPER	USE
MYCIN/NEOMYCIN/TEIRESIAS	Stanford/HPP	Medical diagnosis
EMYCIN	Stanford/HPP	Domain independent diagnosis
GUIDON	Stanford/HPP	Medical diagnosis
SOPHIE	BBN	Electronic circuit diagnosis
STEAMER	BBN	Teaches steam plant operation
SCHOLAR	BBN/MIT	Geography, ARPANET
WHY	BBN	Cause of rainfall
BUGGY	BBN	Diagnosis of arithmetic errors
WEST	BBN	Arithmetic coach
WUSOR-II	MIT	Coach for WUMPUS game

3.2.1 MYCIN Derived Expert Systems

3.2.1.1 MYCIN. MYCIN is an expert system which was designed to assist in diagnosing bacterial infections in the blood, and to recommend treatment. MYCIN contains a list of 100 possible diseases against which to compare symptoms and other data supplied by the user; its diagnosis consists of an ordered list of the most probable diseases.

MYCIN was developed at Stanford University's Heuristic Programming Project (HPP). Like many other expert systems, MYCIN is written in the INTERLISP language; its knowledge base contains 500 IF-THEN rules. The system runs on the XEROX 1100 Scientific Information Processor and is available on the SUMEX-AIM (Stanford University Medical Experiment - Artificial Intelligence in Medicine) network.[8]

MYCIN has been one of the more successful expert systems, its performance having equalled that of human experts as tested at the Stanford University Medical School. MYCIN has also been the basis of other extended expert systems such as EMYCIN, NEOMYCIN, PUFF, and teaching programs such as GUIDON, and GUIDON II.

MYCIN

Purpose:

- To diagnose and prescribe treatment for meningitis and other bacterial infections in blood
- Produces a list of possible diagnoses from 100 available diseases

Background:

- Developed by Stanford Medical School and HPP
- Knowledge base of 500 IF-THEN rules
- Written in INTERLISP
- Origin of several other developments (EMYCIN, NEOMYCIN, PUFF, GUIDON, GUIDON II)
- Runs on XEROX 1100 (LISP machine)
- Performance has equalled that of human experts
- Current status: available on SUMEX-AIM network

MYCIN exhibits a characteristic found in many expert systems, which gives them potential for training applications. This characteristic is transparency, the ability of the system to give the user a trace of its logic and inference processes. In the case of MYCIN, this transparency is provided by TEIRESIAS.

TEIRESIAS is a knowledge-acquisition system whose primary purpose is to assist a human expert in constructing and modifying MYCIN's knowledge base. Through this knowledge-acquisition system a human expert communicates with MYCIN, the performance program, to discover why and how diagnostic errors are being made. In response to the "WHY" and "HOW" commands, TEIRESIAS moves backwards through MYCIN's reasoning steps until the expert identifies the error. The expert can then correct the error by adding to or changing the knowledge base.

However, the "WHY" and "HOW" command can also be used by a student to request an explanation of MYCIN's line of reasoning. In addition, the "RULE" query causes MYCIN to cite and explain the most recently used rule. Although the availability of these commands suggests that MYCIN could be used for instructional purposes, problems were encountered when this use was attempted. These problems will be outlined in the discussion of the GUIDON system.

Transparency in MYCIN

- Lets the user in on its "thought" processes
- TEIRESIAS
  - Knowledge-acquisition system
  - WHY and HOW trace inference procedures
  - Can also be used by student
- Potential for training, but problems encountered

The following three figures illustrate a sample consultation with MYCIN, showing the gathering of patient data, the WHY inquiry, and MYCIN's diagnosis and treatment prescription.

A MYCIN Consultation

```
        NAME                    AGE             SEX             RACE
#) **   JOHN DOE                34              MALE            CAUCASIAN

2)  HAVE THERE ANY POSITIVE CULTURES OBTAINED FROM JOHN DOE?
**  NO

3)  ARE THERE ANY PENDING CULTURES OF JOHN DOE?
**  YES

        SITE            DATE                    STAIN           METHOD OF
                        COLLECTED               EXAMINED?       COLLECTION
                                                                (URINE, SPUTUM ONLY)

  4)  ** CSF            30 JAN 77               YES

              STAIN OF SMEAR  MORPH
              (TYPE NONE IF NO ORGANISM WAS SEEN ON THE SMEAR)
        5)  ** NONE
        (NO OTHER PENDING CULTURES.)

[CONSIDERING ORGANISMS (OTHER THAN THOSE ISOLATED FROM CULTURES) WHICH MIGHT BE PRESENT...]

13) HAS JOHN DOE RECENTLY HAD SYMPTOMS OF PERSISTENT HEADACHE OR OTHER ABNORMAL
NEUROLOGIC SYMPTOMS (DIZZINESS, LETHARGY, ETC.)?
** YES

14) HAS JOHN DOE RECENTLY HAD OBJECTIVE EVIDENCE OF ABNORMAL NEUROLOGIC SIGNS
    (NUCHAL RIGIDITY, COMA, SEIZURES, ETC.) DOCUMENTED BY PHYSICIAN
    OBSERVATION OR EXAMINATION?
** YES

THE CSF CULTURES WILL BE CONSIDERED TO BE ASSOCIATED WITH MENINGITIS
15) PLEASE GIVE THE DATE ON WHICH CLINICAL EVIDENCE (SYMPTOMS, SIGNS, OR
    LABORATORY TESTS) OF MENINGITIS FIRST APPEARED.

PLEASE ENTER CSF FINDINGS IN THE FOLLOWING TABLE

        CSF             PERCENT         PROTEIN         CSF             PERIPHERAL
        RBC             PMN                             GLUCOSE         GLUCOSE
25) **  100             99              900             0               UNKNOWN
```

Gathering Patient Data

A MYCIN Consultation (Continued)

```
34) IS MENINGITIS A HOSPITAL-ACQUIRED INFECTION?
** NOT

[3.0] THIS WILL AID IN DETERMINING THE ORGANISMS (OTHER THAN THOSE SEEN ON CULTURES OR
      GRAMS) WHICH MIGHT BE CAUSING THE INFECTION.  IT HAS ALREADY BEEN ESTABLISHED THAT
        [3.1] THE INFECTION WHICH REQUIRES THERAPY IS MENINGITIS, AND
        [3.2] ORGANISMS WERE NOT SEEN ON THE STAIN OF THE PENDING CSF CULTURE, AND
        [3.3] THE TYPE OF THE INFECTION IS BACTERIAL, AND
        [3.4] NEUROSURGERY IS NOT ONE OF THE RELEVANT ITEMS FROM THE
              HISTORY OF JOHN DOE, AND
        [3.5] THE AGE OF JOHN DOE IS GREATER THAN 10 DAYS
      THEREFORE, IF
        [3.6] THE INFECTION WAS ACQUIRED WHILE THE PATIENT WAS HOSPITALIZED
              THEN THERE IS EVIDENCE THAT THE ORGANISMS (OTHER THAN THOSE
              SEEN ON CULTURES OR GRAMS) WHICH MIGHT BE CAUSING THE
              INFECTION ARE E. COLI ( 75) STAPHYLOCOCCUS-COAG-POS ( 3)
              PSEUDOMONAS-AERUGINOSA ( 3) KLEBSIELLA-PNEUMONIAE ( 5)

[BACK TO QUESTION 34...]
** END
```

The "Why" Inquiry

A MYCIN Consultation (Concluded)

```
        MY THERAPY RECOMMENDATIONS WILL BE DESIGNED TO TREAT FOR ORGANISMS
THAT EITHER ARE VERY LIKELY OR, ALTHOUGH LESS LIKELY, WOULD HAVE A SIGNIFICANT
EFFECT ON THERAPY SELECTION IF THEY WERE PRESENT. IT IS IMPORTANT TO COVER
FOR THE FOLLOWING PROBABLE INFECTION(S) AND ASSOCIATED ORGANISM(S).

INFECTION-1 IS MENINGITIS

  * <ITEM-1>  E. COLI [FROM CLINICAL EVIDENCE ONLY]
  * <ITEM-2>  PSEUDOMONAS-AERUGINOSA [FROM CLINICAL EVIDENCE ONLY]
  * <ITEM-3>  KLEBSIELLA-PNEUMONIAE [FROM CLINICAL EVIDENCE ONLY]
  * <ITEM-4>  DIPLOCOCCUS-PNEUMONIAE [FROM CLINICAL EVIDENCE ONLY]

40) DOES JOHN DOE HAVE A CLINICALLY SIGNIFICANT ALLERGIC REACTION TO ANY ANTIMICROBIAL AGENT?
** NO
[CONSIDERING AMPICILLIN-AND-GENTAMICIN FOR USE AGAINST INFECTION-1...]
41) DO YOU HAVE REASON TO SUSPECT THAT JOHN DOE MAY HAVE IMPAIRED RENAL FUNCTION?
** NO

42) JOHN DOE'S WEIGHT IN KILOGRAMS (OR <NUMBER> POUNDS)
** 70

[REC-1] MY PREFERRED THERAPY RECOMMENDATION IS AS FOLLOWS:
   IN ORDER TO COVER FOR ITEMS <1 2 3 4>
        GIVE THE FOLLOWING IN COMBINATION:
        1) AMPICILLIN
        DOSE: 3.5 G (30.0 ML) Q4H IV (CALCULATED ON BASIS OF 50 MG/KG)

        2) GENTAMICIN
        DOSE: 119 MG (3.0 ML, 80 MG/2 ML AMPULE) Q8H IV (CALCULATED ON
              BASIS OF 1.7 MG/KG) PLUS CONSIDER GIVING 5 MG Q24H
              INTRATHECAL
        COMMENTS: MONITOR SERUM CONCENTRATIONS

        SINCE HIGH CONCENTRATIONS OF PENICILLINS CAN INACTIVATE AMINOGLYCOSIDES,
        DO NOT MIX THESE TWO ANTIBIOTICS IN THE SAME IV BOTTLE.
```

Diagnosis and Treatment

3.2.1.2 EMYCIN. EMYCIN, or "essential MYCIN" is an example of an expert system (MYCIN) which has been generalized to handle additional problem areas. It was developed at Stanford University to transfer the diagnostic and consultation capabilities of MYCIN (the "essential" component of MYCIN, its inference engine) to fields other than the study of blood diseases.[9]

The variety of problem areas that can be investigated by using an existing inference engine and substituting a new knowledge base is illustrated by the fact that EMYCIN was one expert system used in diagnosing an oil spill problem as a part of the Expert System Workshop conducted in August 1980 by the National Science Foundation/Advanced Research Projects Agency (NSF/ARPA). In addition, PUFF, which consists of EMYCIN and a knowledge base of pulmonary diseases, has been used successfully at the Stanford University Medical School to diagnose such diseases.

EMYCIN (Essential MYCIN)

Purpose:

To transfer MYCIN's diagnostic and consultation capabilities to fields other than study of blood infections

Background:

- Developed at Stanford
- Domain-independent version of MYCIN

Current Status: Was one of the eight systems tested in diagnosing an oil spill problem as part of the Expert System workshop in August, 1980 (NSF/ARPA)

- Used successfully in PUFF to diagnose pulmonary disease

3.2.1.3 GUIDON. GUIDON demonstrates that an expert system not intended as a teaching aid can be u. ' for that purpose by augmenting an existing knowledge base. In the case of GUIDON, the MYCIN rule-base is supplemented by a separate knowledge base which contains 200 rules for tutoring. The system teaches by ask.ng questions of the student, and correcting his answers.(10)

GUIDON was written in 1979 by Bill Clancey at Stanford University, and is currently available on the SUMEX-AIM network, although it is not being used. GUIDON has been superseded by an improved version, GUIDON II, which uses NEOMYCIN, a newer version of MYCIN, and which is now under development.(11)

There were two problems associated with using GUIDON with MYCIN's rule base. First, MYCIN's knowledge base is too narrow for teaching a student to be a primary diagnostician. MYCIN presupposes the possibility of a bacterial infection and interprets the results from cultures of the patient's blood and cerebral-spinal fluid. However, MYCIN does not provide, nor quiz the student on, knowledge of when to suspect such an infection. In NEOMYCIN, the knowledge base has been expanded to include this kind of knowledge.

Second, MYCIN does not organize and use its knowledge as a human expert does, for example by investigating a hierarchy of causes. Specifically, it was determined that diagnostic procedures which are imbedded in MYCIN's rules should be extracted from the rules and put into a separate psychological model of human diagnostic procedures so that they can be taught to the student. This has been done in NEOMYCIN.(12)

GUIDON

Purpose:

To use MYCIN as a basis for tutoring

- 2 data bases
  - 200 rules for tutoring
  - Rules for subject area
- Asks questions, corrects answers

Background: Written in 1979 by Bill Clancey at Stanford

Current Status:

- On SUMEX-AIM network, but not being used
- Superseded by NEOMYCIN/GUIDON II, under development
- 2 problems
  - MYCIN's knowledge base too narrow for teaching a student to be a primary diagnostician
  - MYCIN doesn't organize and use its knowledge as a human expert does (hierarchy of causes), needs to separate diagnostic techniques imbedded in rules
- Problems addressed in NEOMYCIN/GUIDON II

3.2.1.4 Potential Training Utility of MYCIN-Derived Expert Systems. MYCIN and the group of expert systems derived from MYCIN demonstrate several principles about expert systems that indicate their potential applicability to training.

In diagnosing real patients, MYCIN's performance has equalled that of human doctors. This performance indicates that an expert system can solve real-world problems, thus increasing the probability that students trained on such a system will also be able to successfully solve problems.

EMYCIN illustrates that the essential inferencing piece of an expert system can be applied to a different subject area. This flexibility could facilitate the construction of training systems for various subject areas.

GUIDON illustrates that with some modification, an expert system can be used in training applications, even if it was designed for another purpose. MYCIN was designed as a diagnostic aid, not a trainer, but after modification of its knowledge base and the addition of a teaching module, the resulting system (GUIDON II) has been used for training.

Usefulness of MYCIN - Derived Expert Systems

- MYCIN illustrates that an expert system can give a performance comparable to that of human experts.

- EMYCIN illustrates that an expert system can be used in more than one subject area.

- GUIDON illustrates that an expert system can be used in training, even if it was designed for another purpose.

3.2.2 SOPHIE

SOPHIE (Sophisticated Instructional Environment) is an expert system which teaches electronic circuit diagnosis. Over the last decade it has been developed in three versions by Bolt Beranek and Newman, Inc.

Intended as a demonstration system, SOPHIE is not now in use. However, the system does illustrate the feasibility of using an expert system for training purposes.

SOPHIE (Sophisticated Instructional Environment)

Purpose: To teach diagnosis of electronic circuits

Background:

- Developed by Bolt Beranek and Newman, beginning in 1973
- Three versions

Current Status: Developed as a demonstration, not in use

The excerpt from a training session with SOPHIE, on the next two facing pages, illustrate SOPHIE's method of instruction. The system poses a problem to the student, providing any necessary description of the situation. A dialogue follows, with the student asking natural language questions which SOPHIE answers, also in natural language. When the student proposes an incorrect solution, SOPHIE attempts to lead him to the correct answer.(13)

A Session with SOPHIE

SOPHIE: The power supply is broken. The current front panel switch settings are as follows...

STUDENT: What is the output voltage?

SOPHIE: The output voltage is 11.7 volts.

STUDENT: What is it in a working power supply?

SOPHIE: 19.9 volts.

STUDENT: What is the voltage between R8 and Q1?

SOPHIE: 47.6 volts.

STUDENT: Replace Q6.

SOPHIE: How do you think Q6 is faulty? Is it shorted?

STUDENT: No.

SOPHIE: Is Q6 an open circuit?

STUDENT: I think so.

SOPHIE: Q6 is not faulted in that manner.

STUDENT: Was it possible that Q6 was an open circuit?

SOPHIE: Yes, that fault is consistent with all of the measurements you have made.

- Poses a problem
- Accepts natural language inquiries
- Evaluates students solution

Transparency in SOPHIE

SOPHIE: Your hypothesis that Q6 is blown is inconsistent with your measurement of the voltage at the base of Q7. If Q6 were blown, the voltage would have been zero; you measured it to be 10 volts.

3.2.3 STEAMER

STEAMER is an expert system developed to teach steam plant operation. The Navy currently uses a working model of a steam plant, which cost $1 million to build, for training in steam plant operation. STEAMER is intended as a lower-cost supplement to this facility.

Bolt Beranek and Newman, Inc. began developing STEAMER in 1981; the system is now ready for testing. It runs on a Symbolics LM-2 computer (a LISP machine) and uses an AED 512 color terminal to display diagrams of the steam plant simulation. These diagrams can be modified by using a graphics editor.(14)

STEAMER

Purpose:

- To teach steam plant operation
- To supplement a working steam plant currently used by the Navy for training

Background:

- Developed by BBN, beginning in 1981
- Runs on symbolics LM-2 computer
- Animated color diagrams of steam plant simulation
- Graphics editor allows student to modify diagrams

Current Status: Preparing for testing

STEAMER consists of three main components, a simulation, a graphics display, and a graphics editor.

The heart of STEAMER is a simulation of the operation of a steam propulsion plant. The system guides the student through the operation, giving step-by-step instructions which the student follows.

The progress of the simulation is displayed by means of detailed color diagrams, which control as well as monitor the simulation. The graphics editor allows the student to modify the diagrams, and thus the simulation.

STEAMER Components

- Simulation of steam plant operations
  - Student follows STEAMER's instructions
- Color graphics portrayal of simulation
  - Diagrams monitor and control simulation
- Graphics editor
  - Student can modify diagrams

While following STEAMER's instructions, the student may at any time interrupt by typing the question "WHY?". SOPHIE will provide a detailed explanation of why the previous instruction was given, and the consequences of not following it.

Transparency in Steamer

5. *Align Drain Valves D1 and D2.*

12. *Open Input Valve I17.*

Why?

Whenever steam is admitted to a chamber, you must align the drains first.

If you don't, water left in the chamber will mix with the steam and high energy water pellets will get thrown downstream.

3.2.4 SCHOLAR

SCHOLAR, like SOPHIE, is an example of an expert system designed as a research model to demonstrate the use of a dialogue between student and computer. It was developed by Bolt Berenek and Newman, Inc., with the Massachusetts Institute of Technology. SCHOLAR was equipped with two separate knowledge bases, to enable it to train students in two areas, geography and ARPANET site information.(15)

Scholar

Purpose: To provide research model for two-way dialogue between student and computer

- Geography
- Information on ARPANET sites

Background: Developed by BBN and MIT

Current Status: Not used

SCHOLAR can either ask questions of the student and correct his answers, or answer questions posed by the students. In answering questions, or correcting answers, SCHOLAR provides appropriate additional information instead of a simple "yes" or "no". Note also, in the sample training session on the following page, that SCHOLAR can handle unexpected responses. Although the system "expects" an answer to its question about the area of Argentina, it does not construe the response "Tell me something about Peru" as an incorrect answer. Instead, it complies with the request, then guides the student back to the original question.

Sample Training Session with Scholar

Machine: What is the language of Argentina?

Student: Probably Spanish.

Machine: Very good. Approximately what is the area of Argentina?

Student: Tell me something about Peru.

Machine: Peru is a country. It is located in South America. The capital of Peru is Lima. Now answer the question you didn't answer before.

Student: The area of Argentina is 1,100,000 square miles.

Machine: Approximately true, but a more exact value is 1,200,000. The country bordering Peru on the east is ...?

3.2.6 WHY

SCHOLAR was a research model of a two-way dialogue between student and computer; WHY extends the scope of that dialogue to include not only facts, but complex systems which involve temporal and casual relationships. The subject area chosen was the cause of rainfall, since rainfall is produced by several inter-related factors.

The approach used in WHY is to model a Socratic dialogue between student and computer, in which the system encourages the student to draw on specific instances of a problem and from them to form a general conclusion. The medium for this modeling of a Socratic dialogue is WHY's set of 24 tutoring heuristics, or rules about how to teach, one of which is quoted below:

> If the student gives as an explanation of casual dependence one or more factors that are not necessary,
>
> Then select a counterexample with the wrong value of the factor and ask the student why his casual dependence does not hold in that case.

In its use of IF-THEN rules about tutoring, WHY is similar to GUIDON. WHY differs from GUIDON, however, in the form of knowledge representation employed in its domain knowledge base: while GUIDON uses IF-THEN rules in its domain knowledge base, WHY uses script-like data structures. This type of structure sequentially outlines the processes involved in causing rainfall, for example:

> First water evaporates from an ocean or sea, then the water-laden air mass is carried over land by winds, which in turn causes the air mass to cool, which causes precipitation.

WHY

Purpose:

- Extension of SCHOLAR's two-way dialogue
- Discusses complex systems, not just facts
  - Subject is causes of rainfall
  - Socratic dialogue (student generalizes from specific instances)
  - Uses tutoring heuristics and script-like data structures

Background:

- Developed in 1977 by BBN

Current Status:

- Demonstration not in use
- Under development to include long-term goals for guiding the dialogue

In 1977, Bolt Beranek and Newman, Inc. developed WHY as a demonstration program; therefore, no attempt has been made to put WHY into routine use. Efforts are continuing toward improvements in tutoring techniques, including the development of goals for guiding the dialogue as a whole, rather than merely the specifics of the interaction. These goals would resemble those generally held by human teachers, such as concentrating on one of the causes of rainfall at a time.

The sample training session on the facing page illustrates the way in which WHY encourages the student to think about the causes of rainfall, to form a general conclusion, and to revise the conclusion if necessary. The heuristics used by the system are included here at the appropriate places, although the student does not see them.

Sample Training Session with WHY

Heuristic 1: Ask for a prediction about a particular case.

Do you think the Amazon jungle has heavy rainfall or light rainfall?

**** Heavy rainfall**

Heuristic 2: Ask for prior causes.

Why does the Amazon jungle have heavy rainfall?

**** Because it's near the Atlantic**

Yes, the Amazon jungle is near the equatorial Atlantic.

Heuristic 3: Ask for intermediate causes.

How does that affect rainfall in the Amazon jungle?

**** The water in the current is warm**

⋮

Heuristic 4: Ask for subsequent causes.

What happens after the moisture-filled air reaches the land?

**** I don't know**

Heuristic 5: Suggest a factor and ask if it is a cause.

Do you think the Andes mountain range has any effect on the rainfall in the Amazon jungle?

**** Yes**

Heuristic 3: Ask for intermediate causes.

How?

**** The prevailing wind hits the Andes and the air cools**

Heuristic 3: Ask for intermediate causes.

How does the fact that the prevailing wind hits the Andes lead to air cooling?

⋮

Heuristic 6: Form a general rule for an insufficient cause.

Do you think that any place with mountains has heavy rainfall?

**** Yes**

Heuristic 7: Pick a counterexample for an insufficient cause, and Heuristic 2: Ask for prior causes.

Southern California has mountains.
Why doesn't Southern California have heavy rainfall?

3.2.6 BUGGY

BUGGY is an expert system which was designed by Bolt Beranek and Newman, Inc. expressly for teaching and has been used in elementary schools. The purpose of BUGGY is to identify a student's procedural errors, or "bugs", in mathematics. (DEBUGGY is a companion system which concentrates on subtraction, and has identified 130 subtraction bugs.) These bugs can be modeled in the LISP language as alterations to a portion of a correct procedure. Analysis of BUGGY's conclusions about the errors made by a group of students will reveal patterns which can be of value in designing instructional programs. BUGGY can also be used as a training device for teachers, giving them practice in diagnosing their students' procedural errors.(16)

BUGGY

Purpose:

- To identify student's procedural errors in mathematics (Debuggy focuses on subtraction)

Background:

- Developed by BBN
- Not interactive with student, but diagnoses error-types
- Results can be used in designing instructional programs
- Useful in training teachers to diagnose errors

Current Status: Has been used in elementary schools

It has been shown that, rather than proceeding unsystematically and thus making random errors, children do follow procedures; their errors result from faulty procedures. The excerpted dialogue on the facing page demonstrates how BUGGY can be used to train teachers in diagnosing these procedural "bugs".

BUGGY selects a bug and replaces a portion of the correct procedure with it, then 'solves' problems which the teachers supply, using the faulty procedure. BUGGY does not have the ability to understand natural language; therefore, when the teachers believe they have identified the bug, they must test their theory by attempting to arrive at the same incorrect answers as BUGGY to a set of five problems, rather than simply describing their bug. When they have correctly identified the bug, BUGGY provides a verbal description of it.

Using BUGGY to Train Teachers·

- Presents incorrectly solved problems to teachers, supervises their diagnosis of the bug.

- Teachers suggest problems which BUGGY solves incorrectly, using the bug.

- Teachers believe they have identified the bug, BUGGY provides sample problems for them to solve to test their theory.

Welcome to Bugland. I have chosen a bug. Here is an example of the bug.

$$\begin{array}{r} 17 \\ +\ 5 \\ \hline 13 \end{array}$$

$$\begin{array}{r} 18 \\ +\ 6 \\ \hline 15 \end{array} \qquad \begin{array}{r} 43 \\ +79 \\ \hline 23 \end{array}$$

$$\begin{array}{r} 21 \\ +39 \\ \hline 51 \end{array}$$

That's not the bug I have.

Using my bug:

$$\begin{array}{r} 21 \\ +39 \\ \hline 15 \end{array}$$

- Teachers correctly identify the bug, prove their theory by obtaining the same incorrect answers as BUGGY to a set of problems.

Very good. My description of the bug is: the student always sums up all the digits, with no regard to columns.

3.2.7 Coaching Systems

One type of expert system which is used specifically for instruction is the computer coach. A coach is a computer-based learning environment in which a student plays a game while the system observes his behavior, offering, at appropriate times, criticism or suggestions for improving his chances of winning.

The purpose behind coaching systems is to assist the student in guided discovery learning; that is, the system must take every opportunity to help the student learn the skills necessary to play the game, but must encourage him to discover principles for himself, and avoid interrupting so often that the student no longer enjoys the game.

Two coaching systems will be briefly discussed here, WEST and WUSOR-II.

Coaching Systems

- Computer-based learning environment
  - Game
  - Criticism and advice from system

Purpose: Guided discovery learning

- Assist at every opportunity
- Encourage self-discovery
- Avoid interrupting too often

3.2.7.1 WEST. WEST is the computer coach for the computer game "How the West Was Won", which is intended to give students drill and practice in arithmetic. The game is similar to the board game "Chutes and Ladders" and requires the player to use the operations of addition, subtraction, multiplication, and division to construct an arithmetic expression from three numbers which he receives from spinners. He moves around the board a number of spaces corresponding to the value of the expression, and attempts to reach the end before the other players. Since landing on certain squares can move a player forward or backward, strategies must be used to capitalize on this feature of the game.

There were two main goals involved in constructing WEST. One was to identify those diagnostic strategies which are necessary for inferring the student's bugs, or misunderstandings, from his behavior as observed by the coach. In this effort, the system is aided by the process of differential modeling. In differential modeling the system compares the student's actions with the actions an expert would take in the same situation, and determines the underlying skills used in both. The other goal was to identify useful tutoring strategies to assist the system in interrupting the student at the right time, and to give him the proper coaching when it does interrupt.

Developed in 1977 by Bolt Beranek and Newman, Inc., WEST has been used in a controlled experiment in elementary school classrooms. The students were divided into two groups, one of which played the uncoached version of "How the West Was Won", and the other played the coached version. It was found that the coached players used a greater variety of arithmetic expressions, and that they enjoyed the game more.

WEST

Purpose: Computer coach for computer simulated board game "How the West Was Won" (arithmetic drill and practice)

- Identify diagnostic strategies needed to infer students "bugs"
  - Differential modeling (student behavior vs. expert)
- Identify tutoring strategies

Background:

- Developed in 1977 by BBN

Current Status:

- Used in elementary schools
  - Students used greater variety of patterns in mathematical expressions
  - Enjoyed coached version more than uncoached version

3.2.7.2 WUSOR-II. WUSOR-II is a coach for the computer game WUMPUS, in which the player attempts to locate a creature called the "WUMPUS" in a maze of treacherous caves, and to kill him. The player is given hints which encourage him to succeed by using logic, probability, decision theory and geometry.

The system is composed of four modules: the Expert, the Psychologist, the Student Model, and the Tutor. The Expert's function is to tell the Psychologist if the player's move is nonoptimal; from this information the Psychologist forms a hypothesis about the skills known to the student; the Student Model uses this hypothesis as an overlay model, to represent the player's knowledge as a subset of the Expert's; finally, the Tutor interacts with the player, based on the context of the Student Model.

WUSOR-II was developed at the Massachusetts Institute of Technology in 1977, and has been tested on a group of twenty players of various ages. The results were judged informally through interviews with the students, who considered WUSOR-II to be valuable learning aids.

WUSOR-II

Purpose: Coach for computer game WUMPUS (student tracks and tries to kill the "WUMPUS")

- Teaches logic, probability, decision theory, geometry
  - Uses four modules:
    - Expert - tells psychologist if student's move is nonoptimal
    - Psychologist - hypothesizes about skills known to student
    - Student model - represents student's knowledge as subset of expert's (overlay model)
    - Tutor - interacts with student, based on student model

Background: Developed at MIT in 1977

Current Status:

- Used with 20 students of various ages, informally judged to be a valuable learning aid

3.3 Conclusions-Expert Systems

Expert systems seem to be well-suited for training applications. The rules and facts in the knowledge base can be modified to reflect changes in the real world, such as the development of new medical treatments or new medical knowledge (the MYCIN group), or the development of new types of equipment (STEAMER). Most are interactive, giving the student immediate reinforcement, and can explain their reasoning processes. In spite of these advantages, few expert systems are currently in use for training. This is most likely due in part to the expense and time commitment required to build such systems, but also because the successes of such systems are fairly recent and few educational agencies are aware of their potential.

Expert Systems - Conclusions

- Expert systems are well-suited to training programs:
  - Rules can be modified to reflect changes in the real world
  - Interactive for immediate feedback
- Few systems are in use for training
  - Takes time and money
  - Educators not yet aware of potential

4.0 GENERAL CONCLUSIONS

Both object-oriented programming languages and expert systems can be valuable teaching aids; however, although there have been many research efforts, there are as yet few such programs, especially ones using expert systems, in routine use. Still fewer of these programs are specifically designed for training, especially military training.

Existing expert systems do show a potential for training use, however, as demonstrated by EMYCIN and ROSS. Such training systems could be oriented toward a military purpose, with the substitution of new knowledge bases. Possible uses include training for equipment maintenance and communications network configuration. Object-oriented programming languages which teach programming skills could be used by the military without alteration.

Artificial Intelligence-Based Teaching Aids

Conclusions

- Many research efforts, fewer programs in routine use
- Few programs specifically military or training-oriented
- Both object-oriented languages and expert systems can be useful aids in teaching
  - Several programs not intended primarily as teaching aids could be used that way (EMYCIN, ROSS)
  - Non-military programs could be oriented toward military applications, with substitution of new knowledge base (equipment maintenance, communication network configuration)

GLOSSARY

ARPA	Advanced Research Projects Agency
AWACS	Airborne Warning and Control System
BBN	Bolt Beranek and Newman
CAI	Computer-Aided Instruction
GCI	Ground Controlled Intercept
HPP	Heuristic Programming Project
ICAI	Intelligent Computer-Aided Instruction
MIT	Massachusetts Institute of Technology
NSF	National Science Foundation
PARC	Palo Alto Research Center
ROSS	Rule-Oriented Simulation System
SOPHIE	Sophisticated Instructional Environment
SUMEX-AIM	Stanford University Medical Experiment - Artificial Intelligence in Medicine
SWIRL	Simulating Warfare in the ROSS Language

REFERENCES

1. Xerox Electro-Optical Systems Division, briefing on the Smalltalk language and system, at The MITRE Corporation, McLean, Virginia, September 16, 1982.

2. William Clancey and Bruce Buchanan, Exploration of Teaching and Problem-Solving Strategies, 1979 - 1982, Report No. STAN-CS-82-910, Stanford University, Stanford, CA, May 1982.

3. Harold Abelson, "A Beginners Guide to Logo", Byte Magazine, Vol. 7, No. 8, August 1982.

4. Adele Goldberg and Joan Ross, "Is the Smalltalk-80 System for Children?" BYTE Magazine, Vol. 6, No. 8, August 1981.

5. Philip Klahr, "Overview of the ROSS Simulation System", Proceedings of the 10th IMACS World Congress on Systems Simulation and Scientific Computation, Montreal, Canada, August 8-13, 1982.

6. William B. Gevarter, Overview of Expert Systems, National Bureau of Standards, Washington, D.C., May 1982.

7. Richard O. Duda and John G. Gaschnig, "Knowledge-Based Expert Systems Come of Age", BYTE Magazine, Vol. 6, No. 9, September 1981.

8. Robin Webster and Leslie Miner, "Expert Systems Programming Problem-Solving", Technology, January/ February 1982.

9. Tom Alexander, "Practical Applications for a 'Useless Science'", Fortune, May 31, 1982.

10. William Clancey, Overview of GUIDON, Stanford University, Stanford, CA, July 1982.

11. William Clancey, telephone conversation, September 1982.

REFERENCES

(Concluded)

12. William Clancey and Reed Letsinger, NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching, Report No. STAN-CS-82-908, Stanford University, Stanford, CA, May 1982.

13. Randall Davis and Charles Rich, "A Tutorial on Expert Systems, Part 2 - Application Areas", AAAI 1982 Conference Tutorial Program, August 17, 1982.

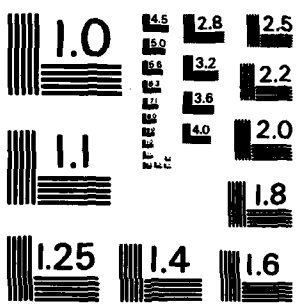
14. Randall Davis and Charles Rich, "A Tutorial on Expert Systems, Part 2 - Application Areas", AAAI 1982 Conference Tutorial Program, pp. 30-37, August 17, 1982.

15. William A. Woods, Bolt Beranek and Newman, Inc., telephone conversation, September 1982.

16. Avron Barr and Edward Feigenbaum, The Handbook of Artificial Intelligence, Volume II, 1982.

AD-A127 702 AN OVERVIEW OF ARTIFICIAL INTELLIGENCE-BASED TEACHING AIDS(U) MITRE CORP MCLEAN VA P K GROVESTON DEC 82 F19628-83-C-0001 2/2

UNCLASSIFIED F/G 9/2 NL



DISTRIBUTION LIST

INTERNAL

A-10 C. A. Zraket

D-10 K. E. McVicar

D-14 D. S. Alberts
J. S. McManus
A. J. Tachmindji
W. B. Woodward

W-30 J. W. Benoit
A. V. Lemmon
J. M. Selander

W-70 G. Carp
E. Famolari
P. G. Freck
R. P. Granato
R. A. Joy
F. W. Niedenfuhr
E. L. Rabben
W. A. Tidwell

W-72 C. W. Sanders

W-73 T. H. Nyman

W-74 T. T. Bean
R. P. Bonasso

INTERNAL (Concluded)

J. R. Davidson
Z. Z. Friedlander
M. Gale
P. K. Groveston
C. R. Holt
E. P. Maimone
R. O. Nugent (10)
W. E. Zeiner

W70 Information Center

EXTERNAL

The Army Model Management Office
U.S. Army Combined Arms Center
ATTN: ATZL-CAN-DO
Major John Strand (10)
Ft. Leavenworth, KS 66027

Army Library
ATTN: ANR-AL-RS
(Army Studies)
Room 1A518
Pentagon
Washington D.C. 20310

DISTRIBUTION LIST

(Concluded)

EXTERNAL (Continued)

Commander
Defense Technical Information Center
ATTN: DDA
Cameron Station
Alexandria, VA 22314

Commander
U.S. Army Intelligence Center and School
ATTN: ATSI-TD-EW (COL Fichtel, CPT D. Velicki)
Ft. Huachuca, AZ 85613

Commander
Soldier Support Center
ATTN: ATSG-DCD-AD
(LTC Crosby)
Ft. Benjamin Harrison, IN 46216

HQ, Department of the Army
SAUS-OR (Mr. Hollis, Dr. Fallin)
DAMO-ZD (Mr. Vandiver)
DAMA-ZD (Mr. Woodall)
Washington, D.C. 20310

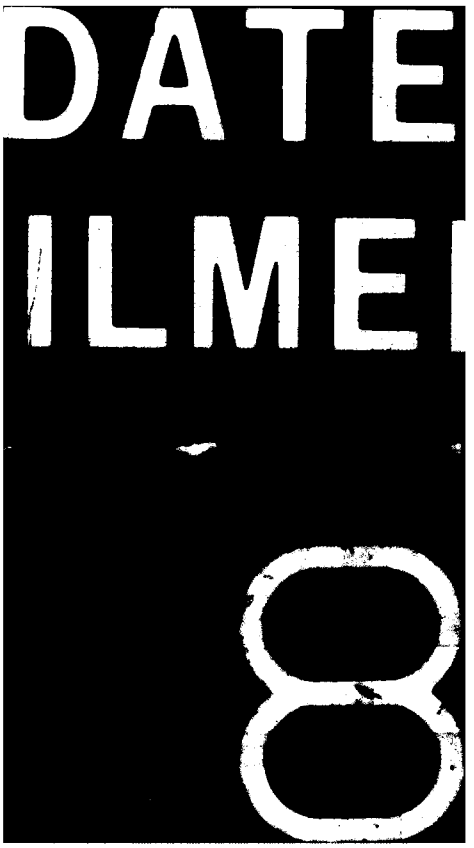
EXTERNAL (Concluded)

Commander
Director, U.S. Army TRADOC Systems Analysis Activity
ATTN: ATAA-TC (Mr. Mathiasen)
White Sands, NM 88002

Commander
Director, U.S. Army Research Institute
ATTN: Dr. Ruth Phelps
5001 Eisenhower Avenue
Alexandria, VA 22333

Deputy Commander, Combined Arms Operations Research Activity
ATTN: ATOR-CAT-D (LTC Childs)
Ft. Leavenworth, KS 66027

Commander
U.S. Army Training and Doctrine Command
ATTN: ATCD-AT (Mr. Goldberg)
Ft. Monroe, VA 23651


DATE
ILME
8