Human-Computer Interaction

in

Distributed Supervisory Control Tasks

Christine M. Mitchell
Principal Investigator
Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
(404) 894-4321
mitchell@chmsr.gatech.edu

Semi-Annual Report
Goddard Space Flight Center
NAG 5-1044

Walt Truszkowski, Technical Monitor
Deliverable No. 1

January 1989

OMIT

Overview

The initial six months of this research grant consisted of two major efforts: extension of existing work on OFMspert, and initiation of research on tutoring principles and models for complex, dynamic system, in particular, applications in Goddard Space Flight Center (GSFC) ground control systems. This research builds on the work done under the previous GSFC contract (ending December 1987) and research supported by NASA Ames Research Center, NAG 2-413 (Everett Palmer, Technical Monitor).

OFMspert Research

At the commencement of this grant, the first phase of the Ally empirical evaluation was performed. Ally is the implementation of OFMspert with control properties and a particular operator interface. The experiment evaluated the effectiveness of a supervisory control team consisting of a human operator and Ally versus a control team consisting of two human operators. The experiment was carried out in the GT-MSOCC (Georgia Tech MultiSatellite Operations Control Center) domain, a Georgia Tech research tool consisting of a high fidelity implementation of the operator interface to MSOCC, a GSFC ground control system.

The Ally experiment was one of the first of its kind: a rigorous empirical evaluation of the effectiveness and dynamics of a cooperative team of a human operator and an expert system. The Ally research comprised the doctoral dissertation for Major James B. Bushman (Air Force); the research was successfully defended in December 1988. The dissertation will be published this spring as a Center for Human-Machine Systems Research technical report. In addition, the research will be prepared for publication in several conference proceedings and two journal papers. Preliminary results of this research were presented at a colloquium in October at Goddard Space Flight Center and at

the annual symposium on the Empirical Foundations of Information and Software Sciences (EFISS). Viewgraphs from the GSFC colloquium and EFISS presentations are contained in Appendix A.

In another aspect of this work, Ms. Patricia M. Jones won the annual Human Factors Society award for the best student paper at the 1988 meeting in October. Her paper, contained in Appendix B, was based on her masters thesis, an extensive empirical evaluation of the intent inferencing capabilities of OFMspert. Appendix C contains a second paper describing OFMspert research and interaction that was presented at EFISS conference and published in the conference proceedings and as Center technical report. A more complete version of her work is in the review process for the *International Journal of Man-Machine Studies.*

The final portion of OFMspert research conducted during this period was the port of the OFMspert/Ally code from Smalltalk-80 to Allegro Common Lisp. For a variety of reasons the OFMspert/Ally system implemented in Smalltalk-80 was unstable and proving very difficult to extend in further research directions. As a result, considerable effort over the last 6 months was spent in converting OFMspert/Ally from Smalltalk-80 to Allegro Common Lisp with PCL (Portable Common Loops) object extensions. The new OFMspert implementation will run on a variety of platforms including the Macintosh II, where it was implemented, a Sun 3/60, and a NeXT machine. The conversion of this research, though time consuming, will allow the rapid extension of the work into new research directions. Copies of the Allegro Common Lisp code are available from the Dr. Christine Mitchell.

Intelligent Tutoring for Space Systems

There are three aspects the tutoring research at Georgia Tech. The first is the conclusion of the ITSSO (Intelligent Tutoring System for Satellite Operators). The next

few months should see the publication of the ITSSO technical report. ITSSO was an interesting piece of exploratory research. Contrary to expectations, ITSSO did not result in enhanced operator performance. The experiment compared subjects trained with ITSSO with a control group whose subjects did not receive explicit tutoring but were allowed to interact freely with the simulated system (GT-MSOCC) for the same amount of time as the ITTSO subjects spent training with the tutoring system. For a range of performance measures, ITSSO-trained subjects performed no better, and in some cases worse, than subjects who were not tutored but only 'played' with the system.

Several interesting points are suggested by this research. First, the results may have been biased by the restrictions of the experimental design used in this research. The experimental design restricted ITSSO interaction to exactly the time allowed for the subjects in the control group to 'practice' with the system (about 5 hours). It may be the case that intelligent tutoring systems require more time than unstructured practice. Although the results looked negative in the context of this experiment, in actual systems a tutoring system that provide a broad and structured experience with a complex system would be worth the additional hours or days of training. Hence, one possibility suggested by the ITSSO experiment is the necessity for improved experimental design.

A second consideration in understanding the ITTSO results is the ITSSO philosophy of training: the ITSSO design was based on a 'bottom-up' perspective. Subjects were introduced to individual 'scenarios', one task or problem at a time. A great deal of the training time was spent working exclusively on a single task, a decision at the heart of the ITSSO design. In the GT-MSOCC domain, as in most supervisory control systems, the operator typically has to handle several tasks concurrently. It may be that the ITSSO training reduced the subjects' abilities to handle multiple competing tasks. The control group, however, interacted with the system using scenarios where there were frequently multiple tasks that needed to be monitored or executed--sessions more typical of the actual GT-MSOCC supervisory control functions.

The combination of the 'bottom up' tutoring decision and the restrictive experimental design in which ITSSO subjects were not allowed sufficient time in which to train in a multi-tasks environment may be the causes of the unexpected experimental results. Future research will examine these finding more closely.

Two other tutoring projects were also undertaken during this past six months. One was the examination of the possibility of extending OFMspert to be both a tutor for novices as well as an assistant for experienced operators. A copy of a preliminary report is included in Appendix D.

Finally, research is underway to assess each of the tutoring models and methodologies that Etienne Wenger (*Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*,Morgan Kaufman Publishers, Los Altos, CA, 1987) reviews in his recent text. The assessment will result in a technical report that will briefly describe each of the intelligent tutoring system (ITS) models, assess its potential contribution to tutoring operators in complex dynamic systems, such as NASA ground control systems, and illustrate its potential contribution in the context of the the GT-MSOCC domain. A preliminary draft of this work is provided in Appendix E.

N89-20693

Appendix A

View Graphs for GSFC Colloquium on OFMspert

October 1988

Human-Computer Interaction

in

Distributed Supervisory Control Tasks

Christine M. Mitchell, Principal Investigator
Center for Human-Machine Systems
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

October 1988

Overview of Georgia Tech Activities

* Operator Function Model (OFM)

* GT-MSOCC (a research laboratory)

* Model-Based Operator Workstations

* Multi-Modal Operator Interaction

* OFMspert (Operator Function Model Expert System)

* Ally: OFMspert with Control Capabilities

* .Intelligent Tutoring System for Satellite Operators
  ITSSO and OFMTutor

TYPES OF APPLICATIONS

Complex, dynamic systems with high costs or risks associated with human error.

--space

--manufacturing

--process control and distribution systems

--military C2

Human operator functions as a supervisory controller

--monitors predominantly automated control systems

--fine tunes in response to unexpected changes in predicted system behavior

--fault detection, diagnosis, and compensation

EXPERIMENTAL ENVIRONMENT

MSOCC:
MULTISATELLITE OPERATIONS CONTROL CENTER

-Actual system at NASA/GSFC

- Coordinates use of shared computer and communications equipment

- System is now manual, moving towards automation

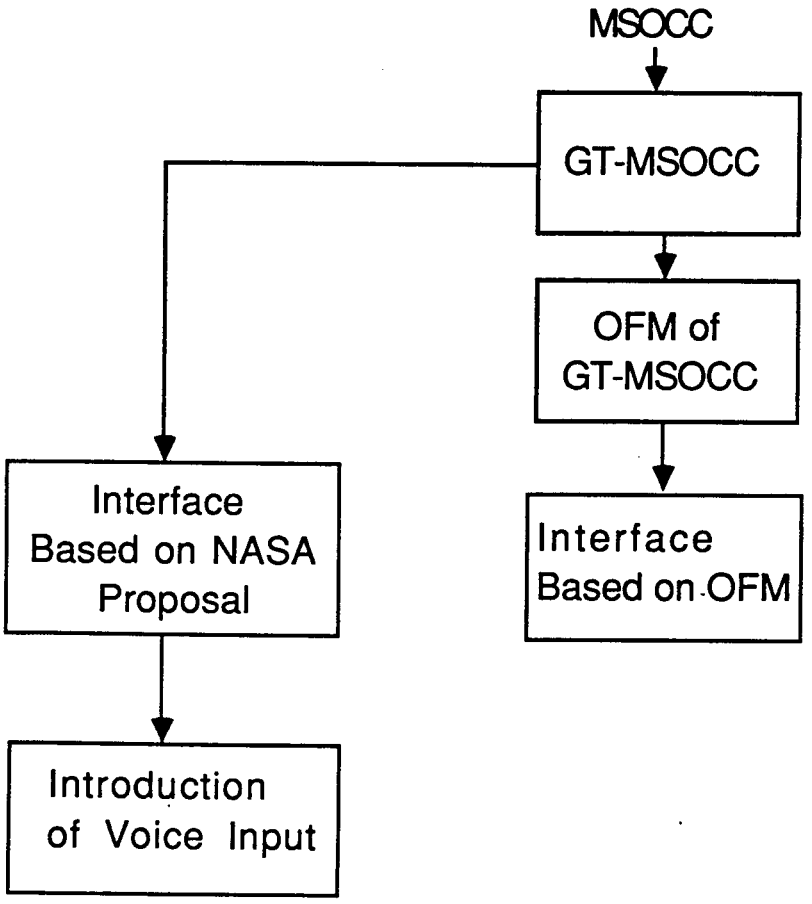
GT-MSOCC

- Developed at Georgia Tech

- Simulation of future automated MSOCC system

- Discrete event, Real Time, Interactive simulation

OVERVIEW OF GT-MSOCC RESEARCH



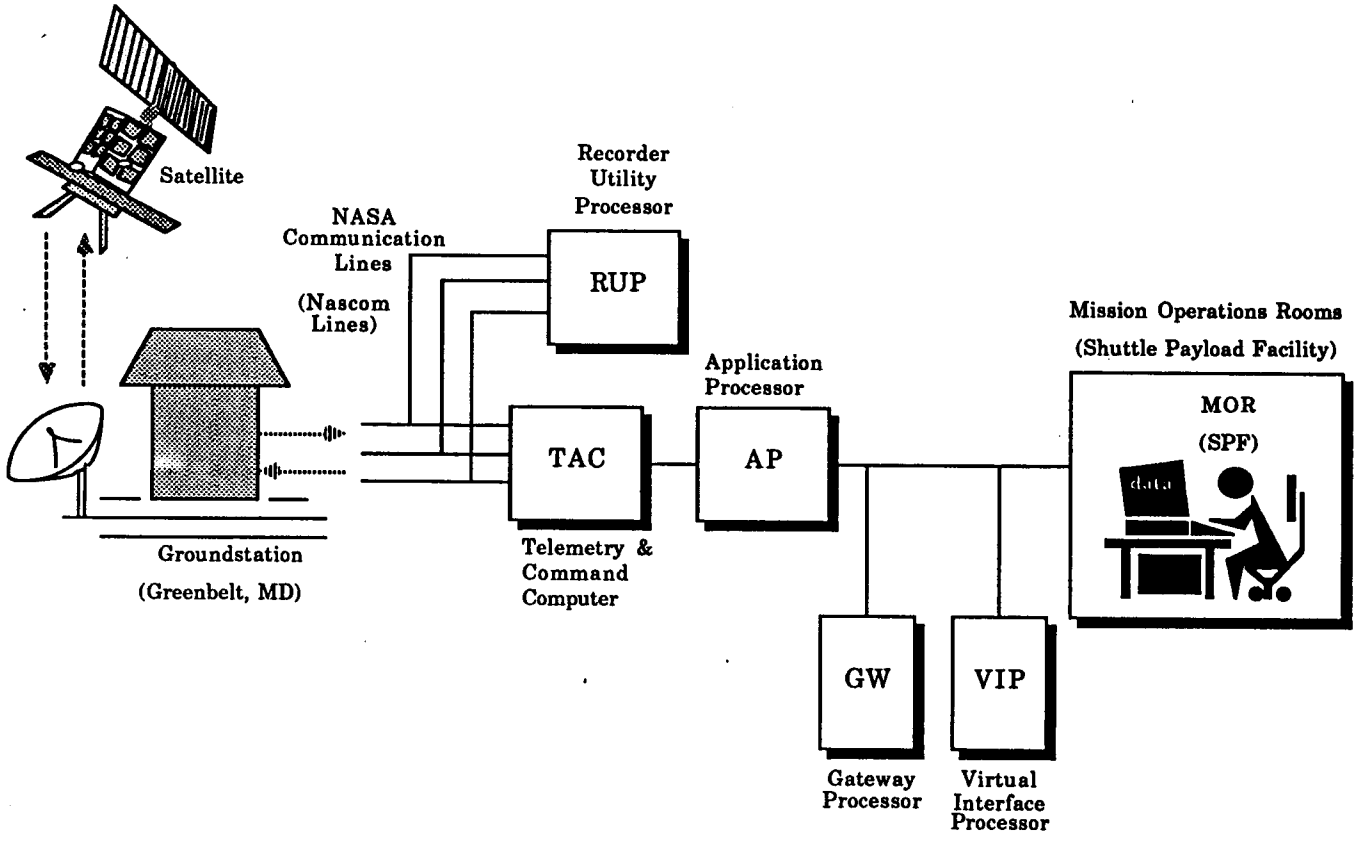


Figure 4. Multisatellite Operations Control Center

GT-MSOCC OPERATOR RESPONSIBILITIES

- SUPERVISE SPACECRAFT CONTACTS CURRENTLY BEING SUPPORTED

- COMPENSATE FOR AUTOMATED SCHEDULE PROBLEMS

- RESPOND TO REQUESTS FOR UNSCHEDULED SPACECRAFT CONTACTS

- DECONFIGURE ALL MANUALLY CONFIGURED EQUIPMENT STRINGS

Operator Function Model (OFM)

* a mathematical tool to represent operator interaction with predominantly automated space ground control systems (cognitive task analysis for system analysis and design).

* OFM's structure represents cognitive as well as physical operator tasks.

* useful for the design of operator workstations and displays (model-based iconic displays).

* useful for the design of an "intelligent" operator's associate (OFMspert and Ally).

* useful to represent the task knowledge in the design of an intelligent tutoring system (ITSSO and OFMTutor).

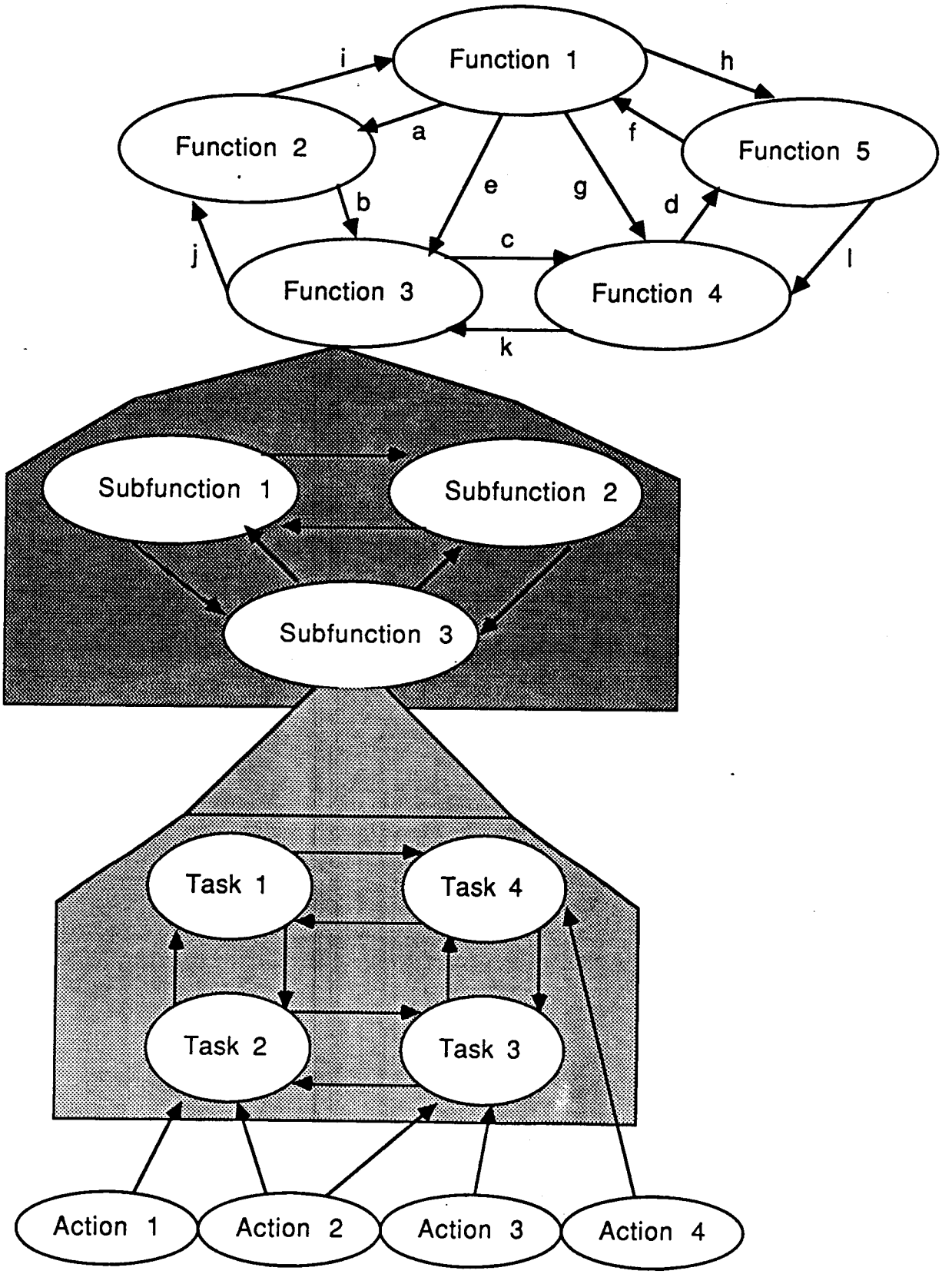


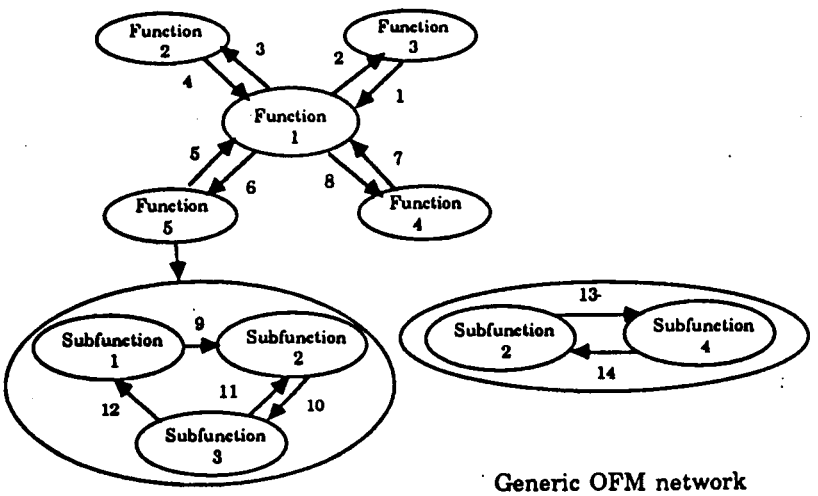
Figure 1. A Generic Operator Function Model

OFM STRUCTURE

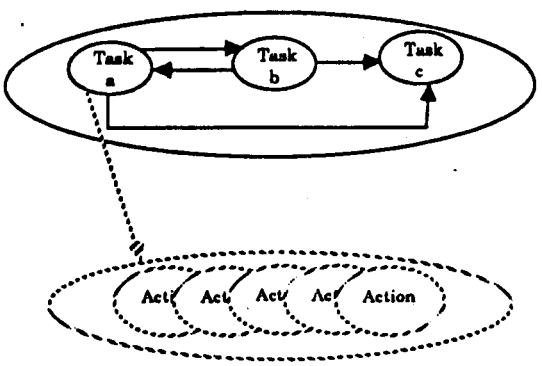
OFM is a network with nodes represented as non-deterministic, finite-state automata.

Higher level nodes represent operator goals; decomposition represents how operator coordinates control actions and system configuration so that acceptable overall system performance is reached.

Next-state transition functions model system triggering events



Generic OFM network



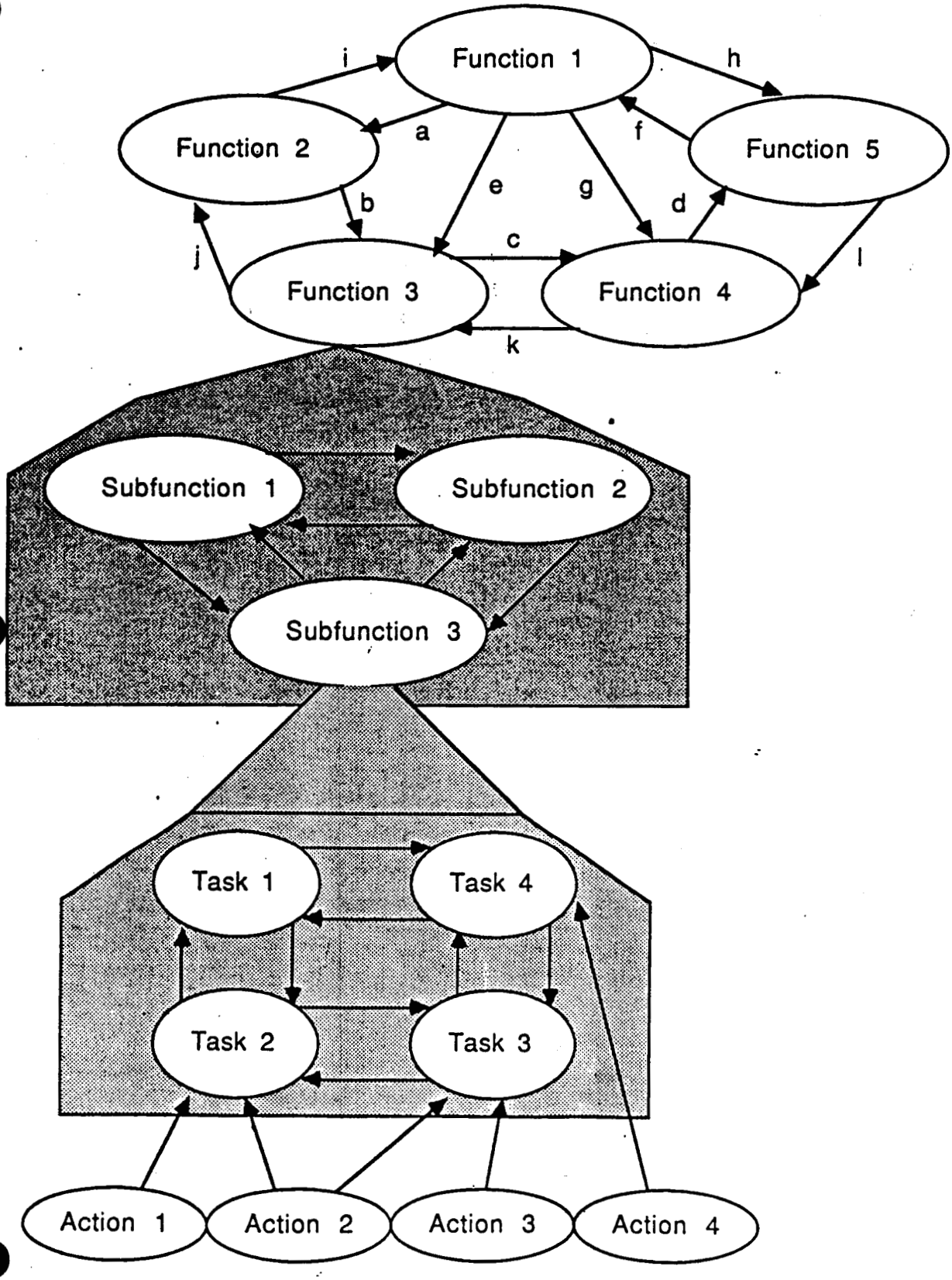
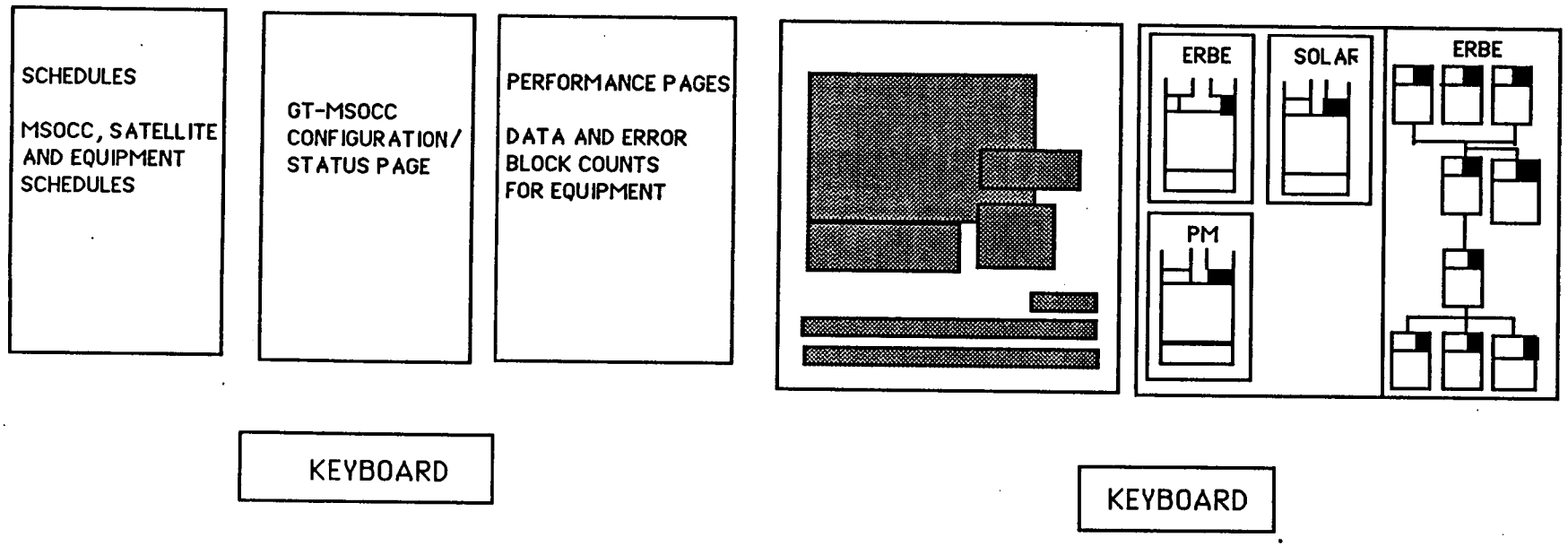


Figure 2. A Generic Operator Function Model



CONVENTIONAL WORKSTATION → MODEL-BASED WORKSTATION

Primary Features of OFM-Based Interface

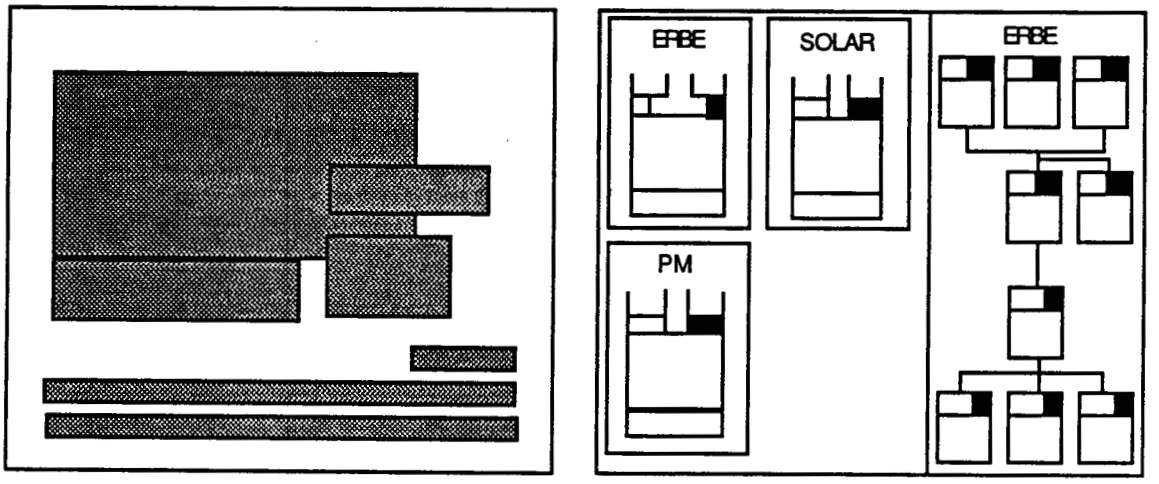
DYNAMIC ICONS

- Qualitative Representation
- High Level View of System Functioning
- Detailed View of Data Propagation

COMPUTER WINDOWS

- Alphanumeric, Overlapping Windows
- Contents Determined by OFM
- Placement Determined by OFM

TWO-MONITOR OFM-BASED INTERFACE



KEYBOARD

EXPERIMENTAL DESIGN

10 SUBJECTS USED EACH INTERFACE

12 EXPERIMENTAL SESSIONS (45 MINUTES EACH)

- 5 TRAINING SESSIONS
- 7 SESSIONS FOR DATA ANALYSIS

INDEPENDENT MEASURES

- DISPLAY CONDITION
- SESSION
- SUBJECT

DEPENDENT MEASURES

- FAULT COMPENSATION (4 MEASURES)
- EQUIPMENT CONFIGURATION AND DECONFIGURATION (5 MEASURES)
- OPERATOR ERROR (2 MEASURES)

Operator Performance Measures

Time to fix system problems:
--fix hardware failures.
--fix each of 3 software failures.
--compensate for automated schedule problems.
--deconfigure manually configured equipment

Number of operator errors:
--operator caused schedule conflicts.
--unnecessary equipment replacements.

Time to respond to ad hoc requests for equipment.

Accuracy of response to ad hoc requests.

MEAN SCORES PER SESSION

MEASURES	ICON/WINDOWS	KEYBOARD	VOICE
Time to detect hardware failures	42.5s*	56.4s	88.0s*
Time to detect SW no flow	56.9s*	312.4s	369.4s
Time to detect SW decreased flow	71.2s*	398.9	438.9s
Time to detect high error count	206.0s*	356.7s	391.7s
Time to deconfigure	11.1s*	22.6s	28.0s
Time to compensate for automated schedule problems	46.5s	75.9s	82.9s
Number of operator-caused schedule conflicts	.16*	.70	.93

MEAN SCORES PER SESSION

MEASURES	ICON/WINDOWS	KEYBOARD	VOICE
# of Unnecessary Replacements	.23*	1.13	1.14
Good Displays Called		45.5	24.5*
Bad Displays Called		2.5	1.1*

THE AUDOPILOT VOICE INTERFACE

- Isolated word, single user recognition
- Template-matching algorithm
- Three background noise levels
- Hierarchical vocabularies
- Up to 64 words per vocabulary
- Manufacturer-reported 98% accuracy
- Desk-top microphone

Use of Artificial Intelligence in Supervisory Control

Human-machine **mix** where artificial intelligence, advanced automation, robotics, and human supervisory control are integrated in an effective human-machine system.

Aaron Cohen & Jon D. Erickson

Johnson Space Center
Advanced Technology Advisory Committee
NASA Technical Memorandum, April 1985

Major Research Issue

How to use artificial intelligence in system control?

Replace human operator

or

Amplify human operator's abilities to monitor the system and detect, diagnose and compensate for system failures?

Objective of OFMspert Research

Design an architecture to provide the human operator with an *intelligent* decision support system

-- to augment, not replace, the versatile human skills with skills provided by machine intelligence.

-- to compensate for known human limitations.

-- to complement individual human preferences

Develop a *theory* of human-computer interaction in the control of complex, dynamic systems (normative, plausible)

Build a *model* of the theory to demonstrate and empirically evaluate the proposed architecture (operator's associate)

Requirements for an

Intelligent Operator's Associate

Operator's Associate must provide information and control abilities at the right time, of the right kind and with the ease of a human associate

- -- understanding
- -- control
- -- interface

Understanding requires a model of the operator and system that can allow the OA to *infer* the operator's current control goals given knowledge of the control task, system functions, and current control state.

OFMspert Characteristics

OFMspert (operator Function Model Expert System) is an intelligent operator's associate based on the operator function model (Mitchell, 1987)

OFMspert uses the Operator Function Model (OFM) to represent knowledge about the operator

OFMspert uses the blackboard model of problem solving (Nii, 1986) to maintain a dynamic representation of operator *goals*, *plans, tasks,* and *actions* given previous operator actions and current system state

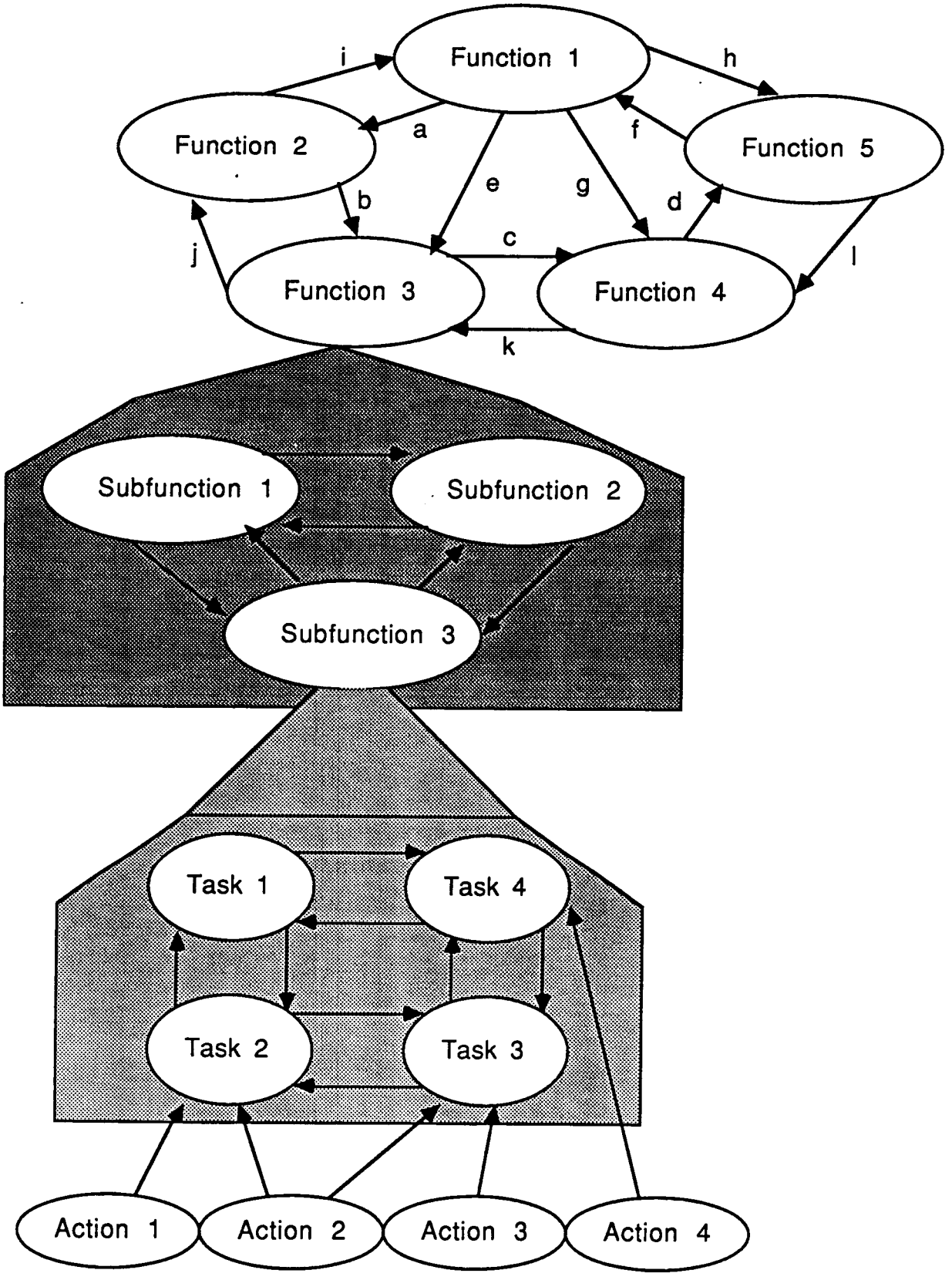


Figure 1. A Generic Operator Function Model

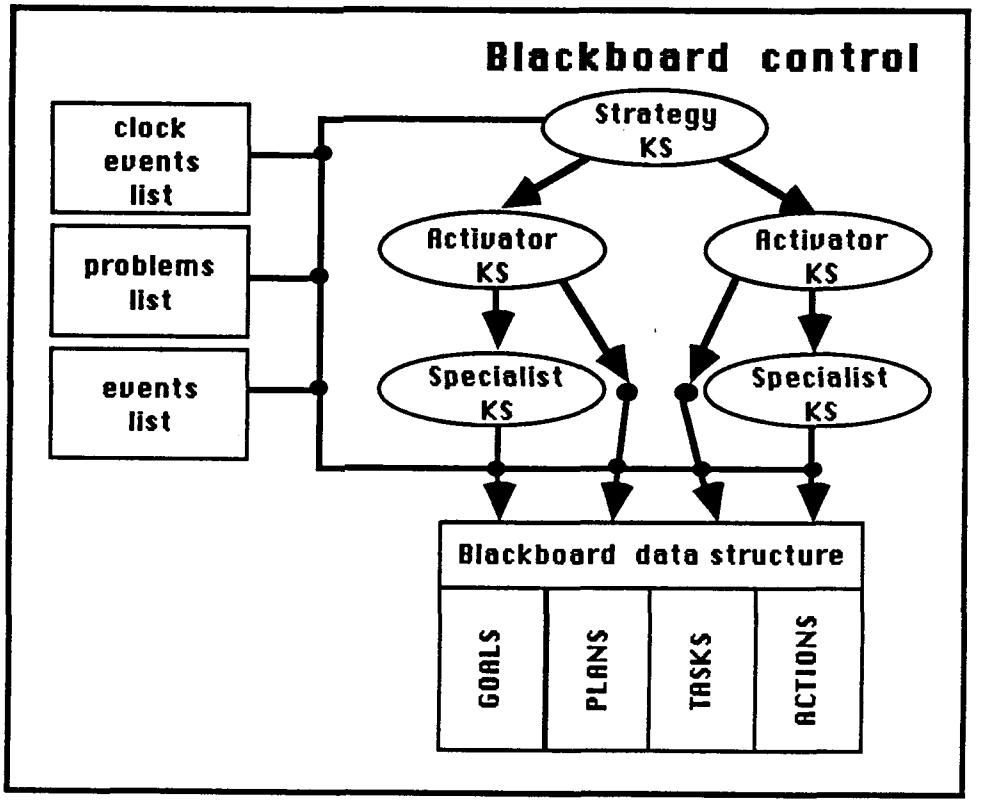
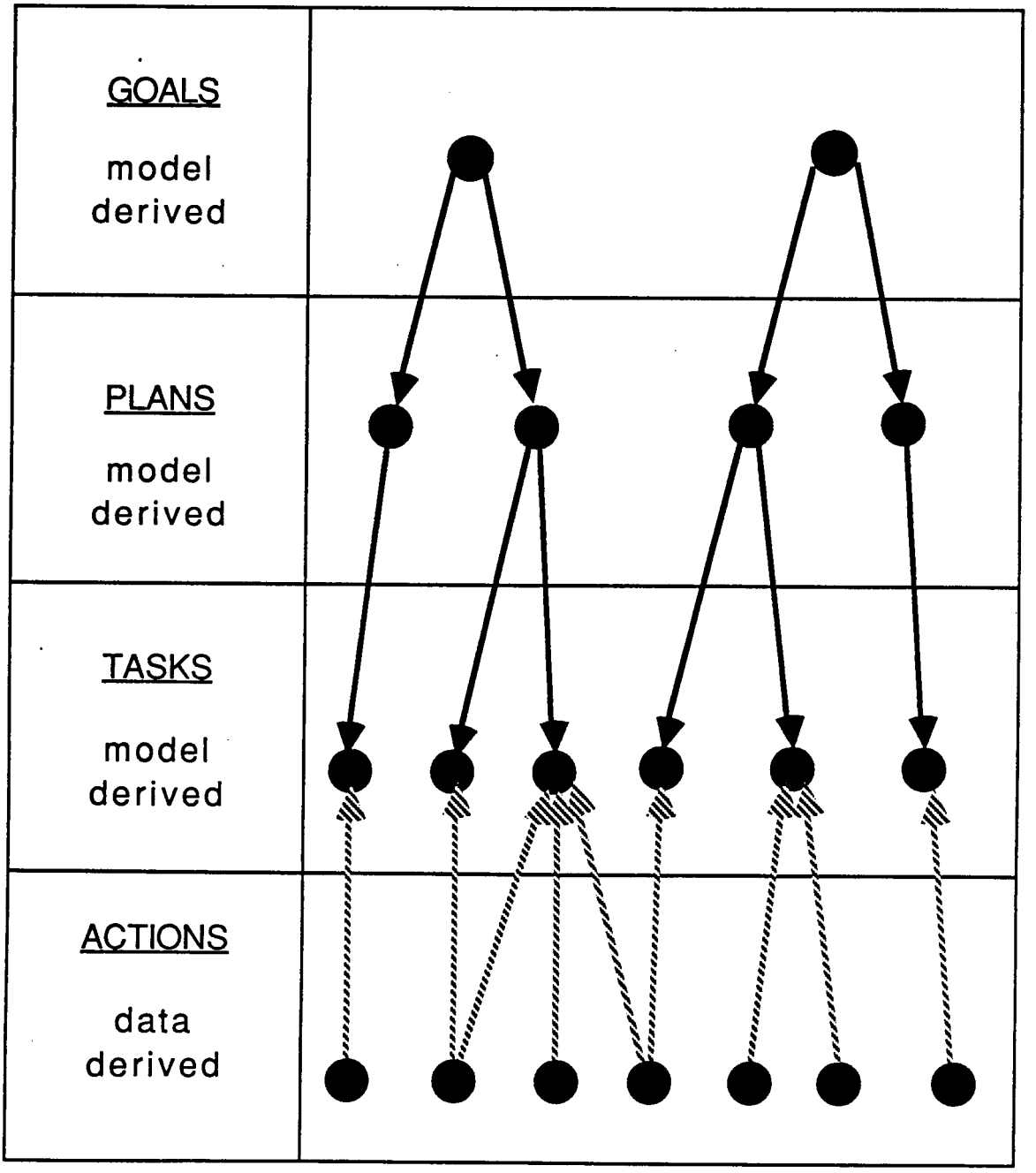


Figure 2. ACTIN's Architecture

Blackboard Model of Interactions

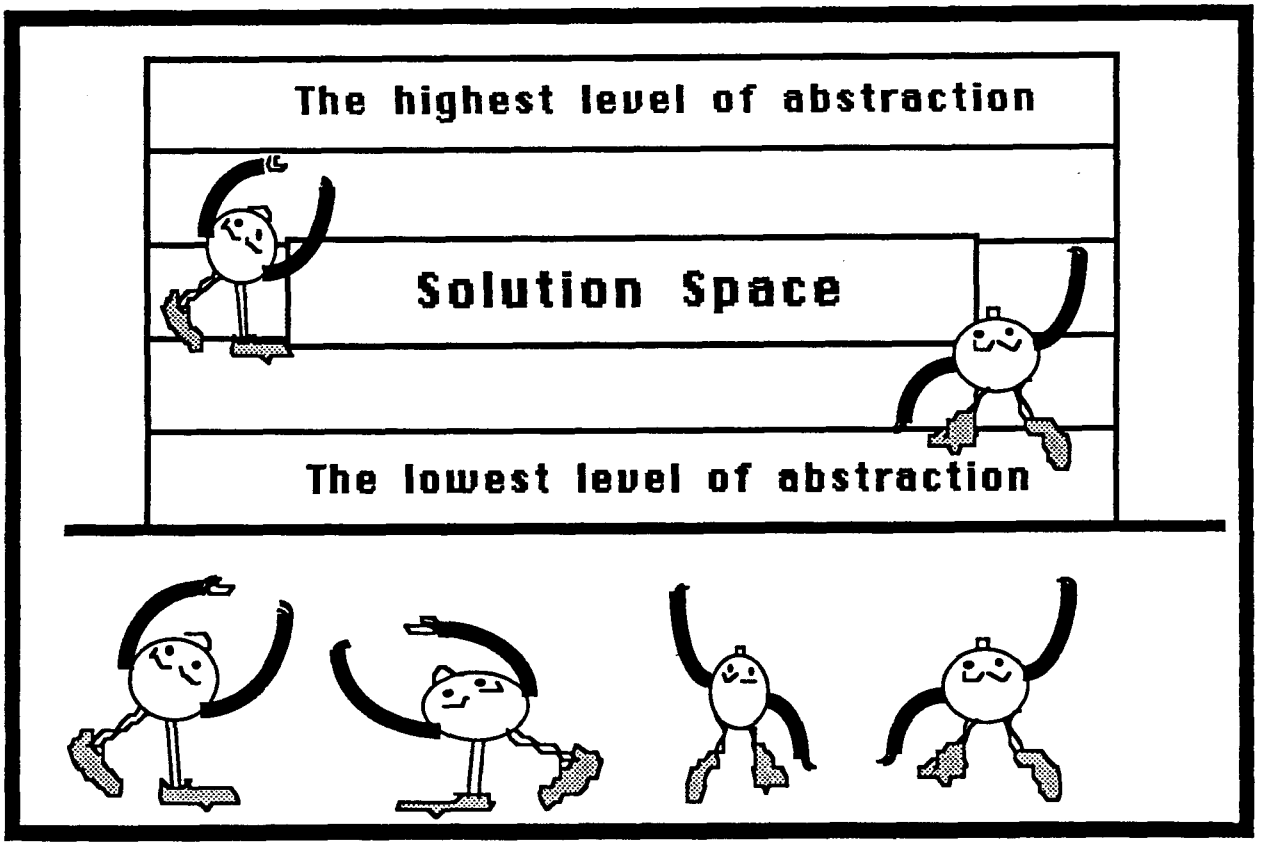


COMPONENTS OF THE BLACKBOARD MODEL:

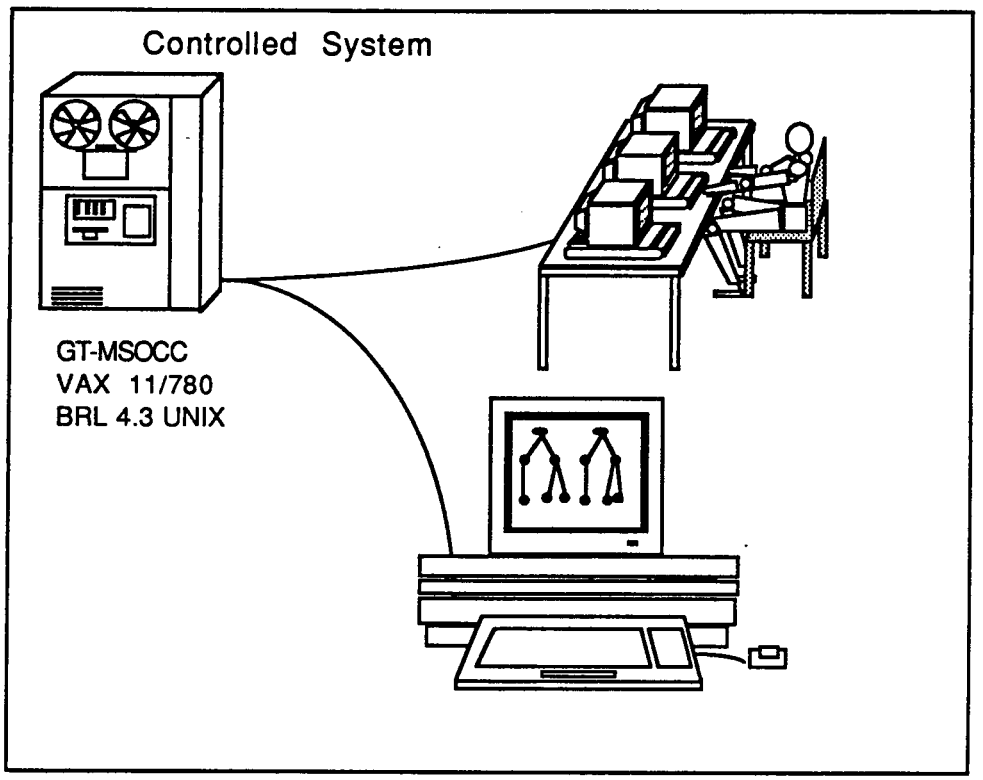
Blackboard data structure

Contains the set of partial and complete solutions known as the solution space.

Divided into levels of information where each level represents a distinct level of abstraction in the solution space.



Blackboard data structure


Controlled System
GT-MSOCC
VAX 11/780
BRL 4.3 UNIX

Experimental Validation

of OFMspert's Intent Inferencing

- Compare a domain expert's interpretations of operator actions to OFMspert's interpretation of those actions.

- Compare verbal protocols of subjects verbalizing their intentions for each action to OFMspert's interpretations of those actions.

Table 3. Experiment 1: Average Percentage of Equivalent Interpretations between ACTIN and a Human Domain Expert. Ordered by rank.

Configure	100%
Endpoint telemetry page requests	100
Deconfigure	97.1
Telemetry page requests	96.3
Answer	91.4
Reconfigure	91.2
Interior telemetry page requests	87.1
Replace	75.3
Mission schedule page requests	66.7
MSOCC schedule page requests	50.3
Equipment schedule page requests	21.8
Events page request	17.7
Pending page request	16.7

Table 5. Experiment 2: Average Percentage of actions matched by OFMspert

Configure	100%
Deconfigure	100
Answer	96.2
Replace	94.8
Equipment schedule page requests	90.3
Mission schedule page requests	85.7
Interior telemetry page requests	84.3
Endpoint telemetry page requests	76.5
MSOCC schedule page requests	75.5
Telemetry page requests	70.2
Reconfigure	60.8
Events page request	53.9
Pending page request	33.3

Table 2. Experiment 1: Proportions of Equivalent Interpretations between ACTIN and a Human Domain Expert

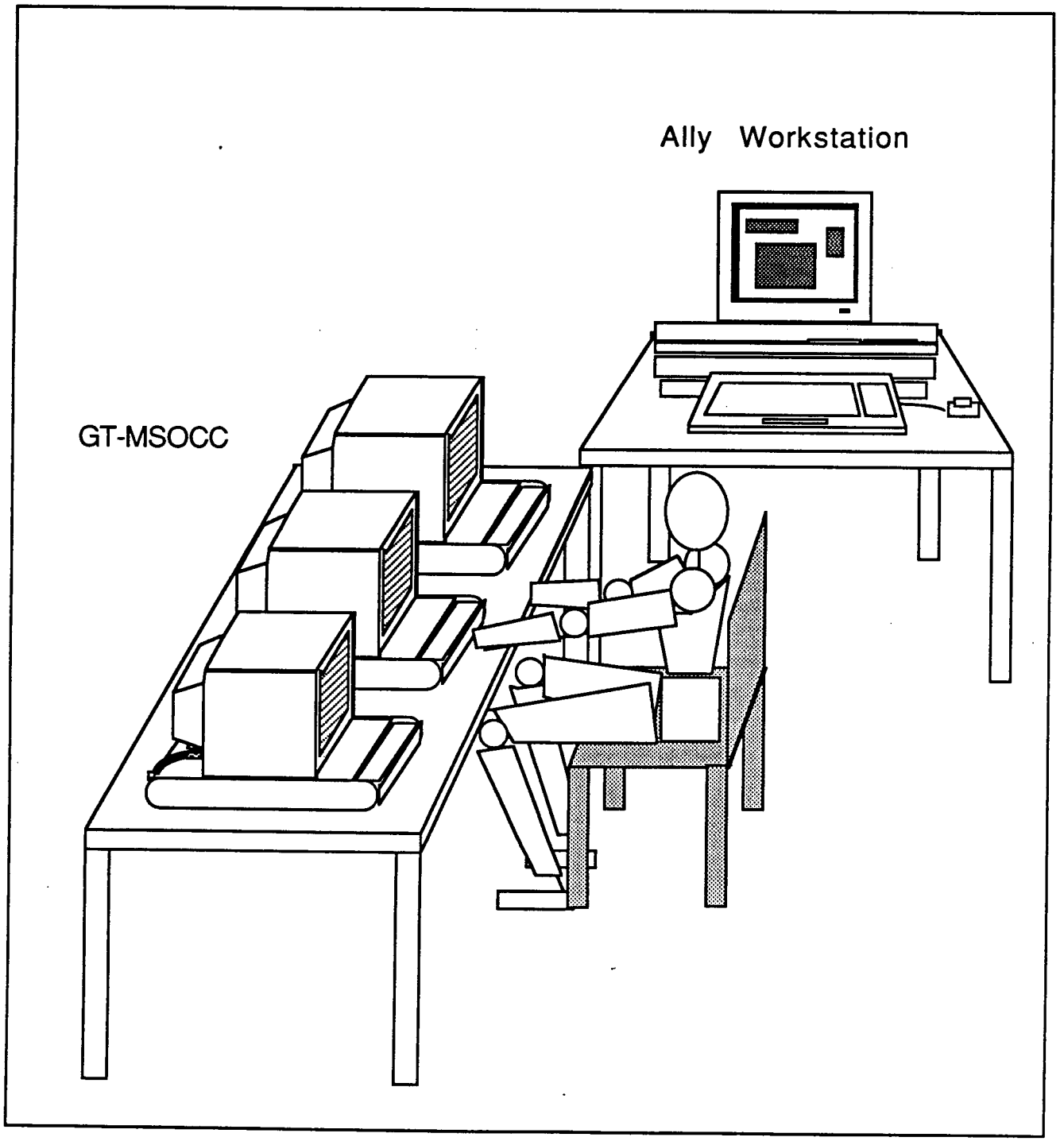
Answer	Subject 1	2	3	4	5	6	7	8	9	10
Telem	7/7 *	32/36 *	11/12 *	22/23 *	40/41 *	11/11 *	39/40 *	18/19 *	28/29 *	29/29 *
Endpoint Telem	2/2	25/25 *	2/2	13/13 *	27/27 *	4/4	24/24 *	8/8 *	1/1	19/19 *
Interior Telem	14/14 *	29/36 *	27/30 *	20/23 *	17/24	20/22 *	25/29 *	18/18 *	42/50 *	22/27 *
MSOCC Sched	14/24	7/11	14/26	10/33	2/27 #	7/26 #	18/37	10/11 *	13/19	6/11
Equip Sched	5/12	1/12 #	10/21	7/87 #	0/13 #	0/7 #	3/11	24/48	10/39 #	2/21 #
Mission Sched	--	--	--	--	--	--	--	--	6/9	--
Events	1/23 #	1/11 #	2/16 #	0/6 #	2/7	3/21 #	5/10	1/17 #	5/26 #	3/9
Pending	--	--	--	0/1	--	0/1	0/1	1/2	1/3	--
Deconfig	12/12 *	16/16 *	11/11 *	15/15 *	12/17	12/12 *	19/19 *	12/12 *	16/16 *	16/16 *
Reconfig	4/4	6/7	1/1	2/3	6/6 *	6/6 *	8/8 *	2/2	3/5	5/5 *
Config	3/3	4/4	3/3	4/4	3/3	--	4/4	3/3	5/5 *	4/4
Replace	12/12 *	13/18	12/17	14/17 *	14/21	12/18	14/22	12/14 *	11/15	15/21
	5/5 *	3/7	5/6	8/8 *	7/8 *	9/9 *	8/8 *	8/8 *	9/9 *	9/9 *

* Significantly good match
\# Significantly poor match

Table 4. Experiment 2: Proportions of Equivalent Interpretations between ACTIN and Verbal Reports

	Subject 21	Subject 22
Telem	30/42 *	40/58 *
Endpoint Telem	33/39 *	26/38 *
Interior Telem	15/19 *	26/29 *
MSOCC Sched	36/45 *	22/31 *
Equip Sched	4/4	25/31 *
Mission Sched	8/8 *	5/7
Events	11/18	7/15
Pending	0/3	4/6
Deconfig	31/31 *	30/30 *
Reconfig	7/15	6/8
Config	5/5 *	3/3
Replace	23/23 *	26/29 *
Answer	12/12 *	12/13 *

* Significantly good match (B> b(0.025,n,0.5)
\# Significantly poor match (B< n-b(0.025,n,0.5)


Ally Workstation
GT-MSOCC

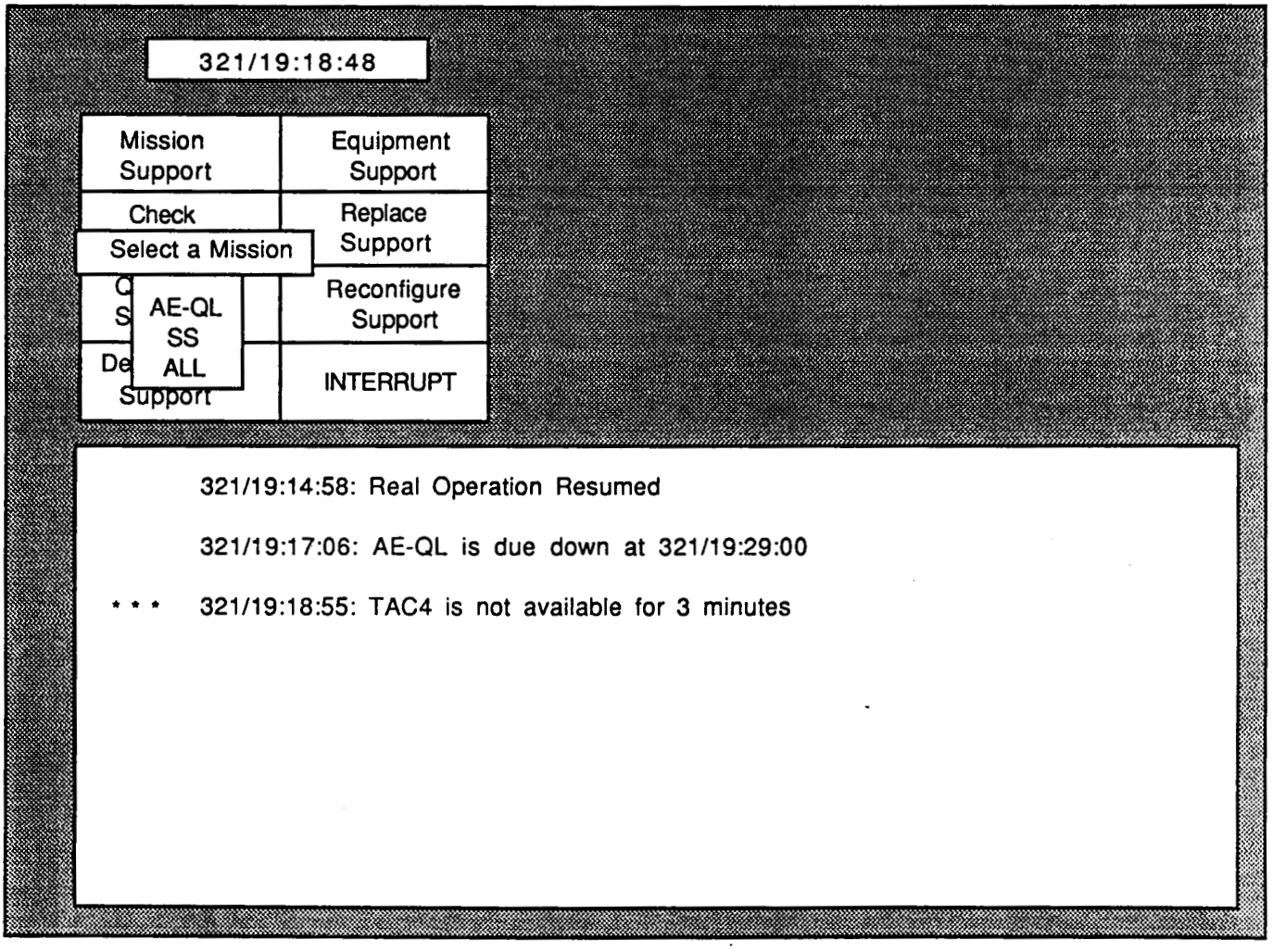
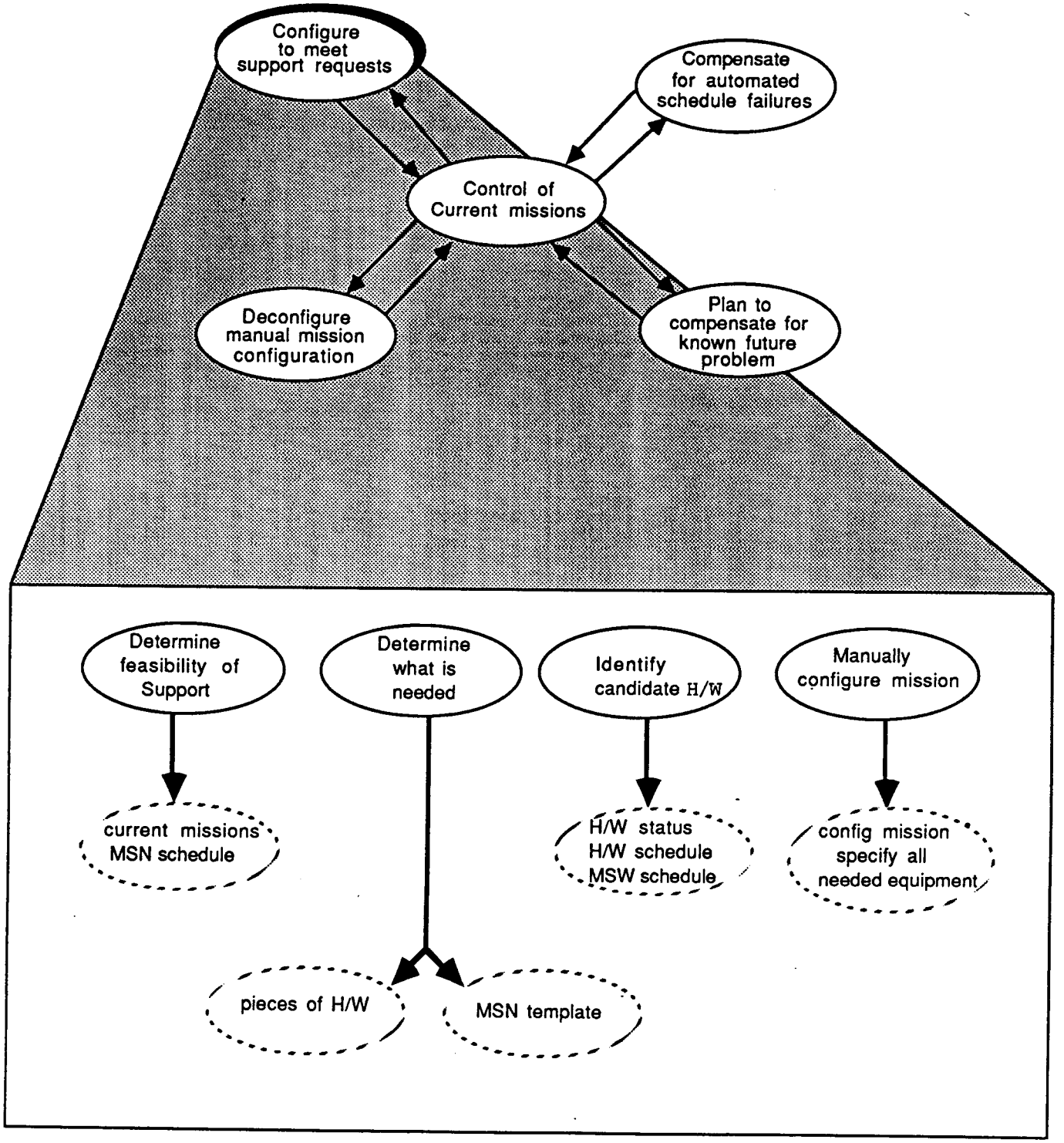
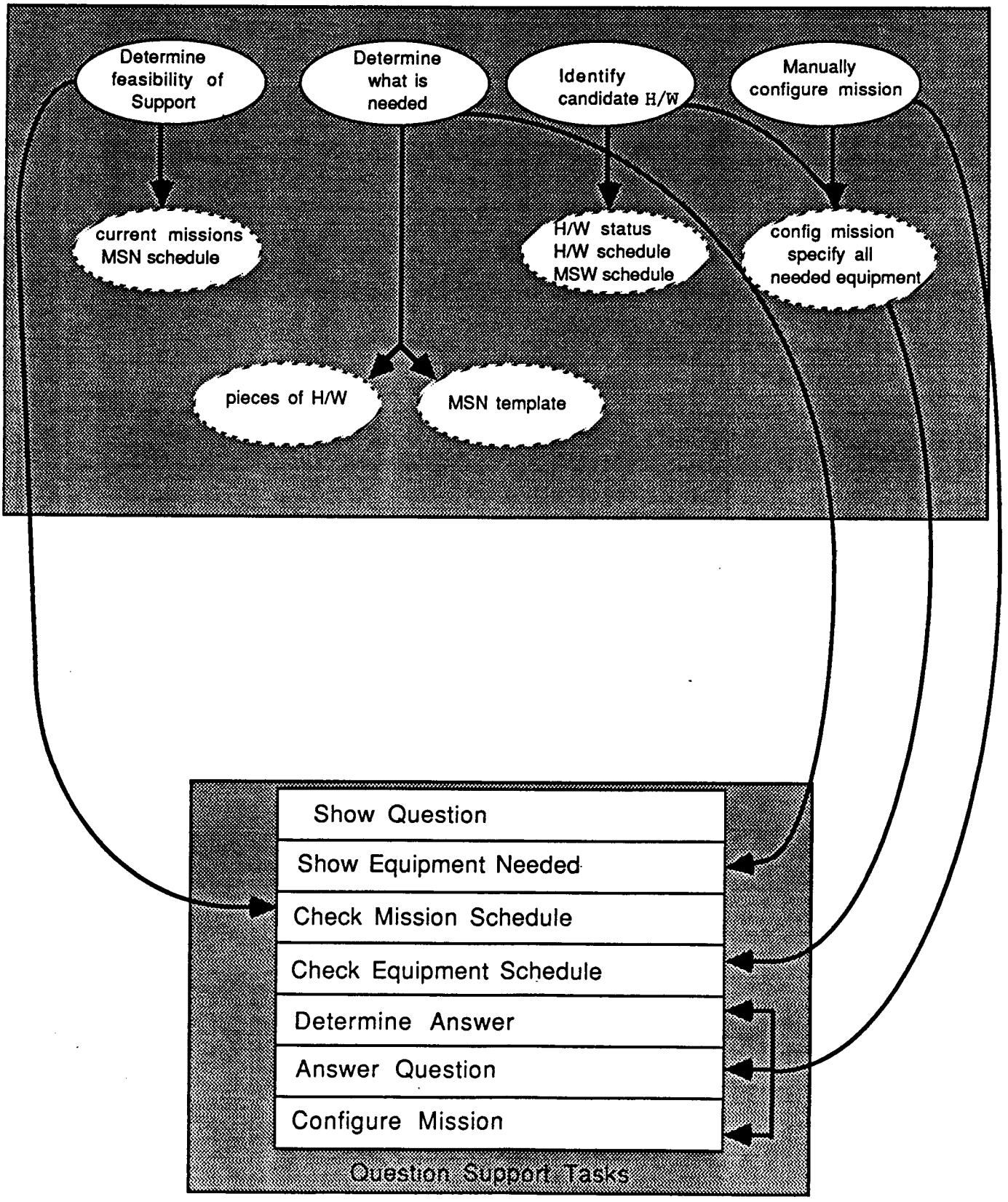
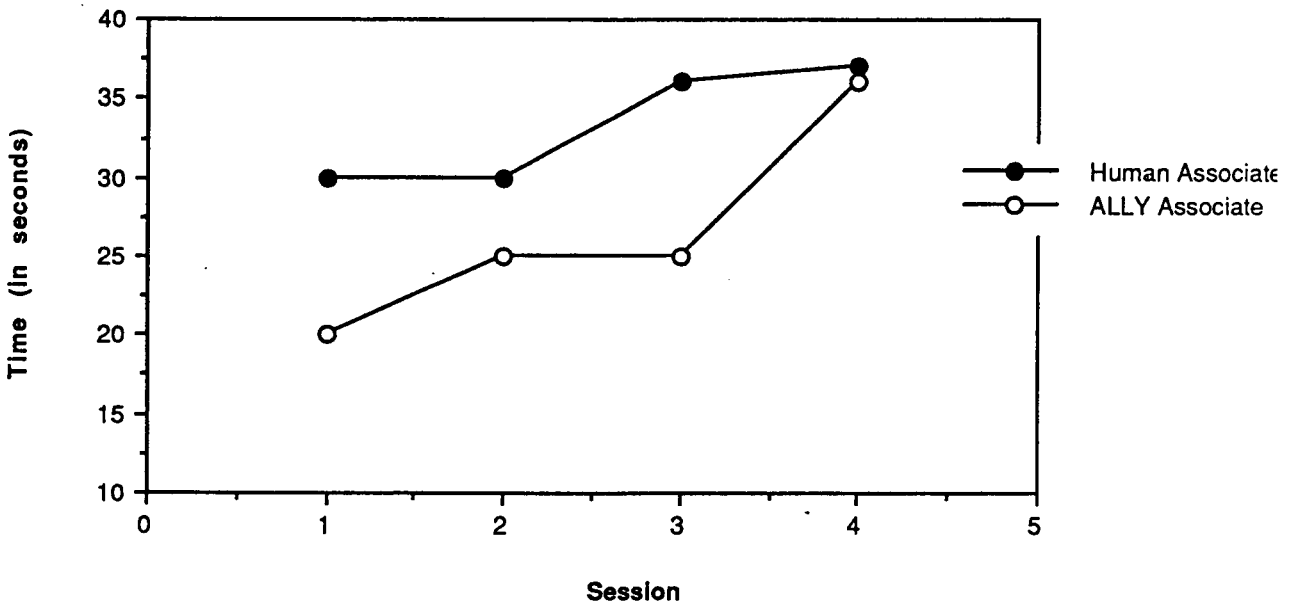


Figure 10. Example of Ally's User Interface

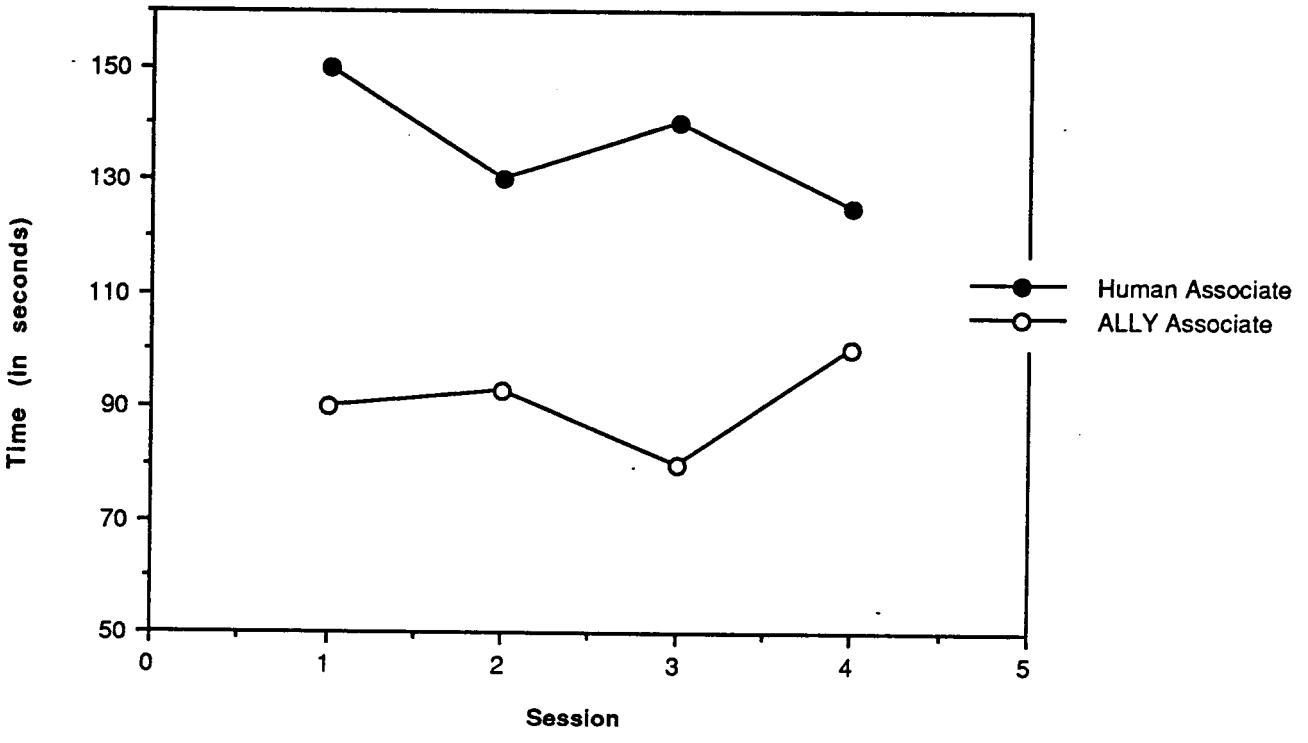

Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
Determine feasibility of Support
Determine what is needed
Identify candidate H/W
Manually configure mission
current missions MSN schedule
H/W status H/W schedule MSW schedule
config mission specify all needed equipment
pieces of H/W
MSN template


Determine feasibility of Support
Determine what is needed
Identify candidate H/W
Manually configure mission
current missions
MSN schedule
H/W status
H/W schedule
MSW schedule
config mission
specify all
needed equipment
pieces of H/W
MSN template
Show Question
Show Equipment Needed
Check Mission Schedule
Check Equipment Schedule
Determine Answer
Answer Question
Configure Mission
Question Support Tasks

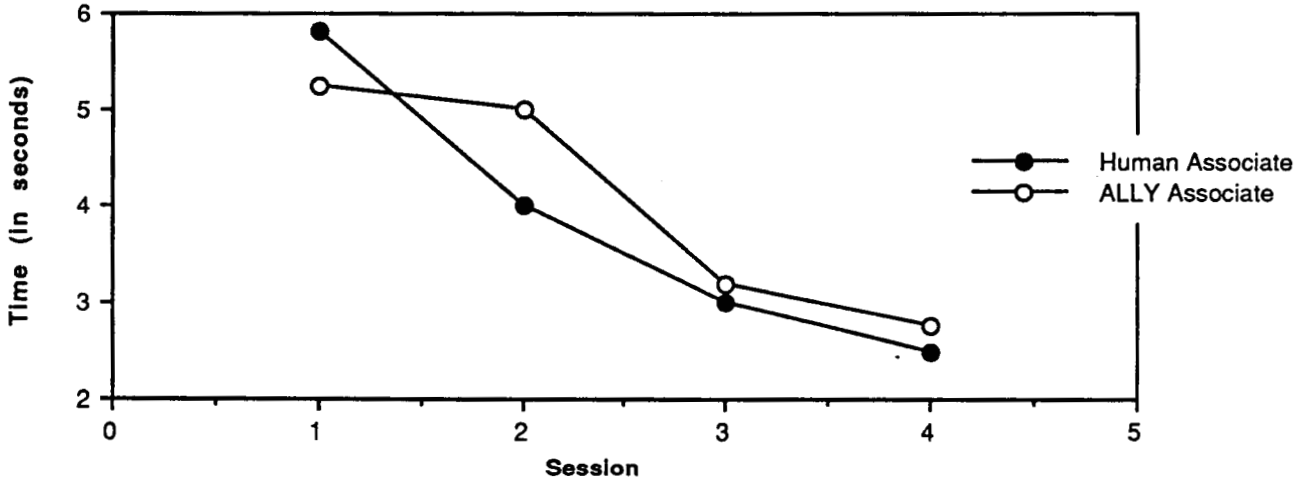
Mean Time to Compensate for Hardware Failure by Session



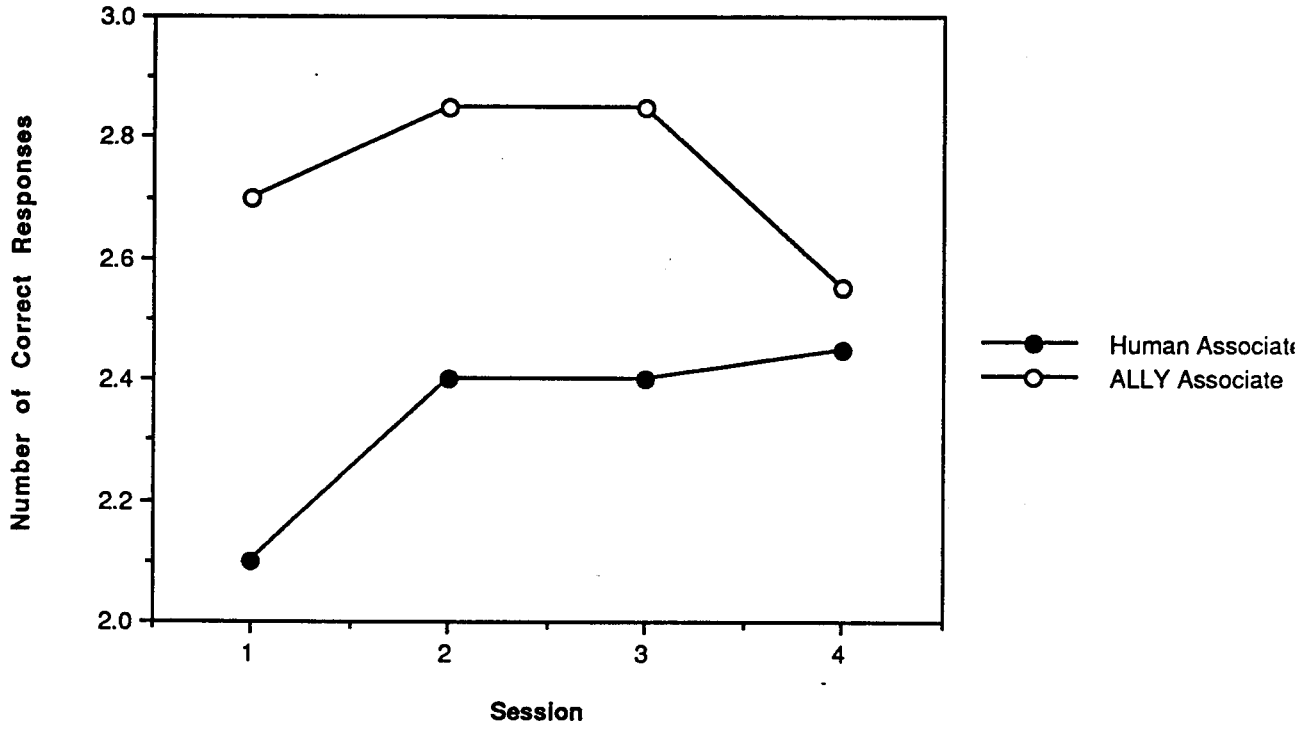
Mean Time to Compensate for Software Type 1 Failures by Session



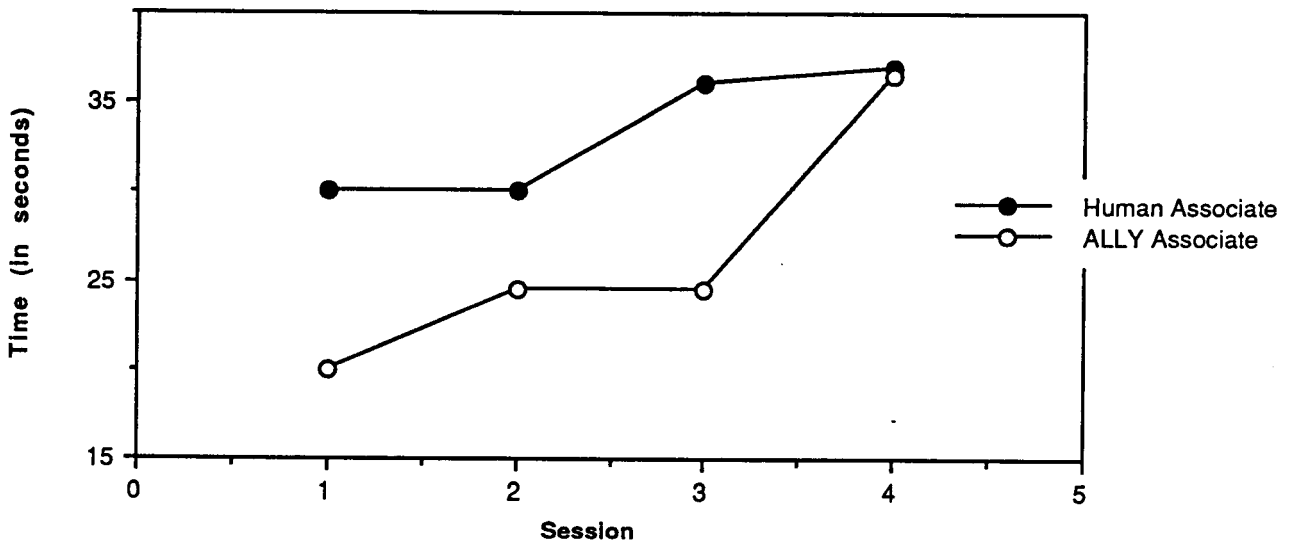
Mean Time to Compensate for Schedule Conflicts by Session



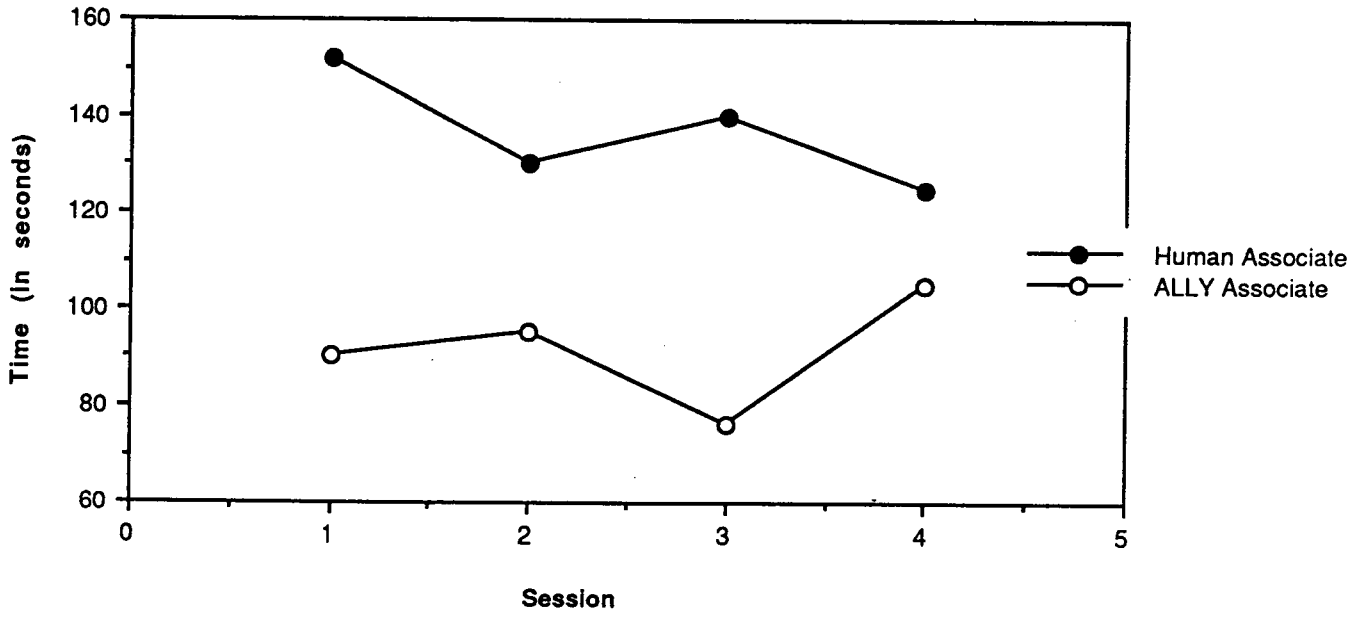
Mean Number of Correct Responses to Support Requests by Session



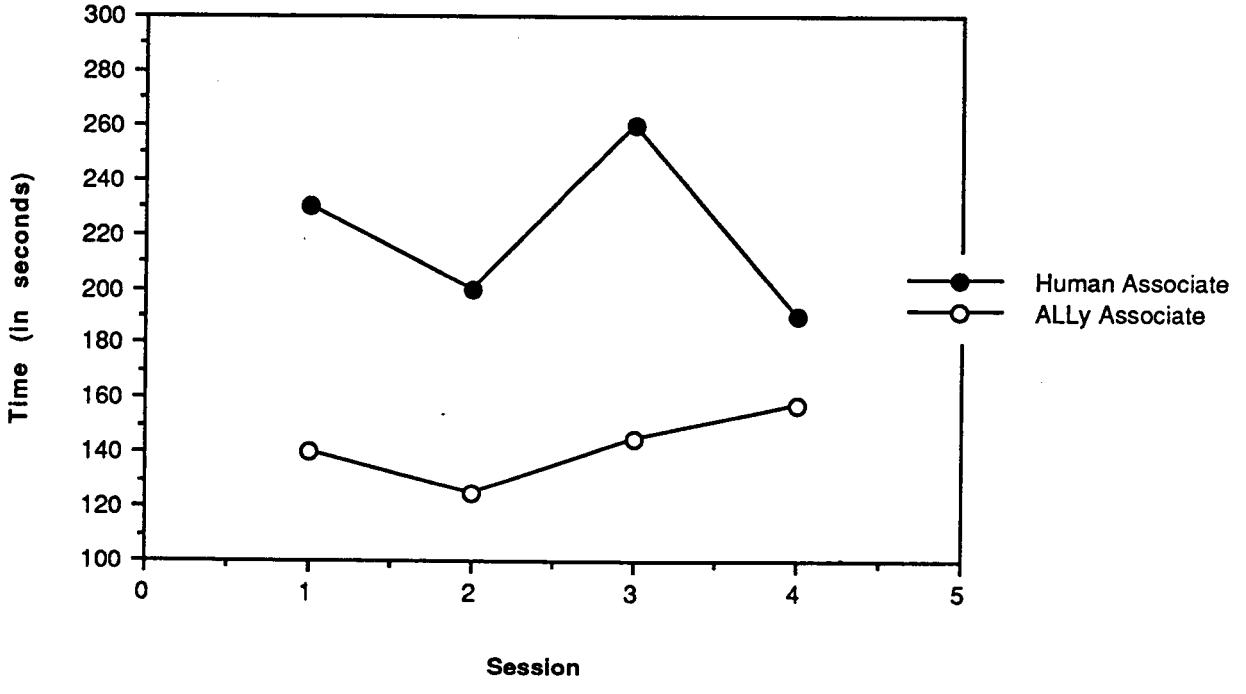
Mean Time to Compensate for Hardware Failure by Session



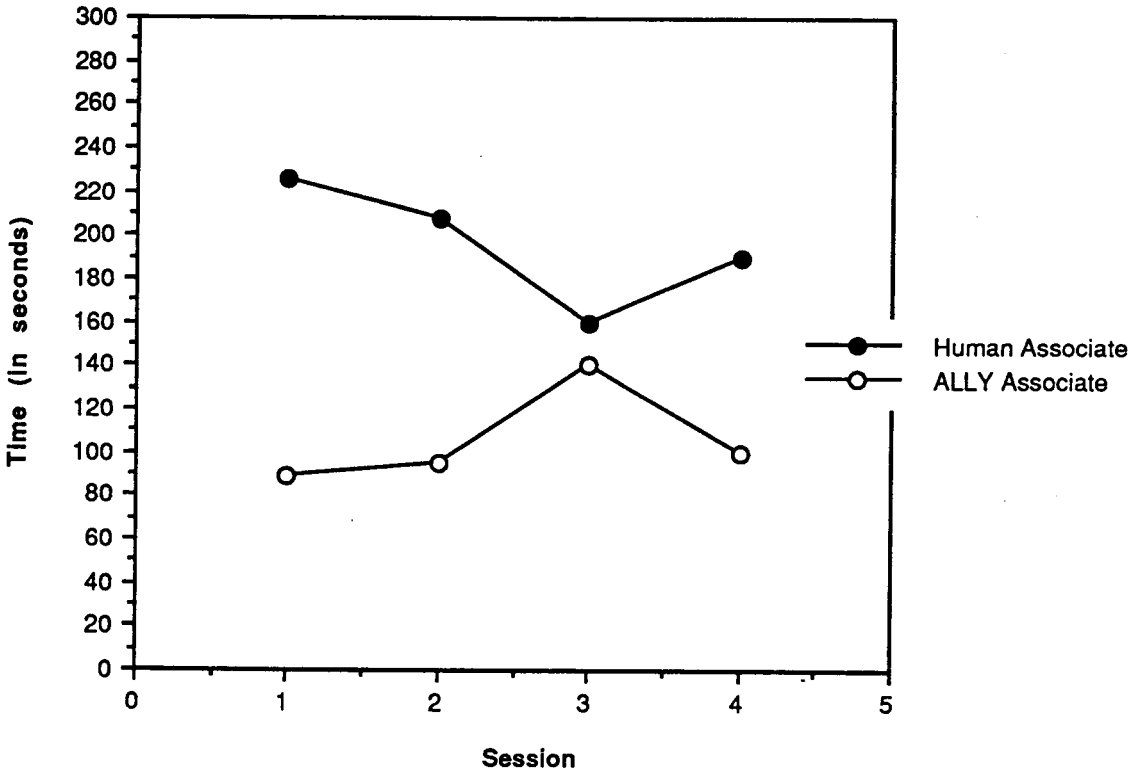
Mean Time to Compensate for Software Type 1 Failures by Session



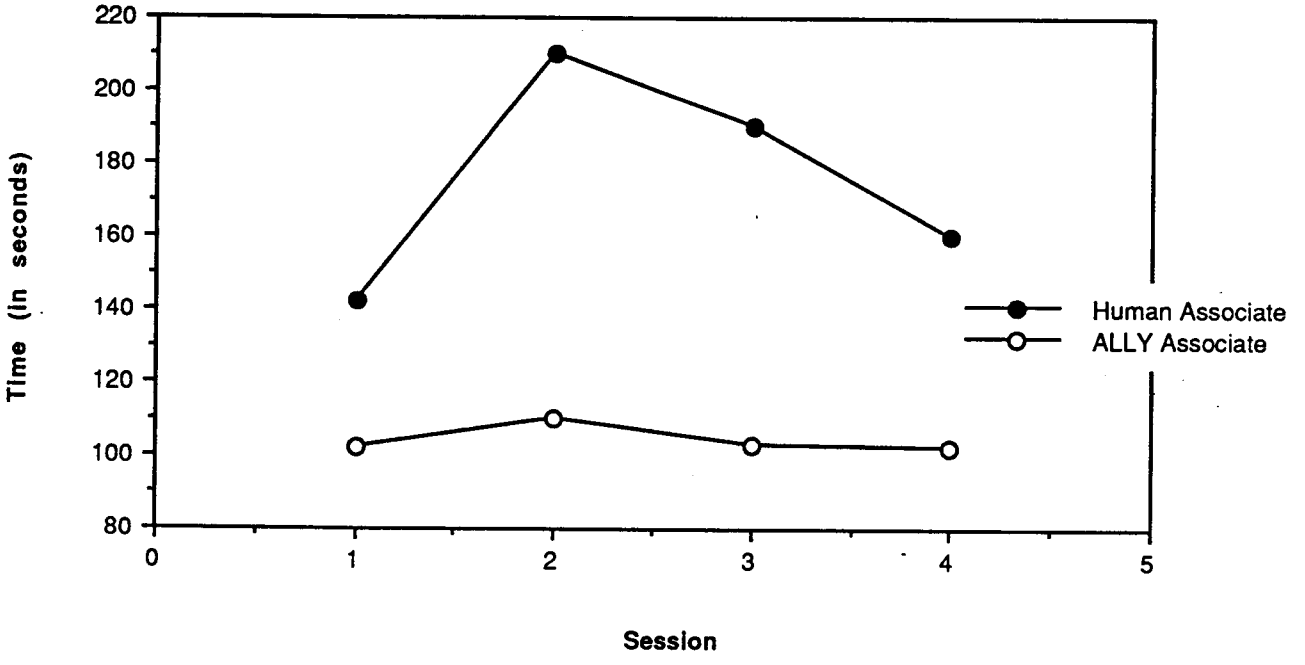
Mean Time to Compensate for Software Type 2 Failures by Session



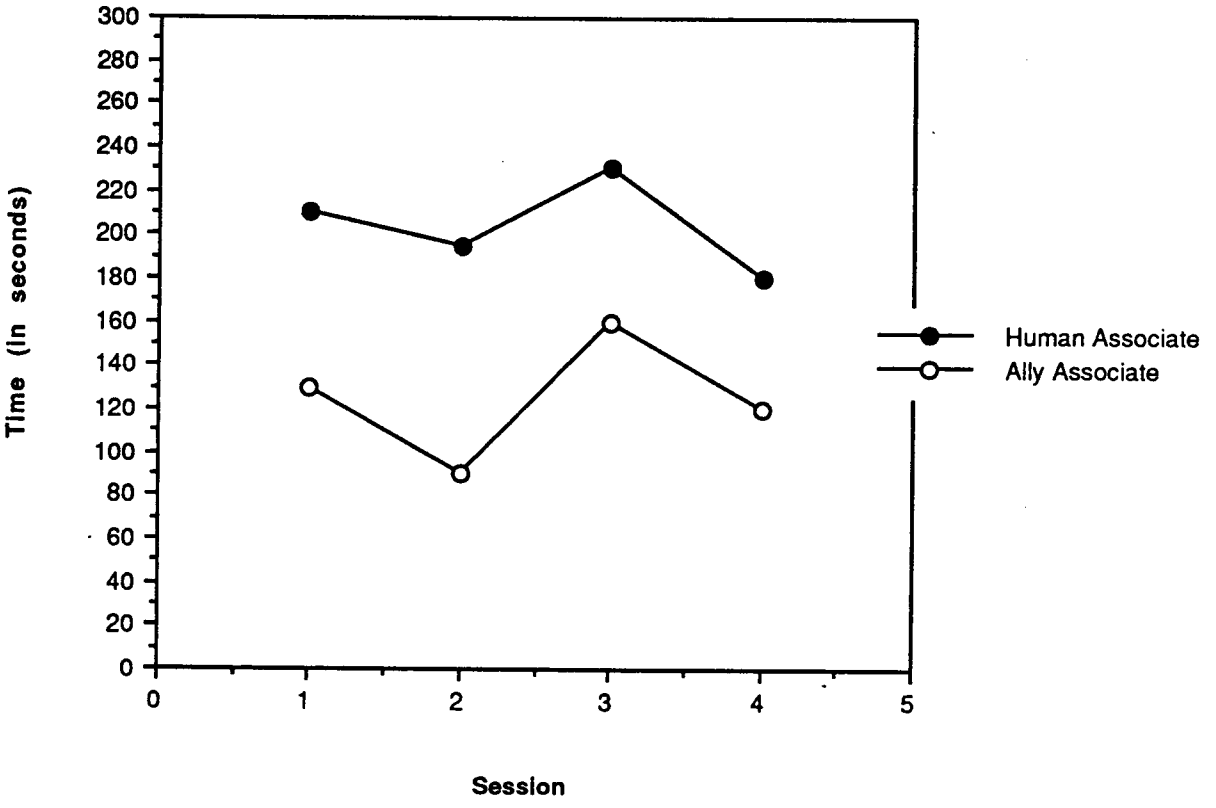
Mean Time to Compensate for Software Type 3 Failures by Session



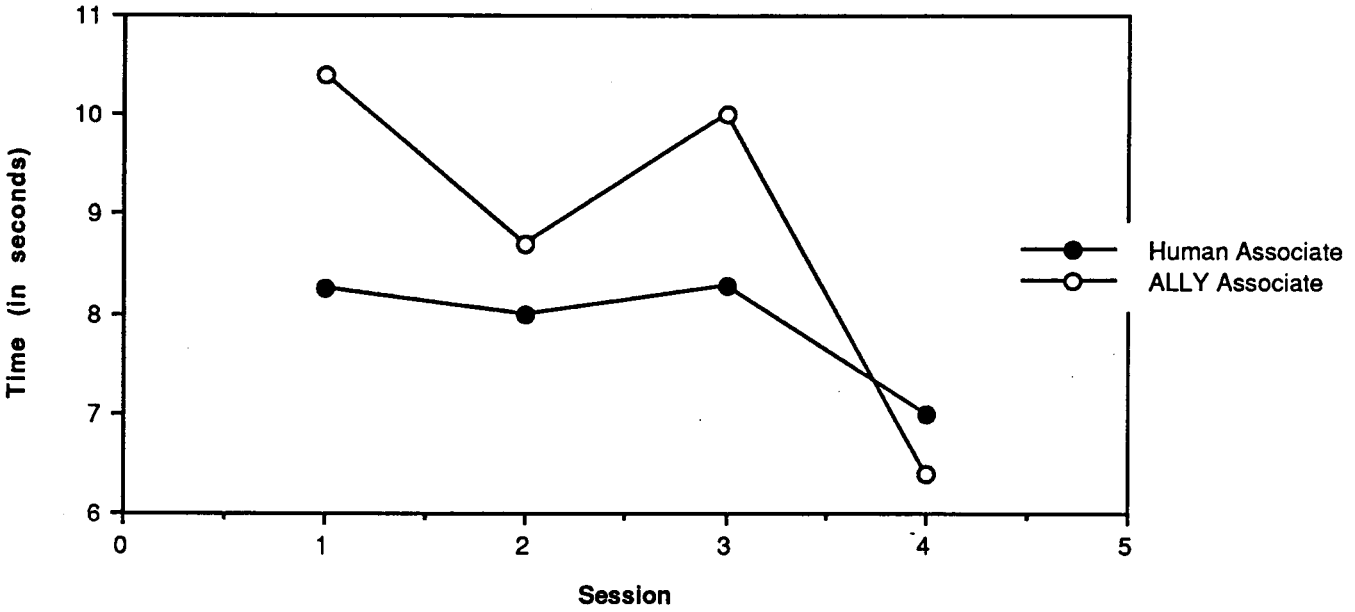
Mean Time to Respond to Support Requests by Session

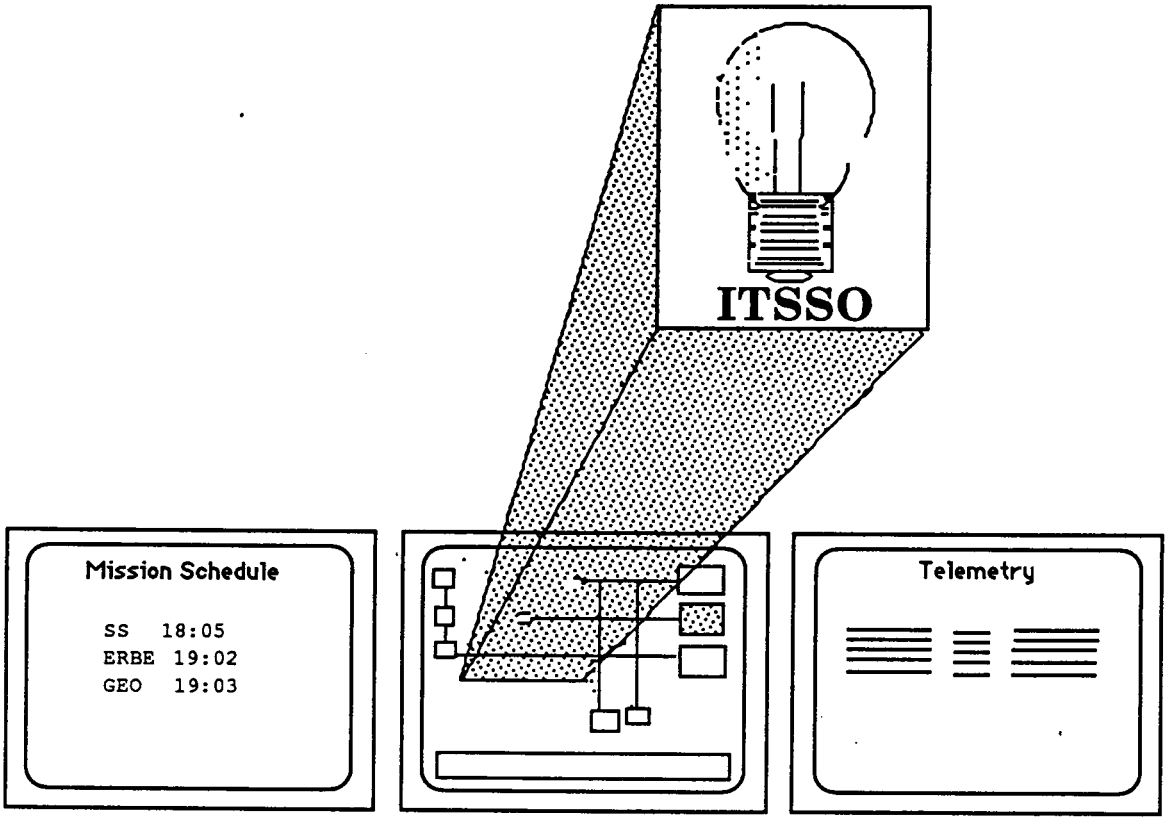


Mean Time to Configure Unscheduled Support Contacts by Session

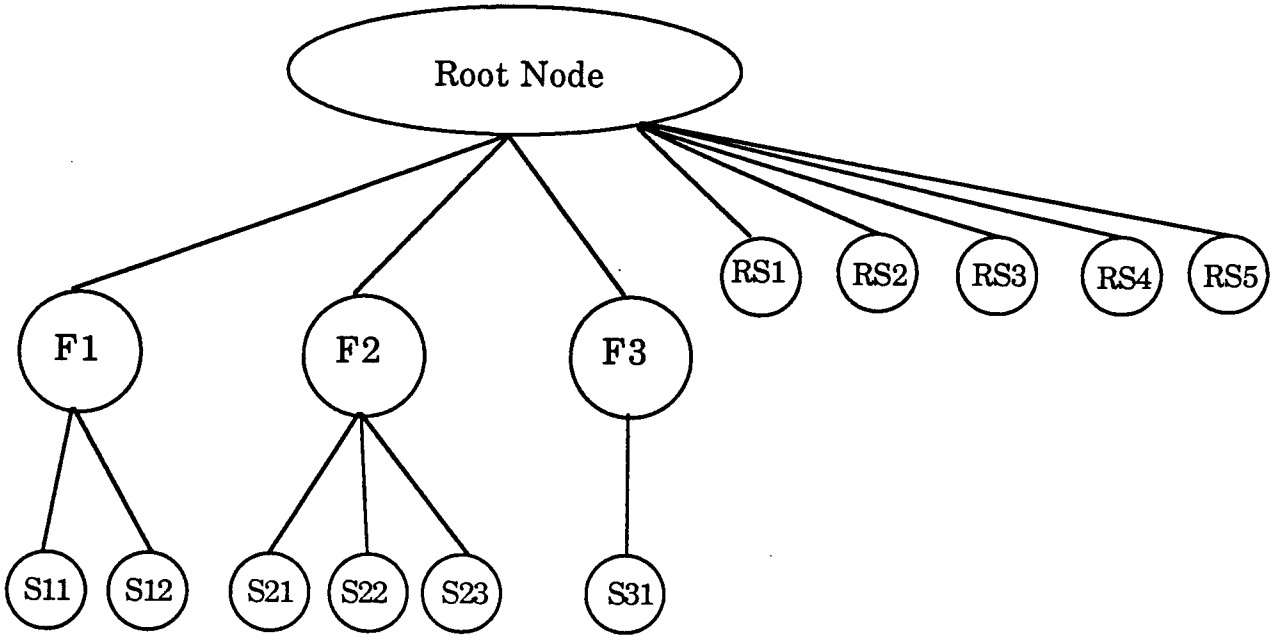


Mean Time to Respond to Deconfigure Requests by Session

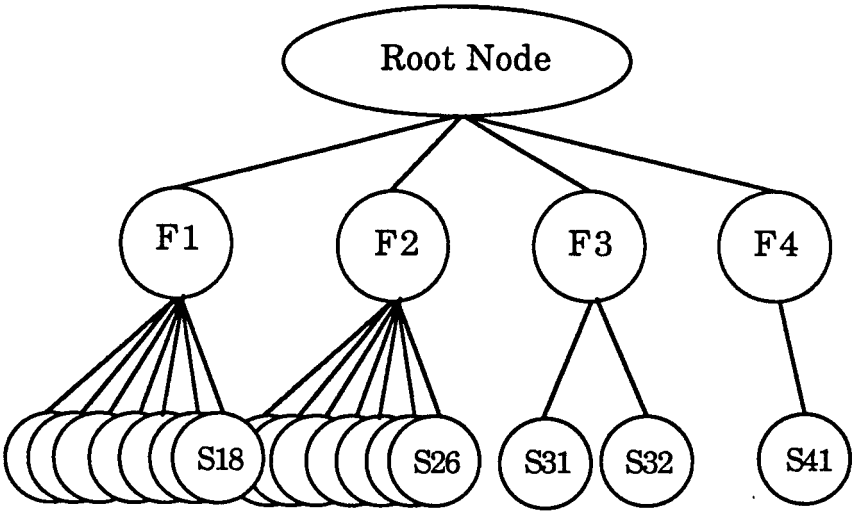




Intelligent Tutoring System for Satellite Operators



A Task Model



Root Node: Supervisory Control of GT-MSOCC

F1: Control of Current Missions

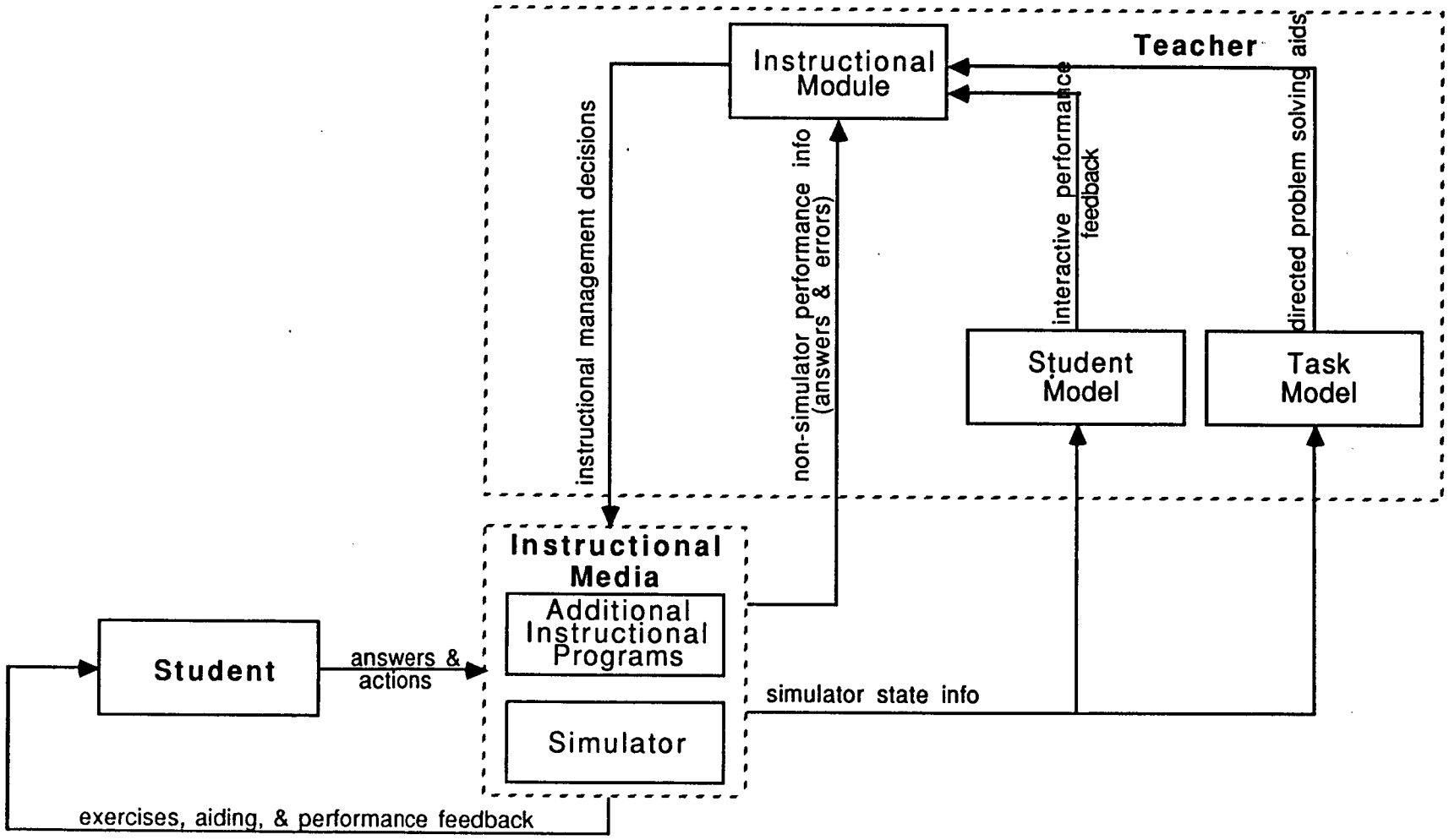
Type of Failure	Replaceable	NonReplaceable
Hardware Failure	S11	S15
No Data Relayed	S12	S16
Half Normal Data	S13	S17
Triple Normal Errors	S14	S18

F2: Configure to Meet Support Request

F3: Compensate for Automated Schedule Failures

F4: Manually Deconfigure a Mission

Figure 3 A Task Model


Teacher
Instructional Module
instructional management decisions
non-simulator performance info (answers & errors)
interactive performance feedback
directed problem solving aids
Student Model
Task Model
Instructional Media
Additional Instructional Programs
Simulator
Student
answers & actions
simulator state info
exercises, aiding, & performance feedback

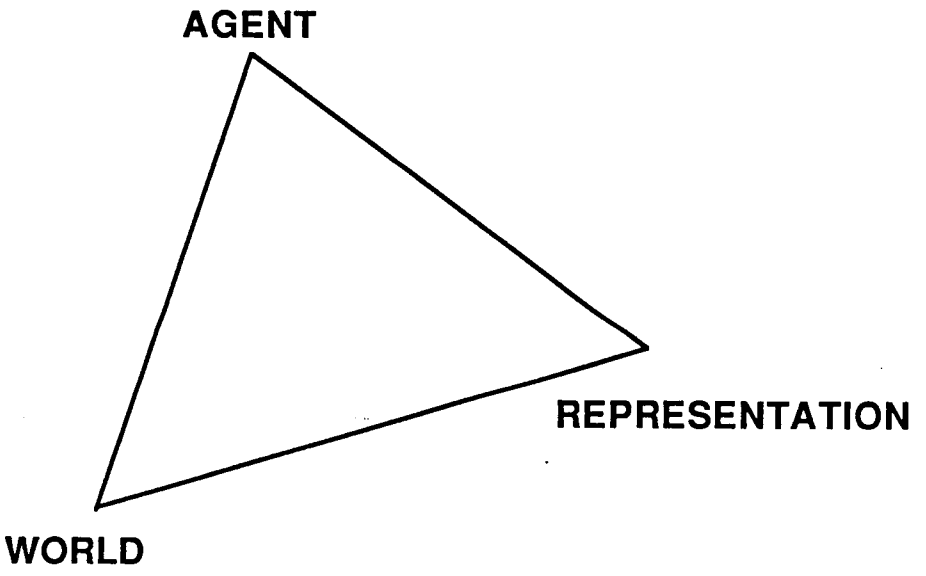
COMPLEXITY AND PROBLEM SOLVING

Three basic elements in problem solving situations:

The **World** to be acted on,

The **Agent** who acts on the world,

The external **representation** of the world utilized by the problem solving agent.



WHAT MAKES PROBLEM SOLVING COMPLEX:

DIMENSIONS OF COMPLEXITY

- Dynamism
- Number of parts and extensiveness of interconnections between parts
- Uncertainty
- Risk

DOMAIN OF INTEREST: COMPLEX DYNAMIC SYSTEMS

- Human operator as a supervisory controller
  - monitoring task
  - troubleshooting task

QUESTIONS TO BE ADDRESSED

What are the specific skills with respect to the four dimensions of complexity that are necessary to carry out the tasks involved in a CDS?

What are the goals of an ITS designed for a CDS? What do we want the operator to learn? Are the goals attainable?

What approaches in each module of an ITS seem appropriate to a CDS and why? How do we translate an approach in the context of a CDS?

What about implementation issues and "do-ability"? How much of the CDS world should be represented in the ITS?

How do we evaluate the ITS (if implemented) to test if the goals are attained?

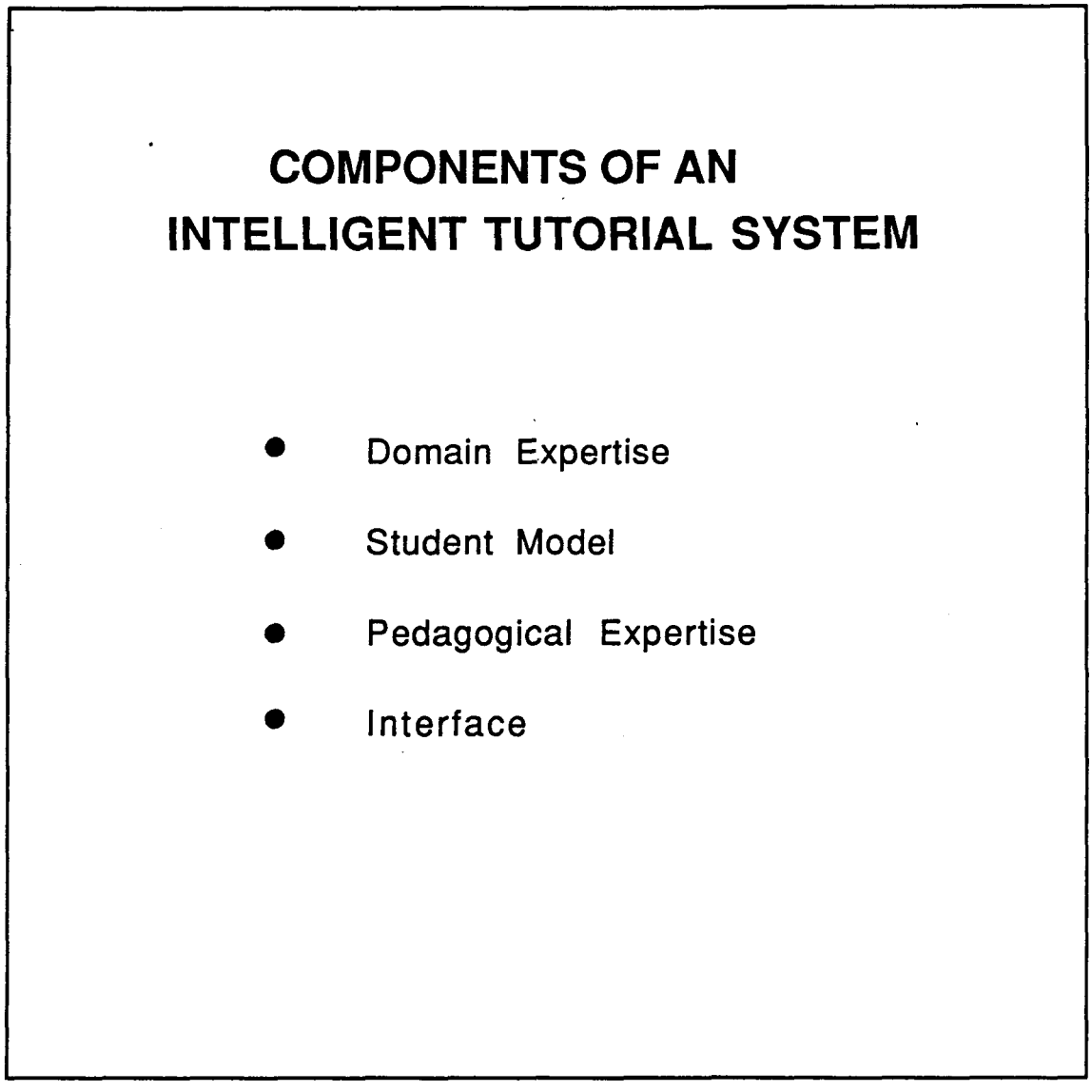


Figure 3. ACTIN's Intent Inferencing Structure

REVIEW OF APPROACHES

Domain Expertise:

- Information-structure-oriented paradigm (SCHOLAR, 1970)
- Hierarchical scripts (WHY, 1977)
- Finite state automata (METEOROLOGY, 1973)
- Multiple representations of procedural and declarative knowledge (SOPHIE I, 1975; RBT, 1986)
- Qualitative modelling (STEAMER, 1984)
- Probabilistic model (INTEGRATION, 1973)
- D-rules (MYCIN/GUIDON, 1979)
- Procedural networks (BUGGY, 1975)
- Generalized AND/OR graph (REPAIR theory, 1980)
- Problem-solving models:
    - Active structural networks (FLOW, 1974)
    - Linguistics theory (SPADE, 1976)
    - Dependency graphs (MACSYMA ADVISOR 1977)
    - Intention-based knowledge structure (PROUST, 1984)
    - Operator function model (AHAB, 1987)

STUDENT MODEL

- Differential model (WEST, 1976)
- Overlay model (WUSOR-II, 1977; GUIDON, 1979)
- Buggy model (BUGGY, 1978; MENO-II; PROUST, 1984)
- Limited bug model (AHAB, 1987)

INTERFACE

- Textual (SCHOLAR, 1970; SOPHIE, 1975; WEST, 1976; GUIDON, 1979; PROUST, 1984; etc)
- Graphical (ALGEBRALAND, 1983; STEAMER, 1984; IMTS, 1986; RBT, 1986; AHAB, 1987)

REVIEW OF APPROACHES (cont'd)

Pedagogical Expertise:

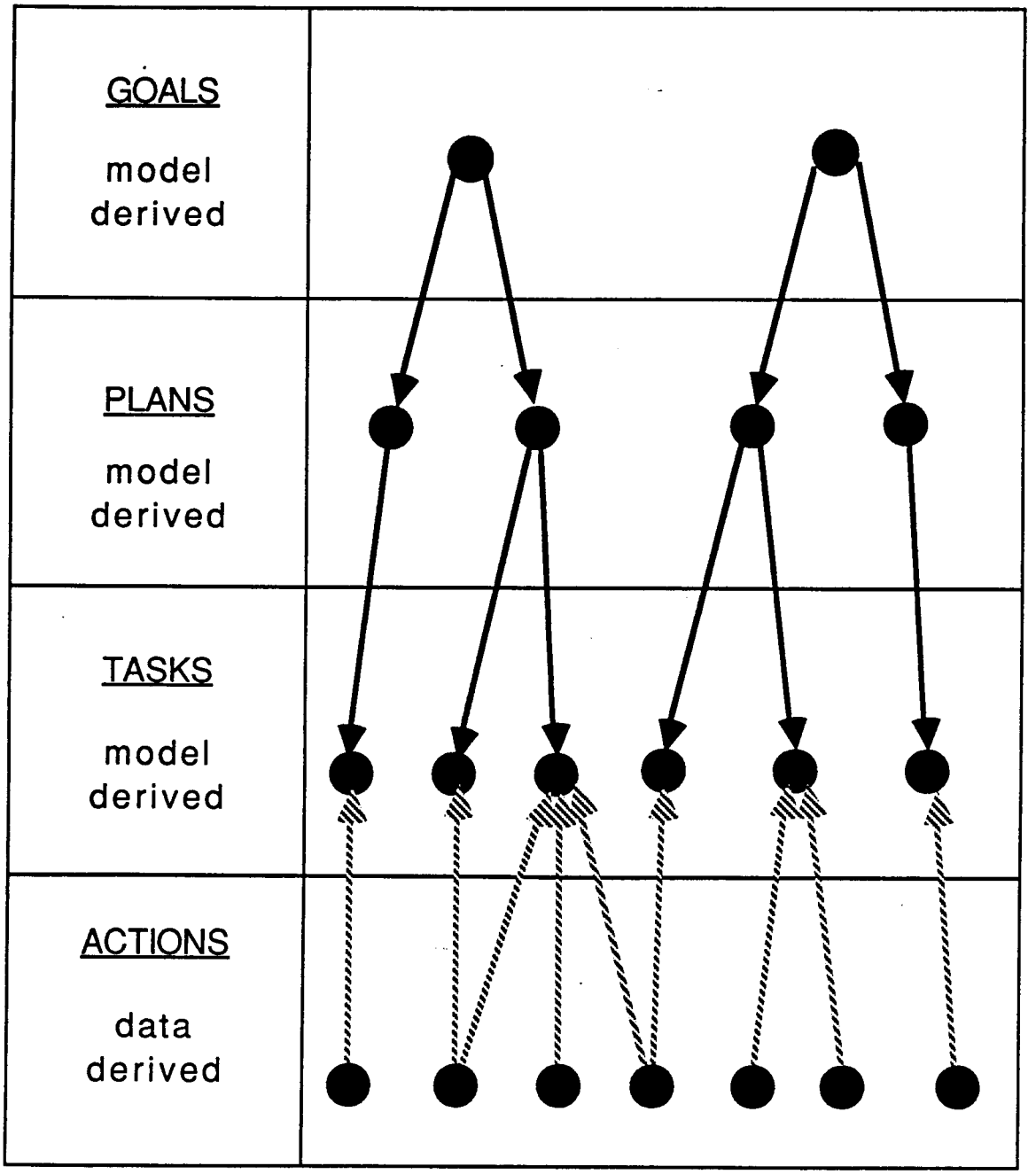
- Socratic method (WHY, 1977)
- Reactive learning environment (SOPHIE I, 1975; MACSYMA ADVISOR, 1977)
- Conceptual fidelity (STEAMER, 1984; AHAB, 1987)
- Progression of O-order qualitative models (QUEST, 1986)
- Curriculum Information Network (BIP, 1976)
- Exploratory learning (LOGO, 1980)
- Issues and examples paradigm (WEST, 1976)
- Increasingly complex microworlds paradigm (Fischer, et al., 1978)
- Expert-based coaching (WUSOR-I, 1976)
- Bite-sized architecture (SMITHTOWN, 1986)
- Layered curriculum and steering test concept (MHO, 1987)
- Discourse management networks (MENO-TUTOR, 1984)
- T-rules (GUIDON, 1979)
- ACT theory (GEOMETRY and LISP tutors, 1984)

OFMTutor

- Intelligent tutoring system for operators of complex dynamic systems

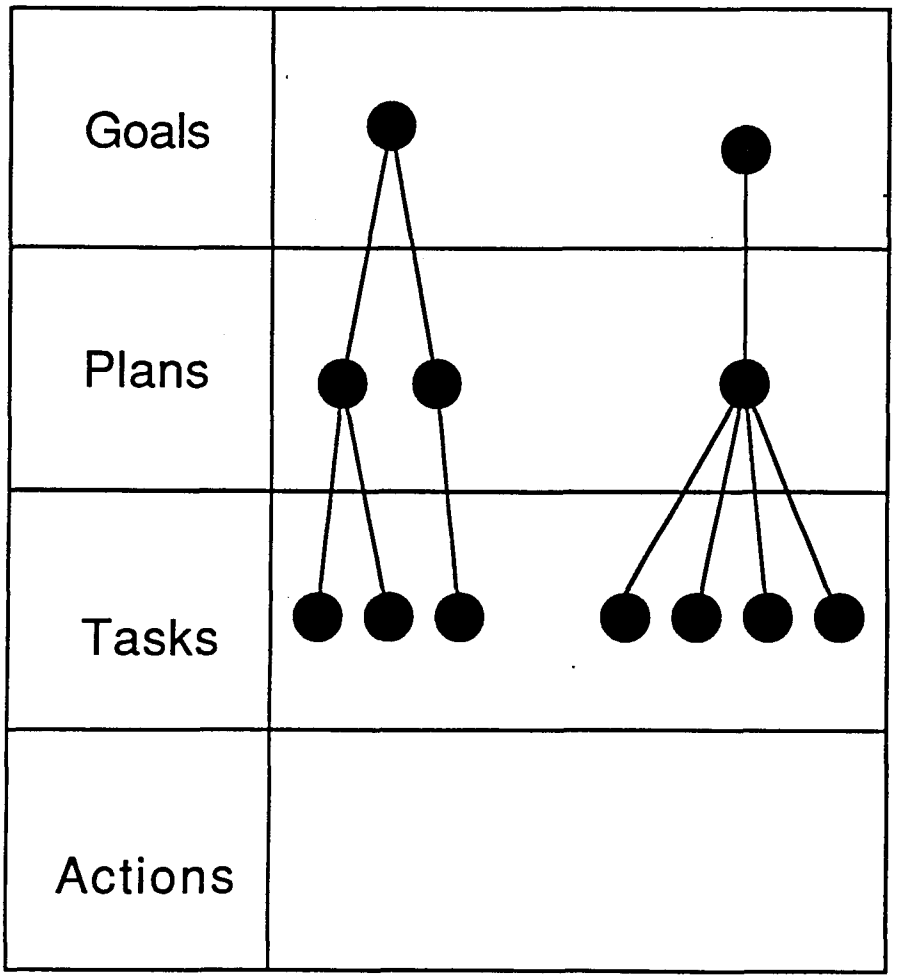
- Based on the Operator Function Model (OFM)

Blackboard model of Interactions



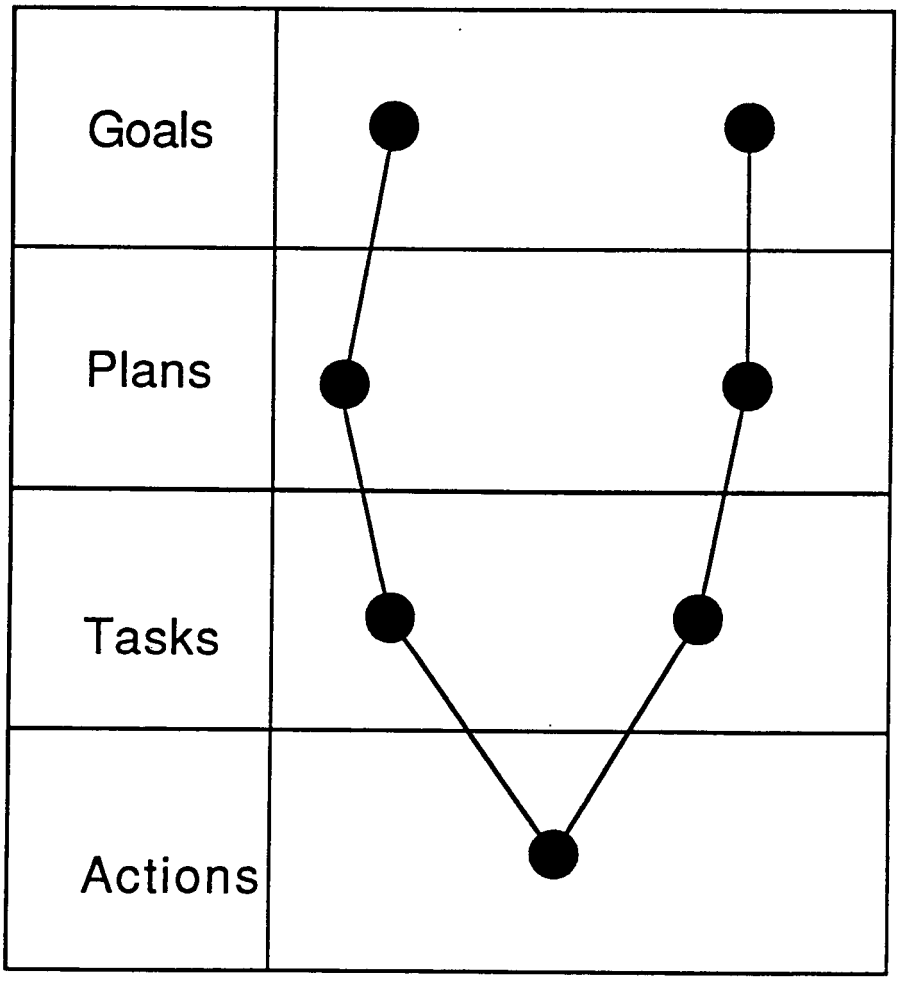
OFMTutor's Model of Expertise

Model derived representative of Goals, Plans, and Tasks



OFMTutor's Student Model

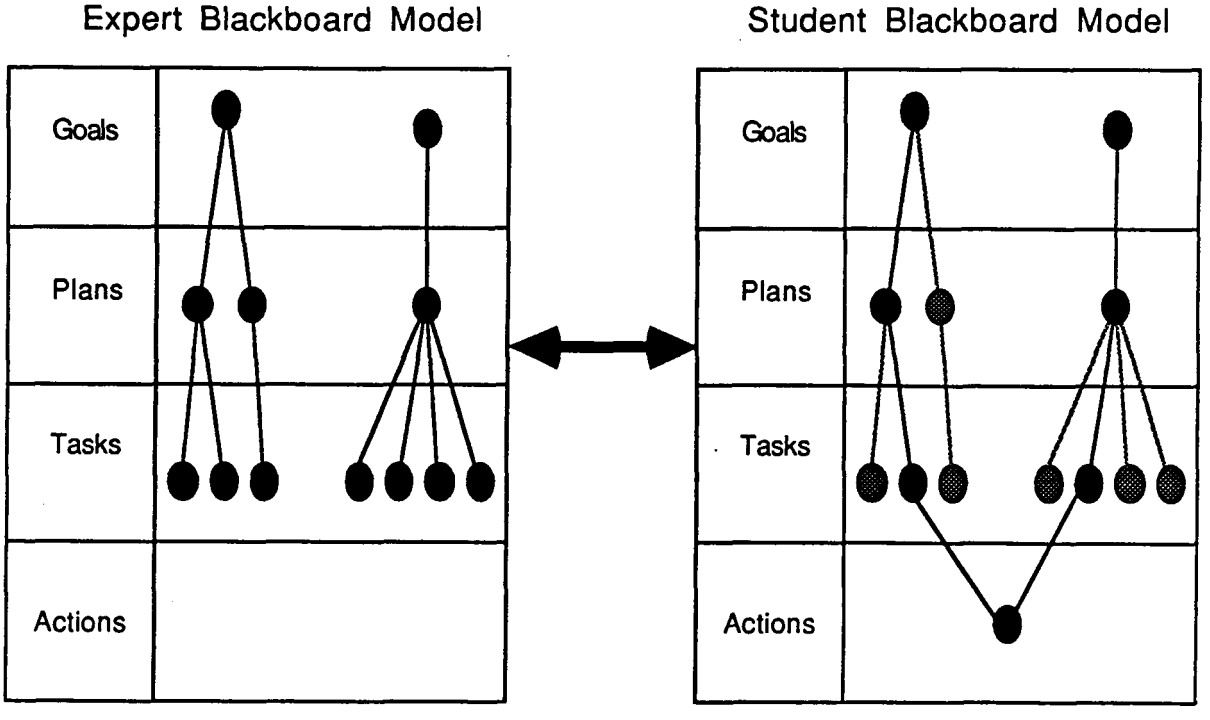
Data derived representation of goals, plans, and tasks based on student's actions



OFMTutor's Pedagogical Strategy and Diagnosis

Guided discovery/coaching in context of system operation

Differential modeling techniques that compare expert and student blackboard models



Work in Process

* Design of a computer-based operator associate that evolves from tutor to assistant as the skills of the human operator change from novice to expert.

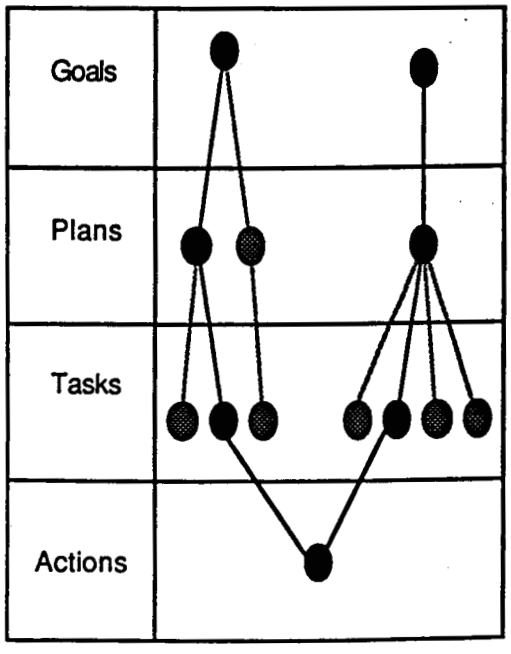
* The refinement of the Ally interaction to allow cooperative problem solving and repair of hypothesis formation.

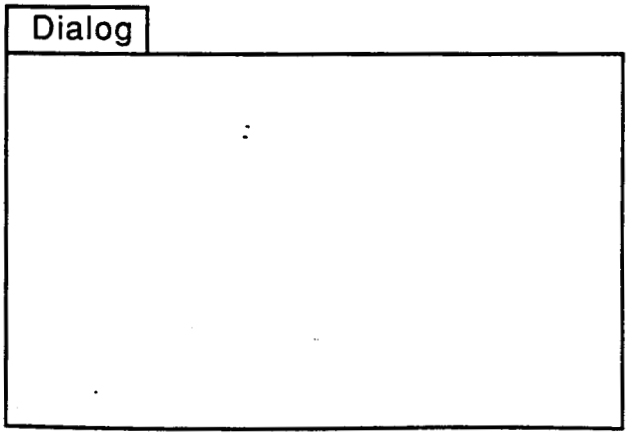
* Evolution of a broader theory of 'good' architectures utilizing human and computer decision makers in interactive control.

OFMTutor's Interface

Supports graphical, inspectable representation of joint hypotheses (expert and student)

Model of discourse enables conversational capabilities and supports repair





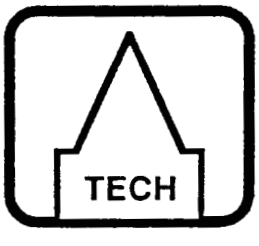
Center For Human-Machine Systems Research

Intent Inferencing with a Model-Based Operator's Associate

Patricia M. Jones
Christine M. Mitchell
Kenneth S. Rubin

Report No. 88-2

August 1988



School of Industrial and Systems Engineering
Georgia Institute of Technology
A Unit of the University System of Georgia
Atlanta, Georgia 30332

INTENT INFERENCING WITH A MODEL-BASED OPERATOR'S ASSOCIATE

Patricia M. Jones, Christine M. Mitchell, and Kenneth S. Rubin*

Center for Human-Machine Systems Research

School of Industrial and Systems Engineering

Georgia Institute of Technology

Atlanta, GA 30332

* Currently with ParcPlace Systems, 2400 Geng Rd., Palo Alto CA 94303

ABSTRACT

This paper describes a portion of the OFMspert (Operator Function Model Expert System) research project. OFMspert is an architecture for an intelligent operator's associate or assistant that can aid the human operator of a complex, dynamic system. Intelligent aiding requires both understanding and control. This paper focuses on the understanding (i.e., intent inferencing) ability of the operator's associate. Understanding or intent inferencing requires a model of the human operator; the usefulness of an intelligent aid depends directly on the fidelity and completeness of its underlying model. The model chosen for this research is the operator function model (OFM) (Mitchell, 1987). The OFM represents operator functions, subfunctions, tasks, and actions as a heterarchic-hierarchic network of finite state automata, where the arcs in the network are system triggering events. The OFM provides the structure for intent inferencing in that operator functions and subfunctions correspond to likely operator goals and plans. A blackboard system similar to that of HASP (Nii et al., 1982) is proposed as the implementation of intent inferencing function. This system postulates operator intentions based on current system state and attempts to interpret observed operator actions in light of these hypothesized intentions. The OFMspert system built for this research is tailored for the GT-MSOCC (Georgia Tech Multisatellite Operations Control Center) simulation. The GT-MSOCC OFMspert has been the subject of rigorous validation studies (Jones, 1988) that demonstrate its validity as an intent inferencer.

INTRODUCTION

Computational representations and models have been constructed for "understanding" human behavior in many applications; e.g., understanding natural language (Winograd, 1972) and understanding stories (Schank and Abelson, 1977). Artificial intelligence has developed many representational formalisms and control strategies that are intended to mimic "intelligent" behavior (cf Cohen and Feigenbaum, 1982). In the field of human-machine systems research, AI techniques offer powerful methodologies for understanding human behavior in the context of human-machine interaction.

Our particular concern is with human-machine interaction in the control of complex dynamic systems (e.g., nuclear power plants). Such systems are highly automated; thus, the human operator acts as a supervisory controller (Sheridan and Johannsen, 1976; Rasmussen, 1986; Wickens, 1984). Supervisory control typically consists of routine monitoring and fine-tuning of system parameters. However, in the event of abnormal or emergency situations, the human operator is expected to detect, diagnose, and compensate for system failures. The ability of a supervisory controller to cope with such situations can be severely limited. Wickens (1984) cites several problems with supervisory control: an increased monitoring load; a "false sense of security" whereby the operator trusts the automation to such an extent that any human intervention or checking seems unnecessary; and "out-of-the-loop familiarity" that implies a reduced ability to cope with non-routine situations.

An important question then becomes how to improve system performance and safety in supervisory control. The answer is not to automate the human out of the system; today's technology cannot match the human's ability to cope with uncertain and novel situations (Chambers and Nagel, 1985). Rather, automated systems must support the human operator. Given that the human will remain an integral part of a complex system, a potential approach to advanced automation is that of "amplifying" rather than automating human skills (Woods, 1986).

The OFMspert (Operator Function Model Expert System) project is an effort to develop a theory of human-computer interaction in supervisory control. OFMspert itself is a generic architecture for a computer-based operator's associate. The operator's associate (and similarly, the Pilot's Associate (Rouse

et al. 1987; Chambers and Nagel, 1985)) represents a design philosophy that allows the human to remain in control of a complex system. The computer-based associate is a subordinate to which the human operator can delegate control activities. The associate also actively monitors system state and operator actions in order to timely, context-sensitive advice, reminders, and suggestions. The intent is to provide intelligent support for the human operator.

The intelligence and utility of the operator's associate rest on its abilities to understand the operator's current intentions in order to provide context-sensitive advice and assume responsibility given for portions of the control task. Models of human-machine interaction offer a variety of frameworks for understanding human behavior (i.e., inferring intentions) in the control of a complex dynamic system (see Jones and Mitchell, 1987, and Jones, 1988, for a review). Knowledge-based problem solving strategies are tools for implementing and reasoning with the knowledge represented in the human-machine interaction model. OFMspert combines a particular human-machine interaction model (the operator function model (OFM) (Mitchell, 1987)) and knowledge-based problem solving approach (the blackboard model of problem solving (Nii, 1986)) to provide the understanding capability necessary for an effective operator's associate (Rubin, et al., 1987). In the next sections, the OFM and the blackboard model of problem solving are described. Next, ACTIN (Actions Interpreter), the intent inferencing component of OFMspert, is discussed, along with a detailed example of how ACTIN infers operator intentions dynamically. Finally, experimental results that validate ACTIN's intent inferencing ability are considered.

THE OPERATOR FUNCTION MODEL

The operator function model (OFM) (Mitchell, 1987) provides a flexible framework for representing operator functions in the control of a complex dynamic system. The OFM represents how an operator might organize and coordinate system control functions. Mathematically, the OFM is a hierarchic-heterarchic network of finite-state automata. Network nodes represent operator activities as operator functions, subfunctions, tasks, and actions. Operator functions are organized hierarchically as subfunctions, tasks, and actions. Each level in the network may be a heterarchy, i.e., a collection of activities that may be

performed concurrently. Network arcs represent system triggering events or the results of operator actions that initiate or terminate operator activities. In this way, the OFM accounts for coordination of multiple activities and dynamic focus of attention.

Historically, the OFM is related to the discrete control modeling methodology (Miller, 1985; Mitchell and Miller, 1986). The OFM is distinguished by its modeling of both manual and cognitive operator actions in the context of system triggering events. Manual actions are system reconfiguration commands. Cognitive actions include information gathering and decision making that are typically supported by information requests.

The OFM is a prescriptive model of human performance in supervisory control. Given system triggering events, it defines the functions, subfunctions, tasks, and actions on which the operator should focus. Used predictively, the OFM generates expectations of likely operator actions in the context of current system state. Used inferentially, the OFM defines likely operator functions, subfunctions, and tasks that can be inferred based on operator actions and system state. Thus, the OFM for a particular domain defines the knowledge needed to perform intent inferencing. What is needed next is a problem solving strategy to use this knowledge.

THE BLACKBOARD MODEL OF PROBLEM SOLVING

OFMspert's intent inferencing component, called ACTIN (Actions Interpreter), uses the HASP blackboard model of problem solving (Nii et al, 1982; Nii, 1986). The HASP blackboard is one of the few artificial intelligence systems that explicitly addresses real-time problem solving in dynamic environments.

The blackboard model of problem solving consists of three components: the blackboard, knowledge sources, and blackboard control. The blackboard is a data structure on which the current best hypothesis of the solution is maintained and modified. The hypothesis is represented hierarchically, at various levels of abstraction, and evolves incrementally over time as new data become available or old data become obsolete. Domain-specific knowledge is organized as a collection of independent knowledge sources. Knowledge sources are responsible for posting and interpreting information on the blackboard. Blackboard

control applies knowledge sources opportunistically; that is, in either a top-down or bottom-up manner, depending on what is more appropriate in the current context.

The blackboard model of problem solving is compatible with the knowledge represented in the OFM. Both models use a hierarchical representation. The blackboard knowledge sources provide a modularity that naturally represents much of the domain knowledge contained in the OFM arcs. The opportunistic control strategy offers the dynamic flexibility necessary for inferring intentions in real time. ACTIN combines the OFM representation of domain knowledge and the blackboard model of problem solving to dynamically construct and assess current operator intentions.

ACTIONS INTERPRETER (ACTIN)

ACTIN's blackboard represents operator intentions as a hierarchy of goals, plans, tasks, and actions that correspond to the OFM's hierarchy of functions, subfunctions, tasks, and actions. Goals are currently instantiated functions, plans are currently instantiated subfunctions, and so on. In some respects, ACTIN is a process model that uses the blackboard problem solving method to build a dynamic representation of current operator intentions based on the OFM's static knowledge (Wenger, 1987).

The general mechanism for the blackboard approach to intent inferencing is as follows. Given an OFM, currently hypothesized goals, plans, and tasks (GPTs) or sometimes additional plans and tasks (PTs) for an existing goal are placed on the blackboard in response to system triggering events. The blackboard incorporates operator actions into the representation with opportunistic reasoning. Thus, actions can be immediately interpreted as supporting one or more current goals, plans, and tasks; and goals, plans, and tasks can be inferred on the basis of operator actions.

Construction knowledge sources are responsible for building the representation of goals, plans, tasks, and actions. These knowledge sources can further be characterized as either model-driven or data-driven. Model-driven knowledge sources are those that post GPT information on the blackboard in response to system triggering events as defined by the OFM. Data-driven knowledge sources are those that post operator actions and attempt to infer support for any current tasks on the blackboard. Data-driven knowledge

sources may also postulate GPT information on the basis of operator actions. Assessment knowledge sources are responsible for evaluating the extent to which operator actions support currently hypothesized goals, plans, and tasks. Assessments are always made in the context of a particular goal or plan which forms the context for possible advice or reminders.

In order to illustrate ACTIN's dynamic intent inferencing, it is first necessary to describe the application domain for which our OFMspert was built: the Georgia Tech Multisatellite Operations Control Center (GT-MSOCC). After describing GT-MSOCC and its OFM, an example of ACTIN's intent inferencing is presented.

GT-MSOCC: APPLICATION DOMAIN

GT-MSOCC is a real time, interactive simulation of MSOCC, a NASA ground control station for near-earth satellites (Mitchell, 1987). MSOCC is a facility for capturing and processing data sent by satellites (see Figure 1). GT-MSOCC is a research domain designed to support theoretical and empirical research on human-computer interaction in the context of a complex dynamic system. It is a high fidelity simulation of the operator interface to an actual NASA ground control system. For more detail, see Mitchell, 1987.

GT-MSOCC operator activities are defined by the GT-MSOCC OFM. At the highest level of the GT-MSOCC operator function model are major operator functions and the system events that cause the operator to transition among functions (see Figure 2). This level of description represents operator goals in the context of current system state. The arcs define system events that trigger a refocus of attention or the addition of a function to the current set of operator duties.

The default high-level function is to control current missions. This involves the subfunctions of monitoring data transmission and hardware status, detection of data transmission problems, and compensation for failed or degraded equipment. Each subfunction is further defined by a collection of tasks, which in turn are supported by operator actions (system reconfiguration commands or display requests).

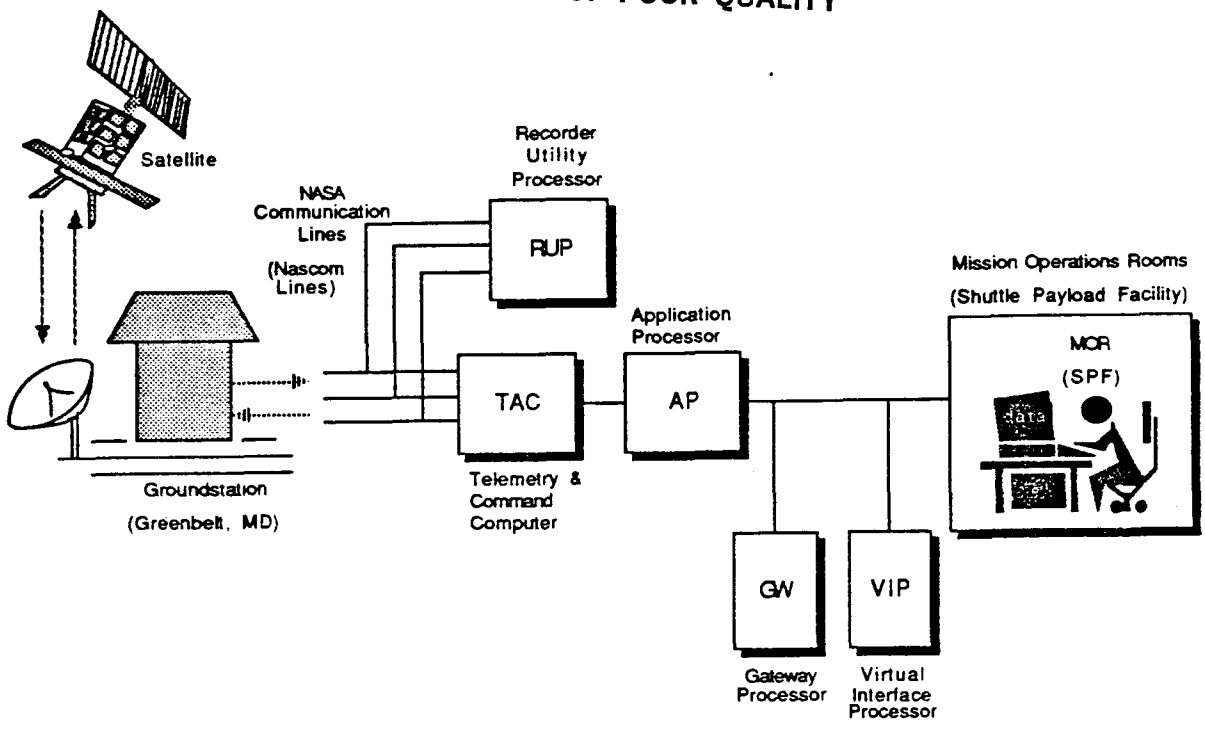
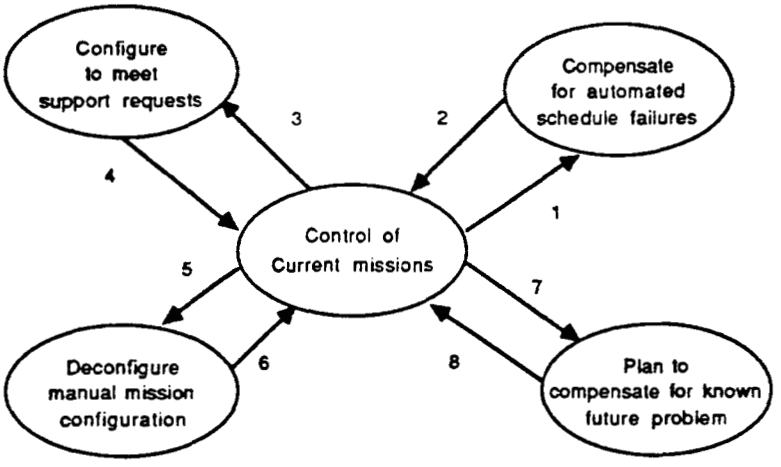


Figure 1. Multisatellite Operations Control Center (MSOCC)

System triggering events cause the operator to focus attention on other high-level functions. An unscheduled support request causes the operator to shift to the "configure to meet support requests" function. An error message from the automatic scheduler causes the operator to transition to the function to compensate for the automated schedule failure. A request to deconfigure a mission causes the operator to shift to the function of deconfiguring a manual mission configuration. Finally, the operator may engage in long-term planning in the absence of other system triggering events. Upon the termination of these other functions, the operator resumes the default control of current missions function. Functions may be terminated by their successful completion or the determination that they cannot be completed.

ACTIN'S INTENT INFERENCING WITH GT-MSOCC

In this section, a detailed example of ACTIN's intent inferencing is provided in the context of GT-MSOCC. Table 1 shows the organization of GT-MSOCC goals, plans, tasks, and actions, as defined by the GT-MSOCC OFM. Given system triggering events, ACTIN's model-driven knowledge sources post the



1. Error message received from the automatic scheduler.
2. Compensation completed or unable to compensate.
3. Unscheduled support request received by the operator.
4. Request configured or unable to meet the request.
5. Message received that a manually configured mission is completed.
6. Deconfiguration completed.
7. Operator summons schedule and/or mission template pages when no other triggering event takes place.
8. Terminate planning function.

Figure 2. GT-MSOCC Operator Functions

appropriate goal, plan, and task (GPT) structures on the blackboard. When operator actions occur, ACTIN's data-driven knowledge sources post actions on the blackboard and attempt to "connect" the actions to tasks which they support. This "connection" between actions and tasks defines ACTIN's intent inferencing capability. The knowledge of appropriate inferences of intent is contained in a data structure that matches actions to task types. Data-driven knowledge sources consult this structure to determine that task type(s) that a current operator action can support. They then search the blackboard's task level of abstraction for those types, and connect the action to all appropriate tasks.

To illustrate ACTIN's dynamic construction of operator intentions, consider the following scenario from GT-MSOCC. The scenario is described in terms of GT-MSOCC system events and operator actions, which then cause activity on the blackboard. ACTIN's intent inferencing results in statements written to a logfile. In the accompanying figures, the current blackboard structure is shown, along with ACTIN's inferences of intent.

1). The PM mission is automatically configured. ACTIN's model-driven knowledge sources post the goal to control the current mission (CCM) for PM. This goal is comprised of two plans: to monitor data

Table 1. GT-MSOCC Goals, Plans, Tasks, and Actions

Goals	Plans	Tasks	Actions
Control current mission CCM	Monitor software (MSW)	Check MOR (CMOR)	telem
		Check endpoints (CEND)	rup/gw/vip/cms telem
	Monitor hardware (MHW)	Check hardware (CHW)	–
Manual Configure Request CCN	Check system constraints (CSC)	Check current number of missions (CCNM)	–
		Check mission schedule (CMS)	msocc sched, msn scheds
		Check scheduled number of missions (CSNM)	msocc sched,pending
	Check mission requirements (CMR)	Check mission template (CMT)	msocc sched, msn scheds
	Identify candidate hardware (ICH)	Find current (FCUR)	–
		Find unscheduled (FUSC)	equip scheds, avails
	Answer question (ANQ)	Execute answer(XAN)	operator answer
		Execute configure (XCON)	manual config. (MCON) events
Compensate for Schedule Failure CFSF	Reconfigure (RCON)	Find duration (FDUR)	telem, pending
		Execute reconfigure(XRCO)	man. reconfig (MRCO), events
		For each equipment:	
		Find current (FCUR)	–
		Find unscheduled (FUSC)	equip scheds, avails
——	Manual Deconfigure Request (DCON)	Execute deconfigure (XDCO)	man. deconfig(MDCO), events
——	Troubleshoot (TBLS)	Check endpoints (CEND)	gw/rup/cms/vip telem
		Check interior (CIN)	nas/tac/ap/modlan telem
——	Replace(HRPL or SRPL)	Find duration (FDUR)	telem, pending
		Find current (FCUR)	–
		Find unscheduled (FUSC)	equip scheds, avails
		Execute replace(XRPL)	replace (RPL)

transmission or software (MSW) and to monitor hardware status (MHW). Each plan is composed of one or more tasks. The monitor software plan consists of two tasks: to check data flow at the MOR (CMOR) and to check data flow at endpoint equipment (CEND). The monitor hardware plan consists of the single task to check hardware status (CHW). This entire GPT structure defines the control of current mission function prescribed by the OFM. When PM is configured, ACTIN's knowledge sources retrieve the control of

current mission GPT structure, fill in mission-specific information (e.g., the name of this particular mission is PM), and post the structure on the blackboard. The resulting blackboard is shown in Figure 3a.

2). Another mission (Geographic Explorer, or GEO) is configured. In the same way the control of current mission GPT was posted for PM, a control of current mission GPT for GEO is also posted. The resulting blackboard is shown in Figure 3b.

3). The operator requests the main telemetry page ("telem"). ACTIN's data-driven knowledge sources determine that the current action type is "telem" and that actions of this type potentially support the tasks of checking the MOR (CMOR) and finding the duration (FDUR) of current missions. Upon examining the tasks level of the blackboard, the knowledge sources find that two eligible tasks are posted: the CMOR tasks for PM and GEO. Thus, the "telem" action is posted and connected to the CMOR tasks. The resulting blackboard is shown in Figure 3c.

4). The operator requests the gateway telemetry page ("GwTelem"). ACTIN's data-driven knowledge sources determine that the current action type is "GwTelem" and that actions of this type potentially

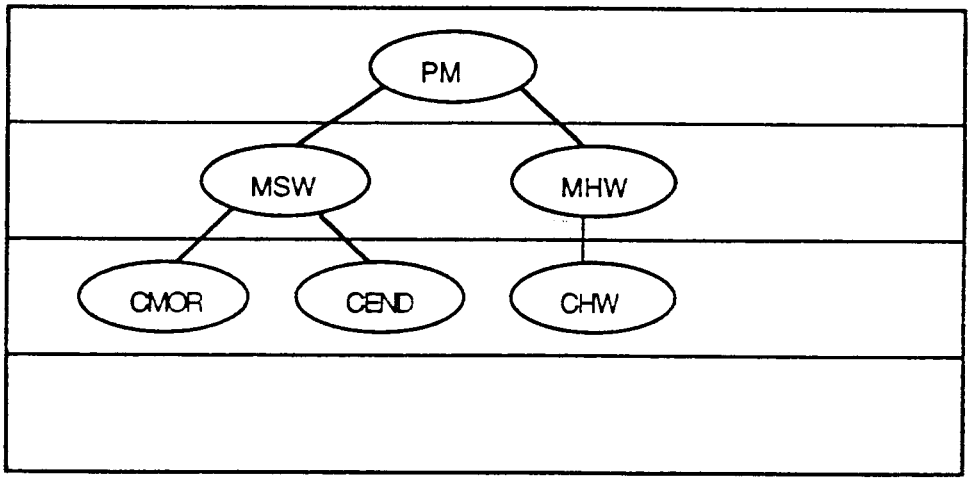


Figure 3a. Blackboard after PM is configured.

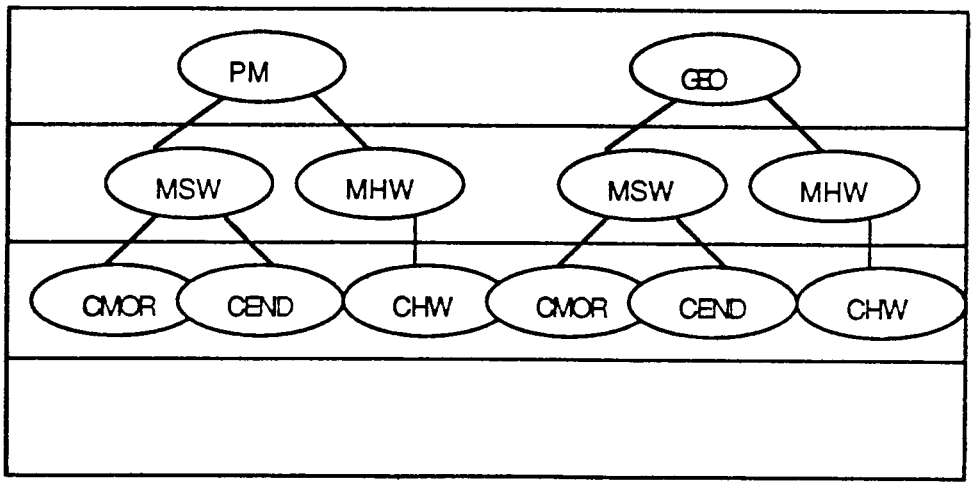
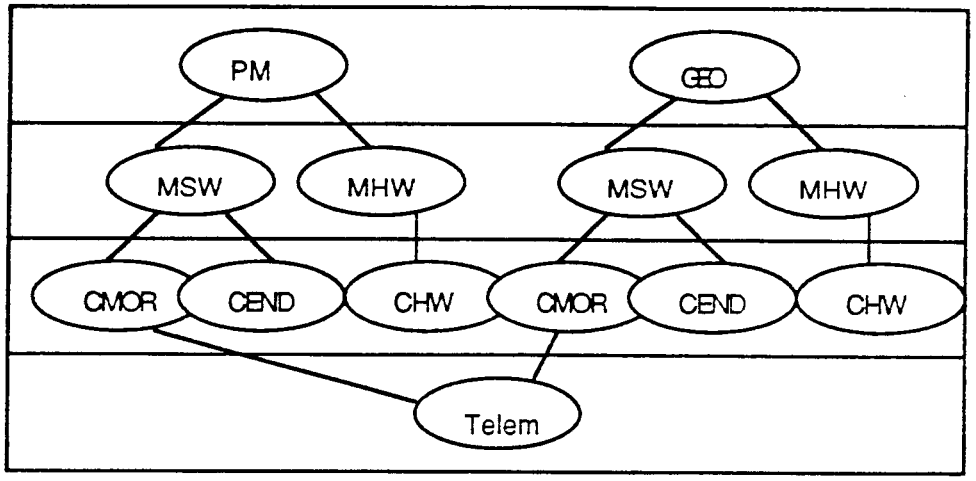


Figure 3b. Blackboard after GEO is configured.



Telem is interpreted as supporting CMOR for PM, CMOR for GEO

Figure 3c. Blackboard after Telem page request.

support the tasks of checking the endpoint equipment (CEND) of current missions. Upon examining the tasks level of the blackboard, the knowledge sources find that two eligible tasks are posted: the CEND

tasks for PM and GEO. Thus, the "GwTelem" action is posted and connected to the CEND tasks. The resulting blackboard is shown in Figure 3d.

5). One of the components used by PM experiences a hardware failure. The component in this example is RUP2. Upon the occurrence of this triggering event, ACTIN's model-driven knowledge sources post a plan to replace the failed component, along with the four associated tasks of finding a currently available replacement (FCUR), finding the duration of the mission (FDUR), finding an unscheduled replacement (FUSC), and executing the replace command (XRPL). The resulting blackboard is shown in Figure 3e.

6). The operator again requests the main telemetry page. This time ACTIN's knowledge sources determine that this action can support three tasks on the blackboard: FDUR for RUP2 and CMOR for both PM and GEO. The resulting blackboard is shown in Figure 3f.

7). The operator requests the schedule for RUP1 ("Rup1Sched"). ACTIN's data-driven knowledge sources determine that the current action type is "Rup1Sched" and that actions of this type potentially support the task of finding unscheduled equipment (FUSC) for RUP components. Upon examining the tasks

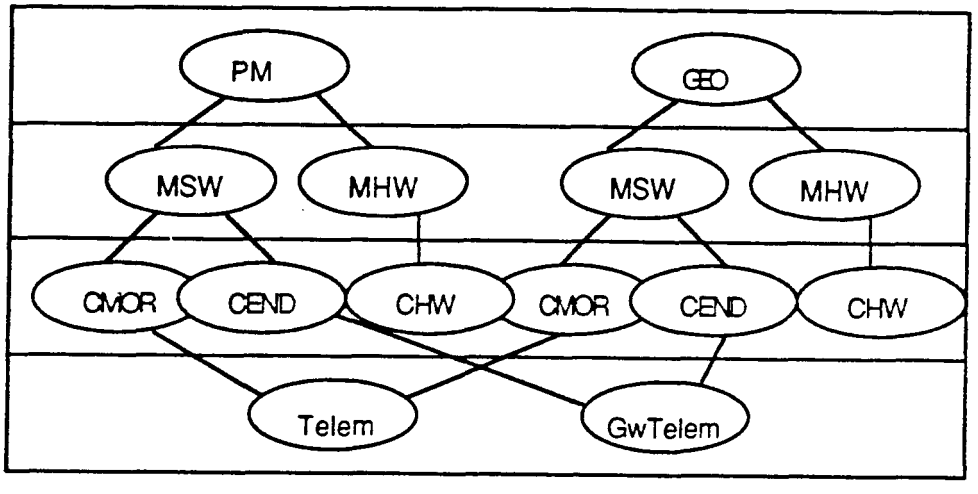


Figure 3d. Blackboard after GwTelem page request.

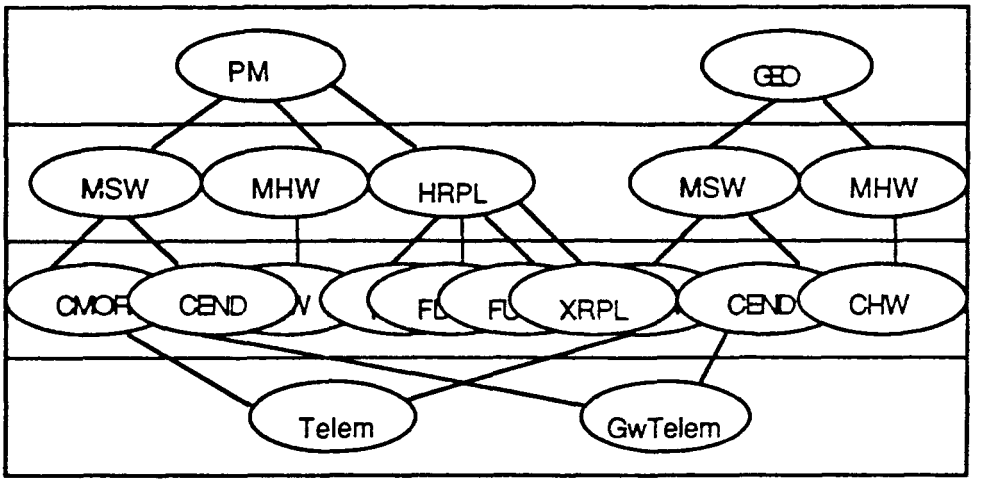
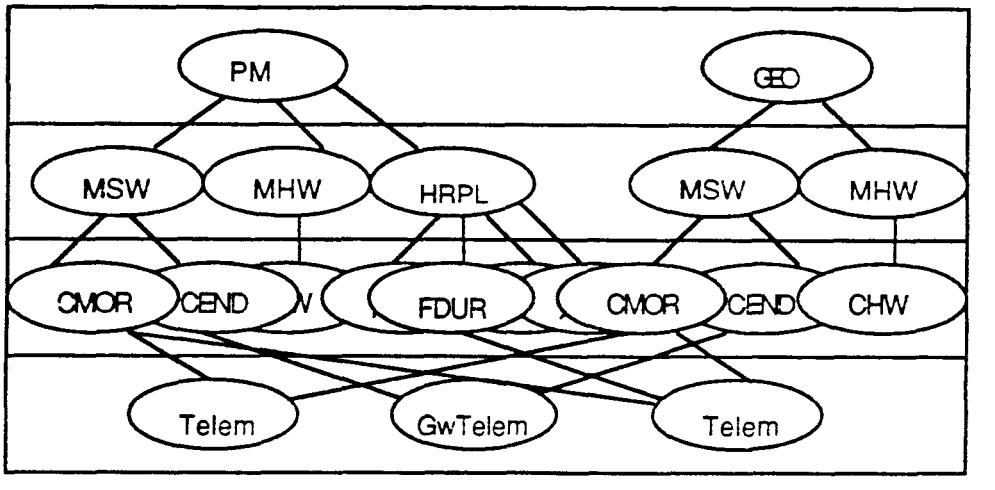


Figure 3e. Blackboard after RUP2 hardware failure.



Telem is interpreted as supporting CMOR for PM, CMOR for GEO, FDUR for RUP2

Figure 3f. Blackboard after Telem page request.

level of the blackboard, the knowledge sources find that one eligible task is posted: the FUSC task for RUP2. Thus, the "Rup1Sched" action is posted and connected to the FUSC task associated with the RUP2

replace plan. The resulting blackboard is shown in Figure 3g.

8). Finally, the operator requests the schedule for NAS5. ACTIN's data-driven knowledge sources determine that this request potentially supports finding unscheduled NAS components (i.e., the FUSC task associated with any NAS component). However, although a FUSC type task is posted, it is not associated with a NAS type component. ACTIN is unable to interpret this request as supporting any current tasks. Thus, the "Nas5Sched" request action is posted, but not connected to any current tasks. Figure 3h illustrates the resulting blackboard.

Several characteristics of ACTIN's interpretation algorithm are notable. First, actions are immediately connected to whatever appropriate tasks exist on the blackboard *at the time the actions are posted.* Connection links are not inferred after the action is posted.

Another important feature is ACTIN's property of maximal connectivity. That is, ACTIN interprets actions in the broadest possible context, assuming that the operator is extracting the maximum amount of information from the display pages requested. In the example above, ACTIN inferred that the second telem

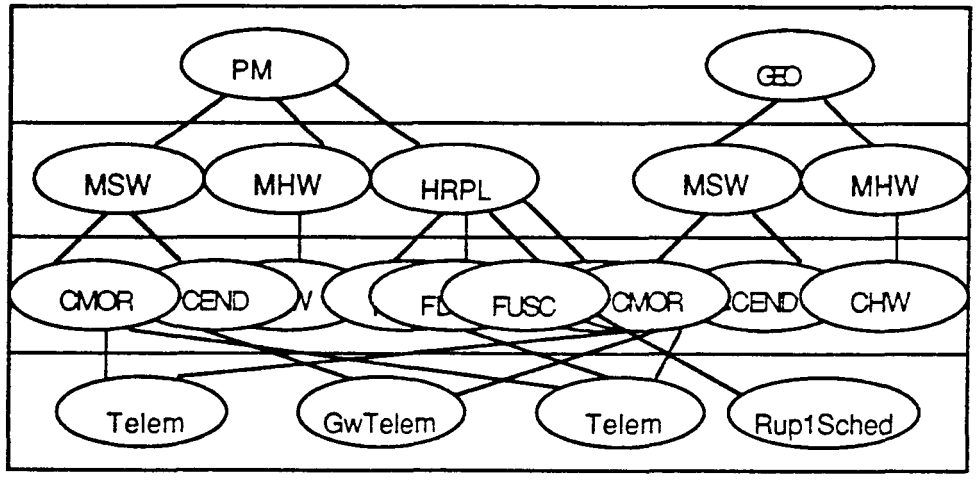


Figure 3g. Blackboard after Rup1 Schedule request.

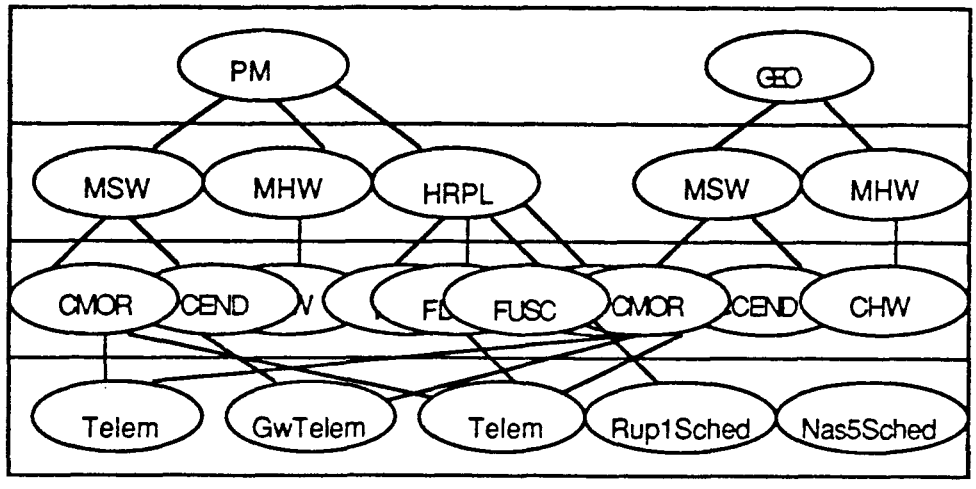


Figure 3h. Blackboard after Nas5 Schedule Request.

action supported all current CMOR tasks as well as the FDUR task for RUP2. Thus, the operator is "given the benefit of the doubt" in the evaluation of performance.

The evaluation of operator performance is performed by knowledge sources that assess the degree to which operator actions support current tasks (and by extension, plans and goals). ACTIN schedules assessments periodically in the context of particular goals or plans. In the example above, ACTIN schedules separate assessments for the control of current mission goals for PM and GEO, and the replace plan for RUP2. Assessments note the number of supporting actions and the time at which those actions occurred. The assessments for PM and GEO would note that the CMOR task is supported by two actions and the CEND task is supported by one action. RUP2's replace plan assessment would state that one action supports the FDUR task and one action supports the FUSC task. The results of these assessments are written to a logfile.

To summarize, the proposed model for intent inferencing uses the OFM methodology to postulate operator functions, subfunctions, and tasks on the basis of current system state and observed operator actions. This model has been implemented using a blackboard architecture. This structure, of which the

scenario described in this section is an example, defines the context for intent inferencing.

The OFM and its implementation in ACTIN is an example of "the middle ground" in theory construction in cognitive science (Miller, Polson, and Kintsch, 1984). The theory has well-defined structures and processes to "support both the instantiation of the theory as an executable computer program and qualitative experimental studies of the theory" (Miller, Polson, and Kintsch, 1984, p. 13). In the next section the validation of the proposed model is explored. A two-stage framework for validation is proposed, and experimental results are briefly discussed.

EXPERIMENTAL VALIDATION

Validation of intent inferencing assures that the system is correctly inferring the intentions of the human operator. Within the context of the OFM structure of intentions, this means that the system infers support for the same tasks (and by extension, plans and goals) as the human, given the same set of operator actions. The "human" in this case can be a human domain expert performing a post hoc analysis, or the human operator giving an (on-line) account of intentions. Thus, the proposed two-part framework for the validation of intent inferencing is 1.) comparison of expert and OFMspert analyses; and 2.) comparison of concurrent verbal protocols and OFMspert analysis (see Jones, 1988, for more detail).

The experimental validation of ACTIN's intent inferencing was conducted in two studies. In Experiment 1, a domain expert's interpretations of operator data were compared to ACTIN's interpretations of those same actions on an action-by-action basis. In Experiment 2, verbal protocols were collected from GT-MSOCC operators while they were controlling GT-MSOCC. Statements of intentions for each action were compared to ACTIN's interpretations.

The results of these studies are discussed in detail in Jones (1988). Overall, the results showed that ACTIN's intent inferencing ability compared favorably to inferences made by a domain expert and statements from verbal reports.

ACKNOWLEDGEMENTS

This research was supported by NASA Goddard Space Flight Center Contract Number NAS5-28575 (Karen Moe and Walt Truszkowski, technical monitors) and by NASA Ames Research Center Grant Number NAG-413 (Dr. Everett Palmer, technical monitor), awarded to Dr. Christine M. Mitchell.

REFERENCES

Chambers, A. B. and Nagel, D. C., 1985, Pilots of the future: Human or computer? *Communications of the ACM, 28,* 11, 1187-1199.

Cohen, A. and Feigenbaum, E. A., 1982, *The Handbook of Artificial Intelligence.* Reading, MA: Addison-Wesley.

Jones, P. M., 1988, Constructing and validating a model-based operator's associate for supervisory control. Report No. 88-1, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Jones, P. M. and Mitchell, C. M., 1987, Operator modeling: Conceptual and methodological distinctions. *Proceedings of the 31st Annual Meeting of the Human Factors Society, 1,* 31-35. Santa Monica, CA: Human Factors Society.

Miller, J. R., Polson, P. G., and Kintsch, W., 1984, Problems of methodology in cognitive science. In W. Kintsch, J. R. Miller, and P. G. Polson (Eds.), *Method and tactics in cognitive science.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Miller, R. A., 1985, A systems approach to modeling discrete control performance. In W. B. Rouse, (Ed.), *Advances in Man-Machine Systems Research,* Vol. 2, 177-248. New York: JAI Press.

Mitchell, C. M., 1987, GT-MSOCC: a research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17,* 553-572.

Mitchell, C. M. and Forren, M. G., 1987, Multimodal user input to supervisory control systems: Voice-augmented keyboards. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17,* 594-607.

Mitchell, C. M. and Miller, R. A., 1986, A discrete control model of operator function: A methodology for information display design. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-16,* 343-357.

Mitchell, C. M. and Saisi, D. L., 1987, Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17,* 573-593.

Nii, H. P., Feigenbaum, E. A., Anton, J. J., and Rockmore, A. J., 1982, Signal-to-symbol transformation: HASP/SIAP case study. Heuristic Programming Project, Report No. HPP-82-6. Heuristic Programming Project, Stanford University, Stanford, CA.

Nii, H. P., 1986, Blackboard systems. *AI Magazine, 7-2, 7-3.*

Rasmussen, J., 1986, *Information processing and human-machine interaction: An approach to cognitive engineering.* New York: North-Holland.

Rouse, W. B., Geddes, N. D., and Curry, R. E., 1987, An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction, 3.*

Rubin, K. S., Jones, P. M. and Mitchell, C. M., 1987, OFMspert: Application of a blackboard architecture to infer operator intentions in real time decision making. Report No. 87-6, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. Also *IEEE Transactions on Systems, Man, and Cybernetics,* to appear.

Schank, R. C. and Abelson, R. P., 1977, *Scripts plans goals and understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Sheridan, T. B. and Johannsen, G., 1976, *Monitoring Behavior and Supervisory Control.* New York: Plenum.

Wenger, E., 1987, *Artificial intelligence and tutoring systems.* Los Altos, CA: Morgan Kaufmann.

Wickens, C. D., 1984, *Engineering Psychology and Human Performance.* Columbus, OH: Charles Merrill.

Winograd, T., 1972, *Understanding natural language.* New York: Academic Press.

Woods, D. D., 1986, Cognitive technologies: The design of joint human-machine cognitive systems. *The AI Magazine,* 86-92.

OFMTutor

An Operator Function Model Intelligent Tutoring System

Patricia M. Jones

ISYE 8706

8 November 1988

INTRODUCTION

This paper proposes the design, implementation, and evaluation of OFMTutor, an Operator Function Model intelligent tutoring system. OFMTutor is intended to provide intelligent tutoring in the context of complex dynamic systems for which an operator function model (OFM) (Mitchell, 1987) can be constructed. The human operator's role in such complex, dynamic, and highly automated systems is that of a supervisory controller whose primary responsibilities are routine monitoring and fine-tuning of system parameters and occasional compensation for system abnormalities (Sheridan and Johannsen, 1976; Wickens, 1984).

The ability of a supervisory controller to cope with abnormal or emergency situations can be severely limited. Wickens (1984) cites several problems with supervisory control: an increased monitoring load; a "false sense of security" whereby the operator trusts the automation to such an extent that any human intervention or checking seems unnecessary; and "out-of-the-loop familiarity" that implies a reduced ability to cope with non-routine situations.

An important question then becomes how to improve system performance and safety in supervisory control. The answer is not to automate the human out of the system; today's technology cannot match the human's ability to cope with uncertain and novel situations (Chambers and Nagel, 1985). Rather, automated systems must support the human operator.

One potentially useful form of support is the use of intelligent tutoring systems to teach the operator about the system and how to function within that system. In the next section, previous research on intelligent tutoring systems (ITS) is considered. Then the proposed design for OFMTutor is presented, and an experimental evaluation is described.

INTELLIGENT TUTORING SYSTEMS RESEARCH

Intelligent tutoring systems are usually described in terms of three modules: an expert module that represents domain expertise; a student model that represents the student's performance record and presumed state of knowledge; and a tutorial module that structures the interaction between the tutor and the

student (Fath et al., 1988; Park et al., 1987; Wenger, 1987). Wenger (1987) considers the interface to be a fourth critical component for the successful implementation of a knowledge communication system. The following discussion of ITS research is divided into these broad categories of domain expertise representations, student modeling, pedagogical strategies, and interface design. Particular attention is paid to efforts that involve complex dynamic systems and/or complex problem solving tasks.

Domain Expertise

According to Wenger (1987), domain expertise forms the source of knowledge to be communicated and the standard for evaluating performance. Thus, the teaching goal is explicitly represented as this knowledge. The degree to which such domain expertise may be articulated is dependent upon the transparency of the expert model's structure and its psychological plausibility. Thus, knowledge whose structure is transparent to the student and whose organization and form are psychologically plausible is knowledge that can be relatively easily communicated to the student.

The SOPHIE project involved a series of tutoring programs for electronics troubleshooting (Burton et al., 1982; Wenger, 1987). The first version of SOPHIE (SOPHIE-I) demonstrates the efficiency and robustness of an inference engine that uses multiple representations of domain knowledge: a simulation-based mathematical model of a circuit; procedural knowledge, organized as a collection of specialists, to act on the circuit model; and declarative knowledge organized as a semantic net of facts. SOPHIE-I was used as a supplemental laboratory in electronics troubleshooting instruction. SOPHIE-II extends SOPHIE-I to include an articulate troubleshooting expert to demonstrate strategies to the student. The emphasis is on articulation of expertise in qualitative, causal terms. Finally, SOPHIE-III is meant to support learner-centered activities, while providing powerful inference capabilities and supporting good explanations. SOPHIE-III's expertise is represented as two separate modules: a troubleshooting expert and an electronics expert. The architecture supports flexible, humanlike reasoning.

Like SOPHIE, the Recovery Boiler Tutor (RBT) (Woolf, 1986) supports a "reactive learning environment" which includes a simulation of the system of interest (a kraft recovery boiler) and in which the student is allowed to propose hypotheses that can be evaluated in real time. Also like SOPHIE, RBT's domain

expertise concerns fault detection and diagnosis. The domain knowledge is represented as a knowledge base of scenarios which describe preconditions, postconditions, and solutions for emergencies or operating conditions.

The AHAB system represents the realization of a proposed ITS architecture for troubleshooting in complex dynamic systems (Fath, 1987; Fath et al., 1988). AHAB works in conjunction with PEQUOD, a marine steam powerplant simulation, to tutor students in troubleshooting strategies. AHAB's domain expertise (a "task model") prescribes troubleshooting actions based on current system state and represents psychologically plausible troubleshooting strategies (i.e., symptomatic and topographic search (Rasmussen, 1986)). The task model is structured as an operator function model (OFM) (Mitchell, 1987), together with representations of declarative and procedural knowledge.

AHAB's OFM provides the richest, most efficient structure for domain expertise of all three systems reviewed here. It accounts for the coordination of strategy and dynamic focus of attention based on current system state. The OFM will be described in more detail in the section on the design of OFMTutor.

Wenger discusses several issues in the representation of domain expertise. The representation should be complete; that is, the expertise should be a process model that has knowledge of the domain and also metaknowledge about how to use it. A process model must be able to solve problems that the student is expected to learn.

Domain knowledge also needs to be relevant to the student. To this end, the process of warranting belief (i.e., justification of new knowledge with respect to previous knowledge and beliefs) is important. It is crucial to motivate the concept to be taught with references to previous knowledge and beliefs held by the student. This serves to justify new knowledge.

Finally, a critical distinction in domain knowledge is whether it is compiled or articulate. Compiled knowledge is "automatic"; it is efficient and simple to use, but no longer possesses transparency and generality. In particular, compiled knowledge does not support the warranting process. Articulate knowledge, on the other hand, is able to support warranting belief via *organization* in terms of decomposition into primitives and configuration of primitives into a model and *justification* in terms of "first principles" (e.g.,

causality, structure, functionality, teleology) and integration.

Student Modeling

Intelligent communication requires understanding of the recipient (Wenger, 1987). Thus, an intelligent tutoring system must possess some model of the student's current state of knowledge. In particular, an ITS uses student actions as data for interpretation and reconstruction of presumed states of knowledge. This requires the explicit consideration of a model of the student. The student model is needed by the tutor to guide the student's problem solving and to organize the learning sequence (Wenger, 1987).

In general, student modeling employs the technique of "differential modeling" -- that is, a comparison of expert and student performance and/or knowledge (Burton and Brown, 1982). Differential modeling requires that the expert and student models have the same structure for unambiguous comparisons to occur. Student modeling techniques may be broadly classified as overlay models or buggy models (Park et al., 1987; Wenger, 1987). An overlay model assumes that the student's knowledge is a subset of expert knowledge; differences between the student and expert models are due to incompleteness of student knowledge. An example of an overlay model is Goldstein's genetic graph (Goldstein, 1982). Buggy models explicitly capture misconceptions as a collection of "bugs" (with or without an accompanying theory of the origin of these bugs); differences between the student and expert models are due to the student's "buggy" deviations (Burton, 1982; Johnson and Soloway, 1985). Buggy models are employed in BUGGY and its variants (Burton, 1982) and PROUST (Johnson and Soloway, 1985).

Two particular student modeling techniques are relevant for our discussion. The first is the idea of the "limited bug model" used in AHAB. AHAB's student model is very similar in structure to the task model, and thus the two can be compared via differential modeling in the spirit of an overlay model. However, AHAB also represents student errors in terms of common or important general types of errors. Thus, errors are not exhaustively enumerated a priori, but broad categories of errors can be used to identify the source of a difference between the student and task models.

The second important consideration is that of student intentions. By utilizing an explicit account of plausible student intentions (i.e., goals and plans), performance can be better understood and thus

diagnosed properly and remedied in context (Genesereth, 1982).

> Plan recognition is a way of using information about the student's actions in dealing with the combinatorics in domains where the number of reasonable solutions and bugs is too large for the expert difference technique to work effectively. ... In addition to helping pinpoint the student's misconception, studying his plan is advantageous in that it enables the tutor to offer remediation in the context of the student's problem and his approach to solving it. (p. 140)

Similarly, Johnson and Soloway (1987) argue that "knowledge of intentions makes it possible to identify more bugs, as well as to understand their causes" (Johnson and Soloway, 1987, p. 50). Thus, it is desirable to account for plausible student intentions explicitly in the design of a student model.

Pedagogical Strategies

Pedagogical strategy defines the organization, sequencing, and form of the student-tutor interaction; it designates what to say and how and when to say it. Many intelligent tutoring systems employ the guided discovery or coaching method in which the student "learns by doing" with the assistance of a non-intrusive coach (Park et al., 1987; Burton and Brown, 1982; Wenger, 1987). The purpose of the coach is to "foster the learning inherent in the activity itself by pointing out existing learning opportunities and by transforming failures into learning experiences" (Wenger, 1987, p. 124).

WEST is one of the earliest and most influential computer coaches. WEST assists students in playing the game "How the WEST was Won" (Burton and Brown, 1982). WEST uses the "issues and examples" paradigm to find issues where a student is weak (via differential modeling) and then to provide examples to illustrate better moves. The guiding principle behind such a pedagogical strategy is to make interventions both relevant and memorable (Wenger, 1987). WEST also employs a number of tutoring principles that govern its intervention capabilities (e.g., "Never tutor on two consecutive moves") (Burton and Brown, 1982).

SOPHIE and RBT both provide a "reactive learning environment" in which students can "play" with the simulation and observe the effects of their manipulations. STEAMER is an inspectable, interactive simulation of a propulsion plant that also allows students to manipulate system conditions and events (Hollan et al., 1987). Fath (1987; Fath et al., 1988) explicitly considers the simulation of a complex system as part of the instructional media; the simulation is important in supporting students' understanding of the system (i.e., building of accurate mental models (Hollan et al., 1987; Wenger, 1987)). Such simulations are important pedagogically in that they can serve as a form of continuous explanation to the student (Wenger, 1987).

An important facet of pedagogy is diagnosis of student misconceptions. Diagnosis updates the student model to reflect issues that need to be addressed in the interaction. Wenger (1987) distinguishes between three levels of diagnostic activities: behavioral, epistemic, and individual. Behavioral diagnosis is concerned with behavior and the product of behavior. It is further characterized as *non-inferential classification* (an evaluation of student performance in terms of correctness) or *inferential reconstruction* that is concerned with the problem solving process. The latter form of behavioral diagnosis is of concern here; inferential reconstruction deals with the use of plans and goals in reconstruction of problem solving behavior. Of especial importance are the PROUST and MACSYMA Advisor systems that explicitly represent student intentions. High-level goals are decomposed into plans and actions; diagnosis is a process that alternates "between model-driven confirmation and data-driven recognition of plans and goals" (Wenger, 1987, p. 374).

An important issue in instructional systems in general is principled curriculum design. While the so-called "frame-based" computer-based instructional methods made an effort to take these considerations into account, much of the research in intelligent tutoring systems does not make use of a theory of learning or instruction (Lesgold, 1988; Park et al., 1987). However, some efforts have been made to consider curricula in ITS research. Wenger (1987) discusses the concept of a *bite-sized tutoring architecture* (proposed by Bonar and his colleagues) in which the system is organized around pedagogical issues called bites. Each bite focuses on a particular aspect of domain knowledge and also includes knowledge of its conceptual and

curricular relations to other bites, the student's mastery of that bite's subject matter, and the abilities to diagnose, generate problems, and generate instructional interventions.

Lesgold (1988) points out that

> Where conventional instruction has an explicit curriculum but fails to have an explicit and complete representation of the knowledge that is to be taught, intelligent instructional systems have tended to represent the target knowledge explicitly but not to represent explicitly that body of knowledge that specifies the goal structure for instruction, the curriculum. (p. 117)

Lesgold argues that an intelligent tutor must represent domain knowledge, curriculum knowledge, and knowledge of metaissues that affect instruction. Domain knowledge includes procedural and declarative (i.e., conceptual) knowledge. Curriculum knowledge is represented as a lattice of goals that are decomposed progressively into subgoals, down to lessons that can be taught completely as a unit. This structure is based upon Gagne's (1971) learning hierarchy, in which the goal of instruction is progressively refined down to the level of individual lessons. The curriculum goal lattice is composed of a number of such goal hierarchies, each of which corresponds to a particular viewpoint of domain knowledge. Finally, the metaissue layer relates to knowledge of student aptitude (e.g., "good at math") and is defined as the topmost goal nodes in the curriculum lattice (i.e., the origins of the various viewpoint hierarchies).

Lesgold emphasizes that the implicit idea of Gagne's learning hierarchy is that the whole is more than the sum of its parts; higher levels in the hierarchy also provide "conceptual glue" that relates lower level knowledge. Furthermore, he argues that the knowledge taught in a lesson depends upon the context in which the lesson is taught. When a lesson is first taught, its "core content" (i.e., a coherent subset of knowledge) should be presented, but if a lesson is remedial, "it is crucial to teach the knowledge that links the core content of the to-be-remediated lesson with the core content of the lesson whose failure produced the need for remediation" (Lesgold, 1988, p. 134).

Gagne's ideas have had a large influence on instructional design practices. He and his colleagues have developed a well-specified approach to instructional design which can be summarized as follows (see Briggs, 1977a):

1. Needs Assessment

This is a process of identifying instructional goals, ranking them by importance, identifying one or more needs, and setting priorities for action.

2. Write Performance Objectives

Performance objectives translate goals into specific behavioral criteria for successful performance. The proposed Gagne-Briggs model of performance objectives distinguishes between action, object, situation, tools and other constraints, and the capability to be learned. The action denotes what observable behavior the student will perform (e.g, writing, running). The object denotes the resulting product of the action (e.g., a poem, a painting). The situation describes the circumstances in which the student will perform (e.g., given the PEQUOD simulation with one introduced fault). Tools and other constraints describe how the action will be carried out (e.g., with a pencil) and performance limits (e.g., without the use of references, within 30 minutes). The capability to be learned is inferred from the action; Gagne and Briggs have proposed a taxonomy of capabilities that distinguishes between intellectual skill, cognitive strategy, information, motor skill, and attitude. This taxonomy is shown in Table 1.

Insert Table 1

about here

An example of a problem-solving performance objective is as follows (see Kibler and Bassett, 1977). Suppose the domain of interest is instructional design, and students are to be taught how to write performance objectives. A reasonable performance objective might be: Given a general statement of the scope and

sequence of topics for a high school course (situation), generate (learned capability: problem solving) appropriate student objectives (object) by writing such objectives (action) within one week (tools, constraints, and special requirements).

3. Analyze Objectives

This is a kind of task analysis in which the taxonomy shown in Table 1 is used. The learning task is analyzed with respect to its essential and supporting prerequisites; such prerequisites define a learning hierarchy. For example, an intellectual skill has as essential prerequisites simpler component intellectual skills and as supporting prerequisites (i.e., those not essential for learning but that can be helpful) attitudes, cognitive strategies, and verbal information.

4. Design the Instructional Strategy

The learning hierarchy provides a prescription of how the instruction should be sequenced; prerequisite skills should be taught first. In other words, instruction proceeds "bottom up". At a more fine-grained level, Gagne distinguishes between nine instructional events (or teaching steps): gain attention, tell the student the objective, stimulate recall of prerequisites, present the stimulus material, provide guidance, elicit the performance, provide feedback, assess performance, and enhance retention and transfer. Of particular interest are Gagne's suggested forms of guidance for learning; these are reproduced in Table 2.

Insert Table 2

about here

---

5. Lesson Planning

The suggested steps in lesson planning are to identify the objective, list the desired instructional events, select ideal media, select materials and activities, analyze materials for events they supply, and plan other means for the remaining events (Briggs, 1977b).

6. Formative Evaluation

Formative evaluation is the process of testing and revising instructional materials while they are still being developed. Three suggested phases of formative evaluation are one-to-one, small group, and field trial evaluations (Dick, 1977a).

6. Summative Evaluation

Summative evaluation is the process of collecting and interpreting data about the quality of a proposed educational product. A suggested five-step approach to summative evaluation is to identify instructional objectives, identify the target population and select students from it, develop evaluation instruments (i.e, objectives-referenced tests, attitude questionnaires, and cost data), document the instructional process, and prepare the final report (Dick, 1977b).

This approach has been quite popular in the educational community; however, it is not without its critics. Novak (1986) argues that Gagne's model of learning (i.e., the learning hierarchy) is founded on a stimulus-response association. As such, this model, with its emphasis on "behavioral objectives" and suggested bottom-up approach to instructional sequences, is an outgrowth of behaviorist psychology. Novak argues for a constructivist, rather than a positivistic, view of epistemology. In other words, rather than an exclusive concern with observable data, we should recognize that "humans construct knowledge using the concepts, principles, and theories they have, and change their *knowledge claims* as new ideas and associated methodologies lead to new constructions of how people and the universe operate" (Novak, 1986, p. 6). Novak also emphasizes the importance of *concepts;* in fact, he states that "'concepts are what we think with.' As we change our concepts and conceptual frameworks in positive ways, we may or may not change our *behavior*, but the *meaning* of our experience changes" (Novak, 1986, p. 8).

Novak argues that a model of human learning is essential for a theory of education. Rather than a behaviorist model of learning, he advocates the theory proposed by educational psychologist David Ausubel. Ausubel contends that the single most important factor in learning is what the learner already knows. Effective instruction ascertains the learner's present state of knowledge and teaches accordingly. Ausubel

views knowledge as a *cognitive structure:* a hierarchical organization of concepts, where specific elements of knowledge are subsumed under more general concepts.

Another important idea is that of concept differentiation: "As new experience is acquired and new knowledge is related to concepts already in a person's mind, these concepts become elaborated or altered, and hence they can be related to a wider array of new information in subsequent learning" (Novak, 1986, p. 25). Thus, a learner's previous knowledge includes relevant concepts and the extent of their differentiation.

Ausubel distinguishes between rote and meaningful learning. Rote learning occurs when the new material is not associated with any existing elements in the cognitive structure. Meaningful learning occurs when new material is linked with subsuming concepts ("subsumers"). The new material is "stored in a somewhat altered form (as a product of assimilation with the subsuming concept(s)) and modifies (differentiates further) the subsumers to which it is linked" (Novak, 1986, p. 26).

Ausubel's theory implies that "concept development proceeds best when the most general, most inclusive elements of a concept are introduced first and then the concept is progressively differentiated in terms of detail and specificity" (Novak, 1986, p. 86). This is in direct contradiction to Gagne's prescription of teaching "bottom up". Ausubel's theory is more persuasive, however, in that "top down" instruction gives a context for learning, and, in Wenger's terms, may serve to warrant belief.

Novak emphasizes the distinction between curriculum (issues) and instructional (teaching) issues. This is similar to the separation of domain and tutorial knowledge that distinguishes the GUIDON system (Clancey, 1987). Novak draws on Johnson's model of curriculum design (Johnson, 1967), a simplified version of which is shown in Figure 1.

Insert Figure 1

about here

It is shown that a curriculum development system uses knowledge available from the culture in conjunction

with structuring and selection criteria to produce a curriculum. The curriculum is an input to structured planning, which also considers instrumental content and teaching behavior repertoire to produce an instructional plan. This plan is administered to learners; their performance is evaluated and relevant feedback is provided to both curriculum and instructional development.

Novak relates Ausubel's theory of learning to Johnson's model of curriculum and instructional development in order to specify the nature of each module. Ausubel's emphasis on concepts implies that the selection criteria for knowledge should be to select major and minor concepts in the field of study. The ordering criteria for knowledge should consider both progressive differentiation and integrative reconciliation:

> Meaningful learning and progressive differentiation require the most general, most inclusive concepts be presented early and subsequent information be provided to clarify meaning and show connections to subordinate concepts....Superordinate learning and integrative reconciliation require that subordinate concepts be presented in a manner that allows association with more inclusive concepts (superordinate concepts), and meanings of apparently disparate concepts will be clarified to show distinctions and relationships between subordinate concepts (integrative reconciliation). (Novak, 1986, pp. 137-138)

Ausubel's theory implies that the curriculum's "intended learning outcomes" (ILOs) should be the concepts to be learned, with associated hierarchical and subordinate relationships. The selected exemplars should be chosen such that "cognitive bridging" is provided (i.e., explicit association between new concepts and the existing cognitive structure). Teaching approaches should to flexible and allow for "hands-on experience" (or, in Piaget's terms, experience with "concrete props"). Actual learning outcomes are a function of the "degree of overall cognitive structure differentiation" and "initial or developed relevant subsumers in the learner's cognitive structure" (Novak, 1986, p. 139). Evaluation can be examined in terms of rate or degree of transfer of learning. The rate of learning depends on the "quality of existing or developed relevant subsumers, and motivation for learning. Transfer of learning to new problem solving situations

will be a function of the degree of concept differentiation, superordinate subsumption, and integrative reconciliation achieved." (Novak, 1986, p. 139). The feedback to curriculum planning may imply the need for "alternative sequences of concept presentation" or "better clarification of relationships between concepts...and/or better description of salient aspects of the concept(s)" (Novak, 1986, p. 139). Finally, the feedback to instruction may indicate the need to select better exemplars (i.e., select those more easily linked to existing cognitive structure), provide better pacing of instruction, or select a better instructional strategy (e.g., one-on-one tutorial assistance when the learners' cognitive development is highly variable).

As will be described later, it is proposed that OFMTutor employ the coaching method of tutoring, with intention-based diagnosis, a modified version of Lesgold's architecture, and an emphasis on relevant, meaningful concepts as argued by Ausubel and Novak.

Interface Design

The interface design is an important part of an intelligent tutoring system, for the student directly experiences interaction with the interface. Two basic routes have been taken in the design of the interface: one concerned with graphical, iconic representations; the other with (textual) dialogue management. Often, graphical representations are used in conjunction with guided-discovery learning or coached activities (e.g., STEAMER, IMTS, RBT), and dialogue systems employ a mixed-initiative dialogue style of interaction (e.g., GUIDON).

Graphical Interfaces. In the domain of complex, dynamic systems, dynamic graphical representations of the system's structure and function are useful. STEAMER's developers felt that a graphical interface to a simulation would be valuable in that it would allow one to view and manipulate the system at a number of different hierarchical levels. The graphical interface is also meant to provide a conceptually faithful representation of the system, in order to foster the development of accurate mental models of the system (Hollan et al., 1987).

The IMTS system provides a number of software tools that enable the development of simulation-based technical training (Towne et al., 1988). IMTS supports both physical and functional views of device components, in order to "promote the student's ability to find, recognize, and manipulate physical elements

in the real system, while maintaining a conception of the functional relationships that cannot be seen directly in the real system" (Towne et al., 1988, p. 18).

RBT "provides tools for reasoning" about the operation of a kraft recovery boiler. These tools include graphs that depict the relationship between various process parameters over time, meters that show the system state at higher levels of abstraction (e.g., safety, efficiency, and reliability), and interactive tutorial dialogues. The system also uses animated graphics to represent a mathematically and physically accurate simulation of the boiler.

AHAB works with the PEQUOD simulation of a marine steam powerplant. PEQUOD uses a qualitative approximation methodology (Govindaraj, 1987) that represents the system at various hierarchical levels. System states are calculated quantitatively but gauge readings, etc. are represented qualitatively. The student can inspect schematics of subsystems that provide information on causal flow and system state. The student also interacts with menus and windows to determine feasible tests, make diagnoses, and gain performance feedback.

GUIDON-WATCH is a graphic interface to NEOMYCIN, a medical consultation system. GUIDON-WATCH allows the student to browse through the database and view reasoning processes for diagnosis (Richer and Clancey, 1985). A number of windows provide information on current hypotheses, causal relations among symptoms, a diagnostic task tree, current evidence, and positive findings.

The importance of a graphical interface is supported by research in human-machine interaction (cf Norman and Draper, 1986). Some forms of interaction are facilitated with a conceptually natural and simple interface composed of iconic representations of objects rather than text. For training and tutoring in domains associated with complex dynamic systems, graphical interfaces allow the student to conceptualize system structure and function in a natural manner and to concentrate on learning rather than the interaction itself.

Toward a Theory of Discourse. When tutorial interactions are necessary, how should the tutor intervene? How does the tutor know when to intervene, what to say, and how to say it? The answer to these questions involves a consideration of discourse, which is intimately tied to the chosen pedagogical strategy.

Burton and Brown (1982) provide specific principles of interaction for a tutorial coach; these are summarized in Table 3.

Insert Table 3

about here

Such rules of thumb are useful guidelines but do not provide any principled theory of discourse conventions in tutoring. Other researchers have focused on how human tutors interact with students (Woolf, 1987; Fox, 1987a and 1987b). Fox (1987a) argues that the tutorial interaction itself (e.g., the way the student responds to questions and the timing of the response) provides diagnostic information and advocates that at least timing information can be utilized by an automated tutor. Thus, a lengthy pause between a question posed by the tutor and the student's response may indicate that the student encounted difficulty in arriving at the answer. Fox also stresses that "repair is an essential factor in natural language interface design" (Fox, 1987b, p. i). In an analysis of face-to-face conversational turn-taking between a human tutor and student, Fox notes that such turn-taking is very flexible, not primarily controlled by the tutor, and offers a fundamental mechanism for repair that is missing from the same type of interaction over a teletype machine. What is lost from the terminal-to-terminal interface is the "opportunity for the hearer to indicate understanding or lack of understanding at the end of every unit; and the opportunity for the speaker to initiate correction on his/her own turn after it was sent" (Fox, 1987b, p. 5). Fox further notes that "certain kinds of interruption are essential for maintaining mutual comprehension" and thus vital for repair (Fox, 1987b, p. 8). Fox's suggestions for dialogue management for an intelligent tutoring system are given in Table 4.

Insert Table 4

about here

Clancey (1982) argues that a case method tutor needs knowledge of dialogue patterns, domain knowledge, and the communication situation in order to carry on a dialogue with a student. Clancey's case method tutor is GUIDON, a tutor that works in conjunction with MYCIN, an expert system for diagnosis of infectious diseases. Augmented domain knowledge allows GUIDON to use metaknowledge to reason about MYCIN's rules and thus to use domain rules in a variety of ways. The communication situation is defined by the student model (an overlay model that represents what topics or rules the student has and has not yet demonstrated that he/she has learned) and the "focus record" that lists goals that the student has inquired about. GUIDON uses "discourse procedures" invoked by tutoring rules to direct and focus the case dialogue. The tutoring rules use knowledge of the communication situation as preconditions; thus, the communication situation drives the tutorial interaction. The tutoring rules are used to select discourse patterns (guide discussion of a domain rule, respond to a student hypothesis, and choose question formats), choose domain knowledge (provide orientation for choosing new goals, measure interestingness of domain rules), and maintain the communication model (update the student model).

Woolf (1987) has investigated the machine representation of discourse conventions in tutoring. She notes that discourse "is often described in qualitative terms along with the *effect* of the utterance on the listener" (Woolf, 1987, p. 250). Discourse analysis often seems guided by implicit rules that are based on the perception of qualitative states such as "what the student already knows". Woolf and her colleagues have begun developing a theory of discourse based on the recognition of qualitative states such as "student is confused" and "topic is generally known". As a first step towards this, Woolf defines conversational *move-classes* "as groups of utterances that have the same rhetorical effect" (Woolf, 1987, p. 253), such as question-topic and provide-example. The choice of a move-class indicates the speaker's intention in that the listener has particular expectations given a certain type of move. Woolf proposes tutoring maxims based on Paul Grice's maxims of conversation: quality (be truthful), quantity (be brief, yet complete), relation (be relevant) and manner (be clear and orderly) (Mura, 1983). Woolf's adaptions of Grice's maxims are shown in Table 5, and the tutorial maxims in relation to move-classes are shown in Table 6.

Insert Tables 5 and 6

about here

The move-classes are also defined with respect to their probable implications. These implications define implicit assumptions that are "taken for granted" in natural conversation. For instance, the choice of the question-topic class implies that the tutor "knows (or attempts to learn) the student's threshold of knowledge", "assumes the student can answer the question", and "thinks the topic is important or is learnable through the discourse" (Woolf, 1987, p. 256). Additional global implications are possible, based on "extended reasoning about sequences of move-classes" (Woolf, 1987, p. 256). Such global implications are assessments such as "student understands" and "topic was complete"; these are inherently uncertain and form the system's current best hypothesis of the student's state of knowledge or current topic. Such reasoning with uncertainty requires the ability to entertain multiple, possibly conflicting hypotheses and to resolve conflicts based on accumulating evidence.

Summary

This section has discussed previous research in intelligent tutoring systems, with emphasis on applications to domains associated with complex dynamic systems. Theories of education, learning, and discourse have also been considered. The following considerations are especially relevant to the design of OFMTutor: the simulation of a complex dynamic domain, a process model of expertise, intention-based diagnosis and student modeling, the coaching/guided discovery paradigm, the importance of concepts in a structured curriculum, a graphical dynamic interface to a complex dynamic simulation, and discourse models that allow for repair.

OFMTUTOR DESIGN

The design philosophy behind OFMTutor is similar to that of the RBT system (Woolf, 1986): the tutor is a "partner and co-solver of problems with the operator" (Woolf, 1986, p. 11). This has much in common with the approach of the "joint cognitive system" described by Woods (1986). The joint cognitive

system paradigm proposes that the computer provide support for the operator, giving context-sensitive reminders and suggestions, rather than dominate the interaction.

This approach has several implications for the design of an intelligent tutoring system for supervisory control of a complex dynamic system. First, the domain expertise and student models must be represented as *process models* (Wenger, 1987) that explicitly capture procedural knowledge and decision making behavior. Second, joint hypothesis formation and decision making imply the need for a pedagogical strategy of coaching in a guided discovery (reactive learning) environment. Finally, the interface design must support the explicit representation of domain expertise and an inspectable model of joint hypotheses.

In the next section I review the theoretical foundations of OFMTutor with a discussion of the operator function modeling methodology. Next, a blackboard architecture that implements the OFM for intent inferencing is discussed. Finally, the design of OFMTutor is presented. The OFMTutor architecture is based on that of OFMspert (Operator Function Model expert system) (Rubin et al., 1987).

The Operator Function Model

The OFM provides a flexible framework for representing operator functions in the control of a complex dynamic system. The OFM represents how an operator might organize and coordinate system control functions (Mitchell, 1987). Mathematically, the OFM is a hierarchic-heterarchic network of finite-state automata. Network nodes represent operator activities as operator functions, subfunctions, tasks, and actions. Operator functions are organized hierarchically as subfunctions, tasks, and actions. Each level in the network may be a heterarchy, i.e., a collection of activities that may be performed concurrently. Network arcs represent system triggering events or the results of operator actions that initiate or terminate operator activities. In this way, the OFM accounts for coordination of multiple activities and dynamic focus of attention. A generic example of an OFM is illustrated in Figure 2.

Insert Figure 2

about here

Historically, the OFM is related to the discrete control modeling methodology (Miller, 1985; Mitchell and Miller, 1986). The OFM is distinguished by its modeling of both manual and cognitive operator actions in the context of system triggering events. Manual actions are system reconfiguration commands. Cognitive actions include information gathering and decision making that are typically supported by information requests.

The OFM is a prescriptive model of human performance in supervisory control. Given system triggering events, it defines the functions, subfunctions, tasks, and actions on which the operator should focus. Used as an expert model, the OFM generates expectations of likely operator actions in the context of current system state. Used as a student model, the OFM defines likely operator functions, subfunctions, and tasks that can be inferred based on operator actions and system state. This ability to infer intentions dynamically is crucial for intention-based diagnosis, and also forms the core of the domain expertise and student models.

Successful application of the OFM to intent inferencing in a supervisory control task (Rubin et al., 1987; Jones, 1988; Jones et al., 1988) demonstrates that the OFM is a viable basis for determining operator (student) intentions in the context of current system state and past operator actions. This application utilized a knowledge-based problem solving methodology known as the blackboard model of problem solving (Nii, 1986). The next section describes this implementation.

The Blackboard Model of Problem Solving

The blackboard model of problem solving consists of three components: the blackboard, knowledge sources, and blackboard control (Nii, 1986). The blackboard is a data structure on which the current best hypothesis of the solution is maintained and modified. The hypothesis is represented hierarchically, at various levels of abstraction, and evolves incrementally over time as new data become available or old data become obsolete. Domain-specific knowledge is organized as a collection of independent knowledge sources. Knowledge sources are responsible for posting and interpreting information on the blackboard.

Blackboard control applies knowledge sources opportunistically; that is, in either a top-down or bottom-up manner, depending on what is more appropriate in the current context.

The blackboard model of problem solving is compatible with the knowledge represented in the OFM. Both models use a hierarchical representation. The blackboard knowledge sources provide a modularity that naturally represents much of the domain knowledge contained in the OFM arcs. The opportunistic control strategy offers the dynamic flexibility necessary for inferring intentions in real time.

Operator intentions may be represented as a hierarchy of goals, plans, tasks, and actions that correspond to the OFM's hierarchy of functions, subfunctions, tasks, and actions. Goals are currently instantiated functions, plans are currently instantiated subfunctions, and so on. The general mechanism for the blackboard approach to intent inferencing is as follows. Given an OFM, currently hypothesized goals, plans, and tasks (GPTs) or sometimes additional plans and tasks (PTs) for an existing goal are placed on the blackboard in response to system triggering events. The blackboard incorporates operator actions into the representation with opportunistic reasoning. Thus, actions can be immediately interpreted as supporting one or more current goals, plans, and tasks; and goals, plans, and tasks can be inferred on the basis of operator actions. In general, actions are interpreted with a strategy of maximal connectivity; actions that can support more than one current task are interpreted as supporting all such tasks. Figure 3 shows the proposed blackboard model of intentions.

Insert Figure 3

about here

Other Modeling and Pedagogical Considerations

The OFM is not enough to define all the knowledge needed for tutoring. The OFM does not explicitly represent the system to be controlled; it specifies operator functions within that system. This level of explanation is reasonable for well-trained operators but is insufficient for tutoring purposes. Like

GUIDON, OFMTutor will have to be augmented with supporting domain knowledge. Based on Novak's theory of education, such supporting knowledge should be in the form of concepts whose subsuming relationships should be clearly explicated. The relevant concepts associated with a complex dynamic system include the system's purpose, function, and structure. In particular, these concepts can be described in accordance with Rasmussen's (1986) abstraction hierarchy. Thus, both knowledge of the system (the abstraction hierarchy) and knowledge of operator functions (the OFM) correspond to a hierarchical arrangement that suggests the course of "top down" instruction proposed by Novak. Furthermore, since knowledge of the system is, in Gagne's terms, an essential prerequisite for knowledge of how to control the system, system concepts should be taught before operator function concepts. Figure 4 illustrates the abstraction hierarchy.

Insert Figure 4

about here

Furthermore, the OFM does not represent errorful behavior. Thus, the inclusion of "buggy" GPT's is necessary as a "limited bug model" similar to AHAB's. In this way, broad classes of misconceptions can be diagnosed and remedied appropriately. In order to build representations of misconceptions, a thorough cognitive task analysis of novice users is necessary. By careful observation of subjects interacting with the system, as well as protocol and off-line analyses, one may be able to characterize misconceptions and their manifestations in action patterns. Then the intelligent tutoring system can be given knowledge of particular action sequences and probable underlying misconceptions associated with them.

The Representation of the Expert and Student

The combination of the OFM and blackboard model of problem solving define a process model (Wenger, 1987) that can be used to represent expertise and student knowledge. The "expert's" goals, plans, and tasks can be represented on one blackboard, and the student's inferred intentions can be represented

similarly on a separate blackboard. Differential modeling can easily assess the difference between the two blackboards. If the student model is missing intentions that the expert has, a reminder or hint can be employed to remedy this error of omission. If the student model includes some intentions not modeled on the expert blackboard, this may signal a misconception to be corrected with the proper intervention.

The expert's goals, plans, and tasks are inferred on the basis of current system state. Thus, intentions are derived from the (normative) operator function model for GT-MSOCC. In contrast to the expert's model-derived intentions, the student's goals, plans, and tasks are inferred based on student actions. The student's intentions may also be modeled with the "buggy" GPT's described previously. Both representations exist in a component of OFMTutor known in general as the Blackboard. The Blackboard actually consists of an expert and a student blackboard, where hypothesized intentions are posted and compared.

Supporting Domain Knowledge

Like AHAB and RBT, OFMTutor will require that students interact with a simulation of a complex dynamic system. Unlike AHAB and RBT, OFMTutor is designed to exist on a computer separate from the simulation itself. This distributed environment supports the clean separation of domain dynamics and tutoring knowledge and strategy. Furthermore, the portability of the OFMTutor architecture is enhanced in that it can, in theory, be placed "on top of" any complex dynamic system simulation for which an OFM can be constructed. Philosophically, it can be argued that intelligent tutoring is but one point on a continuum of intelligent aiding in general, and since our design philosophy of the operator's associate dictates such a distributed environment (Rubin et al., 1987; Jones et al., 1988), it is natural that OFMTutor also requires such an architecture.

The requirement of a distributed environment means that OFMTutor must have an explicit representation of current system state. This representation may be termed the Current Problem Space (CPS). The CPS is needed to give context for the modeling of intentions and for pedagogical interventions. Also, a representation of the current displays to the student is necessary in order to infer what information is currently available to the student. This representation is called the Workstation description. The Workstation maps the names of display pages to their semantic information content.

OFMTutor must also support knowledge of domain concepts. Beyond the procedural knowledge defined in the OFM, an operator must have some grasp of the underlying principles of the system's structure and function. This includes knowledge of the system's purpose, abstract function (e.g., flow of data), function, and physical form. Such knowledge can be captured within the framework of Rasmussen's (1986) abstraction hierarchy. The abstraction hierarchy forms part of the additional knowledge needed for tutoring. This additional knowledge also includes pedagogical knowledge and strategies and the limited bug model goals, plans, and tasks. These enhancements to the knowledge in the OFM define the Enhanced Normative Model (ENM).

Finally, the architecture requires a Communication Interface that communicates with the simulation of the system and with the tutorial interface to the student. Information about system events and student actions is sent from the simulation and the tutorial interface to OFMTutor. A scheduler, called the High Level Controller, manages the various events within OFMTutor. The complete architecture is shown in Figure 5.

Insert Figure 5

about here

Pedagogical Strategy

OFMTutor's pedagogical strategy is guided both by Ausubel's and Gagne's ideas; i.e., concepts and essential prerequisites. Since a concept of the system is an essential prerequisite for learning operator functions to control that system, the instructional process is divided into two broad sections: one on concepts, and one on control. The "conceptual" curriculum is presented top down and organized with respect to Rasmussen's abstraction hierarchy. The student's knowledge is assessed via on-line quizzes and questionnaires.

When the student has mastered system concepts, control functions will be taught. Here, the student will learn procedures for controlling the system. This phase of instruction is organized top down with respect to the operator function model. Also, while the first phase of instruction will take place solely in the context of the tutor, the second "operational" phase is taught in the context of the system simulation. In this phase, the tutorial strategy is one of guided discovery, in which the student explores the system and is given non-intrusive assistance by a "coach." The student has relatively greater control over the interaction here than in the first phase.

Diagnosis occurs by differential modeling of the two blackboards as described earlier. However, the tutor does not give explicit advice or warnings immediately when a discrepancy is noticed. It is important to allow an opportunity for the student to "get back on track" independently.

Tutorial Interface

OFMTutor's interface is a multiwindow environment that allows the student a fair degree of control over the interaction. The student may collapse or open any windows desired at any point in time. In the first phase of instruction, the tutorial interface consists of animated views of the system, text that describes concepts, and multiple-choice and fill-in-the-blank quizzes.

During the second phase of instruction, the interface supports a graphical representation of joint intentions; that is, a representation that explicitly shows the comparison of the expert and student model blackboards. Supporting text windows include a list of expected commands, a list of all commands (so that failures to remember syntax are minimized), and dynamically generated advice and suggestions.

OFMTUTOR EVALUATION

Park et al. (1987) assert that one methodological difference between computer-based instruction and intelligent tutoring systems is that the former pay attention to evaluation procedures, and the latter do not. In this section we examine evaluation procedures from both instructional and industrial-organizational points of view, with the aim of deriving useful evaluation procedures for an intelligent tutoring system such as OFMTutor.

Traditional Instructional Evaluation

As discussed previously, formative and summative evaluation comprise part of the instructional design process. Dick (1977a) describes formative evaluation:

> Formative evaluation may be ... defined as a process of systematically trying out instructional materials with learners in order to gather information and data which will be used to revise the materials. The implication of the term 'formative' is that the evaluation process occurs *while the materials are still being developed.*
>
> ... The sole purpose of formative evaluation is to provide the instructional designer with as much information as possible to revise and strengthen the product which is under development. (pp. 311-312)

The first suggested phase of formative evaluation is one-to-one evaluation. A small representative sample of the target student population (preferably three students -- one of below average ability, one of average ability, and one of above average ability) works through a draft of the instructional materials (including any tests) with the designer. Students give feedback both by their performance and comments to the designer. The designer also asks specific questions in order to discover particular strengths and weaknesses of the materials. This phase is much like a "pilot study" used in traditional experimental situations. The one-to-one phase also includes a review of the materials by a domain expert in order to insure the accuracy of the content. The output from this phase is a set of comments and observations on any difficulties encountered in the use of the materials. The instructional materials are revised with respect to this output, and the revised materials are used in the second phase of formative evaluation.

The second phase of formative evaluation is a small-group evaluation. The purposes of this phase are to evaluate the effectiveness of the first phase's revisions, identify any remaining difficulties, and begin the determination of the feasibility of administering the materials in the field. Dick (1977a) recommends a representative sample of between eight and 24 students for the small-group evaluation. The students take tests and study the materials in a manner similar to that to be used in the field. Questionnaires are also

given, and any helpful comments are solicited as well. The output from the second phase of formative evaluation includes comments, questionnaire data, test scores, and learning times.

The third phase of formative evaluation is field trial evaluation. The major purpose of this phase is to "determine the administrative feasibility of using the instructional materials under normal classroom conditions" (Dick, 1977a, p. 316), as well as to ascertain the effectiveness of the previous revisions. The most critical component of this evaluation phase is that the materials are used in the environment for which they are ultimately intended. Dick (1977a) suggests that the sample size for field trial evaluation should be at least 30 students.

Dick (1977a) notes that no guidelines exist for when to terminate formative evaluation; the decision to terminate formative evaluation "is based almost entirely upon the specific circumstances surrounding the development project" (p. 330). Typically, time requirements or funding are important factors. Designers may also set statistical criteria to be met. For example, the military established an "80/80" criterion for formative evaluation termination. This meant that when 80% of the students achieved 80% of the proposed objectives, the formative evaluation process was judged complete.

In practice, formative evaluation is usually the last stage of the design process. However, very often we are interested in comparing alternative instructional products. Summative evaluation is a process meant to provide data needed this comparison process.

Dick (1977b) reviews several models of summative evaluation. One model, proposed by Gagne and Briggs, has four components: support, aptitude, process, and outcome evaluation. Support evaluation examines the instructional materials, the climate of the teaching environment, parental and peer attitudes, and other factors that may affect learning. Aptitude evaluation, or the evaluation of learner aptitude, is important in that aptitude is significantly correlated with eventual learning outcomes. Furthermore, knowledge of the learner population helps define the generalizability of the summative evalution studies. Process evaluation refers to the documentation of instructional materials and procedures and the formative evaluation process. Outcome evaluation refers to the evaluation and reporting of instructional objectives, criteria for success, and results of product successes and failures.

The basic process of summative evaluation, as described by Dick (1977b), consists of five steps: identification of intended outcomes, identification of the target student population and experimental design, development of the evaluation instruments, documentation of the instructional process, and preparation of the final report. First, one identifies the relevant instructional goals and objectives. Next, one identifies a representative sample of students. Depending on the availability and size of the sample, one has several alternative experimental situations: comparisons between two experimental groups, comparison of one group to norms for a standardized exam, or, in the worst case, to establish criteria for success and administer the appropriate tests to the sample after the completion of their studies. The summative evaluation is also responsible for developing evaluation instruments for data collection. These include means for assessing learning outcomes (e.g., objectives-referenced tests), attitudes about the instructional content and form (e.g., questionnaires), and cost. The instructional process itself must be documented thoroughly, and a clear final report prepared.

Training Program Evaluation

From an industrial-organizational psychology perspective, Landy and Trumbo (1980) review several different approaches to the evalution of training programs. One approach distinguishes between *internal criteria* (i.e., performance in the training situation) and *external criteria* (i.e., performance on the job). Landy and Trumbo give examples of each:

> Internal criteria include objective exams, questionnaires reflecting attitude changes by the trainees, and the opinions of trainees, trainers, or others as to the effectiveness of the program. Comparison of training methods or programs may use the number of hours of training required to reach a common training performance level as an internal measure. A similar criterion -- hours (days or weeks) to reach standard production on the job *after training* -- would be an external criterion. External criteria include measures of quantity or quality of production, time to reach production levels, accident records (for safety training) and other indications of job behavior or training results. (p. 296)

Another approach distinguishes four levels of criteria: reaction, learning, behavioral, and results criteria. The former two correspond to internal criteria, and the latter two correspond to external criteria. Reaction criteria are concerned with the trainees' opinions of the program and typically consist of one or more questionnaires. Learning criteria include final exams, performance tests, and other measures of how much was learned. Learning criteria should reflect the objectives of the training program. Behavioral criteria include performance measures on the job. Results criteria involves the assessment of the utility of training with respect to organizational objectives (e.g., percent increase in job proficiency, percent decrease in accidents or turnover).

Landy and Trumbo also discuss various experimental designs used in the assessment of training programs. Solomon's proposed four group design is one of the most famous and most complete (see Figure 6). The group of prime interest is the experimental group, which is given a pretest, undergoes training, and is then given a posttest. The Control 1 group is given "sham training" during the training period to control for the "Hawthorne effect" (i.e., the effect of *perceived* experimental manipulations on performance; the famed Hawthorne studies showed that workers' productivity increased when they *believed* that working conditions were altered for the better, when in fact the conditions were exactly the same! This phenomena is also known as the "placebo effect"). The Control 2 group is needed to control for the effect of time (i.e., the effect of waiting the duration of the training period before taking the posttest). Finally, the Control 3 group is needed to control for the effect of giving a pretest.

Insert Figure 6

about here

Proposed OFMTutor Evaluation

Formative evaluation is a very important part of any educational project. Because OFMTutor is not intended for simultaneous use by a classroom of students, it is proposed that the one-on-one and small

group evaluations are sufficient for this purpose. The focus of summative evaluation will be on students trained with OFMTutor as compared to those who read over a training manual and operated the system untutored.

Academic research projects typically do not have the opportunity to work with real operators of complex dynamic systems. The OFMTutor evaluation will focus on internal criteria; specifically, reaction (questionnaire) and behavioral (performance) criteria.

REFERENCES

Briggs, L. J. (Ed). (1977a). *Instructional design: Principles and applications.* Englewood Cliffs, NJ: Educational Technology Publications.

Briggs, L. J. (1977b). The teacher as designer. In L. J. Briggs (Ed.), *Instructional design: Principles and applications,* 221-258. Englewood Cliffs, NJ: Educational Technology Publications.

Brown, J. S., Burton, R. R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 227-282. Orlando, FL: Academic Press.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 157-183. Orlando, FL: Academic Press.

Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 79-98. Orlando, FL: Academic Press.

Chambers, A. B. and Nagel, D. C. (1985). Pilots of the future: Human or computer? *Communications of the ACM, 28,* 11, 1187-1199.

Clancey, W. J. (1982). Tutoring rules for guiding a case method dialogue. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 201-226. Orlando, FL: Academic Press.

Clancey, W. J. (1987). *Knowledge-based tutoring: The GUIDON program.* Cambridge, MA: MIT Press.

Dick, W. (1977a). Formative evaluation. In L. J. Briggs (Ed.), *Instructional design: Principles and applications,* 311-333. Englewood Cliffs, NJ: Educational Technology Publications.

Dick, W. (1977b). Summative evaluation. In L. J. Briggs (Ed.), *Instructional design: Principles and applications,* 337-348. Englewood Cliffs, NJ: Educational Technology Publications.

Fath, J. L. (1987). An architecture for adaptive computer-assisted instruction programs for complex dynamic systems. PhD dissertation, Report 87-3, Center for Man-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Fath, J. L., Mitchell, C. M., and Govindaraj, T. (1988). An ICAI architecture for troubleshooting in complex, dynamic systems. Manuscript in preparation.

Fox, B. A. (1987a). Interaction as a diagnostic resource in tutoring. Dept. of Linguistics and Institute of Cognitive Science, University of Colorado, Boulder, CO.

Fox, B. A. (1987a). Repair as factor in interface design. Dept. of Linguistics and Institute of Cognitive Science, University of Colorado, Boulder, CO.

Genesereth, M. R. (1982). The role of plans in intelligent teaching systems. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 137-156. Orlando, FL: Academic Press.

Goldstein, I. P. (1982). The genetic graph: A representation for the evolution of procedural knowledge. In D. Sleeman and J. S. Brown (Eds.), *Intelligent tutoring systems,* 51-77. Orlando, FL: Academic Press.

Govindaraj, T. (1987). Qualitative approximation methodology for modeling and simulation of large dynamic systems: Application to a marine powerplant. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17*, No. 6, 937-955.

Hollan, J. D., Hutchins, E. L., and Weitzman, L. M. (1987). STEAMER: An interactive, inspectable, simulation-based training system. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods*, 113-134. Reading, MA: Addison-Wesley.

Johnson, W. L. and Soloway, E. (1987). PROUST: An automatic debugger for Pascal programs. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods*, 47-68. Reading, MA: Addison-Wesley.

Jones, P. M. (1988). Constructing and validating a model-based operator's associate for supervisory control. M. S. Thesis, Report 88-1, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Jones, P. M., Mitchell, C. M. and Rubin, K. S. (1988). Intent inferencing with a model-based operator's associate. Report 88-2, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. Also to appear in *Proceedings of the Sixth Symposium on Empirical Foundations of Information and Software Sciences*, October 1988, Atlanta, GA.

Kibler, R. J. and Bassett, R. E. (1977). Writing performance objectives. In L. J. Briggs (Ed.), *Instructional design: Principles and applications*, 49-95. Englewood Cliffs, NJ: Educational Technology Publications.

Landy, F. J. and Trumbo, D. A. (1980). *Psychology of work behavior*. Homewood, IL: Dorsey.

Lesgold, A. (1988). Toward a theory of curriculum for use in designing intelligent instructional systems. In H. Mandl and A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems*, 114-137. New York: Springer-Verlag.

Miller, R. A. (1985). A systems approach to modeling discrete control performance. In W. B. Rouse, (Ed.), *Advances in Man-Machine Systems Research*, Vol. 2, 177-248. New York: JAI Press.

Mitchell, C. M. (1987). GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17*, July/August, 553-572.

Mitchell, C. M. and Miller, R. A. (1986). A discrete control model of operator function: A methodology for information display design. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-16*, 343-357.

Mura, S. S. (1983). Licensing violations: Legitimate violations of Grice's conversational principle. In R. T. Craig and K. Tracy (Eds.), *Conversational coherence: Form, structure, and strategy*, 101-115. Beverly Hills, CA: Sage Publications.

Nii, H. P., Feigenbaum, E. A., Anton, J. J., and Rockmore, A. J. (1982). Signal-to-symbol transformation: HASP/SIAP case study. Heuristic Programming Project, Report No. HPP-82-6, Heuristic Programming Project, Stanford University, Stanford, CA.

Nii, H. P. (1986). Blackboard systems. *AI Magazine, 7-2, 7-3*.

Norman, D. A. and Draper, S. W. (Eds.) (1986). *User centered system design: New perspectives on human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Novak, J. D. (1986). *A theory of education.* Ithaca, NY: Cornell University Press.

Park, O., Perez, R. S., and Seidel, R. J. (1987). Intelligent CAI: Old wine in new bottles, or a new vintage? In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods,* 11-45. Reading, MA: Addison-Wesley.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering.* New York: North-Holland.

Richer, M. H. and Clancey, W. J. (1985). GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. *IEEE Transactions on Computer Graphics and Applications,* November, 51-64.

Rubin, K. S., Jones, P. M. and Mitchell, C. M. (1987). OFMspert: Application of a blackboard architecture to infer operator intentions in real time decision making. Report No. 87-6, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. Also *IEEE Transactions on Systems, Man, and Cybernetics,* to appear.

Schank, R. C. and Abelson, R. P. (1977). *Scripts plans goals and understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Sheridan, T. B. and Johannsen, G. (Eds.) (1976). *Monitoring behavior and supervisory control.* New York: Plenum.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., and Wogulis, J. (1988). ONR final report: Intelligent maintenance training technology. Technical Report No. 110, Behavioral Technology Laboratories, Department of Psychology, University of Southern California, Redondo Beach, CA.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann.

Wickens, C. D. (1984). *Engineering psychology and human performance.* Columbus, OH: Charles Merrill.

Woods, D. D., (1986). Cognitive technologies: The design of joint human-machine cognitive systems. *The AI Magazine,* 86-92.

Woolf, B. P. (1986). Teaching a complex industrial process. COINS Technical Report 86-24, Computer and Information Science, University of Massachusetts, Amherst, MA.

Woolf, B. P. (1987). Theoretical frontiers in building a machine tutor. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods,* 229-267. Reading, MA: Addison-Wesley.

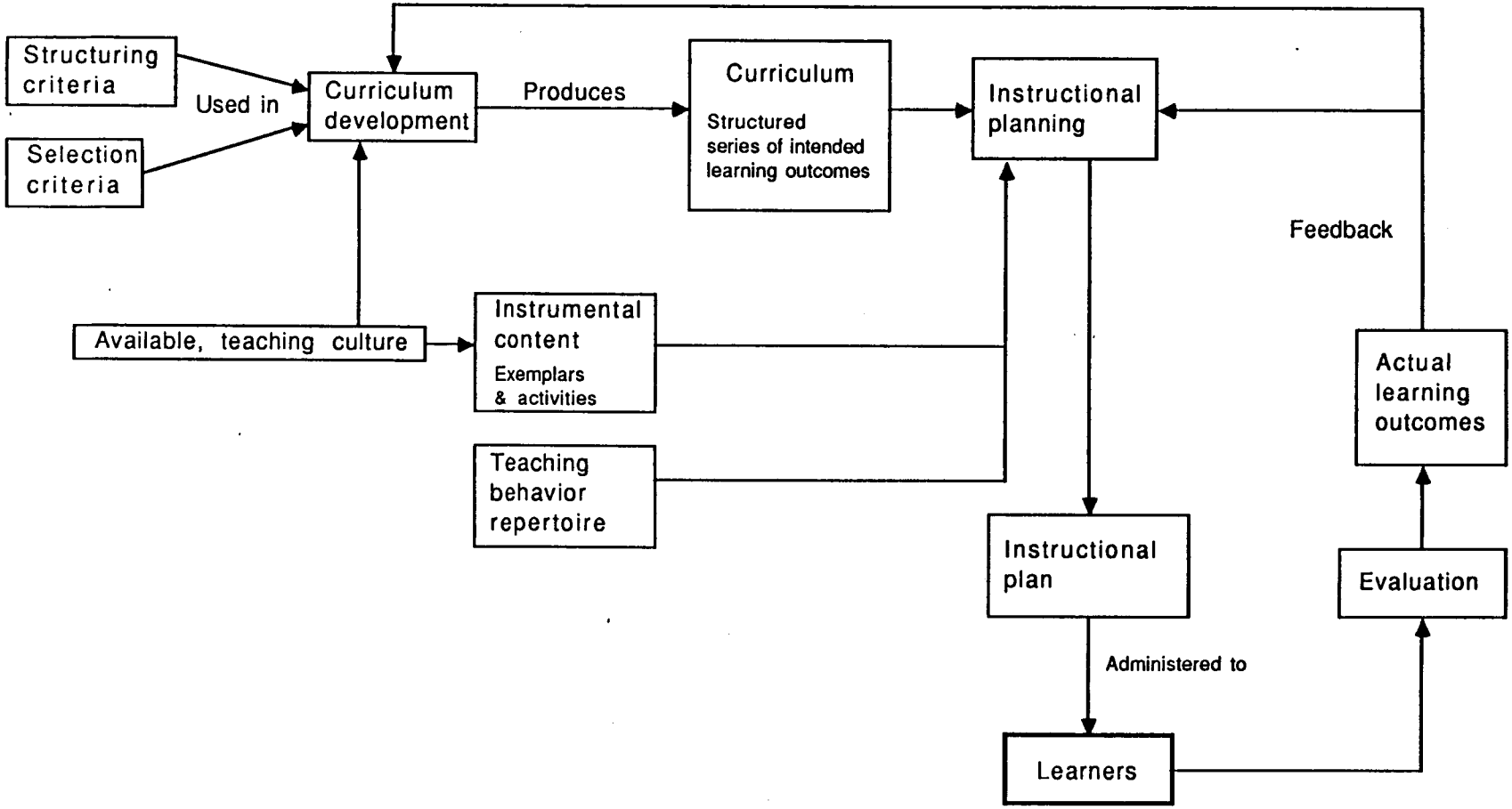


Figure 1. A simplified version of Johnson's model of curriculum. Adapted from Novak, 1986, p. 132.

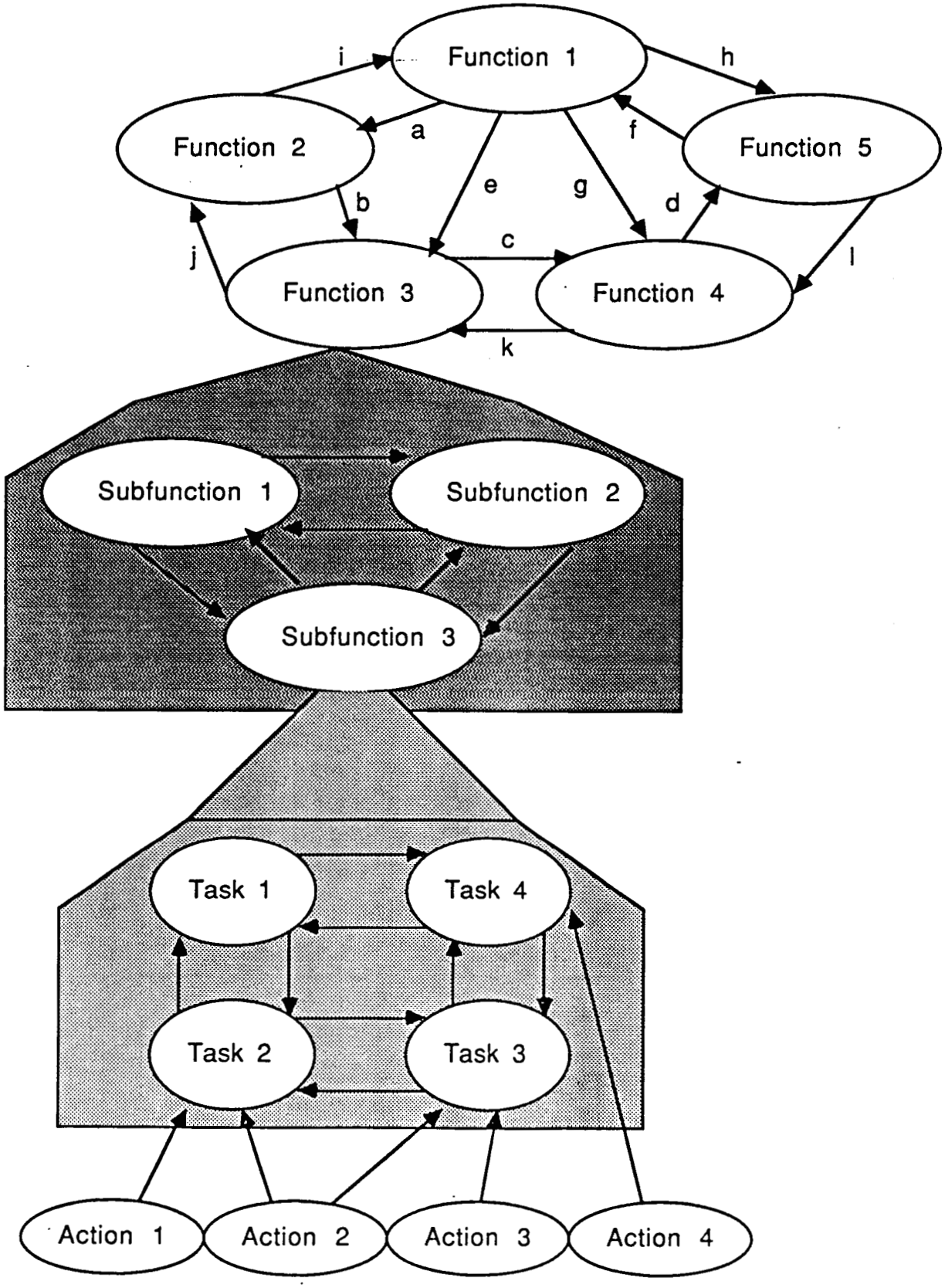


Figure 2. A Generic Operator Function Model

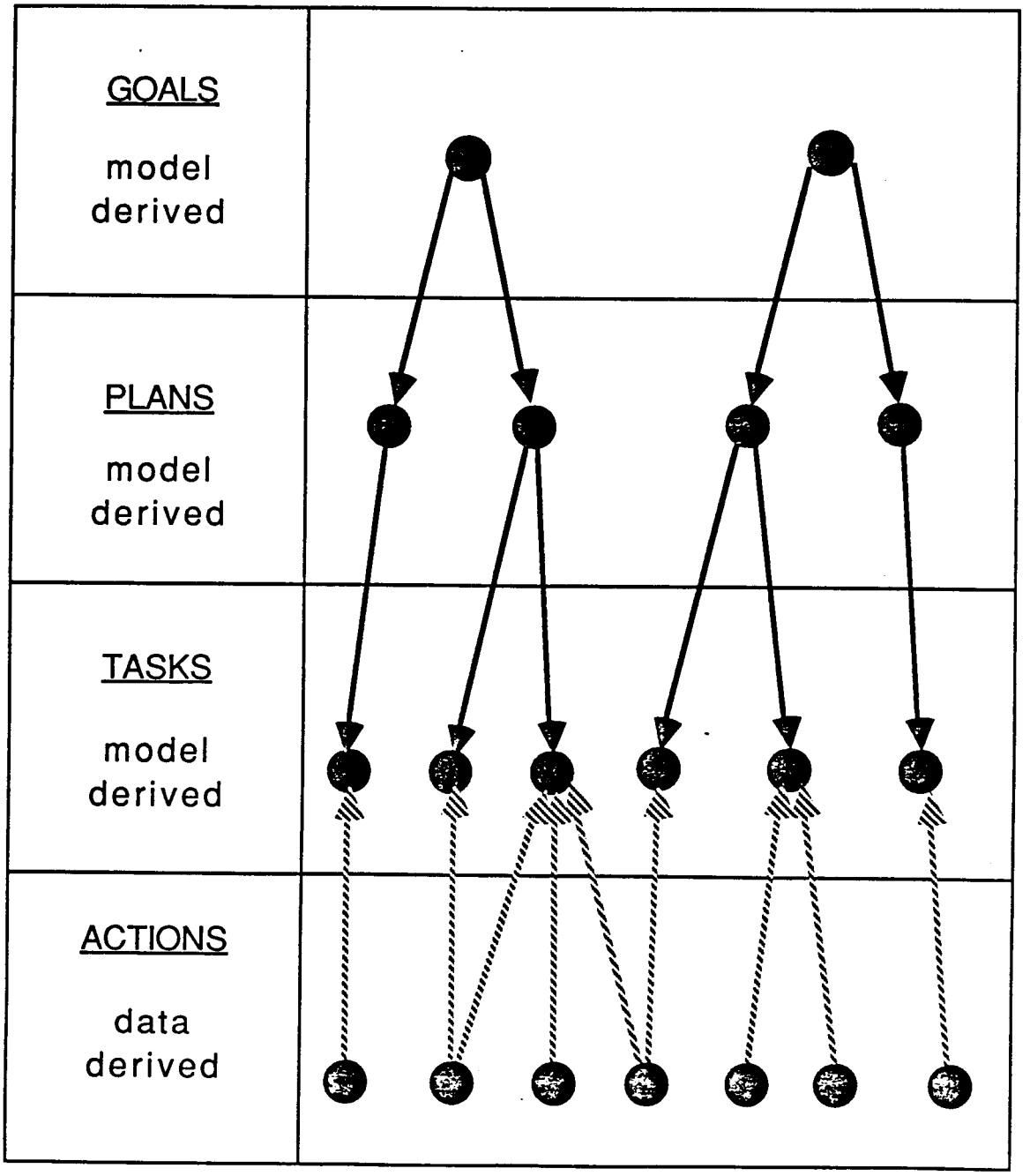


Figure 3. The Blackboard Intent Inferencing Structure

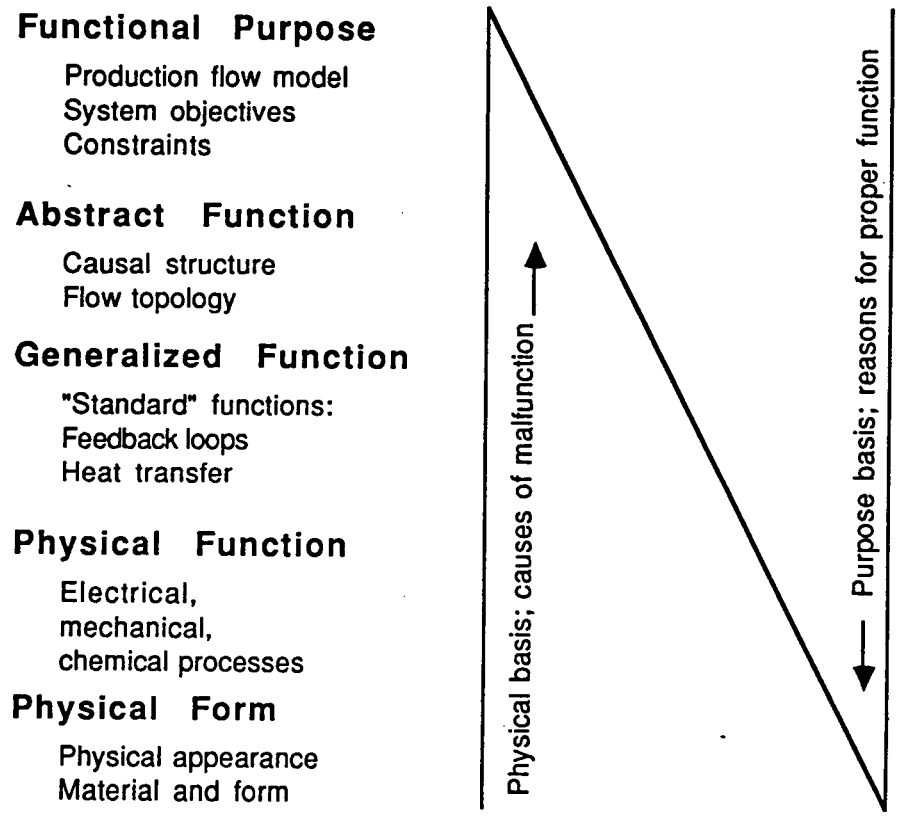


Figure 4. The abstraction hierarchy. Adapted from Rasmussen, 1986, p. 15.

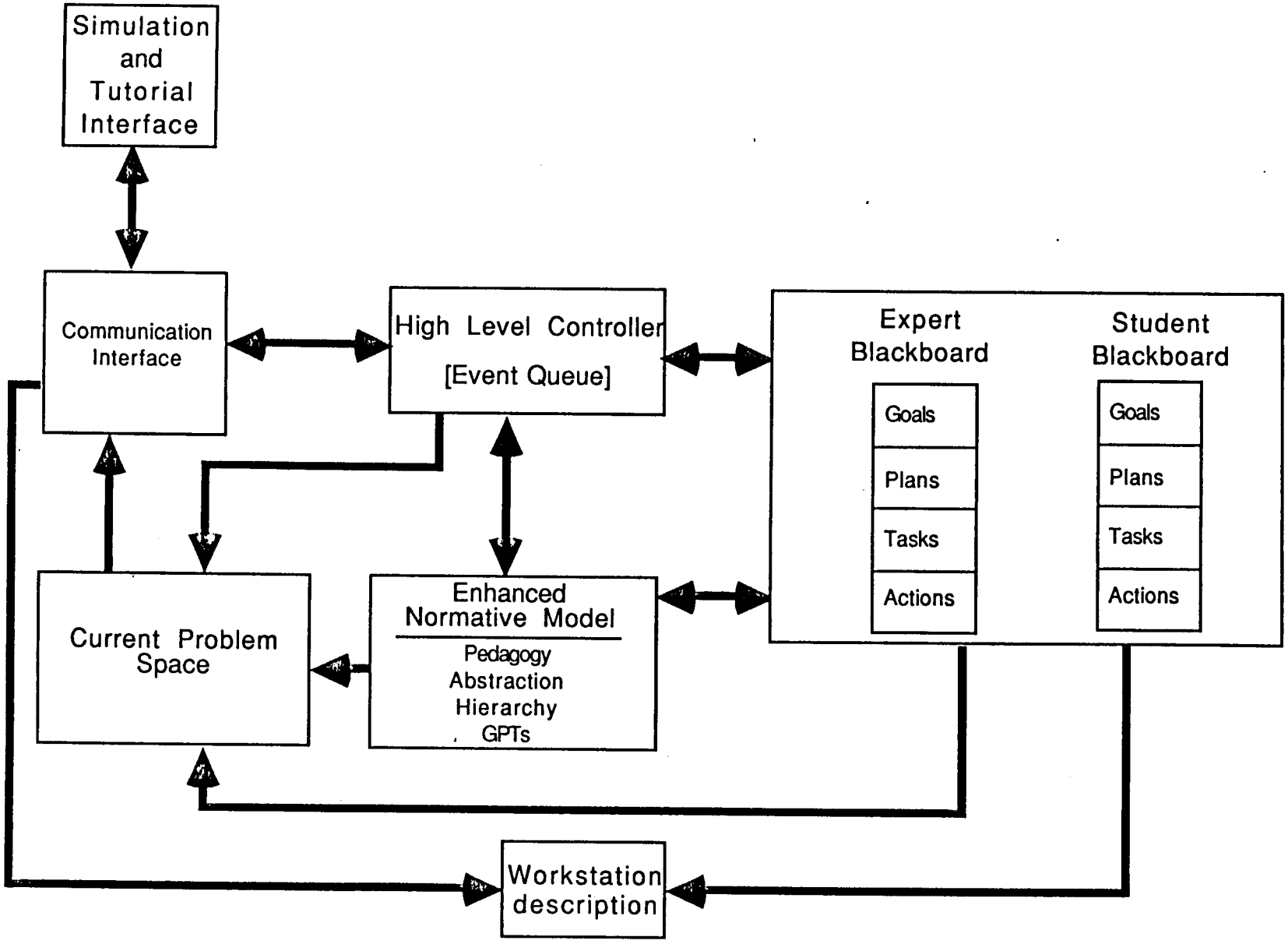


Figure 5. OFMTutor Architecture

Group	Before	Training Period	After
Experimental	Test	Train	Test
Control 1	Test	Placebo activity	Test
Control 2	Test	No train	Test
Control 3	No test	No train	Test

Figure 6. Four-group experimental design proposed by Solomon. From Landy and Trumbo, 1980, p. 300.

Table 1. Taxonomy of capabilities and actions.
Adapted from Briggs, 1977, p. 69.

Capability	Action	Example
Intellectual Skill		
Discriminates	Discriminates	Distinguishes sounds
Concrete Concept	Identifies	Names computer components
Defined Concept	Classifies	Classifies using definition
Rule	Demonstrates	Solves linear equations
Problem Solving	Generates	Synthesizes rules to generate solution
Cognitive Strategy	Originates	Applies model of diffusion to originate solution to reduction of air pollution
Information	States	States current events
Motor Skill	Executes	Drives a car
Attitude	Chooses	Chooses to share toys

Table 2. Suggested guidance for particular learning outcomes. Adapted from Gagne, 1977, p. 211.

Learning Outcome	Suggested Guidance
Discrimination	Point out distinctive features of objects to be discriminated
Concrete Concepts	Gives cues to identifying attributes
Defined Concepts	Present component concepts in proper sequence
Rules	Show how component concepts make up the rule
Problem Solving	Provide minimum cues needed to select and apply rules
Cognitive Strategies	Provide only indirect cues
Names and Labels	Provide coes or memory bridges
Facts	Provide meaningful context
Organized Knowledge	Provide prompting in the context of the organizational framework
Motor Skills	Stimulate recall of sequence of acts; provide practice with feedback
Attitudes	Show human model behavior and how reinforced

Table 3. Some principles of interaction for a coach. Adapted from Burton and Brown, 1982.

Principle 1: Before giving advice, be sure that the issue is one in which the student is weak.

Principle 2: When illustrating an issue, use an example (alternative action) that is dramatically superior to the action taken by the student.

Principle 3: After giving the student advice, allow an opportunity for redoing the action.

Principle 4: If a student is close to making a serious mistake, interrupt and tutor only with advice that will prevent that mistake.

Principle 5: Do not tutor on two consecutive actions.

Principle 6: Allow the student to explore before tutoring.

Principle 7: Praise the student when appropriate.

...

Principle 10: If a student asks for help, provide several levels of hints.

...

Principle 12: Be forgiving of possibly careless errors.

Table 4. Fox's suggestions for dialogue management. Adapted from Fox, 1987b, p. 12.

Turn-taking should not be an on-off option. The interface must allow for each party to participate as they see fit.
It is especially important that during a turn, the other party have the ability to show understanding, initiate repair, etc., at the end of every conversational unit.
The turn-taking mechanism must provide flexibility in turn length.
Correction of the student, or initiation of such correction, should be withheld until the student has had an opportunity to self-correct, or initiate self-correction.

Table 5. Woolf's adaptions of Gricean maxims for discourse. Adapted from Woolf, 1987, p. 254.

Quality	Be committed and interested in student's knowledge. Be supportive and cooperative. Do not take the role of "antagonist"
Quantity	Be specific and concise. Use a minimum of attributes to describe a known concept.
Relation	Be relevant. Find the student's threshold of knowledge. Bring up new topics and viewpoints as appropriate.
Manner	Be in control. Allow both the student and the context to determine the topic.

Table 6. Tutoring maxims supported by conversational move-classes. From Woolf, 1987, p. 255.

Be Cooperative	
Work with student	Explain, summarize, review or repeat, and clearly terminate topics. Release control of dialogue.
Be Committed	
Show interest	Acknowledge answer. Explain topics.
Support student	Outline, introduce topics.
Be Relevant	
Find student's threshold	Question student. Evaluate student hypotheses. Propose and verify misconceptions.
Teach at threshold	Provide analogy examples. Summarize topic.
Be Organized	
Structure domain	Outline, introduce, terminate, review topics.
Complete information	Clearly terminate topics. Teach subtopics and attributes after topic. Teach subgoals after goal.
Be In Control	
Strictly guide discoure	Introduce, describe topic. Question student.

A SURVEY OF

ITELLIGENT TUTORING SYSTEMS:

IMPLICATIONS FOR

COMPLEX DYNAMIC SYSTEMS

BY

ROSE W. CHU

JANUARY 17, 1989

(DRAFT COPY)

Introduction

The purpose of this paper is to provide an overview of the research in the field of intelligent tutorial systems (ITS). More specifically, the various approaches in the design and implementation of ITS will be examined and discussed in the context of problem solving in an environment of a complex dynamic system (CDS). Although there have been several execellent sources of discussion on the work in ITS (Sleeman and Brown, 1982; Wenger, 1987; Psotka et. al., 1988), the motivation for the paper stems from the need to consolidate the findings in the research to a specific domain of interest. In the Center for Human-Machine Systems Research at the Georgia Institute of Technology, one of our interest and focus of research is the application of ITS to complex dynamic systems.

Several relevant topics will serve as the background to the actual study on the numerous ITS. First, issues pertaining to a CDS will be considered. Next, the nature of human problem solving will be discussed, especially in light of a CDS. Then, an overview of the architecture of an ITS will be provided as the basis for the in depth examination of various systems. Finally, the implications for the design and evaluation of an ITS will be discussed along with some concluding remarks and thoughts.

Complex Dynamic Systems

With the advancement of computer technology, the trend towards more complex systems has posed immediate challenges to the field of human-machine interactions due to the changing role of an operator in his work environment. Rasmussen (1986) has cautioned that automation made possible in these systems do not render the human obselete, rather, only the previous responsibility of the human operator in low level system controls have now been replaced. In fact, Wickens (1984) points out three objectives of automation. It allows the execution of functions in a system that an operator cannot perform due to inherent human limitations. Also, automation may take over functions that do not involve the best of human capabilities or are within human limitations but are too taxing. Instead of totally taking over, another objective of automation may be to provide assistance to the operator in achieving the above functions .

An operator's new role as a consequence of automation, has generally been discussed under the term supervisory control. According to Sheridan (1976) "the supervisory control paradigm applies to situations where a person allocates his attention among various graphical or alphanumeric displays and intermittently communicates new programs to a computer which itself is in continuous direct control of a physical process." An operator engaged in supervisory control (thus, he is the supervisory controller) must deal with multi-task, multi-goal and multi-person environments (Baron, 1984). The various activities of a supervisory controller have been characterized in different but consistent ways.

Sheridan (1984) considers the planning, teaching, monitoring, intervening and learning modes of a supervisory control task. Wickens (1984) discusses the control versus diagnostic nature of the operator's task. Baron (1984) catagorizes the activities into planning, monitoring, situation assessment, decision-making, control and communication. Salvendy (1984) breaks down the task into monitor, control, interpret, plan and diagnose. Yet another simple dichotomy of a supervisory control task is that of monitoring versus troubleshooting. Generally speaking, these activities focus on the cognitive behavior instead of the psychomotor performance of the operator.

These tasks imply requirements at a level not considered before (Rasmussen, 1986). For example, an operator must be trained differently in order to meet the demands of his new tasks. An operator must possess knowledge and understanding about the system at a sufficient depth in order to handle both normal and abnormal situations. Moreover, with automation comes a new set of problems (Wickens, 1984). An operator has to deal with an increased monitoring load in face of a more complex system that now have many additional interacting components. On the other hand, an operator may exhibit too much trust in the automated subsystems, resulting in a false sense of security that in turn affects his job performance. There is also the potential problem of "out-of-the-loop familiarity". This problem arises when an operator is taken out of the normal control-loop replaced by automation, and thus interacting less with the system and becoming less familiar with system states. Consequently, the operator may be less able to handle system trouble. Although automation elimates some low-level human error, it also introduces other high-level errors associated with an operator's job. Finally, many tasks that previously involve the cooperation of two human operators may now be replaced by a less personal operator-machine team.

How is a CDS distinguishable from other systems? Baron (1984) cited the following features for a system that require supervisory control:

- the system is very high-tech, large scale, expensive and risky in nature
- the system involves many complex and dynamic processes with many controllable outputs
- the system has many subsystems
- many but not all aspects of the system are automated
- manually controllable variables have slow response, in contrast to automatically controlled and rapid changing variables
- the demands on the system is driven by events
- there is a need to communicate among operators and with other system units
- an operator at times have to follow a predetermined set of instructions during some predictable situations.

A lot of work has been done to model the human supervisory controller (Sheridan, 1984; Baron, 1984; Rouse ? etc). In addition, Rasmussen (1986) has recently provided a valuable framework for understanding and designing supervisory control systems. The next

section discusses a framework for studying the human problem solving behavior in a CDS.

Problem Solving Strategies and Models

Human problem solving has been the subject for research in many aspects of human-machine systems. With respect to a CDS, the tasks of a supervisory controller concern that of solving problems in various situations. Much of the research in this area has focussed on the identification of the different strategies that an operator used in problem solving. Salvendy (1984) cited eleven strategies identified in the literature. A brief discussion of each method is given below.

** NEED TO FIND DEFINITIONS
Backward search (Simon and Simon, 1978) ...
means-end analysis ...
and hill-climbing...(Newell, Shaw and Simon, 1960),
scan-and-search (Simon and Newell, 1971),...
progressive deepening (DeGroot, 1965) ...
and symptomatic search (Rasmussen, 1981; Wortman, 1971) ...

Application of examples (Anderson, 1981) refers to our ability to solve a new problem by referring to an example of an already solved problem. Solving problem by analogy (Mayer, 1981; Gentner and Gentner, 1983; Rumelhart and Norman, 1981; Carroll et al, 1981) involves using solutions in a familiar domain to solve a problem in the new domain. There are some problems that are solved by mental simulation (Hollan et al, 1980). This means that we envision in our mind a scenario surrounding a fact or a problem which may or may not exist. When the problem solving situation is that of fault diagnosis, Rasmussen (1981) points out that an operator may use a strategy called topographic search. In this situation, the operator has a mental model of the normal functions of the system which is mapped against a problem to determine where a system function may have failed. Finally, Rasmussen (1981) also noted three general types of problem solving behavior: skill-, rule- and knowledge-based performance. Skill-based behavior are sensorimotor type performance that is very automatic. Rule-based behavior follows some prescribed procedure in solving a problem. For complex and/or unfamiliar problems, an operator has a goal in mind and plans his actions to achieve the goal based on his model of the environment surrounding the problem. This is knowledge-base behavior.

In the study on human problem solving in fault diagnosis tasks, several models were proposed (Rouse and Hunt, 1984). These models have both prescriptive and predictive value in an attempt to understand the nature of human problem solving. First, models of complexity suggest that measures of complexity should take into account both the problem and problem solver. Second, the theory of fuzzy sets may be used to model the

decision-making component in a problem which involves more than yes/no answers. Rouse and Hunt also proposed a rule-based model where an operator is modelled to solve a problem based on a set of situation-action heuristics. Next, a fuzzy rule-based model accounts for problem solving with highly context-sensitive rules. Lastly, a overall model considers problem solving to consist of three levels of behavior: recognition and classification of the problem situation, planning towards a solution to the problem, and execution and monitoring of the planned actions.

Complexity in Problem Solving

In the previous section, problem solving was discussed from a prescriptive point of view. The question remains as to what is it that makes problem solving complex? Woods' (1988) approach to the psychology of human behavior in complex problems is especially relevant to our interest in ITS. The reason is that his particular approach provides us with insights to determining the goals of an ITS -- what do we want the ITS to teach an operator in a complex dynamic system. The questions that Woods addressed include: what is complexity? how can we map the inherent complexities of particular worlds? what cognitive demands does a world impose on problem solvers? The rest of this section summarizes Woods' discussions and "answers" to these questions.

Complexity is not an entity by itself, it is a characteristic of a situation. Problem solving situations where complexity becomes an issue can be thought of as interactions between three components. First, there is the world or domain of interest to be acted on because of the problem. Next, there are one of more agents acting on the world in an attempt to solve the problem, in other words, the problem solver(s) and finally, the external representation of the world available and perceived by the agent(s). Problem solving situations become complex if the inherent characteristics of the world impose on the agent(s) cognitive demands that affect the adequate performance in various situations.

From the perspective of the world, Woods defines four dimensions of complexity that contribute to the cognitive demands of that world. First, a world can be characterized by its dynamism; this include how event-driven is the world and how much do various tasks compete over time. The number of parts and the extent to which these parts interconnect and interact in a domain provide the second dimension of complexity. A world is also characterized by its level of uncertainty in the data that describes the state of the world. Finally, the amount of risk involved in a world is the fourth dimension of complexity. Thus, every domain or system can be analyzed along these dimensions. With respect to the earlier discussion on complex dynamic systems, it is observed that the four dimensions are consistent with the previous characterization of CDS. In general, a CDS is a world that is very dynamic in nature, has many interconnecting and interacting parts, and involves some degree of uncertainty and risk.

So what is the impact of such a world on the cognitive demands and situations that the problem solver(s) will have to face? That is, a world that is defined relatively high on all the four dimensions above? The rest of the discussion will focus on the consequences of each dimension of domain complexity on the problem solving environment confronted by the agent(s).

In a dynamic and event-driven world, problem solving extends over time and solution to a problem may be long term and changing. Moreover, problems are interrelated: the plan(s) of actions to one problem influence the state or solution to other problems. New events or disturbances may occur at any time to affect a problem and/or how it is being solved. Consequently, a problem solver must have the cognitive skills to cope with the above situations. A dynamic world demands that a problem solver must be adaptive in two major ways. First, the problem solver must be able to make predictions about potential possibilities of how the system may behave. Second, the problem solver must be sensitive to the effects of new events or disturbances and be responsive to these effects in terms of his understanding of the world and his plans towards a problem solution. To suport these skills, the problem solver must possess knowledge about the world, its different states of behavior and its potential changes between states.

When a domain of interest is characterized by many interacting parts, there are several aspects that contribute to the complexity of the problem solving environment. If a problem solver is faced with a system with a large number of parts, he must learn to manage his time among various tasks that involve different parts. The problem of divided attention is intensified when the domain is also dynamic; the problem solver needs good prospective memory that enables him to come back to a task at a later time. However, if the parts in a system are intricate objects by themselves, it becomes very important for the problem solver to have a good understanding of the workings of these parts. In fact, a complex part is a system in itself and serves as a subsystem to the larger, global system.

When numerous components of the domain are extensively interconnected, several consequences are inevitable. First, actions carried out by the system operator to attain a particular effect may produce undesirable side effects. Similarly, errors and faults can propagate within various parts in the system. Also, such a world is a prime candidate for situations with conflicting and competing goals. In order to perform effectively the reasoning involved in such an environment, the problem solver must have knowledge about how different parts interrelate, affect and constraint each other in achieving different goal states. When faced with a situation with multiple faults, a cognitive demand on a problem solver is that of problem formulation. Essentially, the problem solver must be able make judgements about the problem to focus on based on his assessment of the situation and his knowledge about the system and its components. Another cognitive skill that a problem solver should possess

is disturbance management, particularly when the domain is also dynamic in nature. This skill helps the problem solver deal with the effects of the disturbance(s) at the moment and correct the crisis in the long run. Yet another cognitive demand on the problem solver involves diagnostic situations. The problem solver must have sufficient diagnostic skills to avoid errors such as fixation of a single explanation to account for the state of the world, treatment of interrelated problems as independent and oversimplification of the interconnectedness that exists among the various subsystems of the world.

When the domain is high on the uncertainty dimension of complexity, data available to the operator may be unreliable and that a given datum may be evidence to more than one part or state of the world. As a consequence of the former situation, a problem solver must have sufficient inference abilities to collect and integrate the erroneous data in order to explain a particular state of the world. To cope with the latter situation, the operator must have good reasoning skills to correctly map the evidence from the data to the state(s) these data testify to. Thus, the prerequisites to these skills include the problem solver's adequate knowledge on the various mappings of evidence to state(s). If uncertainty is coupled with dynamism, the task of the operator to collect evidence is compounded by two factors. First, not all data about the state of the environment are accessible at a given time. Second, the operator needs to weight the potential benefit of the information to be acquired with the cost or effort in the acquisition process. As a result, the problem solver needs to know different methods for collecting data; that is, he must know when and where to look for data. (** mention about monitoring aspect of the supervisory controller **) He must respond to and check for system events that unfold over time for evidence of a state of the system. Moreover, he must have adequate knowledge about the states of the system to initiate actions that support evidence gathering. In general, the cognitive demand to cope with large amount of data and information is part of problem formulation, where the problem solver must have the ability to discriminate and attend to relevant data in order to arrive at a solution. Correct utilization of the evidence surrounding an incident will avoid the potential of solving the wrong problem.

Finally, when the world is complicated by the presence of risk, the problem solver, in general, is constantly making decisions that takes into account the cost of a particular choice of action(s) to the overall state of the world. In addition, the problem solver must be concerned with not just expected and common situations, but infrequent situations with damaging results to the system.

In the final analysis, Woods emphasizes the importance of the above approach in the understanding the complexity of a problem solving world. The various demands and situations have strong implications on the other two elements of a problem solving situation, namely, the representation(s) of the world to the problem-solving agent(s) and the cognitive processing

capabilities of the agent(s). The breakdown on the different cognitive demands and situations also provide the basis for understanding the effectiveness and appropriateness of the numerous problem solving strategies that were disscussed previously. In accordance to theme of this paper, a global and ideal goal of an ITS designed for a complex dynamic system is to teach an operator all the cognitive skills that he requires to cope with the various cognitive demands and situations that arised due to complexity of the domain. The ITS should also instill into the operator all the knowledge about the system that he will need to support the skills. Questions such as how these skills is taught, and how much of the knowledge should be or can be taught explicitly are yet to be explored and answered.

Architecture of an Intelligent Tutorial System

* basic elements are domain expertise, student model, pedagogical expertise and interface (Wenger, 1987)
* similar breakdown by Fath (1987): task model, student model and instructional module. Interface is part of simulation.

*** according to Wenger

**** domain expertise**

* functions
- has two functions: as a source of knowledge and a standard for evaluating the student's performance
- as a standard, must be able to generate multiple solutions to a problem
- as a source of knowledge, there is a trade off between representing knowledge of expertise as a curriculum (static) versus as a model (dynamic)

* aspects of communicability
- domain knowledge includes pieces of information that are specifically used for instructional purposes (the learning process)
- issue of transparency of the expert module: how inspectable and interpretable are the reasoning steps to the final results
- issue of psychological plausibility of the expert module: how similar is the expert module's performance as compared to the human's.
- choice of viewpoint of the domain to be taken by the expert module should match that of the student. this is a limitation as compared to human expert's adaptability to various student's viewpoints.

**** student model**

* information: how accurate and well covered is the information contained in the student model
- information to interpret a student's behavior
- information to determine the knowledge state of the student based on the interpretation of his action
- explicit representation of the misconceptions a student may have about the domain
- information to explain how these misconceptions may have come about

* representation: language of representation must accomodate for incorrect knowledge of the student. language for expertise is thus not sufficient.
- neutral primitives: granular enough to account for both correct and incorrect knowledge in domain. language itself does not carry "correctness".
- error primitives: enumerative approach-- information about errors and misconceptions for a particular domain of students empirically collected and treated as primitives of the language.
- language is such that the student model should be runnable: model can generate predictions about the behavior of a student in a particular context.

* diagnostic process: accounting for data to form and update student model;
involves formulation and evaluation of competing hypotheses.
- assignment of credit and blame: intrepretation of actions may be top-down or bottom up. search for the student model may be model-driven or data-driven.
- diagnostic process should be robust to noise from three sources: student model is an approximation of the actual student; students are never perfectly consistent in their actions; learning factor may alter the truth about the knowledge state of a student.
- the diagnostic process may be active during a session by taking over a session and requiring the student to do stuff for diagnostic purpose. or the process may be passive; it observes and analyzes the student's action silently in the background. the process may be a mixed too.
- diagnosis may be interactive in nature if a student is involved in explaining his own behavior (but people are not good at doing that) or may be inferential where a student is excluded totally in the diagnostic process. a mix may be prefered.

** **pedagogical expertise**: knowledge about how to communicate knowledge

* didactic process
- represent pedagogical knowledge as rules versus principles
- global decisions affect the sequencing of instructional episodes

- local decisions affect the "when, what and how" of intervention. also includes decisions on guidance in performance, explanations of phenomena and remediation.

* degree of control
- monitor student's actions, but system never takes over
- mixed-intiative: control shared by both student and system
- guided-discovery learning or coached activities: student is in full control

** **interface**: final form of communication

* function
- interface should have conversational capabilitilies between the student and the system
- form of communication may involve language processing
- more popular form due to advanced technology is the use of computer graphics in representation

* desiderata (what is desired in the interface)
- should be clear and understandable in presenting system's topic
- should be explicit about system's capabilities
- should be easy and attracitve to use for the student

****** these breakdown does not neccessarily correspond to distinct modules in an ITS. also decisions about any of these issues in any one component will very likely affect those made for other components

Models of Intelligent Tutoring Systems

The outline for each discussion of a model is organized as follow:

A. Description
Any interesting or important general facts about the model is mentioned here. The methodologies or approaches used for each of the component of the ITS are identified under the following subheadings:
domain expertise
student model
pedagogical expertise
interface

B. Implications for Complex Dynamic Systems
What is applicable and what is not and why with respect to the dimensions of complexity will be addressed in this section.

C. An example in the GT-MSOCC Domain
The issues raised above will be illustrated and discussed in the context of an existing complex dynamic system called GT-MSOCC.

(1) SCHOLAR (Carbonell, 1970)

A. Description

SCHOLAR is considered the first intelligent tutoring system ever developed. Carbonell pioneered the artificial intelligence approach to ITS where knowledge is explicitly encoded. This approach replaced the traditional frame-oriented paradigm.

Domain Expertise

The system applies to the geography of South America. This domain knowledge is represented in a semantic network. The nodes on the network represent relevant objects and concepts that the system knows about. These objects are linked together hierarchically in the network.

Student Model

A early version of the "overlay" model (discussed in more details later) is used. The network can be used to represent the knowledge of an ideal student. Evaluations on a student's actual performance are identified with the concepts in the network that are taught.

Pedagogical Expertise

SCHOLAR does not have any sophisticated tutorial strategies. Its main concern in this respect is to select relevant topics for discussion based on the distance between nodes on the network and the notion of relevance tags of these nodes. Decisions are thus very local and at times random. The student and the system interact in a mixed-initiative dialogue mode.

Interface

The form of communication is textual. A template matching process is use to generate and parse simple sentences.

B. Implications for CDS

For factual knowledge such as geography, the notion of nodes and links can be readily defined. However, for an operator in a complex dynamic system, facts alone are not sufficient; he needs to possess procedural knowledge to carry out his tasks as a supervisory controller. Exactly what the nodes and links mean is not so clear for "how to" type information.

Another important aspect of a complex dynamic system that cannot be represented with a semantic network is dynamism. Specifically, such a network cannot accommodate the passage of time to reflect the potential changing states and behavior of a system. Such knowledge is crucial for an operator in developing his adaptive skills (recall Woods' discussion).

It is conceivable that semantic nets can be used to represent one "viewpoint" of a CDS in an ITS. For example, the complexity of the system in terms of the number of parts and their interconnectedness could be represented by several semantic networks at various levels of abstraction.

C. An Example in GT-MSOCC

One of the operator's function is to manually configure a mission upon request. In order to correctly carry out such a function, an operator must be taught to follow a sequence of plans. Such procedural knowledge would not be adequately represented in a semantic net.

However, part of the training of the operator is to acquire some background knowledge about the system. Factual knowledge such as the various mission configurations, the list of equipments needed by each mission and the maximum number of missions supported at any time could be represented as one or more semantic networks. The goal of the ITS at this point would be to make sure that the operator knows these facts about the system before moving on to the various operator functions. Somehow, the representational scheme used beyond this stage should be connected to the semantic network(s) for smooth transition and consistency.

(2) WHY (Stevens and Collins, 1977)

A. Description

Domain Expertise

WHY represents its domain knowledge in rainfall processes with hierarchical scripts. The authors attempt to capture both temporal and causal relations between typical sequences of events in these meteorological processes.

Student Model

There is no student model. A student's performance is evaluated independently.

Pedagogical Expertise

The tutorial strategy implemented in WHY is the Socratic method. In this method, a tutor asks the student questions to guide him in developing skills and principles for managing hypotheses and drawing relevant inferences from data collect. The strategy is captured in a set of tutorial rules that deals with local decisions about the appropriate questions to ask based on the student's last response. No global tutorial goals are considered in these decisions.

Interface

The dialogues between the tutor and the student is strictly textual. The natural language is processed in a similar fashion as in SCHOLAR.

B. Implications for Complex Dynamic Systems

The issues that evolved from the two major weakness of WHY have been discussed in length by Wenger. These issues will be explored further with respect to complex dynamic systems.

Considering global tutorial goals

In the rainfall domain, Stevens and Collins (1977, 1982) examine the higher-order goals of a human tutor that influence his tutorial decisions. They suggest that such goals must be incorporated into the pedagogical module of an ITS. To consider such goals is then to identify the teaching goals in terms of what a student is supposed to learn. The choice of a pedagogical approach should be consistent with these goals. It is possible and likely, especially with respect to complex dynamic systems, that the approach selected will embody more than one tutorial strategies to achieve all the pedagogical objectives.

In terms of the the kind of cognitive situations an operator will encounter and the type of skills needed to cope with these situations, when and how may the Socratic method be applicable? One possible direction is to

isolate a particular cognitive situation and tutor the operator/student to develop the corresponding skills in a Socratic style. The situation, which is a "case" in Socratic terms, could be presented to the student in a scenario of system events. The tutor proceeds to asks meaningful questions based on the student's actions or responses.

There are several problems that immediately come to mind. In a complex and dynamic world, the various cognitive situations overlap and interact with each other among all dimensions of complexity. Thus, there is no assurance that the skills acquired from two isolated situations will translate to the skills required to manage a single incidence with cognitive demands of both situations. Because the world is dynamic, events are evolving in "real time". As a result, a tutorial dialogue occuring within a scenario must avoid being too obstrusive to the extent of becoming unnatural. Another potential problem is that important events in the scenario may be missed while the dialogue is in progress. Intuitively speaking, it is not feasible to use only the Socratic style of teaching when the domain of interest involves a complex dynamic system. It seems that there may be skills more appropriate than others, and that there may be a more suitable time in the student's learning process than others to apply the Socratic method.

Represent domain knowledge from multiple perspective

The fact that scripts reflect only linear relations between events is even more profound a limitation in complex dynamic systems. Large number of components interact with each other in nonlinear and often unpredictable ways. In order for a student to develop skills to handle problems such as divided attention and prospective memory, the representation scheme chosen for the ITS must account for such nonlinearities.

Another limitation of script-based representation is that only global aspects of a process are captured in temporal and causal terms. The suggested functional perspective of the domain knowledge is particularly relevant in a complex dynamic system. The operator needs to have knowledge about the workings of each component and how it affects and constrains other components in the system. This knowledge supports the operator's many skills such as problem formulation in situations with multiple faults and conflicting goals. That is, both the "x causes y when" aspect and the "how x causes y and why" aspect of the domain knowledge must be captured in the expert model of an ITS.

Besides the above limitations, scripts are not suitable for expressing complex dynamic worlds for reasons characteristic of such worlds. Scripts are good for stereotypical sequences of events. In a complex dynamic system, from the perspective of a supervisory controller, the cause for concern is more for non-stereotypical sequences of events instead. Operators must know not just what normally happens to the system over time, but also what to do in novel situations. Skills in disturbance management and reasoning and inferencing abilities are required of these operators. In any case, the dynamic nature of such a system makes the task of defining all possible sequences of events a very exhaustive and impractical ordeal. Moreover, the uncertainty dimension (in terms of

system behaviors) makes the prediction of all potential sequences of events unrealistic. With regards to the issue of psychological plausibility, it is certainly true that experts do not have a script for every possible situation in order to solve different problems.

To the extend that the idea of multiple viewpoints in the representation of domain knowledge is believable, the form of communication of these viewpoints must go beyond just textual interface. The advance in computer technology make the use of visual and graphical techniques in interface design a very viable option (more on this is discussed in later models).

C. An Example in GT-MSOCC

Consider the possibility of implementing a Socratic style tutor for GT-MSOCC. A session (or a scenario) in GT-MSOCC has the goal of teaching the operator how to troubleshoot endpoints for software failures. The operator's actions and responses are evaluated such that the tutor can pose appropriate questions. The following is a sample list of what might happen:

1. The operator types "display msocc sched". Then there is a long pause...

2. The tutor decides to ask a question:"Do you think you need to check endpoints now?"

3. If the operator answers "yes", the tutor predicts the operator will next execute commands that support the goal to check endpoints (eg. display vip telem).

3a. The tutor then asks "Why do you need to see tac telem page?" to explore the operator's understanding of the task.

3b. The operator may then answer "Because vip3 is an endpoint equipment for the mission ERBE".

.
.
.

4. If the operator answers "no" to question in item 2, the tutor may ask "why not?"

4a. student may answer "because ....."

.
.
.

(3) METEOROLOGY Tutor (Brown et al., 1973)

A. Description

This project launched the work on qualitative models and set the grounds for subsequent research on SOPHIE (next section).

Domain Expertise

As the name of the system implies, the domain of application is meteorology. The core technique represents the causal knowledge about meteorological processes in a qualitative simulation model. Sequences of events in each process are simulated via a finite-state automata. The semantic network approach used in SCHOLAR is also implemented in this system to represent meteorological concepts.

Student Model

No effort is directed to modeling the student here.

Pedagogical Expertise

The tutor is a question-answering system. Questions about factual knowledge from the student are answered in a similar way as in SCHOLAR. To generate explanations for a question about a process, an inference tree is built dynamically from the simulation model. This inference tree describes the temporal and causal relations between events as related to the question.

Interface

The tutorial dialogues between the tutor and the student is carried out in natural language form. A simple process of keyword matching is used to extract the context of a question. Answers to questions about processes are constructed by joining successively predefined text units that reside in each state of an automata.

B. Implications for Complex Dynamic Systems

Operator Control Model (Miller, ?) and Operator Fucntion Function Model (Mitchell, 1987) are two modeling frameworks that involve networks of finite-state automata. The task of predefining all possible series of events is replace by the identification of system states. The dynamism of such systems can then be captured in the state transitions within the network. Thus, the idea of a dynamic process model is especially befitting with regards to complex dynamic systems.

The idea of dynamic generation of explanations may be used to consider a question-answering option for an ITS. The student selects this mode to acquire or review knowledge about the system. Such an option can only be supplementary to the actual teaching that is needed to assist the student in developing the appropriate skills in terms of a complex system.

- need better nlp instead of prestored text. in fact, should be able to take advantage of visual methods in presenting the answer (eg. showing the inference tree where answer is).

- idea of multiple representations supports the idea of multiple viewpoints.

- domain representation affects pedagogical decision and vice versa. that is, teaching goals also affect how we want knowledge to be expressed. what viewpoints or mental models do we want student to develop of the system? physical, functional, causal?

C. An Example in GT-MSOCC

- OFM methodology represents operator functions in a heterarchical/hierarchical network where state transitions reflect system triggering events. A tutor for GT-MSOCC may use the OFM for pedagogical decisions in exploring the student's understanding of the system and his task. Illustrates the dependency between domain representation and pedagogical strategies.

- since we already have OFM, may include a q/a mode operator can choose. Operator may ask questions relating to a system request or message, its effects, and/or how to fix the problem. Answers may be explanations, or even suggested steps or actions. Not really a tutor implemented. do not know if student is actually learning.

- the use of the blackboard for implementing OFM is one way to make model explicit. thus, operator can view the blackboard and see what he is expected to do.

(4) SOPHIE (Brown et al., 1974, 1976, 1982)

A. Description

Domain Expertise

The domain of application for the entire SOPHIE project is the troubleshooting of electronic circuits. Troubleshooting skills involve the ability to collect various measurements, to hypothesize the potential problem areas and to test such hypotheses.

SOPHIE-I and SOPHIE-II represent the domain knowledge in multiple ways. A simulation model represents the mathematical model of the circuit. Procedural knowledge is captured in a set of specialists based on this model, while declarative knowledge is reflected in a semantic network.

In SOPHIE-III, domain knowledge is represented in two separate module: the troubleshooting expertise and the electronics expertise. The troubleshooting expertise has general troubleshooting knowledge for managing a set of hypotheses. The electronics expertise has both general electronic knowledge and circuit-specific knowledge represented in three different levels: components model, production rules and behavior trees each linked with a different reasoning mechanism and input information.

Student Model

Although considered, this portion of the research was never implemented.

Pedagogical Expertise

The pedagogical paradigm of the SOPHIE project is to provide a reactive learning environment for the student. In such an environment, the student has the opportunity to test his ideas and knowledge, and receive constructive feedback and advice.

In SOPHIE-I, pedagogy consists of generating meaningful feedback to a student's action by making inferences based on the knowledge embedded in the simulation model. An articulate expert troubleshooter in SOPHIE-II explains the reasoning and strategies underlying these inferences. The representational scheme in SOPHIE-III works as an inference engine to reflect human-like reasoning. The idea is to use this engine for coaching and modeling the student in an active environment. Unfortunately, this part of SOPHIE-III was never completed.

Interface

SOPHIE and the student interacts via a very robust natural language interface. The natural language processing is implemented with semantic grammers. The idea is to represent a sentence based on domain-dependent semantic catogories instead of its syntax.

B. Implications for Complex Dynamic Systems
C. An Example in GT-MSOCC

(5) STEAMER (William, Hollan, Stevens, 1982)

A. Description

This project pioneered the notion of graphical simulations in training systems. Projects such as the Intelligent Maintenance Training System (Munro et al., 1985) and the Recovery Boiler Tutor (Woolf et al., 1986) have been influenced by STEAMER.

Domain Expertise

The domain of application is operating steam propulsion plants in large ships. The model of the domain knowledge is purely mathematical. From the knowledge communication perspective, STEAMER does not really have a model of the expertise.

Student Model

STEAMER does not have a student model (?)

Pedagogical Expertise

STEAMER presents the steam propulsion plant in an interactive and inspectable graphical simulation form. The student can manipulate various aspect of the simulated plant and examine the effects of his actions. The pedagogical goal is to provide a means for the student to acquire a mental model of a complex physical system and at the same time learn to operate such a system.

To further support this goal, two other modules are implemented. When a student is running a particular procedure, the tutorial module can furnish feedback in the form of explanations based on the graphical abstractions that define the simulated plant. Another module called the feedback minilab allows the student to experiment with different control devices. The student can put together the components for a device and STEAMER will test it by integrating the simulated device with the rest of the system.

Interface

Within the STEAMER's graphical interface, the system and the student interact through simple text processing. (eg. menus and options).

More importantly, the graphical description of STEAMER's simulation model initiated the principle of conceptual fidelity. The goal is to present a conceptual view and not the physical view of a complex system. This view when presented to the student is considered faithful to the actual system if it expresses the same view possessed by experts. Such a view should reflect the mental model that experts use when they reason about the system.

B. Implications for Complex Dynamic Systems
C. An Example in GT-MSOCC