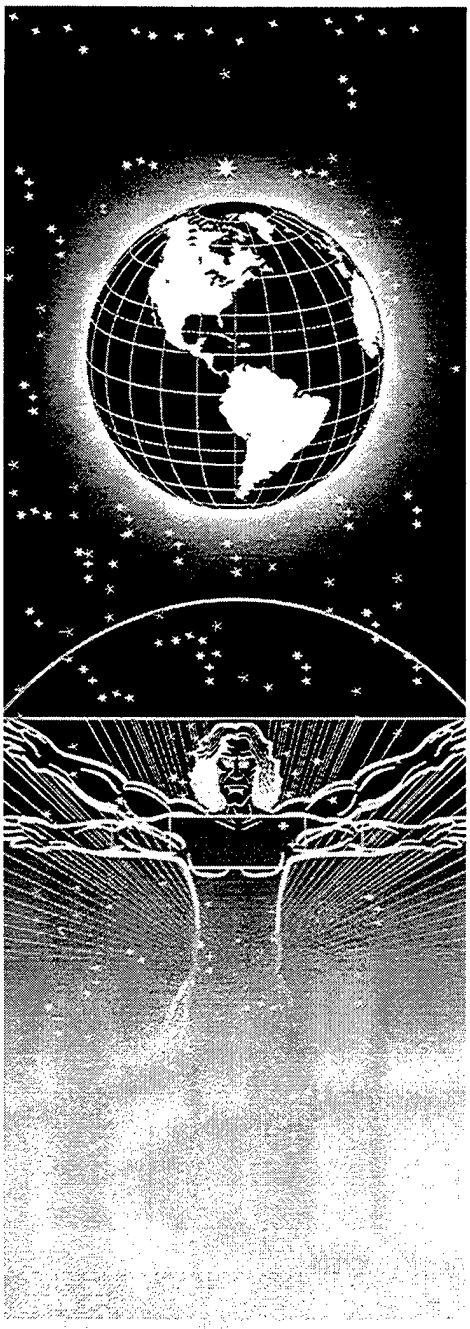
AL/HR-TR-1996-0161



UNITED STATES AIR FORCE
ARMSTRONG LABORATORY

Operability Model Architecture Demonstration Final Report

Stephen E. Deutsch
Jean MacMillan
Nichael L. Camer

BBN Corporation
10 Moulton Street
Cmbridge, Massachusetts 02138

Sonu Chopra

Center for Human Modeling and Simulation
University of Pennsylvania

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION
2698 G Street
Wright-Patterson AFB OH 45433-7604

May 1997

19970616 032

DTIC QUALITY INSPECTED

Approved for public release; distribution is unlimited

Human Resources Directorate
Logistics Research Division
2698 G Street
Wright-Patterson AFB OH 45433-7604

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation, or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

MICHAEL J. YOUNG
Program Manager

BERTRAM W. CREAM, GM-15, DAF
Chief, Logistics Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 1997

3. REPORT TYPE AND DATES COVERED
Final - March 1995 to December 1996

4. TITLE AND SUBTITLE
Operability Model Architecture Demonstration Final Report

5. FUNDING NUMBERS
C - F33615-91-D-0009
PE - 62205F
PR - 1710
TA - D1
WU - 02

6. AUTHOR(S)
Stephen E. Deutsch
Sonu Chopra
Jean MacMillan
Nichael L. Cramer

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BBN Corporation
10 Moulton Street
Cambridge, Massachusetts 02138

Center for Human Modeling and Simulation
University of Pennsylvania

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Armstrong Laboratory
Human Resources Directorate
Logistics Research Division
2698 G Street
Wright-Patterson AFB OH 45433 7604

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AL/HR-TP-1996-0161

11. SUPPLEMENTARY NOTES
Armstrong Laboratory Technical Monitor; Michael J. Young, AL/HRGA, DSN 785-8229

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The Operator Model Architecture (OMAR) is an integrated suite of software tools to support the construction of Human Performance Process (HPP) models. This report documents the use of OMAR to create and test HPP models for Air Traffic Control (ATC) and research in anthropometric human modeling. In addition, this report documents the development of new model analysis tools.

14. SUBJECT TERMS
graphical editing and browsing
human performance modeling
object-oriented simulation
procedural languages

15. NUMBER OF PAGES
83

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

TABLE OF CONTENTS

PREFACE iii
ABSTRACT iv
1. INTRODUCTION 1
2. HPP MODEL DEVELOPMENT AND EVALUATION 3
2.1. Extensions to the Psychological Framework 3
2.1.1. Attention 4
2.1.2. Teamwork 7
2.1.2.1. What is a team? 8
2.1.2.2. Requirements for a Theory of Teamwork 9
2.1.2.3. Small Group Research 10
2.1.2.4. Computer Supported Collaborative Work (CSCW) 11
2.1.2.5. Petri Net Models 11
2.1.2.6. Normative/Descriptive Mathematical Models 12
2.1.2.7. Team Training Research and Mental Model Theories 13
2.2. Behavioral Test of the Psychological Model 16
2.2.1. Method 18
2.2.1.1. Experiment Task and Environment 18
2.2.1.2. Experiment Design and Procedures 19
2.2.1.3. Measures of Effectiveness 21
2.2.2. Results 22
2.2.2.1. MOE Results 23
2.2.2.2. Workload Results 27
2.2.2.3. Difficulty Ratings and Strategy Descriptions 27
2.2.3. Discussion and Implications for the Model 30
3. IMPROVEMENTS TO THE OMAR ANALYSIS ENVIRONMENT 31
3.1. The Post-run Analysis Framework 31
3.1.1. Saving the Data from a Scenario Run 31
3.1.2. Recovering The Data From A Stored Scenario 32
3.2. Measures of Effectiveness 33
3.2.1. Frequency Measures 34
3.2.2. Duration Measures 35
3.2.3. MOE Examples from Behavioral Test of the Psychological Model 36
3.2.4. OMAR Support for MOE Data Collection 37
3.3. The Graphical Holon Analysis Tool and the SCAN Display 39
3.3.1. Design Criteria 41
3.3.2. The Connectivity Graph 43
3.3.2.1. Agent Display 43
3.3.2.2. Procedures Which Transmit Signals 44
3.3.3. Transmitted Signals 44
3.3.3.1. Procedures Which Receive Signals 45
3.3.3.2. Agent Signal Displayer 45
3.3.3.3. "Orphan" Signals 45
3.3.3.4. Agent Filtering 46
3.3.4. Timeline Graph 46
3.3.5. Scrolling in the Scan Display 47
3.3.6. Mouse Documentation Pane 48

4. OMAR HPP MODEL APPLICATIONS: OASYS AND DEPTH 48
4.1. Integrating an OMAR HPP Model into the OASYS Demonstration 49
4.1.1. The Interface Between OMAR and OASYS 50
4.1.2. The OASYS Demonstration ATC Scenario and the OMAR HPP Models 52
4.2. DEPTH Workstation Demonstration 53
4.2.1. The Interface Between OMAR and Jack 55
4.2.2. Maintenance Procedure Development Using an HPP Model 59
4.2.2.1 Jack Animation of an Aircraft Maintenance Procedure 60
4.2.2.2. OMAR Extensions to the Aircraft Maintenance Procedure Representation 62
REFERENCES 63

FIGURES

Figure 1. Roadmap for Literature on Groups and Teams 10
Figure 2. The SCAN Display 42
Figure 3. Stand Graph Display 43
Figure 4. Agent Display 43
Figure 5. Transmitting Procedures 44
Figure 6. Transmitted Signals 44
Figure 7. Procedures Receiving Signals 46
Figure 8. Signal Displayers 46

TABLES

Table 1. Experiment Task Definition—Steps in Processing Each Task 19
Table 2. Measures of Effectiveness Used in Experiment 22
Table 3. Mean Number of Holding Aircraft by Display Condition and Task Load (n=9) 23
Table 4. Mean Acceptance Delay (in seconds) by Display Condition and Task Load 24
Table 5. Mean Release Delay (in seconds) by Display Condition and Task Load 24
Table 6. Spare Release Time (in seconds) by Display Condition and Task Load 25
Table 7. Mean Number of Aircraft Not Welcomed by Display Condition and Task Load 25
Table 8. Mean Workload Ratings by Display Condition 26
Table 9. Mean Difficulty Ratings for Accomplishing Task by Display Condition 28
Table 10. Changes in Strategy When Switching from Unaided to Status Condition 28
Table 11. Changes in Strategy When Switching from Priority to Unaided Condition 29
Table 12. Changes in Strategy When Switching from Status to Priority Condition 29

PREFACE

The work on the Operator Model Architecture (OMAR) was conducted under Delivery Order 8 of the Research, Development, Training, and Evaluation (RDT&E) Support program administered under U. S. Air Force Contract Number F33615-91-D-0009.

The authors wish to thank the Contract Monitor, Mr. Michael J. Young of Armstrong Laboratory's Human Resources Directorate, for his constructive criticism and enthusiastic support of the research effort. The authors also wish to acknowledge the significant contributions made by Dr. Carl E. Feehrer and Dr. Marilyn Jager Adams to previous OMAR-related research.

ABSTRACT

The Operator Model Architecture (OMAR) provides a suite of software tools for the development of Human Performance Process (HPP) models. The principal tools include the Simulation Core Language (SCORE) for the representation of human behaviors and the Simple Frame Language (SFL) as the basis for the object-oriented simulation environment. Graphical editors and browsers and Post-Run Analysis tools support the development of HPP models and the evaluation of their performance. Under the current task, (1) the psychological framework that is the basis for the human operator models was extended in the areas of attention and teamwork; (2) the post-run analysis tools were gathered into a Post-Run Analysis Framework, a new capability was added to specify and collect measures of effectiveness data, and a new analysis tool, the Signal Connectivity Analysis Network (SCAN) display was developed principally to evaluate the performance of holon-based HPP models; and (3) HPP models were developed to operate with the Operability Assessment System (OASYS) and an HPP model developed as an extension of Jack, the University of Pennsylvania's anthropometric model, was used to examine the development of aircraft maintenance procedures.

1. INTRODUCTION

Simulation has been used for many years as a tool to evaluate target system performance, but it is only recently that attempts have been made to include realistic models of the human operators in these evaluations. The Operator Model Architecture (OMAR) is a simulation system that addresses the problem of modeling the human operator directly. Its development focused first on the elaboration of a psychological framework that was to be the basis for the Human Performance Process (HPP) models to be developed, and then on the design of a suite of software tools to support the development of these HPP models. The underlying assumption that was born out was that the software tools that supported the development of the HPP models would be adequate to address the problem of modeling a given target system.

The ability to model human operators and their interactions with target systems has opened up several new areas for investigation through simulation. Procedure development is an important one. Simulation can now have an impact on both operator procedure and maintenance procedure development. It will now be possible to pursue procedure development by "trying out" procedures far earlier in the design cycle than was previously possible. The early evaluation of both operating and maintenance procedures can be fed back into the design process to improve system operability and reduce downstream maintenance costs. An important product of simulation-based procedure evaluation is the capture of a semantically rich computer representation of the procedure under investigation. The richness of the representation is just what is needed to support the substantial documentation effort necessary to field a new set of procedures. Text generators working from this procedure representation can provide paragraph length output to support the authoring of electronic technical manuals for system operation and system maintenance.

As systems become more complex, the range of available user interactions increases, and it is more frequently the case that individual operators are members of a team cooperating to accomplish a given task. They may be working at a single site or, in the case of distributed systems, interacting from remote sites. As the tasks of the human operator become more complex, the HPP models must be further refined to adequately portray the human operator if we are to continue to meet the goal of simulation-based procedure development and evaluation.

It was against this backdrop that Delivery Order 8 was conceived. The psychological framework that has formed the basis for human performance modeling in OMAR was laid down in Delivery Order 1 (Deutsch, Hudlicka, Adams, & Feehrer, 1993a). In Delivery Order 5 (Deutsch, Adams, Abrett, Cramer, & Feehrer, 1993b), the suite of software tools that constitutes OMAR was designed and implemented, and the initial HPP models of human operators were developed. The tasks undertaken in Delivery Order 8 fall into three categories:

- The further development of the psychological framework to extend the range of application of the HPP models and the comparison of HPP model performance with human operator performance;
- The addition of significant new features in the area of post-run analysis; and
- The development of HPP models for specific applications.

Attention and teamwork were areas identified for the further development of the psychological framework. On the one hand, team members can be viewed as resources to assist in the accomplishment of a task, while on the other hand, they can be the source of interruptions to ongoing tasks as they seek assistance in meeting their own goals. Improving the capability of HPP models to accurately represent operators as team members thus makes significant demands on the representation of attention.

Just how well HPP models represent the human operator is a very difficult question to answer. We have begun to answer this question by comparing human performance with HPP model performance using an air traffic control scenario. In the experiment, we have sought to better understand the execution of a task by a human operator and compare that performance with that of the HPP model for the air traffic controller. The ability to develop an OMAR-based user interface that may be operated either by a human operator or by an HPP model was essential to making this comparison possible.

OMAR has a powerful and very general data collection capability and two timeline displays: one that provides data on agent procedure execution and another that presents event data. Each of these displays runs from on-line data. The new Post-Run Analysis Framework enables scenario data to be stored to and retrieved from disk, and enables data from several simulation runs to be reviewed simultaneously. A new display, the Signal Connectivity Analysis Network (SCAN), has been provided to support the

analysis of holon-based human performance models (Young, 1992), and new event types have been provided to ease the process of collecting measures of effectiveness data.

Lastly, OMAR has been used to develop HPP models that have been used in other simulation frameworks. In one case, OMAR was used to develop a human performance model of an air traffic controller for a target system that was modeled using the Operability Assessment System (OASYS). In the second effort, an OMAR HPP model was linked to Jack, the University of Pennsylvania's anthropometric model. The OMAR-Jack integration was then used as the basis for the development of a knowledgeable agent capable of executing aircraft maintenance procedures.

The following sections describe the tasks accomplished in the areas of HPP model development and evaluation, improvements to the OMAR Post-Run Analysis Framework, and the employment of OMAR HPP models in related simulation environments. The research described here was completed under Delivery Order 8 of the Research, Development, Testing, and Evaluation (RDT&E) program during the period March 1995 to February 1997.

2. HPP MODEL DEVELOPMENT AND EVALUATION

2.1. Extensions to the Psychological Framework

The modern workplace, be it military or civilian, is seldom the province of a single person. A person's work is almost always part of a larger effort, linked more or less closely to the work of others, either at a nearby workplace or at a remote site. People working with others at remote sites is becoming more common as the capabilities of networked systems improve to support this mode of operation. As these workplace changes take place, simulation is being used more frequently to develop prototypes of new system designs or evaluate existing systems. In these systems, the human operators must also be modeled if the operation of the system is to be adequately assessed.

Research in this task addressed the demand to provide human performance models of human operators in their capacity as team players. The framework in which the problem was addressed was the basic air traffic control environment which has been used for the past several years. The human players modeled in the environment are the en route air traffic controllers and the aircrews of the aircraft in their sectors. The aircrew members of an aircraft function as team players, as do pairs of air traffic controllers in neighboring sectors. The scenarios developed focused on the hand-off of an aircraft from one air

traffic controller to a neighboring controller. In these scenarios, the aircrew member of the aircraft responsible for communication and the controllers of the current and new sectors for the aircraft also form a team.

The modes of communication modeled included the in-person conversations between aircrew members, the party-line radio communication between the aircrews and the air traffic controllers, and the telephone conversations between air traffic controllers. In this environment, both the in-person and telephone conversations can be interrupted by party-line radio messages. There are frequently situations with multiple demands on auditory attention. In the case of the party-line radio, there are often messages for one aircraft to which the other aircrews attend with less than complete attention.

Given this description of the communication activities among team members, it is very clear that teamwork makes significant demands on the attention of individual team members. Proactive activities require that attention be focused to support the given task. Similarly, reactive demands are made on attention by interruptions that may be auditory or visual in nature. To improve the behaviors of the HPP models, it was important to examine the nature of teamwork (particularly its impact on attention), improve those aspects of the model that support attention, and enable the HPP model to exhibit reasonable behaviors in the teamwork aspects of the tasks being executed.

2.1.1. Attention

Of the several modalities of attention, visual and auditory attention are most important to the current modeling efforts. The work on auditory attention was most concerned with the verbal communications of the air traffic controllers and aircrew members, either in person, via telephone, or over the party-line radio. The work on visual attention focused on the air traffic controller's use of the synthetic radar screen and visual support and coordination of manual workplace tasks. Both visual and auditory attention have reactive and proactive components. Verbal communication is the basis for proactive coordination of flight deck activities, while party-line radio communications are frequent interruptions to ongoing activities. In the visual domain, the appearance of a new aircraft icon on the radar screen may interrupt an ongoing activity, while proactive visual attention is required to accomplish simple manual actions such as button pushes.

The basic architectural components of the human performance model are implemented at the symbolic level. They are modeled on the large-scale structure of the brain as outlined by Edelman (1987, 1989) and Damasio (1989a, 1989b). The performance of a particular

functional capability is typically implemented through the participation of a small number of centers. Subsets of these centers, operating concurrently, will typically have links between them operating in both directions—*reentrant* signals as described by Edelman (1987). The operation of a set of "lower" level centers is "coordinated" by a "higher" center, or *convergence zone* in Damasio's (1989a, 1989b) terms. Damasio's convergence zones are much like Minsky's (1986) hierarchy of agents in *The Society of Mind.*

The Simulation Core (SCORE) language, the procedural language within OMAR, provides the basis for representing functional capabilities. A SCORE procedure is used to represent a simple capability. The procedure is typically in a *wait-state,* pending activation based on pattern matching to a particular form of stimulus. Several such procedures may represent the components of a particular functionality. The links that activate the procedures are the signals that any one of them might generate. For any signal generated, one or more procedures may be enqueued on it, and hence, activated. Due to their pattern matchers, some procedures may respond to a stimulus, while others may ignore it. The pattern of activation in a complex of procedures may differ due to variations in the initial stimulus. A small complex of procedures is used to represent any given functionality.

Within any given complex, several procedures may be running concurrently, some representing automatic processing, others representing components of attended processing. Several layers of processing may be going on concurrently, the initial stimulus initiating the "lowest" processing level, with subsequent "higher" processing layers starting up at the behest of initial "output" from the next lower level. The behaviors of the concurrent processes are based on those discussed by Jackendoff (1987) in *Consciousness and the Computational Mind.*

Within this architectural framework, attention is not simply one component or one complex of procedures. Following Neumann (1987), attention is a "generic term for a number of phenomena each of which is related to a different selection mechanism." In building the model of attention, a selected subset of these phenomena related to air traffic control and aircrew tasks were implemented. The focus was on auditory and visual processing, since they are the most important forms of attention in managing the air traffic control and flight deck workplaces.

The processing of auditory messages is modeled by a set on concurrent processes. At the lowest level is a "hearing" process that is initiated in response to the onset of the auditory input. Shortly after the hearing process is initiated, it in turn triggers a message "understanding" process. Lastly, an "attended" listening process represents the hearer's attending to the spoken message. The nature of the communication over the party-line radio made it necessary to factor in another layer of complexity. From an aircrew's perspective, many of the ATC messages relate to other aircraft in the airspace and can be ignored at some level. Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy (1995) suggest that this is determined as soon as a verbal discriminator appears. The air traffic controllers identify the target aircraft for each message as the first utterance in a message, but it is clearly not the case that both the "understanding" and "attending" processing stop at this early point in message processing. An aircrew member will not initiate a verbal communication with another crew member while the "ignored" message is still coming in. The speculation represented in the model is that the "understanding" process continues processing the incoming message, and it is this process that "flags" the end of the message, so that another verbal message in the person-to-person conversation among crew members may be initiated. Indeed, it may well be the case that the message being heard is a directive for which an immediate response is expected, further delaying the initiation of the person-to-person conversation and reinforcing the speculation that at least the "understanding" level processing must be ongoing through the duration of the message and response.

The presence of the party-line radio means that auditory interruptions are an expected occurrence. In particular, it must be possible to stop an intra-crew conversation at the onset of a radio message and resume the conversation at the completion of the interruption. SCORE priorities assigned to the in-person and radio conversations are used implement the processing of the interruption by the radio message. The intra-crew verbal transactions are typically a statement-response pair. As implemented, an interruption anywhere in the exchange will be resumed, not at the point of the interruption, but by the initial statement of the exchange being made again.

Visual attention, as modeled in OMAR, has reactive and proactive components. The proactive processes are executed primarily in support of related cognitive and manual processes. The reactive processes are concerned with the response to visual events. The air traffic controller's synthetic radar workplace is a visually rich environment. The visual "events" modeled include the appearance of a new aircraft icon on the radar screen as the aircraft approaches the air traffic controller's airspace, the movement of the aircraft

icon across the screen as its position is updated, and a flashing light on the telephone to announce an incoming call from a neighboring controller. The initial response to a visual event is the simple act of identifying the event followed by a sequence of signals that trigger the appropriate procedures to respond to the event. The events that occur are not unexpected and there is typically a goal governing the response to each event. The execution of the response at each stage is mediated by the priority associated with the response procedure and that of the other ongoing procedures.

Visual attention in support of cognitive procedures takes several forms. Probably the most complicated activities take place as the air traffic controller "sits down" at the radar console to "take over" control of the airspace from the previous controller at the start of a scenario. There are several aircraft in the sector and in neighboring sectors. Flight strips, arrayed at the side of the radar screen, provide flight plan information on active and pending aircraft. The air traffic controller's initial acts are to "read" the active flight strips, associate each with the appropriate aircraft icon on the radar screen, and initiate a procedure for managing that aircraft's transit through the airspace. Implicit in the procedure for managing the aircraft is the memory for where the aircraft icon appears on the radar screen. The expert air traffic controller, like the expert chess player, "knows" where the pieces are on the board. In the HPP model, this memory is local to the procedure for managing the particular aircraft, rather than a slot in a "memory" resource (Deutsch et al., 1993a).

The level of visual attention required by simple manual tasks is also modeled. Precise manual actions such as reaching to push a button require visual attention at the target to accomplish the task. Reaching to pick up the telephone receiver, a comparatively large object, is modeled as requiring only a quick glance. Bringing the telephone receiver up to one's ear clearly does not require visual attention.

2.1.2. Teamwork

The operation of complex systems increasingly requires teamwork—multiple individuals must coordinate their actions in order to accomplish a task or a mission. The organization, training, and support of effective teams has become a focus of research in both military and industrial applications. Our goal in this task was to develop a psychological framework, based on the emerging teamwork literature, that would suggest possible approaches for modeling teamwork behavior in the OMAR environment.

2.1.2.1. What is a team?

Recent work on team organization and team training has sharpened the definition of what constitutes a team. Salas, Dickinson, Converse, and Tannenbaum (1992) define a team as having the following characteristics:

- Two or more individuals are involved.
- There is dynamic, interdependent, and adaptive interaction.
- There is a common goal, mission, or objective.
- There is some organizational structure of the team members.
- Each individual team member has specific tasks or functions.
- Task completion requires the dynamic interchange of information, the coordination of task activities, and constant adjustment to task demands.

Teams are often distinguished from "groups" (Orasanu & Salas, 1993) based on characteristics such as the presence of highly differentiated roles, interdependence among members, and the performance of tasks that require coordination among multiple individuals. Examples of teams include aircraft cockpit crews, tank crews, surgical teams, sports teams, and string quartets. Groups, in contrast, have homogeneous and interchangeable (non-specialized) members; the member are independent (coordination is not required), and they perform tasks that could have been done, although perhaps not as well, by one person. Some examples of groups falling under this definition include juries, panels of judges, and the ad hoc problem-solving groups often studied in social psychology experiments on group decision making.

When the above definition is applied to distinguish teams from groups, it becomes obvious that some types of multi-person tasks require teams, while others are best performed by groups, and still others require a hybrid of the two. Tasks that call for precise, quick action by multiple individuals with tight coordination in space and/or time require a team. To successfully perform these tasks, each team member must clearly understand his or her responsibilities, and considerable practice and training may be necessary to achieve the coordination required. In general, team tasks can be defined in advance and team skills can be practiced before they must be applied.

Tasks that require little coordination (so every individual can act independently) or tasks of limited duration (where there is no opportunity for practice or training) may be accomplished by a group. Groups of individuals with diverse backgrounds and perspectives may also be used to explore new territory when tasks are not well defined.

Other types of tasks may require a hybrid approach that combines some characteristics of teams and groups. For example, the tasks of a design team may require multiple specialized skills, so that individuals are not interchangeable and there are well-defined roles. The team's tasks often require creativity, are not repetitive, and cannot be practiced in advance. Although coordination is required in a design team, this coordination is not usually tightly constrained in either time or space. Thus, design teams fall between teams and groups in their characteristics.

2.1.2.2. Requirements for a Theory of Teamwork

In order to be useful for human performance modeling, a psychological framework for understanding team behavior must meet certain criteria. First, the framework must deal with measurable outcomes and defined and observable behaviors, and must be able to relate behaviors to outcomes. In addition, the framework must be based on a theory of the cognitive processes that underlie behaviors. In order to be useful in the OMAR environment, the framework must specify the knowledge structures used by team members and the processes that operate on those structures. For example, a theory that related the "leadership qualities" of team members (as rated by observers) to the quality of the team's solutions (as rated by other observers) is of little value for HPP modeling because the underlying structure and process of "leadership" is not well defined and well structured enough to be modeled.

Figure 1 presents an overview of the major types of literature reviewed for framework development. Literature "clusters" fall along two major axes: 1) whether the literature deals with groups or teams, and whether those groups or teams are artificial (created purely for research purposes) or real (perform a real-world task); and 2) whether the literature deals only with observable behaviors or also examines underlying cognitive processes. Six major types of literature are shown; each is discussed in more detail below. In our review, we found that the most relevant literature for the OMAR framework comes out of the team training research that has focused on real-world teams and deals with the mental models underlying the cognitive processes of those teams.

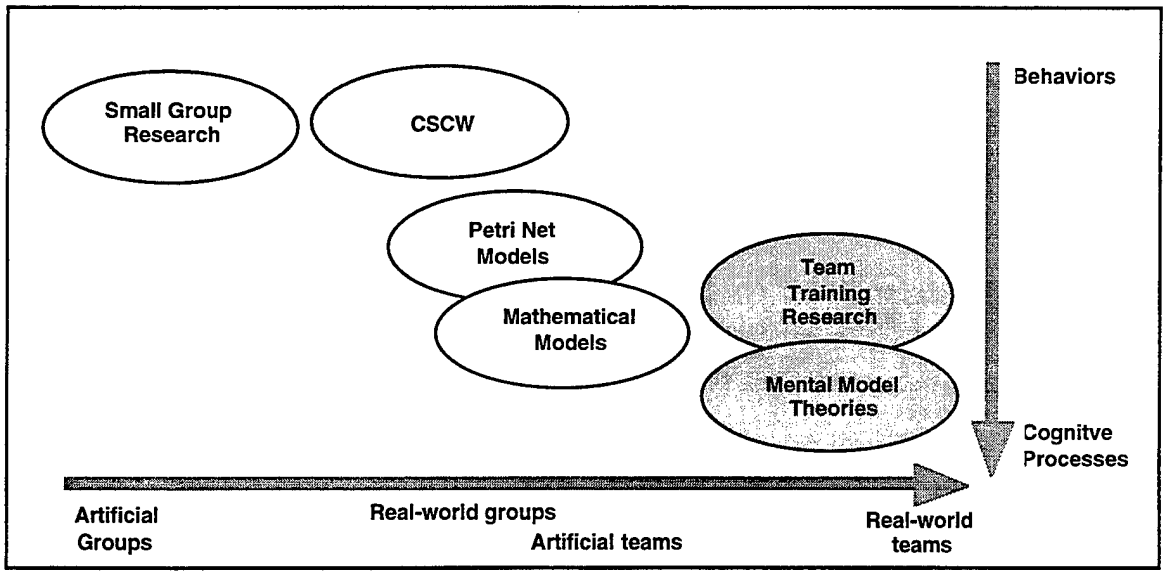


Figure 1. Roadmap for Literature on Groups and Teams

2.1.2.3. Small Group Research

Small groups have been the focus of study in the behavioral sciences for the better part of this century, and a vast literature has been accumulated on both group product and group process (see McGrath, 1984, for a comprehensive review of the research, and Driskell and Salas, 1992, for a high-level review of the value of this research for understanding teams). Studies have examined the effects of group size, composition, structure, and cohesion on group performance as well as the effects of member status, influence, and communication patterns.

A problem frequently cited with small group research (McGrath, 1991) is that it is based on synthetic, ad hoc teams that are created by the experimenter and performing tasks arbitrarily assigned for research purposes. The groups typically studied have no collective history and no context in any larger organization or social unit outside of the experiment. Also, many of the tasks studied are group tasks, not team tasks, under the definition described above—they do not require tight coordination in time and space and may not involve specialized, interdependent roles for group members. Many team researchers feel that the findings of small group research have little applicability to real-world teams (Orasanu & Salas, 1993).

An additional limitation of small group research from the perspective of HPP modeling is that for the most part, it focuses on observable behaviors, rather than on the cognitive processes underlying those behaviors. Orasanu and Salas (1993) comment that "when we

look to the literature for theory that accounts for *team decision making* in complex environments, we find the shelves to be practically bare" (emphasis added). In order to model team interactions in OMAR, we need insight into the individual decision-making processes that produce interactive team behaviors.

2.1.2.4. Computer Supported Collaborative Work (CSCW)

As computer technology has evolved to allow multiple individuals to work together via linked groupware applications, research has flourished on computer-supported collaborative work (CSCW). This research typically examines the effects of groupware technology on group process and group outcomes (see Baecker, 1993; Galegher, Kraut, & Egido, 1990; and Greif, 1988). Typical applications examined include electronic meeting rooms, e-mail and desktop videoconferencing systems, and other distributed collaboration tools that allow individuals in multiple locations to work together.

The major limitation of this work related to developing an OMAR framework for modeling teams is the emphasis on technology and the focus on behaviors rather than cognitive processes. The major independent variable in CSCW research is usually the presence/absence or the features of the technology, and the research examines the impact of the technology on group process or group outcomes. The work often focuses on real, not synthetic, groups or on hybrid groups/teams such as design teams. Team members may be interdependent, and coordination is usually required, but the tasks (e.g., developing a plan, solving a problem, developing a design) do not usually require coordination that is tightly coupled in space or time. Research does not typically concentrate on the cognitive and decision-making processes of the individuals in the group, but only on their observable behavior (e.g., communication patterns) and on the products of the group.

2.1.2.5. Petri Net Models

One approach to understanding teams has been to apply mathematical modeling tools to the team interaction process. This includes the use of Petri nets (discussed below) and the development of normative/descriptive mathematical models, discussed in the next subsection. Petri nets are a representational tool based on graph theory (Peterson, 1981). They can represent logical sequences of decisions, and can represent both conflicting and concurrent processes. Petri nets have been used to model and analyze a variety of complex systems including parallel computing systems, computer-integrated

manufacturing systems, chemical reactions, and legal systems (Coovert & McNelis, 1992).

Over the past 15 years, Levis and his colleagues (Boettcher & Levis, 1982, 1983; Levis, 1989; Tabak & Levis, 1985; Weingaertner & Levis, 1989) have applied Petri nets to study the organizational structure of decision making. Individuals are modeled using a two-stage process of situation assessment and response selection. Individuals are linked together into a team, and constraints are placed on the nets to model bounded rationality, alternative decision strategies, work load, and organizational structures.

Coovert and McNelis (1992) take a somewhat different approach in using Petri nets to model teams. They construct a functional, task level net and "layer" other nets representing communication patterns and individual decision making on top of the functional net. Choices, and the information that goes into those choices, are explicitly represented by places in the net. Activities are represented by transitions. The Petri net captures the interdependencies between information, decisions, and actions in, for example, a three-member team task of transferring supplies from a supply ship to a receiving ship while at sea.

The major limitation of Petri net models for the OMAR framework is that they are a representational tool, not a theory. Petri nets allow explicit representation and communication of many of the phenomena characteristic of team decision making, such as conflict, concurrency, asynchronous behavior, and hierarchical behavior. Petri nets represent only the explicit decision and action points, however. They do not contribute any theory about the cognitive processes underlying the observed decisions and actions, or any theory about how team behaviors are linked to team performance. Thus, the Petri net literature has limited utility in the development of a teamwork modeling framework for OMAR.

2.1.2.6. Normative/Descriptive Mathematical Models

The normative/descriptive approach (Kleinman, Luh, Pattipati, & Serfaty, 1992) to modeling teamwork begins by developing mathematical theories that define optimal, or close to optimal, behavior for well-structured team decision making problems. The behavior of these normative models is then compared to the behavior of teams of human decision makers, and the models are adjusted to fit the data using descriptive parameters that represent human limitations and biases. For example, Bushnell, Serfaty and Kleinman (1988) and Mallubhatla, Pattipati, Kleinman, and Tang (1991) found that team

members systematically undervalue the information that they receive from their partners, as compared to a normative model.

The normative/descriptive modeling approach has two major limitations for an OMAR framework. First, it has been applied and tested only in abstract and artificial environments. Normative models are feasible only for well-defined and simplified abstract problems. The lessons learned from experiments with synthetic teams performing artificial tasks (e.g., underweighting of information from other team members) may apply to real teams in real world environments, but this applicability has not been demonstrated. Second, the theory underlying the models is mathematical, not cognitive. The body of work in normative-descriptive modeling shows that humans systematically fall short of the optimal performance that is possible for highly structured team tasks, but it does not offer any insight into how and why these shortfalls occur, or into the nature of the (non-optimal) process by which the team is operating.

2.1.2.7. Team Training Research and Mental Model Theories

The training of effective teams has become increasingly important in both military and civilian settings over the past two decades. The military is increasingly dependent on teams to operate complex equipment and control centers, while crew performance has become a critical concern in commercial aviation safety. When intensive work began on developing and evaluating team training for these complex real-world tasks, however, researchers found little useful theory or practical guidance available from prior research (Orasanu & Salas, 1993).

Considerable effort has been expended over the past ten years in defining and measuring the behavior of teams, focused on the goal of improving team training (Salas, Dickinson, Converse, & Tannenbaum, 1992). This team-training research has concentrated on identifying the observable behaviors associated with effective team performance in realistic settings, and linking those behaviors to trainable skills. The skills required for effective team performance fall into two broad categories: taskwork skills associated with the performance of individual team members on specific tasks, and teamwork skills (e.g., communication) associated with the overall performance of the team (Orasanu & Salas, 1993; Glickman et al., 1987; Morgan et al., 1986).

Over the last decade, team training research has benefited from technological advances in creating realistic training environments and recording data. Complex, realistic simulation environments allow researchers to observe real-world teams performing tasks that closely

approximate their actual tasks. Flight simulators, for example, allow the observation of cockpit crews as they "fly" a plane with a high degree of realism. Videotapes of crews in these simulated environments allow detailed after-the-fact analysis of both individual actions and team communication patterns linked to events in the simulated world. The availability of detailed communication data linked to events supports the definition and implementation of coordination measures and other measures of team process. For example, it is possible to measure the intelligibility of communication (whether one team member appeared to understand a communication from another team member) as well as the transfer of information and resources among team members in a realistic environment.

Simulated environments also allow the introduction of controlled variations in task load; i.e., the creation of situations that present varying levels of difficulty for the team. These variations allow measurement of the effects of task load on team performance and on the perceived workload of team members. It is also possible to measure the attitudes of team members through questionnaires and to link these attitudes to objective measures of team performance.

Research on team behavior in realistic, simulated environments is revealing complex interrelationships among task load, perceived workload, team coordination patterns, and team performance. Task load is typically an objective measure based on the resource demands imposed by external tasks; e.g., the number or the difficulty of the tasks to be completed. Perceived workload is measured through subjective rating scales such as the SWAT or TLX scales (see Lysaght et al., 1989, for a review of subjective workload measurement). Performance is measured through data collected as part of the simulation, and communication and coordination are measured through ratings based on analysis of videotapes.

Studies have found that team performance does not necessarily decline as external task load increases (Serfaty, Entin, & Volpe, 1993b; LaPorte & Consolini, 1988). One hypothesis is that effective teams *adapt* to increased task load in order to maintain their workload and performance at acceptable levels. There are a number of ways that teams may adapt to increases in their task load: they may change their decision strategies; they may change their communication and coordination strategies, or they may structurally reconfigure the team to handle the increased load.

Serfaty and his colleagues have suggested that teams may shift from an explicit to an implicit coordination strategy as task load increases (Serfaty, Entin, & Volpe, 1993b; Serfaty, Entin, Deckert, & Volpe, 1993a; Entin, Serfaty, Entin, & Deckert, 1993). Under an explicit coordination strategy, team members request resources, information, or actions from other team members. Under an implicit coordination strategy, team members anticipate one another's needs, and supply information, resources, or actions without being asked. Implicit coordination can reduce the number of communications that are required, saving valuable time for the team. Serfaty, et al. (1993a, 1993b) measure shifts between explicit and implicit coordination through an anticipation ratio—the number of information transfers divided by the number of requests for information. Higher anticipation ratios have been linked to better team performance under stress for several types of teams.

A number of researchers have suggested that the concept of *shared mental models* provides a useful framework for understanding team communication and coordination patterns (Cannon-Bowers & Salas, 1991). It has been suggested that more effective teams develop and use a shared, congruent, mental model of the situation and of their common tasks, including the team goals, the current state, and the action needed to reach the goals. Orasanu (1990) found measurable differences in the communications patterns of cockpit crews flying in a simulated environment; the communication patterns of the better-performing crews differed significantly from those of the poorer crews. Specifically, the good crews were much more explicit in defining the problem, obtaining information, and coordinating shared responsibilities. The better crews used their low-workload periods to communicate in ways that updated their shared mental model and helped them plan for contingencies. Orasanu (1990) suggests that creating an explicit shared mental model for the team ensures that everyone is solving the same problem.

Serfaty et al. (Serfaty, Entin, & Volpe, 1993b; Serfaty, Entin, Deckert, & Volpe, 1993a; Entin, Serfaty, Entin, & Deckert, 1993) suggest that the creation of mutual mental models of other team members allows an individual to develop expectations about the actions and anticipated needs of the other team members so that the team is able to rely on implicit coordination (with lower communication requirements) in times of high external task load. Note that there is some discrepancy between Orasanu's (1990) finding that higher explicit coordination was associated with better team performance, and the Serfaty et al. finding that a shift toward more implicit coordination under high task loads was associated with better team performance. The difference may be a function of changing task load. Orasanu found that better teams used slow periods for their explicit-

coordination communications, presumably reducing the need for explicit coordination during high-task-demand periods.

Theories about the role of mental models in team coordination suggest that expectations play a key role in team communication and performance. Expectations reduce the ambiguity of incoming information, reducing the amount of time needed for team members to respond under stress. These expectations are based both on a shared mental model of the situation, the tasks facing the team, and the team's goals, and on accurate mutual mental models of the tasks of the other team members, allowing an individual to anticipate other team members' needs.

For modeling teams in the OMAR framework, mental model theories suggest that it will be necessary to explicitly represent, in models of individual team members, the team's shared understanding of the situation and of their shared goals. Team members can then take individual actions based on this shared information and these shared goals. Each team member will also need to have a representation of each of the other team members—their expected actions, what they can be expected to know, what they need to know, etc. Communication can then be modeled based on discrepancies between the information available to each team member, the information needed by each of the other team members, and the availability of the needed information to the other team members; i.e., if you know something that another team member needs to know, and they probably don't know it, then you should communicate it to them. OMAR-based HPP models of individual operators already represent goals, actions to be taken, and information available. Models of teams in the OMAR environment could build on these representations to create a shared representation for the team, and, for each team member, a representation of each of the other team members.

2.2. Behavioral Test of the Psychological Model

The purpose of this task was to collect human performance data that could be directly compared with performance data for HPP models in order to assess the areas in which the models need to be refined and improved. Detailed differences in human and model process measures, as well as differences in outcome measures, indicate promising directions for increasing model fidelity. This section reports the results of an experiment conducted to collect behavioral data for comparison with the model.

The HPP models of air traffic controllers and flight crews that have been developed in the OMAR project are based on theories about the nature, functions, and limitations of

attention from the experimental psychology literature. The literature does not supply detailed guidance on how we should model attention as it affects performance in a multi-task environment such as air traffic control, however. The human capacity to "juggle" tasks, switching rapidly from one to another to respond to a changing situation without losing track of overall priorities, is not well understood. We therefore focused the behavioral test of the psychological model to further understand how we might more realistically model multi-task performance.

In Adams, Tenney, and Pew's (1994) review of the literature on the cognitive management of multiple tasks, they suggest that, because humans can devote thoughtful conscious attention to only one task at a time, the management of multiple complex tasks consists essentially of working on one task while queuing some number of others. This queue is not a simple list, however, because it must be frequently updated as the situation changes. Adams, et al. conclude that the sources of cognitive workload in multi-task management go well beyond those associated with the performance of individual tasks, and they suggest a framework for understanding multi-task workload. This framework includes 1) maintaining the "stack" or queue of to-be-attended tasks; 2) updating the status of tasks in the queue as the situation changes; 3) resolving conflicts among high-priority goals; and 4) planning the optimal points for transitioning between tasks. Rogers (1996) used a similar framework to study the mental processes involved in flight deck task management. Cognitive task management on the flight deck includes assessing the situation, identifying tasks, prioritizing tasks, assessing and allocating resources, and scheduling tasks.

For the behavioral test of OMAR operator models, we grouped the cognitive tasks associated with multi-task management into two broad areas: 1) creating, maintaining, and updating an awareness of all of the active tasks and 2) choosing actions from among these active tasks, based on overall goals and priorities. One of our primary goals in the experiment was to assess the relative importance of these two aspects of multi-task workload—maintaining an awareness of all pending tasks and choosing the next task to be performed—in multi-task performance.

From a modeling perspective, these two aspects of multi-task management seem to call for different resources and therefore should be modeled in different ways. Maintaining information about active tasks or "keeping track of everything" involves a memory load component. The operator has three alternatives: 1) maintain information in memory about all of the active tasks; 2) re-acquire information about all of the active tasks before

making a choice among them; or 3) rely on an explicit external representation of active task status such as a list or a display as a memory aid. As new information comes in, it must be used to update the list of active tasks in memory or on the display.

Choosing a next action from among multiple competing actions ("what should I do next?") is a decision-making function involving consideration of how each task relates to overall goals and priorities. The most important next action will vary as a function of time as the situation changes. While it is possible to create a "queue" of pending actions and to perform several acts from that queue without reconsidering the situation, this represents a considerable memory load and may not be an effective strategy in a fast-changing situation.

2.2.1. Method

In order to conduct the behavioral test, we designed and developed a multi-task experiment environment, designed an experiment using that environment, and collected performance data from subjects who served as operators handling multiple tasks in that environment.

2.2.1.1. Experiment Task and Environment

We created an experiment environment using the OMAR system to conduct the behavioral test. In order to test human operator versus model performance in a multi-task situation, we needed an environment that:

- requires that the operator process multiple simultaneous tasks using an automated system;
- allows switching of attention between tasks;
- provides flexibility in the order in which the operator processes tasks, allowing the operator to set priorities and use different strategies;
- provides variability in task load and time pressure; and
- supports definition of measures of effectiveness so that it is possible to define and measure a continuum of performance.

To meet these requirements, we created a simulated air traffic control task in which the operator receives messages from controllers in adjacent sectors and from incoming and outgoing aircraft and responds to those messages. The processing of an individual aircraft

through the airspace—from entry to exit—is considered a single task. The operator is required to switch attention between tasks in order to handle multiple aircraft over the course of an experiment session. The design of the task was based roughly on the capabilities of an advanced air-ground data link system for air traffic control as described by den Braven (1992).

Table 1 shows the steps required for the operator to process each aircraft. If the controller did not accept an incoming aircraft, it went into a holding pattern at the edge of the airspace until it was accepted. As soon as the plane entered the airspace, it sent a "hello" message. The controller could respond to this message, but if there was no response the aircraft continued on its flight path. If the controller did not send a release message to an outgoing aircraft before it reached the edge of the airspace, the aircraft went into a holding pattern until it was released.

The experiment environment was implemented using the OMAR system and the display was built with the MIRAGE GUI builder. The display showed a radar-like picture of aircraft location and allowed the operator to view incoming text messages and to construct and send text messages to other controllers and to aircraft. Scenarios specifying the number, location, and flight path of aircraft were created using OMAR. The human subject acted as controller for one of the ATC sectors, with OMAR models acting as controllers in the four adjacent sectors.

Table 1. Experiment Task Definition—Steps in Processing Each Task

Triggering Event	**Operator Action**
Incoming message asking controller to accept aircraft	Accept or refuse aircraft
Aircraft enters airspace and sends "hello" message	Respond with welcome message (optional)
Outgoing plane approaches edge of airspace	Send message to controller of adjacent sector asking that controller to accept the aircraft
Adjacent controller accepts aircraft	Send release message to aircraft clearing it to leave airspace

2.2.1.2. Experiment Design and Procedures

The major independent variable in the experiment was the type of support provided to the operator by the computer display. There were three display conditions:

1. An unaided condition. In this condition, all of the information about the status of the active tasks (planes in the airspace) was to be found in text messages. In order to scan the "queue" of active tasks, the operator was forced either to remember the status of each plane or to scan the list of messages.

2. A color-coded status condition. In this condition, the icons on the radar display were color coded to show the next action (accept, welcome, request transfer, release) that needed to be taken for that aircraft. If no action was needed, the icon was white. This condition provided visual memory support to the operator in remembering the next step to be taken for each plane on the radar screen. There was no indication as to which of the many pending actions was the most urgent, however.

3. A color-coded priority condition. In this condition, only the icon for the most urgent pending task was color coded to show the next action to be taken. In this condition, the display showed the operator what to do next, but did not provide any information (beyond the text messages provided in all conditions) about the status of the other aircraft.

The purpose of the display-condition independent variable was to assess the relative importance of "keeping track of things" versus "deciding what to do next" in a multi-task environment. In the color-coded status condition, the display explicitly represented the "state" of each active task; i.e., the next action to be taken. The operator is not required to remember task status or to scan text messages to acquire this information. The operator must choose among all of the possible actions that might be taken, however. In the color-coded priority condition, an explicit recommendation is made about the next action to be taken. A comparison of performance levels for the status and priority conditions with the unaided condition indicates which aspect of multi-task management was most difficult for operators in this environment. Similar performance for the status and priority conditions indicates that the "keeping track" aspect of task management constitutes almost all of the workload in this environment—once operators know which tasks need to be done, there is little additional advantage in telling them which task to do next. If performance is better in the priority condition than in the status condition, however, this indicates that there is a substantial burden in choosing among active tasks based on overall goals and priorities, and that providing suggestions about which active task to choose can improve performance.

The level of task load was an additional independent variable. Task load was manipulated by changing the number of aircraft that entered the sector during a ten-minute experiment trial. Two scenarios were developed, one representing a "medium" level of task load (19 aircraft entered or exited the sector), and the other representing a higher level of task load (29 aircraft entered or exited the sector). During pilot testing we determined that subjects did not learn or remember the details of the scenarios, so we were able to re-use the same two scenarios in all three display conditions. The addition of task load as an independent variable allowed us to assess whether there was an interaction between display design and task load; i.e., whether some designs were more effective under higher or lower loads.

The experiment used a repeated-measures design in which every subject participated in all three display conditions. Task load was fully crossed with display condition, so that all subjects completed both scenarios in all three display conditions. The order in which subjects experienced the three display conditions was counterbalanced, so that each display design appeared an equal number of times as the first, second, or third condition experienced by the subjects.

A total of nine subjects participated in the study. The subjects were BBN employees who volunteered to participate and were paid for their time. No special training or background was required. Subjects were trained for approximately 45 minutes before beginning the data-collection phase of the experiment. After completing each display condition, the subjects were asked to complete the NASA TLX workload rating scale (Hart and Staveland, 1988). They were also asked to explain how their strategy for the task changed as the display design changed. At the end of the experiment, subjects completed a questionnaire that asked them to compare the difficulty of completing the tasks under each of the three display conditions.

2.2.1.3. Measures of Effectiveness

Measures of effectiveness (MOEs) for the experiment task were defined using the OMAR template approach described below in Section 3.2. Table 2 summarizes the MOEs used in the experiment. The major measure of overall task performance is the number of aircraft that went into a holding pattern at the border of the airspace because they had not been accepted for entry or released for exit. Subjects were instructed to avoid having aircraft go into a holding pattern if possible. Many of the other MOEs measure processing delays—how long it took for operators to respond to various types of messages. We

expected that longer processing delays would be associated with poorer task performance. The number of aircraft not welcomed is a secondary-task measure of workload. We instructed subjects that sending welcome messages was optional, and aircraft did not change their behavior if they did not receive a welcome message. We expected that operators would send fewer welcome messages when they were busier.

Table 2. Measures of Effectiveness Used in Experiment

MOE	Interpretation
Number of Holding Aircraft Number of aircraft entering a holding pattern at the border of the controller's airspace	This is a task performance measure. Aircraft go into a holding pattern at the border of the airspace if the controller is unable to accept or release them in a timely manner. Subjects were instructed to try to minimize the number of times that this occurred.
Time to Accept Elapsed time between receiving a request to accept an aircraft and accepting the aircraft (mean time across all aircraft requesting entry).	This is a measure of processing delay.
Release Delay Elapsed time between receiving a message saying that the next controller has accepted an aircraft and sending a release message to that aircraft (mean time across all outgoing aircraft).	This is a measure of processing delay.
Spare Release Time Elapsed time between time that aircraft receives release to leave airspace and time that aircraft crosses airspace border (mean time across all outgoing aircraft).	This is a measure of the extent to which the operator is able to "act ahead."
Number of Aircraft Not Welcomed Number of aircraft for which the controller did not respond to "hello" messages by sending an (optional) welcome message.	This is a measure of operator overload. We expect that operators will send fewer optional welcome messages if they are overloaded.

2.2.2. Results

The results of the experiment fall into three main groups: 1) MOE results from the simulation, 2) workload results based on subjective ratings by the subjects, and 3) subjects' ratings of task difficulty and their discussions of how the display designs affected their task strategy.

2.2.2.1. MOE Results

The major measure of task performance was the number of aircraft that went into a holding pattern at the edge of the controller's airspace because they were not accepted or released in a timely manner. Table 3 shows the results for this MOE by display condition and task load. There was a large and significant overall effect of display condition. The mean number of holding aircraft in the unaided condition was 5.44; in the status and priority color-coded conditions, the mean number of holding aircraft was only 0.56 and 0.39, respectively. The two aided conditions did not differ significantly from each other in the number of holding aircraft, but both differed from the unaided condition. As expected, there were more holding aircraft in the high task load than in the medium task load scenario. There was also a significant interaction between display condition and task load. An examination of Table 3 shows that task load was an important factor in the unaided condition, increasing the mean number of holding planes from 3 to almost 8, while task load had a negligible effect on the two aided conditions. We conclude that subjects using either the status or the priority displays were able to process almost all aircraft without any holding delays, and that they were able to do so under a high task load.

Table 3. Mean Number of Holding Aircraft by Display Condition and Task Load (n=9)

	Display Condition			
Task Load	Unaided Condition	Color-coded Status Condition	Color-coded Priority Condition	**Mean** (n=27)
Medium	3.00	0.33	0.33	1.22
High	7.89	0.78	0.44	3.04
Mean (n=18)	5.44	0.56	0.39	

Significant Effects (Analysis of Variance): Display Condition, $F(2, 16)=17.72$ ($p<0.001$); Task Load, $F(1, 8)=15.67$ ($p<0.01$); Display Condition x Task Load, $F(2, 16)=11.06$ ($p<0.001$).

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<.01$). No significant difference between Status and Priority.

The timing MOEs show how superior levels of performance were achieved in the two aided conditions. Tables 4 and 5 show the acceptance delays and release delays by display condition and task load. For both types of delay, there was a large and significant effect of display condition—accounted for by the difference between the unaided

condition and the two aided conditions. The difference is especially striking for the acceptance delay, where the color-coded information cut the mean delay time from 34.8 seconds to 13.3 and 14.6 seconds. The two aided conditions appear very similar. Operators were able to act more quickly to accept and release aircraft when they used color-coded information about either status or priority.

Table 4. Mean Acceptance Delay (in seconds) by Display Condition and Task Load

	Display Condition			
Task Load	Unaided Condition	Color-coded Status Condition	Color-coded Priority Condition	**Mean** (n=27)
Medium	35.2	12.5	13.9	20.5
High	34.5	14.0	15.2	21.2
Mean (n=18)	34.8	13.3	14.6	

Significant Effects (Analysis of Variance): Display Condition, $F(2,16)=16.58$ ($p<0.001$)

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<0.01$). No significant difference between Status and Priority.

Table 5. Mean Release Delay (in seconds) by Display Condition and Task Load

	Display Condition			
Task Load	Unaided Condition	Color-coded Status Condition	Color-coded Priority Condition	**Mean** (n=27)
Medium	16.0	10.8	9.6	12.1
High	21.8	12.0	11.0	14.9
Mean (n=18)	18.9	11.4	10.3	

Significant Effects (Analysis of Variance): Display Condition, $F(2,16)=10.50$ ($p<0.01$)

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<0.01$). No significant difference between Status and Priority.

A consistent difference also appears for the "spare release time" MOE, which measures the elapsed time between the time that a plane received a release and the time that it crossed the airspace border. These results are shown in Table 6. The spare time MOE measures how close the aircraft came to reaching the border without receiving a release; i.e., how close it came to going into a holding pattern. Operators in the two aided conditions released planes with about 60 seconds to spare, on average, while the same

operators in the unaided condition achieved only 49 seconds to spare on average. There is also a significant interaction between display condition and task load. Examination of Table 6 shows that operators released planes earlier under lower task loads than under higher task loads in the unaided condition, while task load had little effect on spare release time in the aided conditions.

Table 6. Spare Release Time (in seconds) by Display Condition and Task Load

	Display Condition			
Task Load	Unaided Condition	Color-coded Status Condition	Color-coded Priority Condition	**Mean** (n=27)
Medium	53.9	63.0	60.8	59.2
High	44.7	60.8	62.8	56.1
Mean (n=18)	49.3	61.9	61.8	

Significant Effects (Analysis of Variance): Display Condition, $F(2,16)=7.40$ ($p<0.01$); Display Condition x Task Load, $F(2,16)=3.92$ ($p<0.05$)

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<0.01$). No significant difference between Status and Priority.

Table 7. Mean Number of Aircraft Not Welcomed by Display Condition and Task Load

	Display Condition			
Task Load	Unaided Condition	Color-coded Status Condition	Color-coded Priority Condition	**Mean** (n=27)
Medium	2.11	0.00	0.33	0.81
High	6.43	0.89	3.25	3.52
Mean (n=18)	4.27	0.44	1.79	

Significant Effects (Analysis of Variance): Display Condition, $F(2, 16)=20.09$ ($p<0.001$); Task Load, $F(1,8)=32.8$ ($p<0.001$); Display Condition x Task Load, $F(2, 16)=4.77$ ($p<0.05$)

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<0.01$). No significant difference between Status and Priority.

A final MOE is the number of times that operators did not perform the optional task of welcoming planes to their airspace. We expected that, as task load increased, operators would drop this task to concentrate on accepting and releasing planes so that planes did not go into a holding pattern. Table 7 shows the results for the number of aircraft that

were not welcomed. The was a large and significant effect for display condition, with many more planes welcomed in the aided conditions. For this MOE, the pattern was somewhat different for the status and priority conditions, however. The lowest mean was in the status condition, although the Tukey test did not show a significant difference between the Status and Priority conditions. The mean in the priority condition is probably somewhat higher because welcoming planes was always the lowest priority task, and was suggested as the next task only if there were no other pending tasks. There was a significant main effect for task load, with more planes welcomed in the lower task load condition. There was also a significant interaction effect, accounted for by the larger effect of task load in the unaided condition.

Table 8. Mean Workload Ratings by Display Condition

	Display Condition		
Rating Scale Description (Scale from 1 (best) to 10 (worst))	**Unaided Condition**	**Status Condition**	**Priority Condition**
Mental Demand How mentally demanding was the task?	7.69	4.08	4.92
Physical Demand How physically demanding was the task?	4.75	3.81	3.75
Temporal Demand How hurried or rushed was the pace of the task?	7.53	4.31	4.97
Performance How successful were you in accomplishing what you were asked to do?	4.69	2.58	3.19
Effort How hard did you have to work to accomplish your level of performance?	7.53	4.19	4.03
Frustration How insecure, discouraged, irritated, and annoyed were you?	6.69	3.08	3.08
Mean Total	6.48	3.68	3.99

Significant Effects: Tukey test for pairwise contrasts in Display Condition: Unaided Condition significantly different ($p<0.01$) from Status and Priority Conditions for Mental Demand, Temporal Demand, Effort, Frustration, and Total ratings; no significant difference between Status and Priority Conditions for any ratings.

2.2.2.2. Workload Results

After subjects had completed a display condition, we asked them to rate their perceived workload using the TLX workload rating developed by NASA (Hart and Staveland, 1988). The TLX workload rating has six components that rate different aspects of workload. Table 8 shows the mean score for each rating component by display condition as well as the sum of the six ratings.[1]

The pattern of workload-rating results is quite similar to the pattern of MOE results. Subjects felt that they worked harder in the unaided condition. Four of the six rating scales (mental demand, temporal demand, effort, and frustration) were significantly higher for the unaided condition than for the status and priority conditions. There were no significant differences between the status and priority conditions on any of the scales. It is not surprising that subjects did not perceive differences in physical demand between the conditions—the physical tasks of scanning the screen and moving the mouse did not change. It is surprising, however, that subjects did not perceive differences in their performance in the different conditions given the large differences in performance that were measured by the MOEs. Perhaps subjects felt that they were doing as well as they could, given the display with which they were working, and did not "downgrade" themselves for the larger number of unprocessed planes that were forced to go into a holding pattern in the unaided condition.

2.2.2.3. Difficulty Ratings and Strategy Descriptions

At the end of each experiment session, we asked subjects to rate the difficulty of accomplishing the air traffic control task using each of the three displays. Table 9 shows the mean ratings for each condition. As expected, subjects rated the difficulty of accomplishing the task as much higher in the unaided condition than in either of the aided conditions. There was no significant difference between the difficult ratings for the status and priority conditions. Although the mean difficulty rating was somewhat lower for the status than for the priority condition, the difference is too small to be significant with a small sample size.

[1] Full application of the NASA-TLX methodology involves having subjects make comparative ratings of how well each component applies to the task being rated, and using these scores to weight the components. Because perceived workload was not a primary focus of our study, we did not perform this weighting activity and simply present the unweighted total of the components.

When subjects switched from one display condition to another, we asked them whether their strategy for the task changed when they used a different display. Three switches in condition could occur: 1) unaided to status, 2) priority to unaided, and 3) status to priority. Subjects' descriptions of their strategy changes are shown in Tables 10, 11, and 12 for these three switches.

Table 9. Mean Difficulty Ratings for Accomplishing Task by Display Condition

	Display Condition		
	Unaided Condition	**Status Condition**	**Priority Condition**
Mean Difficulty Rating (n=9) (Scale: 1=very easy to 10=very difficult)	8.3	3.1	4.1

Significant Effects (Analysis of Variance): Display Condition, $F(2, 24)=25.23$ ($p<0.001$)

Tukey test for pairwise contrasts in Display Condition: Unaided significantly different from Status and Priority ($p<0.01$). No significant difference between Status and Priority.

Table 10 shows how subjects described the change in their strategy when they switched from the unaided to the status display. In general, subjects said that they stopped using the text message display and relied on the color coding to tell them the status of each plane. They reported prioritizing the aircraft based on the color coding.

Table 10. Changes in Strategy When Switching from Unaided to Status Condition

What did you do differently?
Disregarded message console to rely on color code.
In Condition 2 (Status) I paid no attention to the "Message History" list but looked only at the icon color coding. In Condition 1 (Unaided) I was constantly referring to the list.
I ignored the text completely. I almost always waited for the symbol to change color before taking any action.
When I knew the colors would persist, I favored hand-offs over acceptances. Both acceptances and welcomes were much, much easier in terms of finding the guilty aircraft.
I didn't use the message history at all.
Prioritized aircraft that displayed colors.

Table 11 shows subjects' comments when they changed from the priority condition to the unaided condition. Subjects reported that, without the color coding, they needed to rely on the message history, which they had previously ignored. Several subjects mentioned having to "remember more" in the unaided condition.

Table 11. Changes in Strategy When Switching from Priority to Unaided Condition

What did you do differently?
For the first two conditions (Status and Priority) I tended to ignore the written log of communications. For this condition (Unaided) I had to read and decode the messages. I spent more time scanning the field and trying to remember what I had already dealt with.
You need to look at the messages a lot more and try to remember aircraft status a lot more.
Constantly looking back and forth between messages and icons. Very difficult.
Although I tried to anticipate requisite action, I did depend on the colors as a reminder when I missed something. It was more relaxed having the safety net, but I did find that using that net left me behind the curve. The colors were most missed in trying to locate inbound traffic.
For Condition 1 (Priority) I did not look at the message history. For Condition 2 (Unaided), I used the message history exclusively.
Paid more attention to message display and "border area"

Table 12. Changes in Strategy When Switching from Status to Priority Condition

What did you do differently?
I went back to read the messages to combine with color code.
I did not change procedure.
I infrequently looked at the text messages to see if I missed anything.
I tended to wait until something changed color before dealing with it. I relied more on the system's judgment of priority. This left me feeling like I was running to catch up more of the time.
Had to remember a lower priority task to finish if it was interrupted by a higher priority task. I would finish a lower priority task if I started to think about it first.
Had to read the messages more.

Table 12 shows subjects' comments about switching from the status condition to the priority condition. These comments are especially interesting because they seem to reflect some preference for the status condition over the priority condition, even though there were no significant differences in performance or workload in the two conditions. One subject reported no change in strategy. Several others reported more reliance on the text

messages to determine the status of aircraft not currently color coded in order to "to see if I missed anything." One subject mentioned relying more on the system's judgment of priority and "running to catch up more" in the priority condition.

2.2.3. Discussion and Implications for the Model

The results of the experiment show clearly that "keeping track of everything," not "deciding among alternative actions" was the predominant source of cognitive workload in the multi-task environment studied. Providing a visual status display of pending actions and letting the operator select the next action to be taken improved performance just as much as providing specific suggestions on which action to take. The difficult part of the simulated air traffic control task was in maintaining and updating information on all of the pending actions, not in choosing among those actions. In fact, there is some suggestion that the status display may be superior to the priority display. Although the difference is too small to be significant with a small sample size, subjects' ratings of the difficulty of the task were somewhat lower for the status than for the priority displays. Also, when they changed from the status to the priority display, subjects reported needing to make more use of the text messages in order not to miss anything, needing to remember their own priorities, and feeling more dependent on the system's priority judgments.

The experiment results have implications for the design of displays and decision support systems for multi-task situations. It appears that supporting situation awareness is more critical than supporting prioritization of tasks. Operators trying to "juggle" multiple tasks need memory support to keep track of changing task status more than they need advice on which tasks are most important. Of course, any generalization from the experiment results is constrained by the details of the tasks that were studied. The experiment tasks were relatively simple, and they were all alike; i.e., the sequence of actions to be taken was the same for every aircraft. Studying the process of juggling different and more cognitively complex tasks—for example, task management by a pilot on the flight deck—might yield different results.

The experiment results suggest that refinements in the OMAR model for multi-task environments should focus on improving the realism of the process by which the model acquires, stores, and updates information about changing task status, not on improving the process by which the model chooses among actions based on overall goals. The OMAR model for the air traffic control task, as currently implemented, functions as

though it has perfect memory for task status and pending actions. Incoming information is used to update the set of pending actions, and the model does not need to refresh or re-acquire that information unless the task status changes. Essentially, the model currently functions in the same way that human operators can function if they have external memory support for situation awareness—the model always has the correct status information immediately available. This produces unrealistic model performance in the unaided condition, where human operators were forced to expend considerable time and effort to acquire and maintain an accurate picture of the status of all of the competing tasks.

3. IMPROVEMENTS TO THE OMAR ANALYSIS ENVIRONMENT

3.1. The Post-run Analysis Framework

In the OMAR system, it is possible to stop a scenario at any point during a run to examine its current state: for example, to view the event or procedure timelines, or to view signal connectivity in the SCAN display. It is also desirable to perform more detailed, less-time critical analyses of completed runs at a later time. To support such analyses, the OMAR system now provides the Post-run Analysis Framework. In the OMAR system, data from a run, in the form of records of simulation events, can be selectively collected during the execution of an OMAR scenario. This data can now be saved to a file. The resulting time-ordered data-sets can then be reloaded using the Post-run Analysis Framework and viewed using a subset of the analysis windows from the standard set of OMAR analysis tools.

3.1.1. Saving the Data from a Scenario Run

While an OMAR scenario is running, data is being collected in the form of records for selected simulation events, events that are recorded on-line during scenario execution. This data can be collected to a disk file to support later off-line analysis by using the "Pause" button on the "OMAR Simulator Control" frame to pause the simulator and then using the menu-item *Save to File...* from the "Scenario" menu on the "OMAR Simulator Control" frame. At this point the user is prompted for a pathname for the file in which to save the data. By default, the name of the file contains the name of the scenario and the directory for the file is the user's home directory. A notification will appear telling the user that the data has been saved to the file.

3.1.2. Recovering The Data From A Stored Scenario

To restore the event data from a previous run, the "OMAR Scenario Analysis" window may be selected by clicking on the "Analysis" menu on the main OMAR toolbar (or by selecting the *Analysis* option on the "Windows" menu on any of the OMAR frames.)

Note that the toolbar on the "OMAR Scenario Analysis" frame is a different color (brown) from that of the other OMAR windows. The toolbars of the post-run analysis windows—discussed below—are also brown. This is to distinguish the analysis windows—which contained fixed data for review at "post-run" time—from the standard OMAR windows—which contain dynamic data and can be used at "run-time."

The data-file of a particular stored scenario may be selected using the *Restore Scenario...* option from the *Analysis* menu on the **OMAR Scenario Analysis** frame. The user will then be prompted for the pathname of the file of scenario data which is to be restored. When OMAR has completed loading the data for the restored scenario, the name of the scenario will appear in the main pane of the **OMAR Scenario Analysis** frame. Clicking on the name of a restored scenario will cause the displays for that scenario to appear.

A set of two displays is provided for each restored scenario: a SCAN Display and an Agent Task Timeline frame. These displays are almost identical to the similar displays in the main OMAR system, however; they differ in the following ways:

- The displays for a restored scenario that contain the "static data" for the restored scenario. The data cannot be used as the basis for continuing a simulation run and is not affected in any way by the scenario currently running in the run-time OMAR system.

- In addition to the standard toolbar menus which appear in the version of these displays in the main OMAR system, these displays also contain a "Scenario" pulldown menu containing commands which are specific to the corresponding restored scenario. (For example, the "Scenario" menu contains commands to show the partner display for the scenario, to hide both displays for this scenario, to hide the displays for all other restored scenarios, etc.)

- As mentioned above, the toolbars of the displays for the restored scenarios are a special color (brown) to set them apart from the similar run-time windows.

In all other aspects, these displays appear and behave like the corresponding displays in the main run-time OMAR system.

3.2. Measures of Effectiveness

A key factor in developing HPP models is assessing the performance of the models on specific tasks. Two distinct but related types of assessment are required:

- An assessment of how well the model performs the task, measured against some absolute criteria for successful task performance, and
- An assessment of how well the model's behavior matches the behavior of human operators performing the same task.

The two types of measures are independent. A model may perform a task with a high degree of success, but in a way that does not resemble human performance and therefore does not predict human performance levels on the task. Both types of assessment require objective, quantitative measures of task performance. We must be able to measure the "success" of both the human and the model in performing a task, where success is defined by how well the human-machine system is meeting its overall goals.

The development of "templates" for "standard" measures of effectiveness (MOEs) for HPP models was a major task for the OMAR project. Our goal was to provide tools for defining MOEs as part of the process of setting up scenarios and experiments, enabling the modeler to easily capture and analyze MOE data from model runs and from experiments with human operators. MOEs are defined as objective, quantitative measures that bear a direct relationship to the overall goal of the task, system, or mission being analyzed.

The development of templates for standard MOEs represents a substantial challenge because the meaning of "success" in a task can be different for different systems and different missions. As Meister (1985) comments, "The context of the data is necessary to understand objective data. That context is a task or job. The frequency with which a switch is thrown is uninterpretable unless we know the task for which throwing a switch is a subtask."

Even though the interpretation of MOEs is domain dependent, the building blocks for MOEs are the same for any task or system. MOEs are built from data about *events* that occur over time. These events may be human (or HPP model) actions, actions by the

automated system (or simulations of those system actions), or actions that occur in the external world (or simulations of those actions in the external world). MOEs may be built from the *relationships* among different types of events; e.g., a human action that follows an event in the external world. Also, the *times* that events occur is almost always a critical factor in defining MOEs.

These three factors—events, relationships between events, and times—are the "raw material" for defining MOEs. Using these factors, we developed a typology for defining MOEs according to the data used to produce them and we defined a generic set of MOE "types" based on the underlying data. We developed a data collection tool that enables the analyst to easily define specific MOEs within each type. Thus, the tool supports the collection of meaningful data for a variety of missions and tasks, even though the interpretation of a specific MOE depends on the domain and context.

MOEs fall into two basic types: *frequency* measures and *duration* measures. Frequency-based MOEs measure the number of times that an event occurs, while duration MOEs measure the amount of times that elapse between two events. Each type can be further differentiated by the origin of the event(s)—the human operator, the automated system, or the external world.

3.2.1. Frequency Measures

Frequency measures are based on counts of events. Frequency counts may be differentiated by *actor*—who originates the action—or by *time period*—when the event occurs, either in an absolute sense or relative to other events. Frequency by actor involves a count of the number of times that a specific action was taken by a human operator or by the automated system, or the number of times that an event occurs in the external world. Frequency by time period measures the number of times that specified actions occurred within a specified time period.

Measurement periods for frequency MOEs may be absolute (e.g., how many planes are cleared to land within a 10-minute period) or they may be relative to other events (e.g., how many planes cross over an airspace border without being cleared to cross by the ATC). The time periods within which the count is taken may be defined based on actions taken by the human (e.g., how many times did Operator 1 acknowledge Operator 2's messages), actions taken by the system (e.g., how many times did the operator respond to a system alarm) or to actions taken by the external world (e.g., how many times did the operator send a message initiating contact when a plane entered his/her airspace).

Some event counts may be measures of *success or failure*; i.e., some actions may be defined to be "correct," while others are incorrect, and we will want to measure how many times the correct action occurred. "Correctness" may depend on information about the operator (e.g., how many times did Operator 1 correctly acknowledge a message from Operator 2 by sending a reply to Operator 2), the system (e.g., how many times did the operator select the correct button to respond to the system's alarm message), or the state of the world (e.g., how many times did the operator/system successfully intercept incoming aircraft).

3.2.2. Duration Measures

Duration MOEs measure the amount of time that elapses between two events. The beginning and the end of the time period are defined by events that involve actions by the human operator, the system, or the external world. Five possible types of duration measures involve the human operator:

- *Human-human duration MOEs* measure the elapsed time between two human actions (e.g., the time elapsed between the time that an ATC operator places an aircraft in a holding pattern and the time the operator releases it from hold).

- *Human-system duration MOEs* may be thought of as system response times—the elapsed time between the time that an operator takes an action and the system takes an action in response.

- *System-human duration MOEs* may be thought of as operator response times—the elapsed time between the time that a system event occurs and the time that the operator responds (e.g., the amount of time before the operator responds to a system alarm).

- *External world-human duration MOEs* capture the amount of time that elapses between the time that an event occurs in the external world and the time that the operator takes an action in response to that event (e.g., the elapsed time between an aircraft crossing a sector boundary and the operator responding to that aircraft).

- *Human-external world duration MOEs* capture the time between an action taken by the operator and an event in the external world (e.g., the elapsed time

between an ATC giving an operator clearance to leave a sector and the time that the aircraft crosses the sector boundary).

Like frequency measures, duration measures may also be linked to success or failure; i.e., correct responses defined in relationship to other events. Duration measures of success include the amount of time required to take a correct action in response to an event, where "correct" has been defined for the specific task or domain being analyzed. Duration measures may be used as measures of success if there are limits on the amount of time available to take an action in response to an event, or if, in general, faster is better.

3.2.3. MOE Examples from Behavioral Test of the Psychological Model

The experiment discussed in Section 2.2 uses a number of MOEs drawn from the typology described above. Examples of MOEs from the experiment serve to clarify the typology as it applies to a specific domain and task. The experiment is based on an air traffic control task, with the operator acting as a controller for an airspace sector receiving messages from and sending messages to other controllers and aircraft. Aircraft must be accepted by the controller before they can enter the airspace, and must be accepted by the next controller and cleared to leave before they can leave the airspace. As an optional task, the controller can send a welcome message to an aircraft in response to a hello message from an aircraft as it enters the airspace.

The MOEs used in the experiment include:

1. The number of outgoing planes that reach the boundary of the controller's airspace without being cleared to proceed to the next sector. This is a frequency measure of success, defined by two events: 1) the aircraft reaches the boundary of the airspace (an external world event) and 2) the controller has sent (or not sent) a message clearing the aircraft to proceed to the next sector (a human operator event).

2. The delay between the time the outgoing aircraft reaches the boundary of the airspace (and goes into a holding pattern) and the time that the controller sends a message to proceed. This is a duration measure defined by two events: 1) the aircraft reaches the boundary of the airspace (an external world event) and 2) the controller sends a message clearing the aircraft to proceed to the next sector (a human operator event). The MOE is further defined by a

relationship—the aircraft has reached the border before the clearance is received.

3. The time that elapses between the time that an outgoing plane receives a clearance and the time that it reaches the border. This is a duration measure, defined by the same events as the previous MOE but in a different relationship.

4. The delay between the time that the controller receives a message from another sector saying that they will accept an outgoing plane and the time that the controller sends a message to the plane clearing it to proceed. This is a duration measure defined by two events: 1) the controller of the adjacent airspace sends an acceptance message (a human operator event) and 2) the controller sends a clearance message to the aircraft (a human operator event).

5. The number of incoming planes that reach the boundary of the controller's airspace without being accepted by the controller. This is a frequency measure of success, defined by two events: 1) the aircraft reaches the airspace border (an external world event) and 2) the controller accepts the aircraft (a human operator event).

6. The number of times that the controller sends a "welcome" message to incoming aircraft in response to their message as they enter the airspace. (This is an optional task in the experiment.) This is a frequency MOE, defined by the relationship between two events: 1) the aircraft sends a message that it is entering the airspace (a human operator event) and 2) the controller sends a welcome message (a human operator event).

3.2.4. OMAR Support for MOE Data Collection

Three factors have been identified as the essential elements of MOEs: events, the relationship between events, and times. To adequately capture MOEs within an OMAR simulation, these factors had to be addressed. The Simulation Core (SCORE) language, the procedural language within OMAR, is the basis for defining agent behaviors, and hence is the generator of events in a scenario. The events may be HPP-agent events, target system events, or outside world events. SCORE, from its very inception, has had formal data recording and post-run processing capabilities know as the *recorder* and the *travelers* based on the work of Manning (1987). The recorder is facilitated by the SCORE

form *defevent* for defining event classes and the form *record-event* to save the data related to a particular event. It is the *recorder* which is the basis for the extensions to OMAR to meet the requirements of collecting MOEs that are described here. The SCORE forms *defevent* and *record-event* and these new extensions form the templates to easily set up the capture of standard MOEs in OMAR.

The capture of events themselves and the time of their occurrence is just what *record-event* does in OMAR. Events can be defined by type as specializations of basic class *sim-event*. The SCORE form *defevent* is provided for this purpose. When events are recorded during a simulation run using the form *record-event*, the principal items saved are the agent executing the event and the time of the event's occurrence. In defining the event class, additional slots may be created, which may then be used at record time to save data related to the event type.

The new capabilities established for MOE collection were concerned with the relationships between events and the time intervals between related events. The principal relationship of concern is just the stimulus-response relationship. An event of a particular type generated by a particular agent occurs, and shortly thereafter, a particular response by the same or another agent is expected. The approach taken to meet this data collection requirement was to implement two new event types: *stimulus-event* and *response-event.* Like all other event types, these are based on the basic simulation event class *sim-event.*

The *stimulus-event* class was defined with two additional slots: *object* and *action.* They are used to identify what the event was and which agent generated the event (e.g., the oil pressure light turned on). Stimulus events differ from other events in that they are stored in the *stimulus-event-table*, a hash table of stimulus events stored by event type.

The *response-event* class, the second new event type, was designed to look back over previous stimulus events to identify and record the particular stimulus event that it was traced back to. A general-purpose matching function, *match-stimulus,* was defined for matching a new response event to a previous stimulus event from the hash table. The *record-event* form for the response event includes the *object* and *action* that it expects to find in the stimulus event. When a match is found, the response event records that stimulus event as the event that it was triggered by, and deletes the stimulus event from the hash table. The time interval between the events is computed and also saved with the response event. Response events, like stimulus events, have *object* and *action* slots so they may be identified during post-run processing.

Stimulus events and response events can take many forms and the relationships between them can be quite complicated. For example, it might be important to record each ring of a telephone as an individual event. In this case, the *match-stimulus* function described above would not function properly when the telephone rings more than once before it is answered. To address this problem, special-purpose matching functions may be written and identified as the particular matching function for a particular response event. The slot *match-method* identifies the matching function to be used for matching the response to a single stimulus event, or perhaps more than one stimulus event as in the telephone example, for a particular response event. By default, the *match-method* slot points to the *match-stimulus* function.

Since stimulus events and response events have been built within the framework of the standard OMAR event and analysis processing, the features of this framework apply to these new event types as well. Existing interactive menus enable the user to control the printing and recording of these new event types, and save and restore functions have been provided so that these event types may be saved to a disk file and restored in the new Analysis framework. They may also be selected for viewing in the Event Timeline display. This display is a particularly effective way to gain insight to the behaviors of these event pairs during a simulation run.

3.3. The Graphical Holon Analysis Tool and the SCAN Display

A great deal of information can be collected during model runs in OMAR. Most of this information is recorded in the form of "event records", or a record of a scenario event. The user can specify which events are to be recorded (based on type of event, the associated agent, etc.). When an event of a specified type occurs , a record of the event is made, recording event-time, triggering agent and procedure, and any parameters associated with the event type. The current "event history" state is available at any point during the run of an OMAR scenario. Moreover the collection of event-data can be stored to a file on disk for further analysis.

Signal events form a special class of events. They record the generation of a signal by the procedure of an agent and the procedures of that agent and other agents that respond to the signal. The connectivity of the agents of a scenario and their procedures through signal passing can form a complex network. In particular, it is this "network" of relationships among the various components in which much of the most important information about the behaviors of the agents of a scenario is stored. That is, in order to

understand what the agents of a scenario are doing, these questions of "connectedness"—which agents communicate with which other agents and in which ways—must be answered.

Holon models (Young, 1992) represent a particular approach to human performance modeling in which signals play a central role. The agents of the model, the holons, each may belong to one or more hierarchies of holons with a complex array of signals passing between them—they are heavy users of the signal passing capabilities of OMAR. And, indeed, holon models were the motivation for the development of a display mechanism to enable the analyst to visualize the pattern of connectivity associated with signal firings in an OMAR scenario.

While trying to understand what has occurred during the run of an OMAR model, the analyst needs to be able to answer the following questions:

Questions of "Connectivity":

- Which components communicate with which other components and in what ways?
- For example, does "Agent X" ever communicate directly with "Agent Y"?
- If so, which procedures in Agent Y are involved?
- With which procedures do "vision procedures" (for example) of Agent X communicate?
- Which Agents communicate only with themselves and with no other agents?
- Are any signals "transmitted" but never "received"?

Questions of "Time":

- How does the behavior of the components change over time?
- During what period of time is a given agent busy?
- How does an agent's "workload" change over time?
- When is a given procedure or set of procedures used? For example, only during initialization? Or is a behavior repeated throughout the model-run at a consistent rate?

Questions of "Causality":

- How does a given signal affect the subsequent behavior of the model?
- What are the (functional) consequences of this signal? For example, what other signals are transmitted as a result?
- Does a signal transmitted by "Agent X" cause "Agent Y" to consequently communicate with "Agent Z"?

Questions of "Clustering" or "Patterns":

- What "set" of behaviors or signals occur repeatedly or in recognizable patterns?
- Can "higher-level" patterns be seen to emerge from the behavior of the model?

Detail/Specific Information:

- In many instances it is important to know the precise context from which the signal was initiated.
- What were the conditions which determined that a signal should be transmitted?
- Which "arguments" determined how and by which procedure the signal was received?
- What is the exact computer code associated with this signal and its transmission or reception?

In short, the OMAR system provides a rich and powerful tool set for modeling human behavior. But one feature of this power is that the resulting data can be both large in scale and complex in structure. Yet, to be useful, this wealth of data needs to be presented in a compact form, understandable by the analyst. As a result, efforts have been made in the OMAR system to supply tools that aid the analyst in understanding and navigating this data.

3.3.1. Design Criteria

The Scenario Connectivity Analysis Network (SCAN) display (see Figure 2) has been designed to present the network of signal connectivity in a meaningful and useful way. In the SCAN display, the signal-connectivity of the agents and their procedures are graphed adjacent to a "timeline" which graphically depicts the specific instances of signal transmission and reception and when they occur. The diagrams in the SCAN display are laid out to take advantage of the natural pattern recognition of the viewer's eye. In this way, patterns inherent in the data become more easily apparent.

(NOTE: The use of color is important in the SCAN display. Furthermore, to enhance this use of color, the SCAN display is drawn against a black background. However, this paper is being reproduced in gray scale and it is being drawn on a white background. Consequently, it is important to note the "false color" of the figures as presented for this discussion.)

In an effort to enhance a more qualitative "high-level" overview of the data, the SCAN display was designed to be very "graphics-intensive". That is, in an effort to present patterns detectable by the eye, the data is graphed in a highly schematic, uncluttered way.

Furthermore, text on the displays --especially in the form of descriptive labels-- has been minimized.

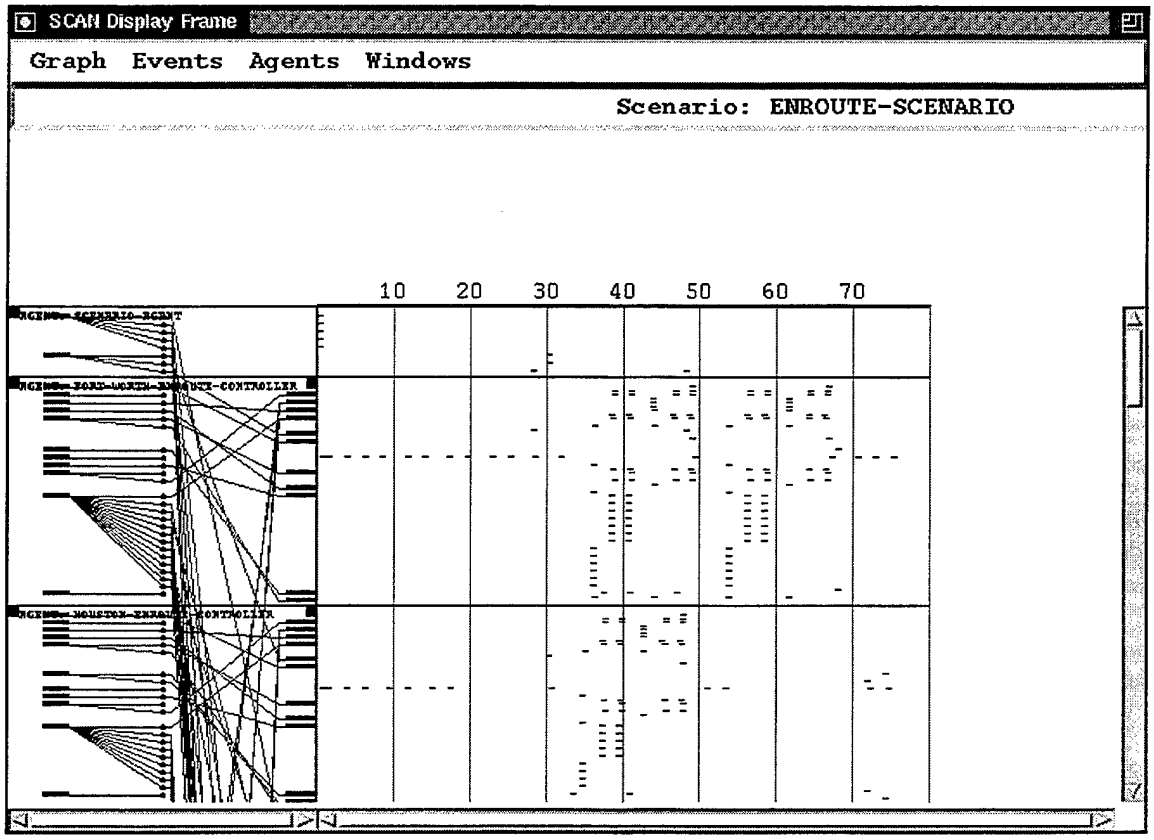


Figure 2. The SCAN Display

While it is desirable to reduce these labels, the information contained in such labels is nonetheless extremely useful. In order to present this information in a much more compact manner, a "mouse-documentation pane" has been used in the SCAN Display. In this way, detailed information about specific entities on the displays are presented in this documentation pane when the mouse is pointed at the object while minimizing the amount of textual clutter that appears in the display.

In a similar way mouse "highlighting" was used extensively in the SCAN display. That is, by pointing at an item in one of the displays, paths of connectivity or other associated information (for example, associated procedures or other representations of the same object on another display) are highlighted graphically.

Finally, a small palette of colors have been used to depict consistently and distinctly the various classes of objects shown throughout the SCAN display.

3.3.2. The Connectivity Graph

On the left side of the SCAN Display is the Connectivity Graph which shows the graph of signal-connectivity among the various agents and the procedures governed by those agents.

Figure 3 is a complex graph laid out in a more traditional graph-like display. Such graphs can be quite useful, especially if there are few nodes in the graph. However, as the number of nodes grows and the connectivity among the nodes becomes increasingly complex, such displays can become increasingly cluttered, quickly making it all but impossible for the eye to discern even the simplest patterns. And while the problem of such clutter can be helped to some degree by allowing the nodes to be laid out "by hand", for a large number of nodes such a "hand-crafted" solution quickly becomes impractical.

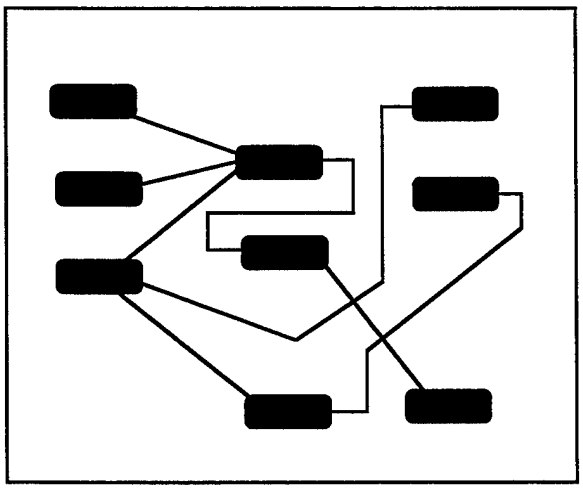


Figure 3. Stand Graph Display

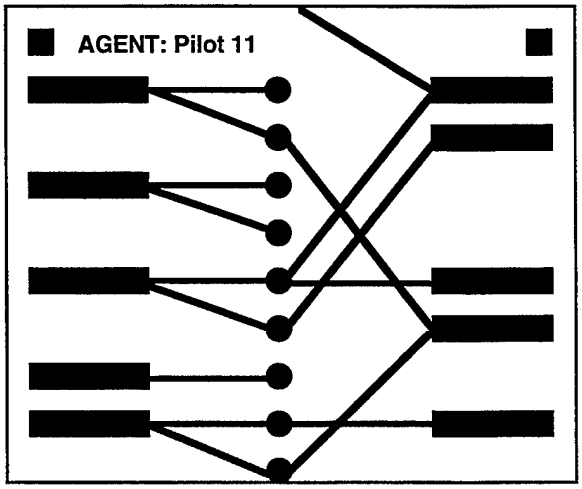


Figure 4. Agent Display

An even more critical problem remains. In such standard graph-displays it is extremely difficult to display the passage of time in any meaningful way. Nodes and arcs can, for example, be made to flash to depict the transmission or receiving of a signal, but it is very hard to extract qualitative information from such transient displays or their patterns.

With these considerations in mind, the Connectivity Graph of the SCAN Display was designed. The details of the Connectivity Graph are explained below.

3.3.2.1. Agent Display

First, as shown in Figure 4 each horizontal band across the Connectivity Graph corresponds to a single Agent and its associated Procedures.

3.3.2.2. Procedures Which Transmit Signals

Highlighted in Figure 5 are a series of blue rectangles along the left side of the display. Each of these blue rectangles corresponds to a single procedure (in the current agent) which has transmitted a signal during the course of the current model-run. (Note that these procedures are displayed in alphabetical order.)

Moving the mouse over one of the blue rectangles will cause all the information associated with signals transmitted from this procedure to be highlighted; i.e., the signal-arc (from transmitting-procedure to receiving-procedure) and the corresponding signal-instance in the Timeline Display (see below). (Furthermore, detailed information about the corresponding procedure will appear in the Mouse-Documentation Pane.) Finally, note that each procedure can submit more than one type of signal.

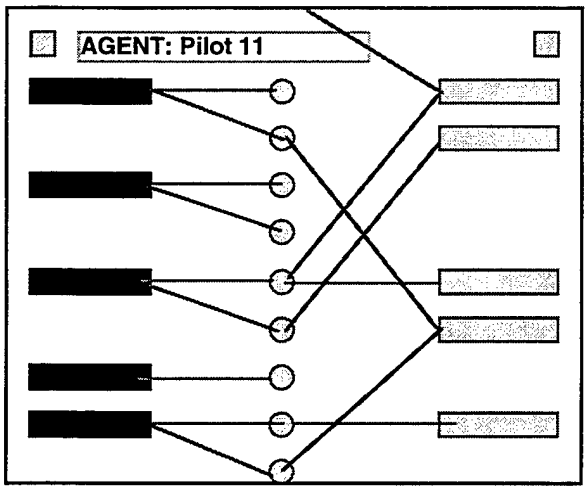


Figure 5. Transmitting Procedures

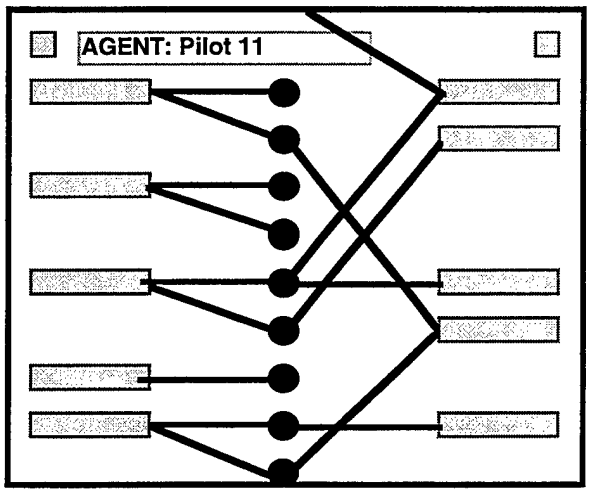


Figure 6. Transmitted Signals

3.3.3. Transmitted Signals

Aligned vertically down the center of the display (see Figure 6) is a row of yellow circles. Each of these points corresponds to a specific signal connecting two procedures which have occurred during the current model-run. Arcs are drawn from the yellow circle to the respective transmitting and receiving procedures. These arcs appear in their normal (un-highlighted) state as light gray.

Moving the mouse over a yellow signal-circle will cause the all relevant information about this signal to be highlighted; i.e., the signal-arc (from transmitting-procedure to receiving-procedure) and the corresponding signal-instance in the Timeline Display (see

below). (Furthermore, detailed information about the signal type will appear in the Mouse-Documentation Pane.)

Note that a single arc corresponding to each signal consists of two straight line-segments which are distinctive and are easily followed by the eye, especially when the signal-arc is highlighted.

3.3.3.1. Procedures Which Receive Signals

Highlighted in Figure 7 are a series of red rectangles along the right side of the display. Each of these red rectangles corresponds to a single procedure (in the current agent) which has received a signal during the course of the current model-run.

Note that these procedures are displayed in alphabetical order. Furthermore, a procedure in the current agent that both transmits signals (a "blue rectangle") and receives signals (a "red rectangle") will be located at the same vertical position.

Moving the mouse over one of the red rectangles will cause all the information associated with signals received by this procedure to be highlighted; i.e., the signal-arc (from transmitting-procedure to receiving-procedure) and the corresponding signal-instance in the Timeline Display (see below). (Furthermore, detailed information about the corresponding procedure will appear in the Mouse-Documentation Pane.)

3.3.3.2. Agent Signal Displayer

At the upper left corner of the Agent's display is a small blue square. Moving the mouse over this square will cause the information for *all* signals which are transmitted by procedures by this agent to be highlighted (see Figure 8). This is effectively equivalent to simultaneously placing the mouse over all the blue rectangles in Agent's display.

Similarly, in the upper right corner is a small red square which will perform the corresponding function for all the signals received by this Agent's procedures.

3.3.3.3. "Orphan" Signals

At the bottom of the list of Agents is a display labeled "AGENT: ORPHAN". This corresponds to all signals which were transmitted but which were not then "received" by any procedure.

Note in Figure 3 the signal which has been transmitted by the second procedure from the bottom but which is not received. (This is depicted by a yellow "signal-circle" which has an "incoming" signal-arc on the left but no corresponding "outgoing" signal-arc on the right.)

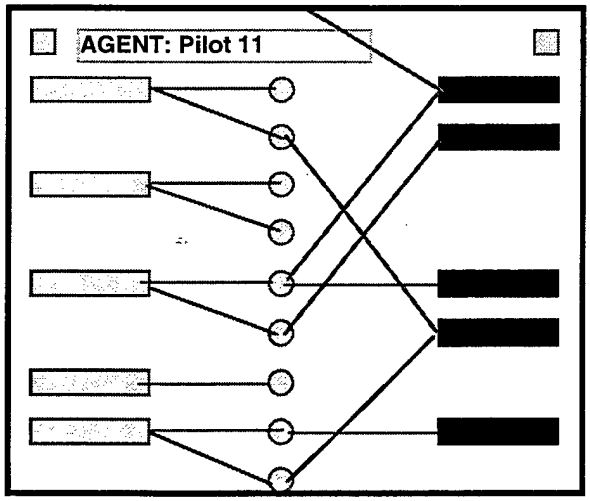


Figure 7. Procedures Receiving Signals

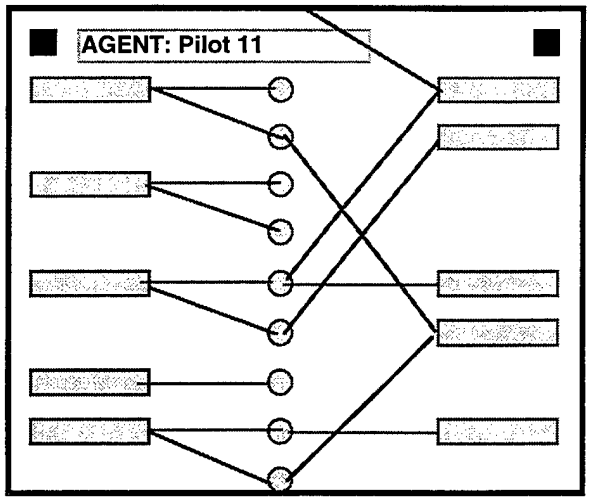


Figure 8. Signal Displayers

3.3.3.4. Agent Filtering

The list of displayed Agents can be "filtered" by selecting the command "Filter Displayed Agent..." on the "Agents" menu.

When this command is chosen the user is presented with a list of all available agents. The user then selects the agents for which he wishes the signals to be displayed. The procedures for the Agents are displayed according to the following rule: If an agent is among the set of agents selected by the user, all of the procedures for that agent are displayed, both procedures transmitting signals and procedures receiving signals. (Furthermore, the name is of the agent is displayed in yellow.)

For all other agents—those not in the "selected set"—the procedures for those agents are shown *only* if those procedures either transmit signals to or receive signals from one of the selected signals.

3.3.4. Timeline Graph

To the right of the Connectivity Graph in the main part of the display is the Timeline Graph portion of the SCAN Display (see Figure 2). In this graph, the actual specific

occurrences of a signal (simultaneous transmitting and receiving of a signal) for each signal type is plotted against time.

Across the top of the Timeline is a time scale (in units appropriate to the specific application model, usually seconds) plotted horizontally, increasing to the right.

Below the time scale are horizontal bands across the Timeline corresponding to each of the Agents which are currently being displayed.

Scattered across the Timeline are a number of small horizontal lines, corresponding to each Signal firing in the model run. Small yellow horizontal lines correspond to the transmitting of a signal; the vertical position of the yellow horizontal line corresponds to the vertical position of the "signal-circle" in the Connectivity Graph which is associated with this signal.

Similarly, a small red horizontal line corresponds to the receiving of a signal, its vertical position corresponding to the vertical position of the procedure receiving the signal in the Connectivity Graph.

Moving the mouse over one of these signal marks will cause the following:

- The corresponding signal-arc will highlight in the Connectivity Graph.
- The corresponding signal-mark will highlight in the Timeline Graph (i.e. the "red receiving mark" will highlight if the mouse has been moved over the yellow "transmitting mark" and vice versa.)
- Detailed information about this particular signal occurrence will appear in the mouse-documentation pane; i.e., the time of the signal, names of the transmitting and receiving procedures and agents, and the actual arguments to this signal instance.

3.3.5. Scrolling in the Scan Display

Scrolling with the vertical scroll bar (on the right side of the display) will cause both the Connectivity Graph and the Timeline Graph to scroll, revealing any non-visible agents and their procedures.

Scrolling with the horizontal scroll bar (on the bottom of the display) will cause the Timeline Graph to move forward and backward along the time axis.

3.3.6. Mouse Documentation Pane

In the upper part of the SCAN Display, below the menu bar and the main label, is a region of the screen used as a "Mouse Documentation Pane". As discussed above, as the mouse moves over the various items in the SCAN Display, detailed information about that item appears in this area (see Figure 2).

4. OMAR HPP MODEL APPLICATIONS: OASYS AND DEPTH

Systems of almost all types have become more expensive to build and more complex to operate. It has become increasingly important to "know" that these systems are going to achieve the objectives for which they were designed. Simulation, the modeling the behaviors of the equipment, has been used for a long time as one means to address the system assessment problem. But systems are more than just the equipment. Several individuals may be involved in the operation of a system, often at a single site, but more recently operators collaborate from remote sites. The operators interact with the system and frequently also interact with one another. The tasks taken on by the operators can be quite complex, and most notably, system automation frequently places a high cognitive load on the operators.

The HPP models produced using the OMAR software development environment are an important "product" of OMAR. They portray the perceptual, cognitive, and psychomotor skills of the operators that they emulate. It is now possible to include models of the system operators in the simulation system used to evaluate system performance. The range of possible applications is very extensive. Within this project, OMAR HPP models have been integrated to run with the Operability Assessment System (OASYS) (BBN, 1996) and the anthropometric model Jack (UPenn, 1995). In the OASYS environment, OMAR provided HPP models that interacted with OASYS models of the operator workplace. The HPP models were complete perceptual, cognitive, and psychomotor models. In the Jack simulation world, the OMAR HPP model is primarily a cognitive model that uses the Jack anthropometric model for perceptual and psychomotor functions. The boundary between the HPP model and the "world" is in a very different place in each of these applications. The use of OMAR HPP models in each of these environments is described in the following sections.

OMAR HPP models are also being used in two other simulation efforts. In the first, Young (1992; 1993) is using OMAR to implement a holon-based (Koestler, 1976) architecture. Here, the holons or agents are each part of a hierarchy. The holons

communicate using SCORE signals to pass quanta of activation. A holon whose activation reaches a specified threshold will generate further signals to holons downstream in the communication path. The SCAN display was designed specifically to provide a graphical presentation tracing the activation of holons in the multiple holon hierarchies of a holon-based model.

The second OMAR HPP model application was done for the NASA-Ames Research Center (Deutsch, Cramer, & Clements, 1996). HPP models were developed for the captain and first officer of a commercial aircraft and the air traffic controller managing the top-of-descent transition. OMAR was also used to model a subset of the aircraft flight deck instruments: the Mode Control Panel (MCP), the Control Display Unit (CDU) for the Flight Management System, and the Data Link message interface. The scenarios covered the data-link-based top-of-descent negotiation between the aircrew and the air traffic controller. The aircrew managed communication with the air traffic controller using the data-link system modeled in OMAR as well. The CDU and MCP were interfaced with the NASA-Ames miniACFS aircraft simulation running on an SGI machine. In contrast to the OASYS demonstration where two simulations were running on a single machine, for the NASA-Ames OMAR application, the HPP models and the aircraft model were running on separate machines.

4.1. Integrating an OMAR HPP Model into the OASYS Demonstration

Before the integration of OMAR and OASYS, OMAR had been used as a complete and self-contained simulation system. HPP models were one component of an environment that typically included models of the workplaces or target systems operated by the HPP models and models of the environment beyond the workplaces that the workplaces reflected. Within these simulation environments, all the "human" players were represented as HPP models. To take advantage of the efficiency of the OMAR simulator, simulations were always run in "fast-time" rather than real-time.

Using OMAR to develop HPP models for the OASYS demonstration placed a number of significant new demands on the system. The OASYS system was to provide a separate simulator that would include a model of each operator workplace and the environment beyond the workplaces. The workplace models and the workplace environment had typically been developed as part of the OMAR simulation environment itself. In the OASYS framework, the workplaces were to be used not only by the HPP models, but also by human players. The new version of OMAR had to communicate with another

simulator, the OASYS simulator, and it had to operate in real-time to accommodate the presence of human players. Communication between the simulators was necessary at two levels: at the system level to coordinate the operation of the two simulators and at the application level to simulate the observation and use of the workplace by the human performance model.

OMAR, built as a modular software system, was modified to accommodate these changes. The OASYS demonstration, which took place at Armstrong Laboratory on Thursday, June 22, 1995, included a scenario with four ATC workplaces. In the primary scenario there was one human operator and three OMAR-provided ATC HPP models managing the remaining workstations. It was then possible to interchange human players and HPP models at any of the workstations. The major OMAR tasks accomplished were to link the simulators for simultaneous operation, provide HPP models for the ATC demonstration, and support the demonstration.

4.1.1. The Interface Between OMAR and OASYS

A key element in supporting the OASYS demonstration was the integration of the operation of the OMAR and OASYS simulators. To provide an environment suitable for human players, the simulators had to run in real-time. The OASYS team made the decision that the simulators would operate as individual processes within a single Lisp image. The code for the interface between the two processes was developed so that it could easily be adapted to operate with OMAR and OASYS running on separate machines communicating via Unix sockets. To make this future transition from communication between processes within a single image to socket-based communication as simple as possible, text strings were chosen as the basis for the protocol for messages to be passed between the simulators.

Both OMAR and OASYS support the signaling of and enqueing on events. That is, in each system, a named signal with arguments can be announced and procedures that are enqueued on the signal object can examine the signal's arguments to decide whether or not to further process the signal. In OMAR, signals take the form of lists, while in OASYS they are particular Lisp class objects. To make the communication possible on the outbound side, the OASYS class objects or OMAR lists had to be converted to strings, passed between the processes, and then converted to lists when OMAR was receiving them and class objects when OASYS was receiving them.

In OMAR on the receiving side, signals were now being received from two asynchronous processes, the OMAR simulator process and the "socket" process receiving events from the OASYS simulator "socket" process. To ensure that these processes did not conflict when putting events in the OMAR event queue, the SCORE form to install an event in the queue, *signal-event,* was set up to run *without-interrupts.* The OMAR event enqueing routine can be called from either OMAR itself or from the process that translates the OASYS event string into an OMAR list. Within OMAR there is no distinction between enqueing on internally (OMAR) and externally (OASYS) generated events. From this point on, within SCORE, an OASYS generated event is just another *signal-event* generated SCORE event. Events from either source are enqueued on using the SCORE forms *asynchronous-wait* or *with-signal*; it is the event lists themselves that differ in content between internally and externally generated events.

Events originating on the OMAR side of the interface are to be acted on immediately within OMAR. In the case where OMAR is currently busy processing an event, the new OASYS-generated event is simply taken as the next event from the event queue to be processed when processing for the current event has been completed. However, it is also possible that the next OMAR-scheduled queue item is not to be processed until a future time. In this case, OMAR would be in a *wait-state* pending the time of the next scheduled item and is not actively processing its event queue. To address the wait state problem, a tight delay loop, using *without-scheduling* for efficiency, was adopted to replace the wait state. The delay loop checks a variable that is set as a new OASYS-generated event is inserted in the OMAR event queue. On finding the variable set OMAR processes the newly-arrived event.

On the outbound side, OMAR has to be able to insert events into the OASYS event queue. The process on the outbound side is somewhat simpler. OMAR has the SCORE language *signal-event* form that is used to generate events locally. A new form, *signal-external-event,* was developed as the basis for generating events to be communicated to OASYS. On the OMAR side, the new event form invokes the procedure that translates the OMAR event list to a string to be immediately communicated to OASYS. On the OASYS side, the string is picked up and translated to an OASYS-style event object for processing in OASYS.

With respect to the synchronization of the simulators, either side may come up first and be joined by the other. In practice, OMAR is usually brought first. The OMAR HPP models are then initiated and essentially "wait" for something to happen. Within this

framework, it is the responsibility of OASYS to start an experiment trial which then generates the first workplace events that the HPP models are to respond to.

4.1.2. The OASYS Demonstration ATC Scenario and the OMAR HPP Models

As part of the final demonstration of OASYS, the OASYS team built an air traffic control (ATC) simulation for a team of four operators controlling adjacent sectors of the airspace. Each ATC operator performed his or her tasks at a workplace consisting of a synthetic radar screen and screen panels to manage the sending and receiving of messages. ATC operators exchanged structured messages to transfer aircraft between sectors and were able to accept or deny these transfer requests as a function of their workload. An OASYS script controlled the number, route, and speed of aircraft, subject to the commands of the ATC operators. Two OASYS scripts were implemented to create different levels of workload for the operators by varying the number of planes in the airspace. An OMAR HPP model was developed to handle the ATC task. The HPP model was able to fill one or more of the operator positions, while human operators filled the remaining positions. Comparative data were collected for all four positions, with three positions filled by HPP models and one by a human operator. Measures of performance included the number of times that aircraft were delayed waiting for a clearance, the mean length of the delays, and the mean time between receiving a clearance and sector crossing. The data file produced by OASYS was transferred to an Excel spreadsheet for analysis.

This demonstration was designed to illustrate many of the key features of the OASYS experiment system: the ability to quickly create a simulation of a target system, the ability to easily create dynamic GUI prototypes, the ability to simulate multiple operator stations for a multi-person task, the ability to interchangeably use human operators and human performance models in an experiment, the ability to specify performance data to be collected during the experiment for both humans and models, and the ability to create an experiment data file that can be analyzed with COTS software.

The OMAR HPP model for the air traffic controller, developed to operate in the OASYS ATC demonstration scenario, was developed with two distinct components: a basic human operator model and a set of capabilities particular to the specific OASYS ATC task. The ATC component of the HPP model detects aircraft in the airspace, monitors their progress toward the area-of-responsibility boundary, and manages the hand-off of the aircraft to the neighboring air traffic controller. The HPP model also manages requests from neighboring air traffic controllers to transfer control of their aircraft into

the sector. The HPP model makes extensive use of the OMAR-OASYS interface at the application level. The data from an OASYS event is translated to an OMAR event form and properly inserted into the OMAR simulator event stream. Information on aircraft location, maintained by the OASYS simulator, is provided through the interface. The incoming data includes the appearance and location of planes on the radar screen and the arrival of new messages for the ATC. The OMAR HPP models generate external events that use the OMAR-OASYS interface to create and activate OASYS events. These are primarily ATC actions to generate messages for neighboring ATCs.

The ATC component of the HPP model was integrated with the basic operator model developed in Delivery Order 5 to complete the HPP model for the OASYS demonstration. This layer provides basic human-like proactive and reactive behaviors representative of the actions that take place at an ATC workplace. A proactive scan activity monitors the progress of aircraft icons across the screen, while reactive activities respond to the appearance of new aircraft icons. Similar proactive and reactive activities are used to deal with the construction and sending of messages, and generating the response to newly-arrived messages. Coordinated hand-eye actions are provided, particularly for the mouse and keyboard operations to create messages for other controllers. The scenarios and the ATC tasks were designed so that not all tasks had the same priority. Priority schemes in the HPP model were developed so that task execution reflected the differences in task priorities among the tasks.

OMAR includes a simple human figure model, KATE, seated at a radar console. The ATC HPP model was connected to KATE during the OMAR-OASYS demonstration. It was then possible to follow the hand and eye actions of the model as it executed the ATC tasks in response to the OASYS-generated events of the scenario.

4.2. DEPTH Workstation Demonstration

A major feature of the Design, Evaluation, Personnel, Training, and Human Factors (DEPTH) program (Ianni, 1995) has been the capability to model the execution of aircraft maintenance procedures using the University of Pennsylvania's anthropometric model, Jack, to animate the execution of these procedures. The Jack simulation environment also includes the capability to use CAD files as input to populate the environment with models of the target systems, usually an aircraft or the equipment used to service an aircraft. Jack is an interactive system and it is through this interactive capability that sequences of maintenance procedures have been developed.

The computer capture of the maintenance procedures and visualization of the maintenance procedure execution has many potential applications. The most immediate impact is on the development and evaluation of maintenance procedures themselves. As maintenance procedures are developed or existing procedures are revised, the modeling process provides the basis for an earlier and more thorough evaluation of proposed procedures than would otherwise be possible. In a fully developed system, issues such as time requirements for procedure execution, access to confined spaces and tool requirements for a specific task could be addressed.

The modeling of aircraft maintenance procedures also has the potential to impact the design process. If maintenance procedure development is conducted concurrently with aircraft design, "difficult" maintenance tasks can be identified at a time in the design process at which they can be addressed as design problems rather than being uncovered at a later point in the system development process as long-term and costly maintenance problems. Evaluating problems of physical access and tool usage in restricted spaces are probably most important here. Easy access to subsystems requiring frequent maintenance should be assured, as well.

There is also a series of activities downstream from the actual development of the maintenance procedures themselves that can be addressed starting from a declarative computer representation of the procedures. Here, the most important is probably the generation of multimedia electronic technical manuals. The authoring and updating of these manuals is a time consuming, laborious process even with the authoring tools available today. The maintenance procedure process has the potential to help in at least two areas: providing textual descriptions of maintenance procedures and providing CAD material as graphical input to the authoring process. A very closely-related area is computer-based tutoring for maintenance operations. The procedure representations that have the potential to support the authoring of technical manuals can similarly be used to support the development of tutoring systems for aircraft maintenance.

From the foregoing, it is clear that the modeling of aircraft maintenance procedures has the potential to a have a significant, positive impact affecting the areas of design for maintenance, better evaluation of and more rapid development of the maintenance procedures themselves, and the authoring of electronic technical manuals supported by tutoring systems. The basic challenge has been to realize this potential.

The basis for this task was the recognition that OMAR complements existing DEPTH and Jack capabilities in several important ways. Most important was that OMAR has the potential to provide an HPP model of the maintenance person possessing a representation of the cognitive skills necessary to execute the maintenance tasks. As the HPP model becomes "smarter" about executing maintenance tasks, the user will have to do less to build the representation and animation of a given new or revised maintenance task. Secondly, the declarative representation that is built in OMAR to represent the execution of the maintenance task is just the representation that is needed to support the downstream functions related to electronic technical manual authoring. The Simple Frame Language (SFL) -based representation of maintenance procedures has been used in the past as input to Spokesman (Meteer, 1989), a powerful text generation system capable of generating full paragraph descriptions of maintenance procedures.

This task has addressed the initial complex steps in building a maintenance procedure development system based on DEPTH and an integration of OMAR and Jack. The first step has been to provide an interface between OMAR and Jack so that the cognitive HPP model of the maintenance person can interact with its physical realization operating with CAD-generated aircraft maintenance world objects. The second step was to build a representation of a maintenance procedure based on the B-1 environment provided by Armstrong Laboratory. The task was a joint effort by BBN and the University of Pennsylvania.

4.2.1. The Interface Between OMAR and Jack

At one level, both OMAR and Jack are simulation systems, each designed to operate as stand-alone systems where each has the capability to model human players, target systems under study, and outside world objects that the human players or target systems interact with. In integrating the systems to run together, it was no longer necessary to model either the target systems or the outside world in OMAR. This task was simply turned over to the Jack simulator. Issues surrounding the new human performance model were more complicated. Basically, the cognitive tasks of the maintenance person model were to be taken over by the OMAR HPP model and expanded. The interactions of the human performance model with the target systems were to continue to be animated by Jack. The resulting HPP model for the maintenance person exists in part in the OMAR simulator and in part in the JACK simulator. To accomplish this, communication between OMAR and Jack takes place at two levels: the systems level and the application level.

The substructure that supports communication between OMAR and Jack is the C-language Application Interface, or C-API, that the Jack system provides. The interface provides access to the Jack simulator at several levels. It allows access to the Jack Command Language (JCL) (UPenn, 1995), access to the Lisp components of Jack, and access to Jack PaT-Nets (Douville, 1995), a language for controlling Jack actions based on a finite state machine architecture layered on Lisp. The C-API, based on the Remote Procedure Call (RPC) protocol, makes it possible for the client system to run either on the same machine as Jack or another machine, either locally or at a remote site. When the interface was first brought up, the initial testing was done using an SGI machine running Jack at Armstrong Laboratory and OMAR running on a Sun at BBN in Cambridge, Massachusetts. The client RPC software for the Sun was adapted from the standard SGI code by the University of Pennsylvania. On the OMAR side of the interface, the Franz Lisp foreign function call capability was used to build the interface software between OMAR and the C-API client.

Under a separate task, the OMAR system was ported from the Sun workstation to run on an SGI machine. Once this was accomplished, it was possible to run OMAR and Jack entirely in an SGI environment, either on a single machine, or using one SGI machine to run OMAR and another to run Jack. The C-API interface is used in both run-time configurations.

With the basic system level interface in place, it was then possible to address application level interface issues. From the OMAR side of the interface, the two major areas to be addressed were controlling Jack actions and establishing the basis for referencing objects in the Jack world.

Human performance modeling typically requires the concurrent execution of several activities, a straightforward example being the coordinated hand and eye motions used to accomplish a simple manual task. The SCORE language for modeling behaviors has a series of forms dedicated to managing and arbitrating concurrent activities. In Jack, the PaT-Nets finite state machines provide a similar capability. Providing the capability for OMAR to manage Jack's multi-tasking performance across the interface will be possible, but addressing the level of complexity on the Jack side of the interface for this initial effort was judged to be not cost effective. Since the initial maintenance tasks to be developed were largely sequential in nature, the added complexity of providing a full multi-tasking capability across the interface was deferred in favor of devoting more effort to the development of the maintenance procedures themselves.

The initial OMAR-Jack interface implementation provides for the sequential execution of Jack tasks as they are placed on a Jack queue by OMAR. This capability is provided by a Jack Lisp function, *add_task_to_q,* that takes the name of the Jack figure to execute the operation, the operation to be executed, and the arguments to the operation as input. The calls made via this interface to control Jack actions are primarily calls to Jack Lisp functions or to PaT-Nets. The University of Pennsylvania provided a number of shared object libraries supporting Jack actions required by the maintenance procedures that are described in the next section. The Jack *register_module* and *load_module* functions were to used to access this libraries.

The remaining interface issue to be discussed is that of referencing Jack world objects from OMAR. The Jack world objects, whether anthropometric models or target system objects, are figures comprised of segments and joints. Particular locations may be associated with segments and labeled as sites. Jack figures, segments, joints, and sites have names and pointers. In the Jack world, the pointers are the primary means to reference objects. Jack provides a complete set of functions to obtain names from pointers and pointers from names. Names can be absolute, referring to a particular figure, or relative, referencing for example the palm of a right hand, but not identifying a particular figure to which it belongs. The naming structure is hierarchical, with each element in the full name separated a period. To gain access to the Jack routines for dealing with Jack names and pointers from OMAR, a series of OMAR functions were defined that call the Jack routines using the C-API interface.

The names used for Jack world objects work well for the users of Jack—the "pwr_control" is readily recognized to be a power control unit and in the context of the B-1 aircraft it is easily recognized as a line replaceable unit (LRU). The "pwr_control" has a handle, a jack for a cable to plug into, and points that would typically be grasped to carry the LRU. When interactively building Jack commands to create an animation of a maintenance procedure, the user can readily select the proper references either by name or by pointing at the desired object. As long as there is a user building the procedures and time is available to construct in each step in the required detail, this process works very well.

An important goal in adding cognitive skills to the maintenance HPP model by using OMAR has been to explore ways to aid the user in constructing the maintenance procedure representations. If the HPP model "knows" how to execute some of the steps of the maintenance procedure being developed, it can relieve the user from explicitly

specifying those steps. Providing the HPP model with "type" information that will enable the model to reason about the objects before it is an important step toward accomplishing this goal. The type information states that the Jack object named "pwr_control" *is a* power-control-unit and that it *is a* line-replaceable-unit. Similarly, the objects that are named as handles and jacks are defined as handles and jacks.

The Simple Frame Language (SFL) in OMAR was designed to address the specific task of defining the objects of the simulation world so that they would be available to be reasoned over by HPP models. The type definition, or *is-a* relation in SFL terms, is the most important relationship expressed in SFL. When the Jack object named "pwr_control" is defined to be a power-control-unit and a line-replaceable-unit, these are *is-a* relationships, and taken together, this is just an example of multiple inheritance in SFL terms.

Hence, the "reference" problem has been addressed at two levels: first as an interface problem and then by using SFL to provide a declarative representation of the Jack world objects. The declarative representation of Jack world objects provides the HPP model of the maintenance agent with the information that the human user has when he or she views the "pwr_control" with a "handle." From the SFL representation, the HPP model now "knows" that the Jack object with the name "pwr_control" is a power-control-unit and a line-replaceable-unit and that "handle" is a handle that can be used to move the LRU. The named graphical objects of the Jack world have been tagged with semantic labels.

For the current task, the process of "semantic labeling" is a programming task accomplished using the Graphical SFL Editor. SFL types, or concepts in SFL terms, are defined and particular named Jack objects are identified by type. Once this is done, OMAR can step in and obtain the Jack pointers through the interface to Jack. The OMAR definitions of Jack entities also address the distinction between those entities which are figures or segments and those which are sites. The HPP model is then able to reference the entities without regard to whether they are figures, segments, or sites. In the future, it will be important to provide this same referencing capability to user of the OMAR-Jack HPP model of the maintenance agent. The interactive user defining maintenance procedures should not have to know that the handle on an LRU is a Jack segment and the grab point to move the LRU is a Jack site. Future work in providing graphical semantic labeling of the Jack world entities will make this possible.

4.2.2. Maintenance Procedure Development Using an HPP Model

At an intuitive level, it is clear that integrating an OMAR HPP model with the Jack anthropometric model ought to provide the basis for building a maintenance agent model that will make it easier for the interactive user to pursue the development of aircraft maintenance procedures and support the downstream tasks, such as the authoring of technical manuals. Jack provides not only the anthropometric model, but also provides the capability to capture CAD data as the basis for target system models of the aircraft and maintenance support equipment. OMAR provides the ability to enhance the anthropometric model with a representation of the cognitive skills of the maintenance agent. The long-range goal of this integration effort is to demonstrate that this intuition is correct. The goal in the immediate effort described here has been to carefully select the first steps toward this goal, implement those steps, and demonstrate how they move us along the way toward realizing the long-term goals.

Jack is both an interactive system and a programming environment. As an interactive system, the user can develop sequences of operations, which in this environment constitute the execution of an aircraft maintenance procedure. The variety and number of possible actions that an aircraft maintenance person might be called upon to execute is huge, and hence, there is a constant demand to add to the envelope of Jack's capabilities. Adding to the envelope of capabilities is typically a programming task. A representative example of a required capability is stepping backward, having lifted an LRU from the mounting rack on a shelf. On a larger scale, many new capabilities will be required to manage the use of a broad range of tools used in maintenance operations.

The development of HPP models in OMAR is also a programming task. To manage the complexity of the models, a considerable effort has been devoted to the development of tools to support HPP model building. Graphical editors and browsers are available to provide support in generating and managing the code that constitutes the HPP models and an extensive set of post-run analysis tools serve as debugging aids to support model development.

Given the integration of OMAR and Jack, the next step was then to develop a representation and animation of a simple aircraft maintenance procedure. This has been primarily a programming task, with the programming work being done both in OMAR and Jack. The maintenance procedure representation and the animation are the two distinct end products of this initial effort. The animation provided is that which is

typically expected from Jack today. It is the explicit representation of the aircraft maintenance procedures themselves that enables the addition of a new range of capabilities related to maintenance procedure development and technical manual generation.

Even the relatively simple maintenance task undertaken for the demonstration project has provided a look at what these new capabilities include. The task involves the removal and replacement of an LRU. The B-1 environment for the demonstration was provided by Armstrong Laboratory and subsequently modified by the University of Pennsylvania to take advantage of new Jack features not previously available.

4.2.2.1 Jack Animation of an Aircraft Maintenance Procedure

The OMAR/Jack integration effort illustrates the untethering and replacement of a power supply unit. The following basic Jack actions combine to form higher level tasks:

- Grasp
- Reach (fast inverse-kinematics)
- Locomotion
- Hose dynamics
- Visual search
- Task-appropriate attention.

These actions are encapsulated as parameterized PaT-Nets (Parallel Transition Networks). Rather than invoking these nets directly, the OMAR process puts each PaT-Nets request into a queue of requests. A queue manager, maintained for each virtual agent in the Jack environment, consumes and animates task actions. As discussed in a previous section, the OMAR-to-Jack Lisp communication is done through the Jack C-API. A queue of requests is maintained to facilitate the animation of visual attention. Since a queue allows lookahead of downstream tasks, we allow the potential to *interleave* or *anticipate* where attention may be directed. Also, a queue manager allows the spawning of an attentional net or process for a particular *type* of action. For example, if OMAR requests that Jack animate a grasp, attention will automatically be directed to the site relevant for that grasp. The animation generated uses only one explicit "focus" command that directs an agent's line of sight. Whenever possible, attention is invoked automatically in relation to the type of action being executed.

In order to facilitate the animation of human grasps, the Jack system maintains a list of grasp sites and hand orientations in the Jack Object Specific Reasoner (OSR) Table (Levison, 1996). Objects in the environment are tagged with an abstract OSR type. Each OSR type can in turn be associated with a use. Different uses are associated with sites on an object, hand orientations, and type of grasp. The OSR stores information regarding palm and finger orientation relative to a site for a particular use. For example, pulling the handle on the power supply requires maintaining which site on the handle is relevant for pulling (usually the center of the handle), a grasp type which closes the hand and fingers about the handle and the approach orientation of the palm and fingers relative to the handle.

Animation of human reaching is done with a fast inverse kinematics module in Jack (Tolani & Badler, 1996). When a Jack grasp is requested, or if a lifting or pushing action is required, the inverse kinematics routine determines an analytic solution for the required arm position. This routine is called iteratively to generate the entire arm reaching motion.

Human locomotion and path planning in Jack are implemented using attract and avoid behaviors (Reich, 1996). For the replace power supply scenario, we instantiate a PaT-Net that attaches an attract behavior from the agent to the power supply. Within this PaT-Net, we also pass the information that the agent should avoid particular types of obstacles (walls or racks in our procedure). We also pass a final orientation vector that indicates which direction the agent should be facing when the attract location is reached.

Hose dynamics (Foster & Metaxas, 1996) in Jack are used to animate the motion of cables in our simulation. In our scenario, a cable is tethered at one end to the power supply and to the aircraft ceiling at the other end. We animate the agent grasping the cable, detaching it from the power supply and letting it trail according to gravity from the ceiling. Rather than using constraints or attach commands in the Jack system, our hose dynamics uses a physics-based approach. While constraints may illustrate the motion of cable hoses when their end positions remain static, they are not as suitable for scenes where the human figure is dragging or manipulating the hose or if the end of the hose is constrained to a moving object.

Visual search, task-related attention, and explicit focus commands are supported in Jack (Chopra, 1995). At the start of our simulation, we invoke a search net which causes the agent to scan the environment until a particular feature or object is observed (in our scenario, the power supply). Since task requests are placed on a queue, the queue

manager for an agent automatically invokes the appropriate attentional behavior for each type of task or task mix on the queue. For example, while the agent is walking to the power supply, the agent will look at the power supply and occasionally glance at his feet (when a memory uncertainty threshold is exceeded or when the agent is in close proximity to an obstacle such as the rack). When grasping an object, the queue manager invokes an attentional process that accesses the OSR. This process determines the relevant site for the grasp and directs attention appropriately. Explicit focus commands that direct an agent's line of sight are also supported.

4.2.2.2. OMAR Extensions to the Aircraft Maintenance Procedure Representation

Interactive building of a procedure such as the remove and replace for an LRU has typically been done for a particular LRU with a particular configuration. The number and type of fasteners, the number of cables attached, and the number of handles available must each be factored into the procedure for the particular LRU. The ability to develop the remove and replace LRU procedure in OMAR makes a significant difference. In the HPP model that has been developed, one remove and replace LRU procedure addresses each of these configuration specific issues. As the HPP model executes the remove and replace procedure, it determines the number of fasteners and cables that have to be released and the number of handles available for pulling the LRU out in preparation for removing it. The type of fastener to be released is also determined, and in the case where the cables are to be disconnected and reconnected in a specific order, that specific ordering is used as the procedure is executed.

The use of SFL for the semantic labeling of the Jack representation of the LRU is the central element that enables a single OMAR procedure to be developed which addresses these many varied cases. The OMAR HPP model of the maintenance agent "knows" how to adapt to the several variations in the configuration of an LRU by referencing the SFL representation of the particular LRU as the relevant steps of the procedure are executed. Notes and cautions are also important aspects of the documentation of maintenance procedures. A typical caution relates the weight of an object that is to be lifted to the requirement that the lift be executed by two people. The SFL representation of the LRU includes a slot for the weight of the LRU. This weight is checked before the lift of the LRU is executed. If the weight requires a two person lift, a cautionary message is generated as part of the execution of the procedure.

REFERENCES

Adams, M. J., Tenney, Y. J., & Pew, R. W. (1994). *State-of-the-art report: Strategic workload and the cognitive management of advanced multi-task systems* (CSERIAC-SOAR 91-6). Wright-Patterson Air Force Base, OH: Crew System Ergonomics Information Analysis Center Publication Series.

Baecker, R. M. (1993). *Groupware and computer supported cooperative work: Assisting human-human collaboration.* San Francisco, CA: Morgan Kaufman Publishers, Inc.

BBN (1996). *Operability Assessment System (OASYS) Final Report.* BBN Report 8106. Cambridge, MA: BBN Corporation.

Boettcher, K. L. & Levis, A. H. (1982). Modeling the interacting decision maker with bounded rationality. *IEEE Transactions on Systems, Man, and Cybernetics, 12,* 334-344.

Boettcher, K. L. & Levis, A. H. (1983). Modeling and analysis of teams of interacting decision makers with bounded rationality. *Automatica, 19,* 703-709.

den Braven, W. (1992). *Design and evaluation of an advanced air-ground data-link system for air traffic control* (NASA Technical Memorandum 103899). Moffett Field, CA: NASA Ames Research Center.

Bushnell, L. G., Serfaty, D., & Kleinman, D. L. (1988). Team information processing: A normative-descriptive approach. In S. E. Johnson & A. H. Levis (Eds.), *Science of command and control: Coping with uncertainty*. London: AFCEA International Press.

Cannon-Bowers, J. A. & Salas, E. (1991). Cognitive psychology and team training: Shared mental models of complex systems. *Human Factors Society Bulletin*, p. 1-4.

Chopra, S. (1995). *Strategies for Simulating Direction of Gaze and Attention*. HMS Center, University of Pennsylvania.

Coovert, M. D. & McNelis, K. (1992). Team Decision Making and Performance: A review and proposed modeling approach employing Petri nets. In Swezey, R. W. and Salas, E. (Eds.) *Teams: Their training and performance*. Norwood, NJ: Ablex Publishing Corporation.

Damasio, A. R. (1989a). The brain binds entities and events by multiregional activation from convergence zones. *Neural Computation, 1,* 123-132.

Damasio, A. R. (1989b). Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition, 33,* 25-62.

Deutsch, S. E., Cramer, N., & Clements, R. (1996). *Vehicle/aircrew procedure and control modeling*. BBN Report 8121. Cambridge, MA: BBN Corporation.

Deutsch, S. E., Hudlicka, E., Adams, M. J. & Feehrer, C. E. (1993a). *Research, Development, Training, and Evaluation (RDTE) Support: Delivery Order #1 - Computational Cognitive Models* (AL/HR-TP-1993-0072). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Deutsch, S. E., Adams, M. J., Abrett, G. A., Cramer, N. L. & Feehrer, C. E. (1993b). *Research, Development, Training, and Evaluation (RDTE) Support: Operator Model Architecture (OMAR) Software Functional Specification* (AL/HR-TP-1993-0027). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Douville, B. J. (1995). *PaT-Net user's guide (Draft).* University of Pennsylvania Working Paper. Philadelphia, PA: University of Pennsylvania.

Driskell, J. E. & Salas, E. (1992). Can you study real teams in contrived settings? The value of small group research to understanding teams. In Swezey, R.W. and Salas, E. (Eds.) *Teams: Their training and performance.* Norwood, NJ: Ablex Publishing Corporation.

Edelman, G. M. (1987). *Neural Darwinism: The theory of neuronal group selection.* New York: Basic Books.

Edelman, G. M. (1989). *The remembered present: A biological theory of consciousness.* New York: Basic Books.

Entin, E. E., Serfaty, D., Entin, J. K., & Deckert, J. C. (1993). *CHIPS: Coordination in hierarchical information processing structures: Experiment Report* (TR-598). Burlington, MA: Alphatech, Inc.

Foster, N. & Metaxas, D. (1996). Realistic Animation of Liquids, *Graphical Models and Image Processing, 58(5),* pp. 471-483.

Galegher, J., Kraut, R. E., & Egido, C. (1990). *Intellectual teamwork: Social and technological foundations of cooperative work.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Glickman, A. S., Zimmer, S., Montero, R. C., Guerette, P. J., Campbell, W. J., Morgan, B. B., Jr., & Salas, E. (1987). *The evolution of team skills: An empirical assessment with implications for training* (Technical Report No. NTSC87-016). Orlando, FL: Naval Training Systems Center.

Greif, I. (1988). *Computer-supported cooperative work: A book of readings.* San Mateo, CA: Morgan Kaufman Publishers Inc.

Hart, S. G. & Staveland, L.E. (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In P. A. Hancock & N. Meshkati (Eds.) *Human mental workload.* Amsterdam: Elsevier.

Ianni, J. (1995). *Maintenance Simulation: Research & Applications* (AL/HR-TP-1995-0019). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Jackendoff, R. (1987). *Consciousness and the computational mind.* Cambridge, MA: The MIT Press.

Kleinman, D. L., Luh, P. B., Pattipati, K. R., and Serfaty, D. (1992). Mathematical models of team performance: A distributed decision-making approach. In Swezey, R. W. and Salas, E. (Eds.) *Teams: Their training and performance.* Norwood, NJ: Ablex Publishing Corporation.

Koestler, A. (1976). *Janus: A summing up*. New York: Random House.

LaPorte, T. R. & Consolini, P. M. (1988). *Working in practice but not in theory: Theoretical challenges of high reliability organizations.* Annual Meeting of the American Political Science Association, 1-4, Washington, D.C.

Levis, A. H. (1989). Generation of architectures for distributed intelligence systems. *Proceedings of the 1989 IEEE International Conference on Control and Applications* (ICCON '89). Jerusalem, Israel: IEEE Press.

Levison, L. (1996). *Connecting Planning and Acting Via Object Specific Reasoning.* Ph.D. Dissertation , HMS Center, University of Pennsylvania.

Lysaght, R. J., Hill, S. G., Dick, A. O., Plamondon, B. D., Linton, P. M., Wierwille, W. W., Zakland, A. L., Bittner, A. C., Jr., & Wherry, R. J. (1989). *Operator workload: Comprehensive review and evaluation of operator workload methodologies* (Technical Report 851). Alexandria, VA: United States Army Research Institute for the Behavioral and Social Sciences.

Mallubhatla, R., Pattipati, K. R., Kleinman, D. L., & Tang, Z. B. (1991). A normative-descriptive model for a team in a distributed detection environment. *IEEE Transactions on Systems, Man, & Cybernetics, 21(4),* 713-725.

Manning, C. R. (1987). *Acore: The design of a core actor language and its compiler.* Master's thesis, Massachusetts Institute of Technology. Cambridge, MA: MIT.

McGrath, J. (1984). *Groups: interaction and performance.* Englewood Cliffs, NJ: Prentice Hall.

McGrath, J. E. (1991). Time, interaction, and performance (TIO): A theory of groups. *Small Group Research, 22*, 147-174.

Meister, D. (1985). *Behavioral Analysis and Measurement Methods*. New York: John Wiley & Sons.

Meteer, M. W. (1989). *The Spokesman natural language generation system.* BBN Report 7090. Cambridge, MA: BBN Corporation.

Minsky, M. (1986). *The society of mind.* New York: Simon and Schuster.

Morgan, B. B., Glickman, A. S., Woodard, E. A., Blaiwes, A. S ., & Salas, E. (1986). *Measurement of team behaviors in a Navy environment* (Tech. Report No. NTSC TR-86-014). Orlando, FL: Naval Training Systems Center.

Neumann, O. (1987). Beyond capacity: A functional view of attention. In H. Heuer & A. F. Sanders (Eds.), *Perspectives on perception and action* (pp. 361-394). London: Lawrence Erlbaum Associates.

Orasanu, J. (1990). *Shared mental models and crew decision making* (Tech. Rep. No. 46). Princeton, NJ: Princeton University, Cognitive Sciences Laboratory.

Orasanu J. & Salas, E. (1993). Team decision making in complex environments. In Klein, G. A., Orasanu, J., Calderwood, R. and Zsambok, C. E. (Eds.) *Decision Making in Action: Models and Methods*. Norwood, NJ: Ablex Publishing Corporation.

Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. New York: Prentice hall.

Reich, B. D. (1996). *An Architecture for Behavioral Locomotion*. Ph.D. Dissertation, HMS Center, University of Pennsylvania.

Rogers, W. H. (1996). *Understanding flight deck task management through pilot interviews.* Cambridge, MA: BBN Corporation.

Salas, E., Dickinson, T. L., Converse, S. A., & Tannenbaum, S. I. (1992). Toward an understanding of team performance and training. In Swezey, R. W. and Salas, E. (Eds.) *Teams: Their Training and Performance*. Norwood, NJ: Ablex Publishing Corporation.

Serfaty, D., Entin, E. E., Deckert, J. C., & Volpe, C. (1993a). Implicit coordination in command teams. *Proceedings of the 1993 JDL Symposium on Command and Control Research*, Washington, D.C.

Serfaty, D., Entin, E. E., & Volpe, C. (1993b). Adaptation to stress in team decision making and coordination. *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting*. Santa Monica, CA.

Tabak, D. & Levis, A. H. (1985). Petri net representation of decision models. *IEEE Transactions on Systems, Man, and Cybernetics, 15*, 812-818.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science, 268,* 1632-1634.

Tolani, D. & Badler, N. I. (1996). *Real-time Inverse Kinematics for the Human Arm*. HMS Center, University of Pennsylvania, to Appear in PRESENCE.

UPenn (1995). *Jack user's guide.* Philadelphia, PA: University of Pennsylvania.

Weingaertner, S. T. & Levis, A. H. (1989). Analysis of decision aiding in submarine emergency decisionmaking. *Automatica, 25*, 349-358.

Young, M. J. (1992). *A Cognitive Architecture for Human Performance Process Model Research*. (AL-TP-1992-0054). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Young, M. J. (1993). *Successively Approximating Human Performance* (AL-TP-1993-0026). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.