E-24-652

Operator Function Modeling: Cognitive Task Analysis, Modeling and Intelligent Aiding in Supervisory Control Systems

Final Report

NASA Ames Grant NAG 2-413

Dr. Everett Palmer, Technical Monitor

Christine M. Mitchell, Principal Investigator

Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
404-894-4321
cm@chmsr.gatech.edu

September 1990

Acknowledgements

This research spanned a four year period and and its success depended on the participation, insightful comments, and periodic review of many people. The support, both conceptual and financial of NASA research and operations personnel, particularly Ev Palmer, Walt Truszkowski, and Mike Shafto, was greatly appreciated. T. Govindaraj was a major contributor to the original OFMspert conception and provided continuous support as the project evolved. Kenny Rubin and Patty Jones designed, implemented, and debugged the first OFMspert. Patty's award winning M.S. thesis demonstrated OFMspert's intent inferencing capabilities. Patty, Rose Chu, Suyin Uang, Todd Callantine, Julie Chronister, and Serena Smith were vital participants in our ability to change OFMspert's hardware and software platforms. Jim Bushman's award winning doctoral thesis demonstrated that the OFMspert concept was a viable architecture to provide useful assistance in complex dynamic systems. Finally, Richard Robison kept the whole computer infrastructure running--a great feat with research involving multiple computer systems, languages, and processes.

Executive Summary

This research addressed the design, implementation, and empirical evaluation of task-analytic models and intelligent aids for operators in the control of complex dynamic systems, specifically aerospace systems. The work carried out under the sponsorship of this grant * includes three related activities. First, we studied the use and development of models of operator decision making in complex and predominantly automated space systems. The primary representation was the operator function model (OFM). Second, the OFM was used to represent operator activities in a NASA Goddard satellite ground control system, the MultiSatellite Operations Control Center (MSOCC). Finally, and most significantly, OFMspert (Operator Function Model Expert System), the third portion of this research addressed the development of an operator's assistant: a stand-alone knowledge-based system that interacts with a human operator in a manner similar to a human assistant in the control of aerospace systems. OFMspert is an architecture for an operator's assistant that uses the OFM as its system and operator knowledge base and a blackboard paradigm of problem solving to dynamically generate expectations about upcoming operator activities and interpreting actual operator actions. An experiment validated the OFMspert's intent inferencing capability and showed that it inferred the intentions of operators in ways comparable to both a human expert and operators themselves. Next, OFMspert was augmented with control capabilities. An interface allowed the operator to interact with OFMspert, delegating as much or as little control responsibility as the operator chose. With its design based on the OFM, OFMspert's control capabilities were available at multiple levels of abstraction and allowed the operator a great deal of discretion over the amount and level of delegated control. An experiment showed that overall system performance was comparable for teams consisting of two human operators versus a human operator and OFMspert team.

Overall, this research has been very productive. In addition to the empirically validated proof-of-concept demonstrations of intent inferencing and operator aiding, this grant supported two Ph.D. theses, a master's thesis, and an undergraduate senior design project. Furthermore, the research received three awards and was the topic of dozens of invited conference presentations and a range of publications in international journals, newsletters, and technical reports. A summary of the papers and presentations is contained in Appendix A. Appendix B contain copies of the primary papers and technical reports.

* This research was also supported in part by NASA Goddard Space Flight Center grant NAS5-28575, Walt Truszkowski, technical monitor.

Introduction

The human's role as a supervisory controller of a complex, predominantly automated, dynamic system often leads to problems, including (1) an increased monitoring load; (2) a false sense of security whereby the operator trusts the automation to such an extent that any human intervention or checking seems unnecessary; and (3) "out of the loop" familiarity, i.e., a supervisory controller who acts primarily as a passive monitor rather than an active controller and who is less likely to respond as quickly or appropriately to system failures (Wickens, 1984). These and other difficulties with the increasing proliferation of automation have serious implications for the ability of operators to cope with emergency situations.

Although one path is to pursue increasingly sophisticated automation, eventually replacing human decision makers in complex systems, there is widespread acknowledgement that within the foreseeable future, humans will continue to play a critical role is ensuring system safety and efficiency (e.g., Chambers and Nagel, 1985). Thus, a major automation design issue is the use of automation to *enhance*, rather than *replace*, the human in the control and decision process. The goal is to use automation to amplify the human's strengths and compensate for the human's limitations. A complementary issue is to design automation so that the human can take advantage of the power of automated tools and systems, and yet remain alert to inherent and transient automation limitations.

This research explored the design of a computer-based assistant that amplifies the human's expertise and awareness of system evolution, yet compensates for known human limitations. It was based on the assumption that while some control tasks and functions can be fully automated, many important control functions require a design that incorporates human overrides. Thus, a computer-based assistant interacts dynamically with a human operator. However, as the name 'assistant' implies, the relationship between human and computer decision makers is one of superior to subordinate, with the human operator always in control. The computer serves as an assistant to whom the operator can dynamically delegate as few or as many control activities as s/he chooses.

This research proposed an architecture for a computer-based assistant that embodied these properties. Implemented in a NASA satellite ground control application, empirical evaluation demonstrated the extent to which the operator's assistant could dynamically *understand* operator intentions and correctly *interpret* operator actions. Using its understanding and interpretation, the assistant could then offer context-sensitive advice, reminders, and assistance in carrying out the control functions.

An operator's assistant raises many research issues (see for example the discussion in Chambers and Nagel (1985) and Rouse *et al.* (1987)). A critical issue is the requirement for a model of the human operator. This may be the single most important design issue because its successful resolution is a necessary condition for the rest of the system. The operator model provides the intelligence or the knowledge that an adaptive, computer-based assistant needs to assist intelligently a human operator in the control of a complex, dynamic system. The computer assistant uses the operator model to estimate correct and predicted operator state, i.e., to assess and predict operator functions, intentions, and performance given current system state.

This research used and extended the operator function model (OFM) methodology to define its knowledge about operator behavior. This report provides a brief summary of the operator function model (OFM) methodology, particularly how it is used in a computer-based operator's assistant. Next, because a proof-of-concept and subsequent validation depend on a domain of application, the NASA Goddard Space Flight Center satellite ground control system, MSOCC (Multisatellite Operations Control Center), is described together with its operator function model. Finally, the remainder of the report describes the operator-assistant architecture--OFMspert (Operator Function Model Expert System). First, an overview of the architecture is presented. A summary of an empirical study that validated OFMspert's intent understanding capability follows. Finally, ALLY, OFMspert augmented with control capabilities, is described together with the results of an experiment demonstrating that an OFMspert-human team controlled the satellite ground control system as well as a team comprised of two experienced human operators.

Operator Function Model

The operator function model (OFM) provides a flexible framework for representing operator activities in the context of dynamic systems (Mitchell, 1987). The OFM is a representation of how an operator might decompose and coordinate system control functions to meet system objectives and ensure system safety. An OFM represents the interrelations between dynamic system states and operator activities. Figure 1 depicts a generic OFM structure.

The OFM is a network in which nodes represent operator activities. Activities are structured hierarchically, representing primary operator control functions at the highest level and individual control actions at the lowest. Typical decomposition of activities is function to component subfunctions, subfunction to component tasks, and task to component actions. Actions can be both physical (e.g., an information query or system control command) or cognitive (e.g., information gathering, information processing, and decision making).

The OFM network is also heterarchic; that is, at the same level, there may be several activities that, given system state, are undertaken concurrently. The heterarchy accounts for the coordination and concurrent nature of operator activities as well as the operator's dynamic focus of attention.

The OFM represents the dynamic nature of the system-operator interaction by network arcs. The network arcs represent system events or the results of operator actions that initiate or terminate operator activities at various levels of the network hierarchy.

The operator function model is a prescriptive model that specifies nondeterministically a set of plausible operator functions and related activities given current system state and recent operator actions. As such, it provides two necessary components of a computer-based assistant: (1) the structure to represent knowledge about the system and operator activities; and (2) a mechanism to define expectations of operator activities given current system state. In other applications, the OFM has been successfully used to model, design, and control user interfaces (Mitchell and Saisi, 1987; Dunkler *et al.* 1988). In this research, the OFM provided the structure to organize the knowledge about the controlled system and related operator activities required by the computer-based operator assistant.

GT-MSOCC

In order to demonstrate and test the modeling and aiding techniques developed in this research a realistic test-bed was required. We used GT-MSOCC, the Georgia Tech-Multisatellite Operations Control Center. GT-MSOCC is an interactive, real-time simulation of MSOCC, a ground control system for NASA near-earth satellites located at NASA Goddard Space Flight Center in Greenbelt, Maryland. GT-MSOCC is a high fidelity simulation of the operator interface to the actual control system and was designed to support a range of research topics on operator modeling, training, and aiding (e.g., Mitchell, 1987; Mitchell and Saisi, 1987; Mitchell and Forren, 1987). The GT-MSOCC operator monitors the data transmitted by satellites to ensure data integrity, compensates for equipment failures and schedule anomalies, and responds to *ad hoc* support requests.

GT-MSOCC Configuration

GT-MSOCC supports 17 spacecraft (16 near-earth satellites and the Space Shuttle). Individual spacecraft have different requirements for the number and types of equipment needed to support communication and data transmission. An overview of the MSOCC equipment network is given in Figure 2. In general, all spacecraft use several NASA communication lines (Nascom lines) to transmit their data through a variety of computer and communications networks for data processing and recording. These configurations may include a Recorder Utility Processor (RUP), a Telemetry and Command computer (TAC), one or more Application Processor computers (AP), a Gate Way processor (GW), a Command Management System computer (CMS), and a Virtual Interface Processor (VIP). Finally, data are sent to a Mission Operations Room (MOR) or to a Shuttle Payload Facility (SPF). MORs and SPFs are spacecraft specific control rooms where operators monitor and control the spacecraft. It should be noted that RUPs, CMSs, GWs, and VIPs do not transmit data to subsequent components; rather they are 'endpoints' in the equipment configuration.

GT-MSOCC Operator Function Model

At the highest level, the GT-MSOCC operator function model depicts the major operator functions and the system events that cause the operator to transition among the functions or pursue concurrent functions (Figure 3). This level of description represents operator goals in the context of current system state. The arcs define system events that initiate a refocus of attention or the addition of a function to the current set of operator duties. The GT-MSOCC operator function model is presented in detail because this model defines the knowledge used to infer intentions and understand operator actions, and, subsequently, to identify the control abilities that the operator's assistant can offer.

Control of Current Missions. The default high-level function is to control satellites that are currently transmitting data (Figure 4). This function involves two primary (default) subfunctions: monitor the data flow at the equipment endpoints and monitor the hardware status. If a hardware failure occurs, the operator initiates a fault compensation subfunction to replace the faulty equipment. While monitoring data flow, if the operator suspects a problem with the amount or integrity of the data at one of the terminal points in the equipment network, s/he will initiate a troubleshooting/fault detection subfunction. To troubleshoot the operator examines individual components in the equipment network attempting to locate the cause of the problem; if a suspect component is identified, a fault compensation subfunction is initiated. Each subfunction is further defined by a collection of tasks which in turn are supported by operator actions (e.g., system reconfiguration commands or display requests).

Support for Unscheduled Requests. System events cause the operator to focus attention on additional or alternative high level functions. A request to the operator to configure the necessary equipment for an unscheduled spacecraft contact causes the operator to initiate the "configure to meet support requests" (Figure 5). This function consists of a variety of subfunctions including (1) checking the overall system to ensure capacity is not exceeded (GT-MSOCC can support up to five missions concurrently.); (2) checking the equipment requirements of the spacecraft in question; (3) attempting to identify available equipment; and (4) if all the conditions are met, indicating that the support request can be met and manually configuring the spacecraft's network. As in the control of current mission function, each subfunction is further defined by a collection of tasks which in turn are supported by operator actions.

Compensate for Automated Schedule Failures. Figure 6 depicts the subfunctions and tasks associated with the compensate for schedule conflicts operator function. The automated schedule that controls the allocation of specific pieces of GT-MSOCC equipment to specific spacecraft passes is always dated, i.e., the schedules are often as old as twelve hours and, as such, do not reflect the most recent system conditions. As a result, recently failed equipment or equipment originally scheduled but currently supporting another spacecraft is not taken into account by the automated schedule and control system. When the automated control system finds that the scheduled equipment is not available, the operator receives a request to manually reconfigure the equipment network, specifying an alternative component. Three tasks comprise the reconfigure function. First, the operator identifies the hardware components that are causing the conflict. Second, the operator attempts to find replacement equipment. Third, if successful, the operator uses this equipment to configure the equipment network. Since the spacecraft contact is relatively short (approximately ten minutes), it is important that the operator configure the equipment network as quickly as possible to avoid delays in contacting the spacecraft.

Deconfigure Manually Configured Network. Figure 7 depicts the subfunction and tasks associated with the deconfigure operator function. When the operator manually configures or reconfigures an equipment network for a spacecraft, the operator must manually deconfigure it. The system notifies the operator that the satellite contact is completed and tells the operator to deconfigure the equipment network manually. The operator types the appropriate deconfigure command and the equipment network is deconfigured. This is the operator's highest priority function since equipment is not available for use by other spacecraft until it is deconfigured. Thus, although the deconfigure operator function appears somewhat simple, the deconfigure function is critical to overall system effectiveness.

Browsing/Planning. Although the high-level representation of the GT-MSOCC operator function model includes a high-level planning/browsing function (e.g., Figure 3), the planning/browsing function was not implemented in OFMspert. The browsing/planning hierarchical decomposition is less straightforward than other high-level operator functions. For example, one approach accounts for *all* actions that cannot be interpreted by other activities as browsing/planning activities. Current research is beginning to examine browsing and planning functions and suggest mechanisms to support intent inferencing for these functions.

OFMspert

Background

An operator's assistant supports natural, real-time interaction with the human operator. Our goal is to design the computer component of the supervisory control system so that it mimics the functions that a human assistant performs. A computer assistant should be able to swiftly assume responsibility for control tasks that the human operator may delegate and to offer the human operator context-sensitive suggestions, advice, and reminders. The operator's assistant design can be characterized in terms of three principles: a stand-alone cooperative subordinate, dynamic task allocation, and dynamic intent inferencing. The stand-alone property is a characteristic of knowledge-based systems. Dynamic task allocation is a philosophy that underpins how human operators and knowledge-based systems cooperate in the control of complex dynamic systems. Dynamic intent inferencing is the component that provides intelligence. As such, it is at the very heart of the OFMspert design.

Dynamic Intent Inferencing

The intelligence and utility of the operator's assistant rest on its abilities to understand the operator's current intentions and to provide context-sensitive assistance in the form of operator aids (e.g., suggestions, advice, or reminders) or by assuming responsibility for portions of the control task. To ensure generalizability, the operator's assistant requires a well-defined knowledge structure that represents information about the controlled system and operator functions, as well as a problem solving structure to build a dynamic representation of operator intentions which reflects current system state and recent operator actions. There are several candidate models that might be used for both of these requirements (Geddes, 1985; Jones, 1988; Jones *et al.*, 1990). The OFMspert research uses the operator function model (Mitchell, 1987) to organize knowledge about the controlled system and related operator activities, and the blackboard model of problem solving (Nii, 1986) to build a current hypothesis of operator intentions. The next section summarizes how these models are used in the Actions Interpreter (ACTIN), the understanding component of OFMspert.

ACTIN (Actions Interpreter): OFMspert's Understanding Component

The Actions Interpreter (ACTIN) is the OFMspert component that is primarily responsible for dynamic intent inferencing. ACTIN dynamically builds a model (or "current best hypothesis") of operator intentions in the context of current system state and attempts to "interpret" operator actions in light of this understanding. The operator function model (Mitchell, 1987) forms the basis of ACTIN's knowledge about how system events trigger likely operator activities (e.g., a failure may initiate activities to compensate for that failure). Using the OFM, operator activities are structured in a hierarchy of functions, subfunctions, tasks, together with operator actions undertaken to support the activities.

The ACTIN model of intentions is implemented as a blackboard (Englemore, Morgan, and Nii, 1988; Nii, 1986). Thus, ACTIN consists of a blackboard data structure which contains the evolving representation of current operator intentions (i.e., functions, subfunctions, tasks, and actions), and a collection of knowledge sources that constructs, maintains, and assesses the blackboard data structure. As system triggering events occur in real time, ACTIN posts new functions, subfunctions, and tasks on the blackboard. As operator actions occur, they are posted on the blackboard and "connected" to any current tasks which they support. This process of "connection" is intent inferencing and provides OFMspert's understanding. Figure 8 depicts ACTIN's intent inferencing structure. More detail about OFMspert's blackboard is given in the section that follows.

OFMspert Architecture

The generic structure of OFMspert is depicted in Figure 9. Six functional components comprise the system. Each of these components performs certain functions necessary to an operator's assistant.

In general, the arrows in Figure 9 represent message-sending paths within OFMspert. A one-way arrow represents unidirectional communication; the tail of the arrow denotes the component that sends the message and the head of the arrow denotes the component that receives it. The receiver of a message may return a value to the sender. However, the reply path of a message is not shown. For example, the workstation component has no arrows leaving it, indicating that the workstation is a passive component that can only reply to messages but cannot generate any of its own. A two-way arrow represents bi-directional communication between the two components; in this case both components are capable of initiating communication. The message type in one direction may be of a different type than the message type of the other direction. A message between any two components may be a request for information or a request for the receiving component to carry out an internal event. Each OFMspert component defines its own internal events, and therefore appears as a black box to other system components.

In general, all new messages from the controlled system or information about operator actions enter OFMspert through the OFMspert interface. The interface decodes the messages, and new activities, called events, are sent to OFMspert's high level controller (HLC). The high level controller schedules and manages the execution of OFMspert's internal events. HLC events can be one of three types. The first is an update to the current problem space (CPS), OFMspert's representation of the controlled system. The second type of event is an enhanced normative model (ENM) event. The enhanced normative model contains normative information derived from the OFM. This module also contains OFMspert 's control properties. The third type of event is a blackboard event that changes ACTIN, OFMspert's blackboard.

The final OFMspert component depicted in Figure 9 is called the workstation, and it contains a semantic description of the actual workstation the human operator uses in the control of the system. The workstation itself does not initiate any events or activities; rather, it contains information other OFMspert components may need. The remainder of this section provides a detailed description of the OFMspert modules and control processes.

OFMspert Interface. The OFMspert interface, from an abstract point of view, is simply a black box that provides the logical communications between OFMspert and the controlled system (and human operator). At a very low level, there exists some form of hardware communications between the computers supporting the controlled system and OFMspert (if they are located on physically separate machines). At a higher level, the interface is responsible for decoding messages sent to OFMspert and encoding messages sent by OFMspert back to the controlled system. When the interface decodes a message received from the controlled system, it creates an event based on the message type and posts the event in the high level controller's event queue to be processed at the earliest time possible. For example, if an event occurs in the controlled system (e.g., an equipment failure or an operator action) that instantiates a new operator function, a message is sent to the OFMspert interface. The interface then creates a high level controller event that, when processed, will instanciate a function/subfunction/task structure that is placed on the blackboard. Abstractly, the OFMspert interface is an endless loop that continually decodes messages, creates events, and posts events in the HLC event queue.

High Level Controller (HLC). The high level controller is the central scheduler for events within OFMspert. HLC events are the result of activities initiated by either the operator or the controlled system itself. Messages from the controlled system are decoded by the OFMspert interface and cause OFMspert events to be created and scheduled for execution during OFMspert system cycles. A system cycle begins when the HLC initiates the execution of a scheduled event and ends when the sequence of actions required to carry out the event are executed. New events created during the current system cycle and placed in the HLC event queue to execute at future times are not considered part of the current system cycle. On each system cycle, the HLC executes the first event in its queue whose time is before or at the current system time. As seen in Figure 9, the arrows that point to the HLC originate at components that are capable of placing an event in the HLC event queue. Modules with arrows initiated at the HLC are OFMspert components that can execute an event when the HLC deems that one is ready.

Current Problem Space (CPS). OFMspert, in most facets of its operation, requires knowledge of the current status of the controlled system. This information is used to hypothesize operator functions, verify the semantics of operator actions, and assist in blackboard assessments. OFMspert's current problem space maintains an internal representation of the most prominent features of the current state of the controlled system. The CPS receives a message from the OFMspert interface whenever there is a relevant state change in the controlled system and uses this information to update its representation. Some less important status information may not be continuously stored within OFMspert due to space or speed constraints. When OFMspert needs additional status information about the controlled system it may ask for and receive this information on an as-needed basis. The latter interaction is depicted in Figure 9 by the arrow from CPS to the OFMspert interface.

Enhanced Normative Model (ENM). The enhanced normative model contains normative information about the controlled system and the OFM-derived information about operator functions and procedures. This component plays a critical role in intent inferencing and in OFMspert's ability to interact with the controlled system. The final ENM implementation contains all necessary information for both intent inferencing and system control.

The ENM contains the function, subfunction, and task activity trees that are used by ACTIN for intent inferencing. Activity trees are static knowledge stored in an ENM data base and indexed by system state changes. System events that initiate operator state changes are derived from the OFM and are therefore also static information. When a relevant system state change is decoded by the interface, an ENM event is placed in the HLC event queue. This event is executed on the next system cycle and the ENM uses the proper system state change index to retrieve the appropriate activity tree. Then, an ACTIN event to update the blackboard representation is placed in ACTIN's event list. System events that cause new task information to be sent to ACTIN are referred to as initiating conditions.

Operator actions, which are encoded into messages and sent to OFMspert by the controlled system, are decoded by the interface, placed on the HLC events queue, and eventually sent to the enhanced normative model. The ENM converts these actions to the proper blackboard form and creates an ACTIN event to update the representation.

ACTIN (Actions Interpreter). ACTIN is OFMspert's blackboard and it is responsible for the intent inferencing functions. Like most blackboards, ACTIN has three primary components: a blackboard data structure, knowledge sources, and blackboard control. Figure 10 depicts the ACTIN component in more detail.

ACTIN's blackboard is a hierarchical structure of nodes defining functions, subfunctions, tasks, and actions. Blackboard nodes on the higher three levels, i.e., function, subfunction, and task nodes, are usually model-derived; thus, some system event, i.e., initiating condition, triggers an OFMspert cycle that posts nodes defined by an enhanced normative model activity tree. Action nodes are always data-derived; thus, a blackboard action node is always the result of an actual human operator action that was decoded at the interface, processed by the HLC, interpreted by the ENM, and posted and processed by ACTIN's event list and corresponding knowledge sources. Occasionally there are data-derived function, subfunctions, or task nodes. Data-derived nodes are used by ACTIN to infer a function, subfunction, or task from one or more operator actions not fully understood in the context of the current blackboard representation.

ACTIN, like HASP (Nii *et al.*, 1982), contains three hierarchically related types of knowledge sources (KSs): strategy, activator and specialist. The specialist KSs contain the domain-specific knowledge needed to manipulate the blackboard data structure; these KSs construct the blackboard representation (current best hypothesis) and perform blackboard assessments. The activator KSs select the specialists and together form part of the blackboard control structure.

Within the blackboard there are two major types of events: construction and maintenance of the operator representation, and assessment of the representation to evaluate operator performance with respect to the normative procedures prescribed in the ENM. Every time the HLC schedules an ACTIN cycle, the strategy knowledge source is the first control entity called. The HLC has no control over what type of event the blackboard executes; this control resides in the strategy KS. Every time the ENM schedules a blackboard cycle, the strategy KS determines which type of event to focus on next. After selecting an event, the strategy knowledge source calls an appropriate activator knowledge source. Events are one of two types: maintenance or assessment. For each event type, there is a corresponding activator knowledge source. The activator KS chooses the most appropriate specialist KS to process the event.

The strategy KS analyzes three lists to determine what event to focus on next. These lists are the clock-events list, the events list, and the problems lists. Clock-events exert the greatest influence on the blackboard control process. The clock-events list contains events scheduled for future execution, for example, a periodic assessment of some control task. All events in the clock-events list that are scheduled to perform at or before the current time are immediately executed. The events list contains events that are generated by the ENM while interpreting system state changes and operator actions. All new information is placed in the blackboard events list and, thus, provides the strategy KS with a central location for finding new events on which to focus. During a single ACTIN cycle, all events on the events list are processed. The problems list contains all operator action nodes that could not be understood when they arrived in ACTIN, i.e., actions that were posted on the action level but could not be connected to one or more task nodes at the task level. Unconnected actions are put in the problems list in the hope that future operator actions or system events can help to disambiguate their meaning. The problems list is examined after all ready clock-events and events list events in the current cycle have been processed. Any item in the problems list that is subsequently explained is removed and processed.

Information Fusion. The first requirement of the intent inferencer is to construct a representation of the operator's current state. To do this, both model-derived and data-derived information are posted and manipulated on the blackboard data structure by knowledge sources. The relationship between the objects at different levels is specified by named links generated by the knowledge sources. The objects and links between them generally form a representation that pictorially resembles a forest of rooted trees. Each 'tree' represents a function and its associated subfunctions, tasks, and actions. When a new action enters ACTIN and is placed on the blackboard, the KSs attempt to connect it to all possible tasks that the action may support. An action that connects to tasks located in different activity trees is assumed to support all active functions. However, this may or may not be true. When new information enters ACTIN, there is often insufficient information to determine which task(s) the action is intended to support. Our policy is to *maximally connect* new actions, i.e., connect an action to all possible tasks that it might support. The problem solving strategy opportunistically disambiguates the situation at a later time.

Information Removal. An important issue in constructing and maintaining the operator's current state representation within ACTIN is that of knowing when to remove information from the blackboard. At some point, the utility of individual pieces (or groups) of information becomes negligible, i.e., old information becomes outdated or obsolete. To facilitate current maintenance and assessment operations, information with low utility should be removed. The dilemma arises in determining when information has negligible value. Removing information that is still needed may cause future assessments to hypothesize incorrectly that an operator error has occurred. To prevent this situation, information removal is governed by a *strategy of least commitment* in which the decision to remove information is delayed until it is absolutely certain that the information has no value.

OFMspert uses well-defined system events as the primary means of determining when information should be removed from the blackboard. The enabling conditions for transitions between nodes at the heterarchic level of the OFM include those that cause information removal. Within OFMspert, enabling conditions that terminate an operator function cause an assessment of the function, subfunction, or task and information removal of the corresponding blackboard nodes. Action nodes are removed only when they are no longer connected to any current tasks, i.e., no longer in support of any current functions or subfunctions. Maximal connection of actions ensures a conservative information removal strategy.

Blackboard Assessment. In OFMspert, knowledge sources, derived from the OFM of the controlled system, carry out assessments. Assessment knowledge sources are invoked by blackboard control to determine the extent to which operator actions support currently hypothesized functions, subfunctions, and tasks. Assessments are always made in the context of a particular functions or subfunctions. Initially, the result of an assessment is a detailed evaluation written to a file. The second phase of this research, OFMspert with control capabilities, uses assessments in real time to provide the basis for active operator aiding.

An Example of OFMspert Intent Inferencing Operation. A general example of OFMspert intent inferencing is presented below. However, first we must distinguish between initiating and terminating conditions. Initiating conditions "start something" in the controlled system and thus will cause the posting of new operator function, subfunction, and tasks on the blackboard. Terminating conditions "finish something" in the controlled system and thus will cause the assessment and removal of now-obsolete functions, subfunction, tasks, and any connected actions from the blackboard. It is possible that operator actions and changes in the controlled system are neither initiating nor terminating, e.g., an information request. It is also possible that the same action or system change can be both terminating and initiating--that is, finish one thing and start something else in the controlled system, e.g., a manual configuration action terminates the configure function and initiates a control of current mission function.

Suppose the operator executes an action in the controlled system. This action is coded into a message and sent to the OFMspert interface, which parses the message and schedules the appropriate enhanced normative model event for handling this input and, if necessary, schedules another event to update the current problem space. "Scheduling" here means adding an event to the high level controller's event queue in time-sorted order. The event queue is repeatedly checked to see if it is time for its first event to "fire." When that time comes, the message to begin processing will be sent to the ENM. The ENM will generate events to be processed by the blackboard. The exact nature of these events depends on whether the operator's action was initiating or terminating. For any operator action, the ENM will always add a "post action" event to the blackboard's event list. If the action is initiating, the ENM will also generate the appropriate functions subfunction, and task structure and add a "post activity tree" to the blackboard's event list. If the action is terminating, the ENM will add "assess" and "information removal" events to the blackboard's event list. If the action is both initiating and terminating, the ENM will create and add "assess", "information removal", and "post activity tree" events to the blackboard's event list. After this direct interaction with the blackboard, the ENM schedules an event in the high level controller's event queue to actually carry out the events just added to the blackboard's event list. Subsequent OFMspert system cycles update the current problem space and the blackboard.

Summary. The generic OFMspert consists of six major components. The blackboard architecture permits a hierarchical representation of the operator's inferred current functions, subfunctions, and tasks. This dynamic and hierarchic organization of the blackboard parallels the structure of the operator function model. The blackboard data structure naturally and efficiently represents operator actions and controlled system events as a structure of functions, subfunctions, tasks and actions. The knowledge sources are convenient, well-organized structures that represent domain knowledge and can assess the overall effectiveness of how the operator coordinates control actions to meet higher level system goals. Details of the software engineering design and specification are given in Rubin *et al.* (1988).

The effectiveness of an operator's assistant depends on the validity of its model of operator intentions and its interpretation of operator action. Thus, the next step in this research project addressed the validity of OFMspert's intent inferencing component.

Validation of OFMspert's Intent Understanding

This phase of the OFMspert research assessed the degree to which OFMspert possessed the knowledge or understanding to intelligently assist an operator. Validation of intent inferencing assures that the system is correctly inferring the intentions of the human operator. Within the context of ACTIN's structure of intentions, this means that the system infers support for the same tasks (and by extension, subfunctions and functions) as the human, given the same set of operator actions. The "human" in this case can be a human domain expert performing a *post-hoc* analysis, or the human operator giving a concurrent verbal account of intentions. Thus, the experimental validation of ACTIN's intent inferencing was conducted in two studies. In Experiment 1, a domain expert's interpretations of operator data were compared to ACTIN's interpretations of those same actions on an action-by-action basis. In Experiment 2, concurrent verbal protocols were collected from GT-MSOCC operators. Statements of intentions for each action were compared to ACTIN's interpretations.

In experiment 1 a domain expert hypothesized intentions from the data of ten GT-MSOCC operators. These ten operators were the original GT-MSOCC subjects (Mitchell and Saisi, 1987; Mitchell and Forren, 1987) in a GT-MSOCC control condition. The last three sessions of each subject were used in this analysis, yielding a total of 30 hours of experimental data. The data from these subjects consisted of various logfiles that detailed the events that occurred during the experimental sessions. Perfect state information (i.e., what missions were currently configured, what equipment failures existed) was available, as well as every action by the operator. The domain expert used these logfiles as the basis for interpretations.

The second experiment compared subject verbal protocols to ACTIN interpretations. This experiment used verbal data as a measure of subjects' intentions in controlling GT-MSOCC. Verbal protocol data have been extremely useful in the development of human-machine models. Verbal data can be treated as any other class of data that proposes a correspondence between observed behavior and predictions of a model; in fact, verbal reports may be a preferred source of data because of the richness of information available (Anderson, 1987, Ericsson and Simon, 1984; Miller, Polson, and Kintsch, 1984).

The data in Experiment 2 consist of verbal protocols from two subjects for seven GT-MSOCC sessions. Both subjects were trained in the standard control condition (see Mitchell and Saisi, 1987). The subject participated in 12 experimental sessions. The first five were considered training. During sessions 6 through 12, the subject controlled the GT-MSOCC system while verbalizing intentions, actions, and problem solving activities. The subjects were instructed to verbalize why they performed every action in the system; occasionally the experiment prompted the subjects with "Why?" when they failed to verbalize an intention for an action.

The verbal protocols were transcribed and interpreted by the experimenter. (Complete segmented protocol transcriptions are available in Jones, 1988.) Intentions were coded from the verbalizations in several ways. The most straightforward was a direct statement of intent (e.g., an utterance of the type "I'm asking for this display because I want to find out this."). A variation of this straightforward verbalizing was of the type "I want to do this", immediately preceded or followed by the subject's typing in the relevant command. A less direct method of inferring intentions involves examining what information the subject used as a result of the action.

The data from the two experiments consist of corresponding sets of interpretations for the same actions. One set of interpretations is from ACTIN, the other from a human. These data can be considered paired observations, since for every action there are two interpretations; the same entity (action) is observed under two experimental conditions: ACTIN and human interpretations.

Data summarizing the results of these two experiments are given in Figures 11 and 12. Overall, ACTIN's intent inferencing ability compared favorably to human interpretations of the same actions, both in the expert's analysis of data files and the verbal protocol analysis. The observed differences were primarily due to model error and can be remedied in part by some extensions to the operator function model and to ACTIN. Many mismatches occurred because the GT-MSOCC OFM did not represent planning and browsing (e.g., information requests to support upcoming events). Certain classes of actions--notably important system configuration commands--were very well-matched. More detail is available in Jones (1988) and Jones *et al.* (1990).

ALLY: OFMspert with Control Properties

OFMspert components coordinate their functions to build a representation of the operator's current functions and associated subfunctions, tasks, and actions. In the initial phase, OFMspert had the knowledge about *how* to control the system, e.g., how to troubleshoots or compensate for failures, but did not have control capabilities. Given an effective model of operator intentions, the next step in the OFMspert research made OFMspert less passive, enabling it both to engage in system control and to interact with the operator in the mode of an assistant. The next sections describe ALLY, OFMspert enhanced with control properties, and the empirical evaluation of ALLY as an operator's assistant.

Characteristics of an Operator's Assistant.

The two primary characteristics of an effective operator's assistant are understanding and control. ACTIN, OFMspert's understanding component, was shown to be an effective architecture for postulating and interpreting operator activities. Given a reliable understanding component, the next OFMspert phase focused on providing OFMspert with system control properties which the human operator could initiate, refine, and terminate. OFMspert control properties were intended to be as effective as those of a human assistant and include interactive refinements between OFMspert and the human operator that emulated the manner in which experienced teams of human operators interact.

ALLY, like OFMspert itself, is both a theory of interaction and an architecture in which the theory is implemented and evaluated. The theory underpinning the ALLY architecture is based on a literature review and a case study of two human operators jointly controlling a dynamic system

The literature suggests three principles of effective cooperation. First, operators use multiple mental models to represent knowledge of the physical system, their own activities, and their knowledge of other team members. These models are maintained at multiple levels of abstraction. The appropriate level is dynamic and determined by a cooperation strategy. The second principle is that cooperation includes "cognitive balancing"--dynamically balancing the workload among team members given current system demands and operator availabilities. Finally, the literature suggests that cooperation is flexible. Activities between operators are dynamic and interactive.

Case Study of a Team of Human Operators.

ALLY is designed to assist the GT-MSOCC operator in carrying out all of the GT-MSOCC supervisory control functions. The design was based on a model of the GT-MSOCC operator control functions and attempted to duplicate the capabilities of a human assistant observed in the case study. The case study documented the interaction of a team of two experienced operators controlling GT-MSOCC. During operation, verbal protocols of the two-person team were collected.

In the case study, the relationship between the human operator and the human assistant was one in which the operator supervised the assistant. The assistant, however, was not passive. The assistant understood the cognitive complexities of the operator functions and actively monitored the system for failures, and, when necessary, initiated fault detection and compensation activities. The assistant helped the primary operator by issuing reminders of incomplete activities . The primary operator dynamically delegated the tasks to the assistant. At times, the responsibility for a whole function would be given to the assistant; at other times, the second operator assisted the primary operator in performing a function. The two operator team effectively controlled GT-MSOCC. Together, the two operators controlled the system such that overall system performance was better than performance for a single operator.

ALLY Architecture

The operational concept in ALLY's design is that ALLY functions in a manner similar to a human assistant. The operator has complete control over ALLY and can delegate as few or as many of the control responsibilities to ALLY as desired. ALLY is not passive, however; it also actively monitors the system and initiates troubleshooting activities when necessary.

ALLY interacts with the GT-MSOCC system in a distributed fashion (Figure 13). The distributed architecture simulates the environment of a human assistant. ALLY, like the human assistant, performs independently of the GT-MSOCC system. This architecture is consistent with the concept of an assistant that executes autonomously in its own environment.

ALLY has the same information as the human operator. ALLY receives messages from the GT-MSOCC system indicating changes in system state. As with the human operator, ALLY's knowledge of system events always lags somewhat behind the actual state of the system. For example, if the operator replaces a failed component, ALLY does not update its representation until GT-MSOCC finishes the replace and notifies both the operator and ALLY of the change.

ALLY receives some information automatically, primarily information about changes in system state. ALLY requests other information from the system. Time and speed problems is a distributed architecture prevent an autonomous agent from having and maintaining complete knowledge about the controlled system. For the GT-MSOCC application, ALLY, like the human operator, requests satellite and equipment schedule information on an "as needed" basis. When ALLY needs schedule information to perform a specific activity (e.g., find a replacement), ALLY requests the appropriate schedule from the GT-MSOCC system.

ALLY Operator Interface

In order to interact with ALLY, the three monitor GT-MSOCC workstation was augmented with an ALLY workstation. The ALLY workstation consists of a computer, a CRT and a mouse. The operator uses the workstation to delegate tasks to ALLY and ALLY uses it to communicate with the operator.

The ALLY display consists of three primary windows (see Figure 14). The top window displays the current time. The middle window consists of control buttons that the operator uses to delegate control tasks to ALLY. The bottom window is the Message Transcript window. ALLY uses this window to communicate with the operator. In the Message window, ALLY precedes each message with a time stamp indicating when the message was written; for critical messages, ALLY uses an audio signal to notify the operator and precedes the message line with asterisks.

ALLY's Control Capabilities

The operator delegates activities to ALLY by clicking the mouse on one of the control buttons. The tasks defined in the control buttons are based on the operator function model of the GT-MSOCC operator. The "Check Telemetry", "Failure Support", "Question Support", "Reconfigure Support", and "Deconfigure Support" control buttons relate directly to the five control functions defined by the GT-MSOCC operator function model. In addition, ALLY provides "Mission Support" and "Equipment Support" information to the operator. These classes of support were suggested by the case study. The operator uses the "Interrupt" control button to stop ALLY from carrying out a task. This interrupt capability provides the operator with complete control over ALLY. Not only can the operator decide which tasks to delegate to ALLY, the "Interrupt" control button provides the operator with the capability to flexibly 'de-allocate' tasks. Gaines and Shaw (1983) described this "reset" capability as an important part of a user interface; Fox (1987) identifies it as an essential part of interaction in problem solving and tutoring.

Each of the control functions defined by the control buttons, except for "Interrupt", has an associated set of subtasks. These tasks reflect different levels of abstraction and/or aggregation at which the operator can interact with ALLY. The operator can delegate to ALLY as much or as little responsibility as desired.

ALLY uses a series of "pop-up" windows to define the range of subtasks. When the operator selects one of the control buttons, ALLY displays a submenu. If at any point during task specification, the operator makes a mistake or changes his/her mind and decides not to have ALLY perform the task, the operator can click outside of the menu and ALLY stops the task specification process. This "repair" capability keeps the operator in complete control of the conversation (Fox, 1987).

When ALLY completes an assigned task, it checks to see if the overall operator or control function the task was supporting has been completed. If the function is incomplete and ALLY knows that it can now complete the function, ALLY offers to do so. For example, assume that Application Processor 4 (AP4) failed. The operator tells ALLY to find a replacement for AP4. ALLY determines that AP1 can be used and tells the operator; then, ALLY offers to perform the actual replacement task. The operator can either authorize ALLY to perform the task or do it him/herself.

The principle is that ALLY understands the operator's functions in the system and knows that a related task will need to be undertaken eventually. While ALLY only performs delegated system control tasks, it understands the overall control functions and thus, can assess the degree to which they are completed and offer timely assistance. This behavior is similar to the interaction between the two human operators controlling the GT-MSOCC system. The human assistant would consistently offer to complete a function if only part of the tasks were performed. This flexibility does not reduce any of the operator's control over ALLY. Rather, it permits the operator to balance the workload in the context of current system state.

The following sections describe the functionality of each ALLY control button and how the operator uses the buttons to delegate tasks to ALLY. The relation between control buttons and the GT-MSOCC operator function model is also described.

Mission Support. The operator uses "Mission Support" to request ALLY to provide information about a specific mission. At this time, "Mission Support" consists of one task; the operator can ask ALLY to identify the time a current mission is scheduled to be completed.

"Mission Support" can be used to assist in several operator tasks defined in the operator function model. For a mission with a failed component, ALLY can determine how long a replacement component is needed by checking the duration of the mission(s). The duration of current missions also supports the "Check System Constraints" subfunction of the "Respond to Unscheduled Support Requests" (i.e., determine if the maximum number of concurrent missions supported by GT-MSOCC will be exceeded).

Figure 15 shows the menus ALLY uses for "Mission Support". When the operator clicks on "Mission Support", ALLY displays a menu showing the tasks that the operator can delegate to ALLY. When the operator selects "Show Mission Time Down", ALLY displays a list of the current missions and asks the operator to select one. For the selected mission, ALLY determines its termination time from the OFMspert Current Problèm Space and reports the time to the operator in the Message Transcript window.

Equipment Support. The operator uses "Equipment Support" to ask ALLY to provide information about equipment and classes of equipment. Figure 16 shows the various ALLY menus for this function. These tasks, i.e., "Check if Equipment Available" and "Find a Free Equipment", support several of the GT-MSOCC control functions, including "Identify Replacement Equipment" for both the "Fault Compensation" and "Compensate for Schedule Conflict" functions, and "Identify Equipment" for the "Unscheduled Support Request" function.

For "Check if Equipment is Available", ALLY determines if a specific piece of equipment is available for a specified period of time (e.g., AP1 available for 3 minutes). In a series of pop-up menus, ALLY asks the operator to define the equipment class, component number, and duration. ALLY then asks GT-MSOCC for the equipment schedule for the designated component. When ALLY receives the schedule, it checks to see if the equipment is available for the time desired and tells the operator the answer in the Message Transcript window.

For "Find a Free Equipment", ALLY identifies an available component of a specified class for a specific duration. Again, ALLY uses a series of pop-up menus with which the operator defines the equipment class and duration. Then, ALLY requests a schedule from GT-MSOCC for all components of that equipment class. ALLY searches the schedules to identify a component that is free for the duration requested. The results of ALLY's search, either successful or not, are written in the Message Transcript window.

Check Telemetry. In the GT-MSOCC operator function model, "Control Current Mission" consists of three major subfunctions: "Monitor for Hardware Failures and Software Problems", "Detect the Cause of Software Problems", and "Compensate for Failed or Degraded Hardware". These subfunctions are, in turn, abstracted into several operator tasks . ALLY decomposes "Control of Current Missions" into two activities: "Check Telemetry" and the "Failure Support".

Figure 17 depicts the relationship between the operator activities described in the operator function model and ALLY's "Check Telemetry" function. "Check Telemetry" is divided into two activities, monitor the network endpoints (e.g., mission operations room (MOR)) and troubleshoot. The operator can delegate either of these activities to ALLY. The operator can delegate two monitoring tasks to ALLY: "Monitor Endpoints (e.g., RUP3)" and "Monitor MORs/SPFs". Both activities directly relate to the OFM monitor subfunction. The operator can delegate three troubleshoot tasks to ALLY: "Troubleshoot Interior Points", "Troubleshoot All Equipment", and "Troubleshoot a Specific Equipment". The troubleshoot tasks relate to the operator function model's "identify degraded hardware" subfunction.

Taken together, these ALLY activities provide the operator with the flexibility to choose the extent of the troubleshooting activity that s/he delegates to ALLY. With the exception of the "Troubleshoot a Specific Equipment" task, when the operator delegates any of the monitor or troubleshoot tasks to ALLY, ALLY asks the operator to specify which of the current missions to check. ALLY maintains a collection of the current missions in OFMspert's Current Problem Space and provides the operator with a list of all of these missions, together with an option to check all of the missions. The operator can, therefore, ask ALLY to focus on a specific mission or on all of the missions.

Failure Support . "Failure Support" consists of three activities: "Find a Replacement", "Replace a Failed Equipment", and "Issue an Alert Message". Figure 18 depicts the relationship between OFM and ALLY. The ALLY activities are structured to provide the operator with the capability to delegate a range of fault compensation tasks to ALLY.

The first ALLY activity in Figure 18, "Find a Replacement", corresponds to the "Identify Replacement Hardware" operator control activity. When the operator delegates this task to ALLY, ALLY requests the appropriate equipment schedules from the controlled system and attempts to identify a replacement. In the "Find Replacement" activity, ALLY does not replace the component; rather, ALLY examines the equipment schedules and reports the results to the operator. As with a human assistant, if ALLY finds a replacement, ALLY then offers to replace the failed component. If ALLY cannot find a free replacement, it offers to send the appropriate alert message to the GT-MSOCC system.

"Replace a Failed Equipment" corresponds to the "Manually Reconfigure" operator control activity in the operator function model. When the operator delegates this task to ALLY, ALLY first checks to see if it has already found a replacement (i.e., the "Identify Replacement Hardware" task). If ALLY finds a replacement, it issues the replace command. If ALLY does not find a replacement, it offers to send the appropriate message back to the controlled system. When the operator selects the "Issue alert" option, ALLY tells the controlled system that no replacement equipment is available.

When the operator delegates any of these three tasks to ALLY, ALLY uses a series of pop-up menus to allow the operator to identify the failed component (Figure 19). ALLY assumes that the failure is one of its currently hypothesized failures and therefore displays a list of the currently hypothesized equipment failures. The operator, however, might know about other failed components; consequently, the "Other" menu option provides the operator with the capability to identify a failure that ALLY does not list. When the operator selects "Other", ALLY uses pop-up menus to let the operator identify a new failed component. If ALLY does not know about any failures, ALLY immediately goes to these menus to have the operator specify the failure.

"Failure Support" also allows the operator to update ALLY's set of suspected failures. Occasionally, ALLY may misdiagnose a software failure. A failure identified by ALLY may have been the normal fluctuations in the system. The operator uses "Remove a Failure" to tell ALLY to remove a component from its failure list.

Question Support. "Response to Unscheduled Support Requests" is a function that consists of four related activities: "Determine Feasibility of Support", "Determine Equipment Needed", "Identify Equipment", and "Manually Configure Mission". Each of these activities are supported by various tasks under ALLY's "Questions Support" control button. "Question Support" consists of a range of activities that the operator can ask ALLY to perform: "Show Question", "Show Equipment Needed", "Check Mission Schedules", "Check Equipment Schedules", "Determine Answer, "Answer Question", and "Configure Mission". Figure 20 depicts how each of these tasks supports the corresponding subfunctions in the operator function model.

"Question Support" has a range of activities from very simple support (e.g., "Show Question") to the complete set of activities required by the function (e.g., "Answer Question"). In this manner, the operator decides how much or how little support ALLY provides. When the operator selects "Show Question", ALLY restates the support request. "Show Equipment Needed" corresponds to the "Determine Equipment Needed" subfunction. When asked, ALLY provides the operator with the mission's equipment requirements. "Check Mission Schedules" corresponds to the "Determine Feasibility of Support" subfunction. When the operator delegates this activity to ALLY, ALLY check three system constraints. First, ALLY checks the current system state to see if five missions are already being supported. If so, the request cannot be supported. Next, ALLY requests the GT-MSOCC schedule to see if scheduling this mission will cause more than five missions to be supported concurrently in the time frame under consideration. Finally, ALLY requests the mission schedule to see if the mission is already scheduled during the time frame of the support request. If ALLY determines that the request cannot be supported, ALLY tells the operator and offers to answer "NO" to the question. If the mission can be supported, ALLY reports this to the operator in the Message Transcript window.

The "Identify Equipment" operator subfunction corresponds to ALLY's "Check Equipment Schedules". When requested, ALLY attempts to identify the available equipment that can be used to support the mission. For each class of equipment needed by the mission, ALLY requests schedules from GT-MSOCC and identifies specific components that are free and unscheduled. If ALLY finds that all of the equipment is available, ALLY reports this result in the Message Transcript window together with the specific pieces of equipment. If ALLY finds that some of the necessary equipment is not available, ALLY tells the operator what is not available and offers to answer "NO" to the question.

ALLY's "Determine Answer" activity corresponds to two operator subfunctions: "Determine Feasibility of Support" and "Identify Equipment Needed". When the operator selects "Determine Answer", ALLY first checks the system constraints. If ALLY finds that the constraints are satisfied, ALLY proceeds to check the equipment schedules to identify the specific components that can be used to support the mission. If ALLY finds the necessary equipment, ALLY then indicates to the operator that the mission can be supported and specifies an equipment network that can be used. ALLY then offers to configure the mission; the operator may ask ALLY to configure the mission or do it him/herself. If ALLY finds that the mission cannot be supported, ALLY reports the result and reason in the Message Transcript window and offers to answer "NO" to the question.

"Answer Question" is similar to the previous activity, except that in this case, ALLY answers the question, i.e., ALLY sends a message to GT-MSOCC. Then, if the answer is "yes", ALLY offers to configure the mission.

The "Configure Mission" task corresponds to three of the operator subfunctions: "Determine Feasibility of Support", "Identify Equipment Needed", and "Manually Configure Mission". When the operator selects "Configure Mission", ALLY performs the same activities as "Answer Question", except, when the answer is affirmative, ALLY automatically configures the mission. If the mission cannot be supported, ALLY reports to the operator and offers to send an alert message to GT-MSOCC.

ALLY activities requiring interaction with the controlled system are somewhat difficult for ALLY (see the discussion of ALLY limitations in a subsequent section), thus, care was taken to structure the tasks to allow the operator to delegate a range of responsibilities. In this way, the operator can choose how to use ALLY's advice and has control over the type of support ALLY provides. For example, the operator may ask for a recommendation or may actually delegate the entire replacement task to ALLY.

Reconfigure Support. The operator uses "Configure Support" to delegate activities to ALLY related to the "Compensate for Automated Schedule Failure" operator control function. This function consists of three subfunctions: "Determine What Hardware Component is Unavailable", "Identify Replacement Hardware", and "Manually Reconfigure". Each of these subfunctions are supported by menu options in "Reconfigure Support". The menu options are "Find Replacement Equipment", "Reconfigure the Mission", and "Issue an Alert". Figure 21 depicts the relationship between the OFM and ALLY for reconfigure support.

ALLY maintains a list of pending requests. When the operator delegates any of these activities to ALLY, ALLY asks the operator to identify the pending mission (i.e., the mission that was unable to be configured) from the list ALLY displays.

"Find Replacement Equipment" corresponds to two operator subfunctions: "Determine What Hardware Component is Unavailable" and "Identify Replacement Hardware". When the operator selects "Find Replacement Equipment", ALLY identifies the equipment that is unavailable from information contained in OFMspert's Current Problem Space. ALLY then attempts to identify replacement equipment. For a failed component, ALLY requests schedules for that component's equipment class from GT-MSOCC. If a replacement cannot be found, ALLY informs the operator and offers to issue the appropriate Alert message. If a replacement is found, ALLY tells the operator and offers to reconfigure the mission.

"Reconfigure Mission" combines all three of the operator subfunctions. When the operator delegate this activity to ALLY, ALLY first identifies the unavailable equipment, then attempts to find replacements, and, if a replacement is found, configures the mission. If ALLY cannot reconfigure the mission because there is no replacement component, ALLY tells the operator and offers to issue the appropriate Alert message.

Finally, when the operator selects "Issue an Alert", ALLY sends the appropriate Alert message to GT-MSOCC. This activity supports the "Configure Mission" subfunction since a task for this subfunction is to issue an alert if the mission cannot be configured.

Deconfigure Support. "Deconfigure Support" corresponds to the "Manual Deconfigure Mission" function in the operator function model. This function consists of two activities:

"Identify Hardware String" and "Remove Components". "Deconfigure Support" has only one activity, "Deconfigure Mission". This task corresponds to both of OFM deconfigure activities; Figure 22 depicts the relationship between ALLY and the OFM deconfigure function.

When the operator selects "Deconfigure Mission", ALLY asks the operator to specify the mission to be deconfigured. ALLY generates a list of missions from OFMspert's currently hypothesized deconfigure subfunctions. ALLY then issues the deconfigure command to the controlled system and tells the operator that the mission is deconfigured.

Interrupt Button. The last control button is "Interrupt". The operator uses "Interrupt" to stop ALLY from performing a delegated activity. The operator interrupts ALLY by clicking the "Interrupt" control button. ALLY stops processing the current task and does not report any intermediate results to the operator.

Summary. The control buttons were designed with specific principles in mind. First, and foremost, the operator is provided a great deal of flexibility to decide how much or how little support ALLY gives. The operator has complete control over the extent of ALLY's system control activities. The operator may ask ALLY to determine an answer and then the operator may carry out the function him/herself; or, the operator may ask ALLY to perform the entire function.

Second, the definition of the system control activities is well defined. ALLY only performs the task that the operator delegates, and nothing more. For example, "Answer Question" means to answer the support request question and nothing more. It does not imply that ALLY should configure the mission if the answer is yes. In this manner, both the operator and ALLY understand exactly what is meant by the set of mutual activities and there are no hidden meanings.

Third, while ALLY only performs the activities that the operator requests, ALLY's model of the operator and the operator control function also permits ALLY to offer assistance and reminders with respect to the current operator functions. For example, if a control activity is incomplete, ALLY offers to complete it, if it can. "Respond to Support Request", for example, does not end with answering "YES" to the support request question; the mission must also be configured. ALLY anticipates that the operator might request ALLY to configure the mission and offers to do so. It is important to note, however, that timely offers of assistance or reminders do not limit any of the operator's control flexibility; the operator may always say "NO" to ALLY's offer of assistance.

ALLY's Automatic Tasks

In addition to system control activities requested by the operator, ALLY also performs several operator support tasks automatically. For the GT-MSOCC domain, ALLY monitors, and when appropriate, troubleshoots the equipment networks. ALLY also automatically monitors deconfigure a mission's equipment network, manually reconfigure the network of a scheduled mission, and manually configure an unscheduled

After some period of time, ALLY checks to see if the appropriate operator action has been completed. If it has not been completed, ALLY reminds the operator of the missed system request and offers to perform the task. The operator may then tell ALLY to perform the task or choose to do it him/herself.

If the operator tells ALLY not to perform the task, ALLY does not provide additional reminders. Repeated reminding was intentionally not implemented to prevent ALLY from becoming a nuisance or a nag, if for some reason the operator intentionally choose not perform a task (Knaeuper and Morris, 1984).

ALLY's Limitations

As with any cognitive system, either human or artificial, ALLY has strengths and limitations. ALLY's strengths are speed and computational processing capabilities. ALLY limitations in the GT-MSOCC domain are incorrect identification of software failures and a limited ability to successfully undertake planning.

ALLY does not accurately determine all software failures; it makes both Type I (i.e, false alarms) and Type II (i.e., missed failures). Since ALLY does not have perfect knowledge of the state of the system, it makes incorrect inferences about the data quantity and quality in the equipment networks. Errors occur when ALLY is testing hypotheses about the status of a particular component (i.e., normal or failed). ALLY can generate false alarms when an equipment has not failed and can miss a failure that has occurred. These errors are not intentional but are due to misinterpretation of the random noise in the system associated with normal fluctuations in the data flows.

ALLY's other limitation is planning. In GT-MSOCC, ALLY does not always accurately perform activities that support unscheduled support requests. These activities require ALLY to identify equipment that will be available for the duration of the support request. To determine feasibility of support, ALLY needs to know three things: 1) *exactly* when the configure command will be issued; 2) how long it takes to identify all of the needed equipment: and 3) much time the operator takes before issuing the configure command.. ALLY must estimate these time. These estimates, plus the duration specified in the support request, define an exact planning window that ALLY uses to determine if the support request can be scheduled.

To carry out unscheduled configure support requests, ALLY takes a snapshot of the current system state and checks to see if the mission can be supported throughout the planning window. If any of the system constraints or any of the equipment requirements are not satisfied during any portion of the planning window, ALLY concludes that the mission cannot be supported. ALLY does not have the capability to "slide" the planning window to determine if waiting a few seconds changes the answer. A human operator, on the other hand, will notice that the mission could be supported if the configure command was delayed for 30 seconds, for example. ALLY is unable to do this type of sensitivity analysis. Consequently, ALLY can commit an error by indicating that the mission cannot be supported when in fact it can.

ALLY can also indicate that a mission *can be* supported, but, by the time the operator actually issues the configure command, a conflict exists. This occurs when the planning window ALLY used was not long enough to include operator delays.

Both of these errors are examples of the "brittle" trait of knowledge-based systems. Even state-of-the-art systems are not as flexible as a human decision maker in novel or ambiguous situations.

Although ALLY has limitations, these limitations do not hinder its capability to function effectively as an operator's assistant. As with any joint cognitive system, each cognitive agent must recognize the strengths and limitations of the other agents. In order for a joint cognitive system to perform effectively, cognitive "impedance matching" must occur (Moray, 1986; Woods, 1986). With respect to ALLY, ALLY's limitations in planning are compensated by the human operator's planning capability. In addition, ALLY's strength in computational and recording capabilities compensate for the human operator's limitations in monitoring and troubleshooting data flows.

Experimental Evaluation of Ally

Experimental Design

An experiment was conducted to evaluate the effectiveness of ALLY as an operator's assistant. The experiment compared the performance of a team of two human operators with a team comprised of an operator and ALLY (Figure 23). Performance measures included those in the original GT-MSOCC experiments (Mitchell and Saisi, 1987; Mitchell and Forren, 1987).

Ten undergraduate Air Force ROTC cadets from Georgia Institute of Technology participated as subjects for the experiment. Subjects participated in both experimental conditions: control of GT-MSOCC with a human assistant and control of GT-MSOCC with ALLY. Several questionnaires were used during the experiment to collect subjective data. At the end of each experimental session, the subjects completed a questionnaire to attempt to elicit the interaction/cooperation strategy subjects used with either the human or computer-based assistant. In addition, the subjects completed an ALLY Exit questionnaire and a Human Exit questionnaire at the end of their last data session with the respective assistant. The purpose of exit questionnaires was to elicit opinions about important aspects of the assistant. Finally, at the end of the experiment, the subjects were asked to complete a Subjective Comparison Rating questionnaire to compare their opinions about the two assistants.

The subjects participated in twenty-four sessions: eight sessions of baseline GT-MSOCC training, three sessions of human assistant training, four sessions human assistant data collection, five sessions of ALLY training, and four sessions of ALLY data collection. A total of 240 hours of data were collected. The sessions were approximately 45 minutes in length. The sessions were typically run on consecutive days with one session per day. Occasionally, subjects missed a day and made up the session by running multiple sessions in a single day.

Analyses of variances were performed to determine the effect of each of the four independent variables (Condition, Group, Session, and Subject) on each of the sixteen dependent measures. A significance of .10 was used to detect significant effects. In addition to the statistical analysis, the results of the questionnaires and analysis of audit logs of the subjects' activities were examined to provide additional insight into the individual interaction strategies used by the subjects. These analyses, in conjunction with the statistical analyses, were used to evaluate the effectiveness of ALLY as an operator's assistant and to evaluate the proposed theory of cooperation as it was implemented in ALLY.

Experimental Results

The experimental results are summarized in Figure 24a, b, c. These data indicate that, on the average, a human-ALLY team performed comparably to a team of two human operators. Only two performance measures yielded a significant difference: time to compensate for software failures characterized by termination of data flow and the number of correct responses to unscheduled support requests. For these measures, the human-ALLY team performed more effectively. On all other performance measures the ALLY team performed as well as the team of two human operators. A more detailed discussion is provided in Bushman (1989).

Overall, the performance of the subjects using ALLY as an assistant was as effective as performance with the human assistant. Individual strategies enabled some of the subjects to perform better with ALLY than with the human assistant. The primary area that was affected by personal strategies was in detecting and compensating for software failures. Several subjects were able to develop a style of interacting with ALLY that enabled them to detect software failures before either the operator or ALLY could on their own. This enabled them to detect the failures faster and to correct a larger percentage of the total failures.

Since ALLY does not have the capability to anticipate schedule conflicts, it is not able to plan for these events in advance. The subjects that relied on ALLY's capability to respond to schedule conflicts did not take advantage of their own planning ability. The subjects that performed best with ALLY did not rely exclusively on ALLY, but used their own capabilities to anticipate and plan for these events.

An unexpected result was a side-effect associated with the difficulty ALLY has with planning. ALLY performed as well as the human assistant in responding to unscheduled support requests. However, because the subjects knew that this was an area in which ALLY made mistakes, they regularly checked ALLY's answers. The additional checking resulted in more correct responses to support requests with ALLY.

Subjective preferences indicated that subjects liked different aspects of the two assistants. They found ALLY to be more efficient and the human assistant to be more natural.

Summary

This experiment provides strong support for the assumption that a computer-based assistant based on a model of the operator's function can perform as well as a human in a supervisory control team. As with any cognitive system (either human or artificial), ALLY had strengths and limitations. The subjects that performed the best with ALLY were able to capitalize on its strengths and compensate for its weaknesses. The result was an overall increase in the system performance.

This research demonstrated that a computer-based assistant founded on the identified principles of human-machine cooperation and an operator function model of the supervisory control task can achieve performance compatible with a human assistant. In addition, this research has provided a "starting-point" from which a finer theory of cooperation can be developed. The significance of this research is that it has provided empirical data about the nature and effectiveness of human-machine cooperation in supervisory control applications.

Quantitative experimental data demonstrated the feasibility of the architecture for a computer-based assistant. Qualitative data, in the form of subjective evaluations, identified some of the individual interaction and cooperation strategies.

These quantitative and qualitative analyses may form the basis of a more refined theory of human-machine cooperation. Since no theory exists, exploratory research is essential to develop a more definitive theory of cooperation.

Conclusions

Overall, this research has been very productive. It pioneered research into the possibility of constructing an intelligent operator's assistant. An architecture for a model-based intent inferencer was designed and implemented. Once running, the ability of the system to correctly maintain a model of operator intentions as postulated functions, subfunctions, and tasks, and to critical events and reminds the operator when it appears that the operator may have forgotten or overlooked an event.

The type of automatic support that ALLY provides will vary with the domain; the principle, however, is that automatic activities are those that exploit the power of a computer-based assistant. When faced with multiple tasks, the human operator typically performs them serially. A computer, on the other hand, can perform tasks concurrently. Consequently, since the design objective of the computer-based assistant is not only to replicate the human assistant, but to provide a tool that the operator could use, ALLY takes advantage of the computer's processing capabilities.

ALLY performs automatic activities in addition to activities that the operator delegates. In this manner, the operator remains in control of GT-MSOCC and ALL. In addition, the operator has a tool to assist in performing the most cognitively demanding activity, i.e., monitoring the equipment networks, without having to ask specifically for help.

ALLY's Automatic Fault Detection. ALLY, using the power of a computer, continually monitors the data transmissions at the network endpoints for each currently supported mission. If ALLY detects a problem, ALLY informs the operator and automatically begins to troubleshoot the network to identify the cause of the problem. Once ALLY has identified the cause, ALLY informs the operator that it suspects a component failure. ALLY does not initiate replacement activities, however, unless the operator directs ALLY to do so.

ALLY's automatic monitoring and troubleshooting activity is based on a cognitive task analysis described in the operator function model. The operator function model describes the operator functions in levels of abstraction. The most cognitively demanding task is monitoring and troubleshooting. This activity is very difficult because of the format of the telemetry information displayed to the operator and because not all of the information necessary to perform the task is immediately or simultaneously available.

Part of the nature of a cooperative problem solving team is that both operators understand the cognitive nature of the task and act accordingly. The operators attempt to "balance" the workload between them. ALLY understands the cognitive nature of the supervisory control functions and attempts to provide assistance for tasks that the operator needs help in performing (i.e., monitoring and troubleshooting). The primary purpose of an assistant is to reduce workload and improve overall system performance. By aggregating and abstracting the raw telemetry data to something more meaningful to the operator--a task that a computer performs effortlessly--ALLY reduces the human operator's cognitive workload in the control of GT-MSOCC.

Reminding Capability. ALLY's other automatic activity is to remind the operator of critical events that might have been missed or overlooked. In the GT-MSOCC system, ALLY, when necessary, reminds the operator of three types of events, all three are system requests asking the operator to manually change the system state. For GT-MSOCC these requests are manually interpret actual actions in that context was extensively evaluated. Given a valid model of operator intentions, OFMspert was augmented with control properties. Again, an extensive empirical evaluation demonstrated that a human-ALLY team controlled a simulated satellite ground control system as effectively as a team comprised of two human operators.

This research showed that for complex dynamic systems, such as satellite ground control, the operator function model (OFM) provided a compact representation of functions, intentions, and operator activities in complex dynamic system. Furthermore, the OFM was a successful organization for information that OFMspert could use to hypothesize and interpret current operator activities. OFMspert's ACTIN, blending the OFM and a blackboard paradigm for problem solving, proved to be an effective means of dynamically constructing and maintaining a model of operator intentions. Finally, the OFM guided the design of OFMspert's control capabilities. The interactive, flexible functionality of ALLY (OFMspert with control) was shown to be as effective an assistant to the human controller as another experienced operator.

The OFM and the OFMspert structures show strong promise for application in a variety of domains in which a task-analytic description of operator activity is available and where there is an interest in providing expert system capability to augment human operator capability. Finally, OFMspert exists as an alternative use of artificial intelligence (i.e. its Blackboard model of problem solving) in complex systems. Rather than replacing operator control activities or running in parallel, OFMspert was designed to interact with the human operator and act as a assistant . The intention was not to design two, parallel decision makers, but rather a human-computer symbiosis that acts in similar ways to effective teams of human decision makers.

This research resulted in many papers and presentations and several research awards. Listings of the papers and presentation are in Appendix A. Copies the major papers and technical reports follow in Appendix B.

References

Anderson, J.R. (1987). Methodologies for studying human knowledge. *Behavioral and Brain Sciences*, 10, 467-505.

Bushman, J. B. (1989). Identification of an operator's associate model for cooperative supervisory control situations. Ph.D. dissertation, Report 89-1, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Chambers, A. B. and Nagel, D. C. (1985). Pilots of the future: Human or computer? *Communications of the ACM*, 28, No. 11, 1187-1199.

Digitalk (1986). *Smalltalk/V: Tutorial and Programming Handbook.*

Dunkler, O., Mitchell, C.M., Govindaraj, T., and Ammons, J.C. (1988). The effectiveness of supervisory control strategies in flexible manufacturing systems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-18, No. 2, 223-237.

Englemore, R.S., Morgan, A. J., and Nii, H. P. (1988). Introduction. In R. Englemore and T. Morgan (Eds.), *Blackboard Systems*, Reading, MA: Addison-Wesley, 1-24.

Ericsson, K. A. and Simon, H. A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

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

Gaines, B. R. and Shaw, M. L. G. (1985). Dialog engineering. In M. E. Sime and M. J. Coombs (Eds.), *Designing for Human-Computer Communication.* London: Academic, 23-53.

Geddes, N.D. (1985). Intent inferencing using scripts and plans. *Proceedings of the First Annual Aerospace Applications of Artificial Intelligence Conference.*

Goldberg, A. (1983). *Smalltalk-80: The Interactive Programming Environment.* Reading, MA: Addison-Wesley.

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

Jones, P. M., Rubin, K. S., and Mitchell, C. M. (1990). Validation of intent inferencing by a model-based operator's associate. *International Journal of Man-Machine Studies*, in press.

Knaeuper, A., and Morris N. M. (1984). A model-based approach for on-line aiding and training in a process control task. *Proceedings of the 1984 IEEE International Conference on Systems, Man, and Cybernetics*, 173-177.

Miller, J. R., Polson, P. G., and Kintsch, W. (1984). Problems of methodology in cognitive science. In W. Kintsch, J. R. Miller, and P.G. Polson (Eds.), *Method and Tactics in Cognitive Science*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1-18.

Mitchell, C. M. (1987). GT-MSOCC: A research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, 553-570.

Mitchell, C. M. and Forren, M. G. (1987). Multimodal user input to supervisory control systems: Voice-augmented keyboards. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, No. 4, 594-607.

Mitchell, C. M. and Saisi, D. L. (1987). Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, No. 4, 573-593.

Moray, N. (1986). Modelling cognitive activities: Human limitations in relation to computer aids. In E. Hollnagel, G. Mancini, and D. D. Woods (Eds.), *Intelligent Decision Support in Process Environments*, Berlin: Springer-Verlag, 211-226.

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

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

ParcPlace Systems (1988). *The Smalltalk-80 Programming System*. ParcPlace Systems Inc.

Reichman, R. (1986). Communication paradigms for a window system. In D.A. Norman and S. W. Draper (Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, 285-313.

Rouse, W. B., Geddes, N.D., and Curry, R. E. (1987). An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction*, 3, No. 2.

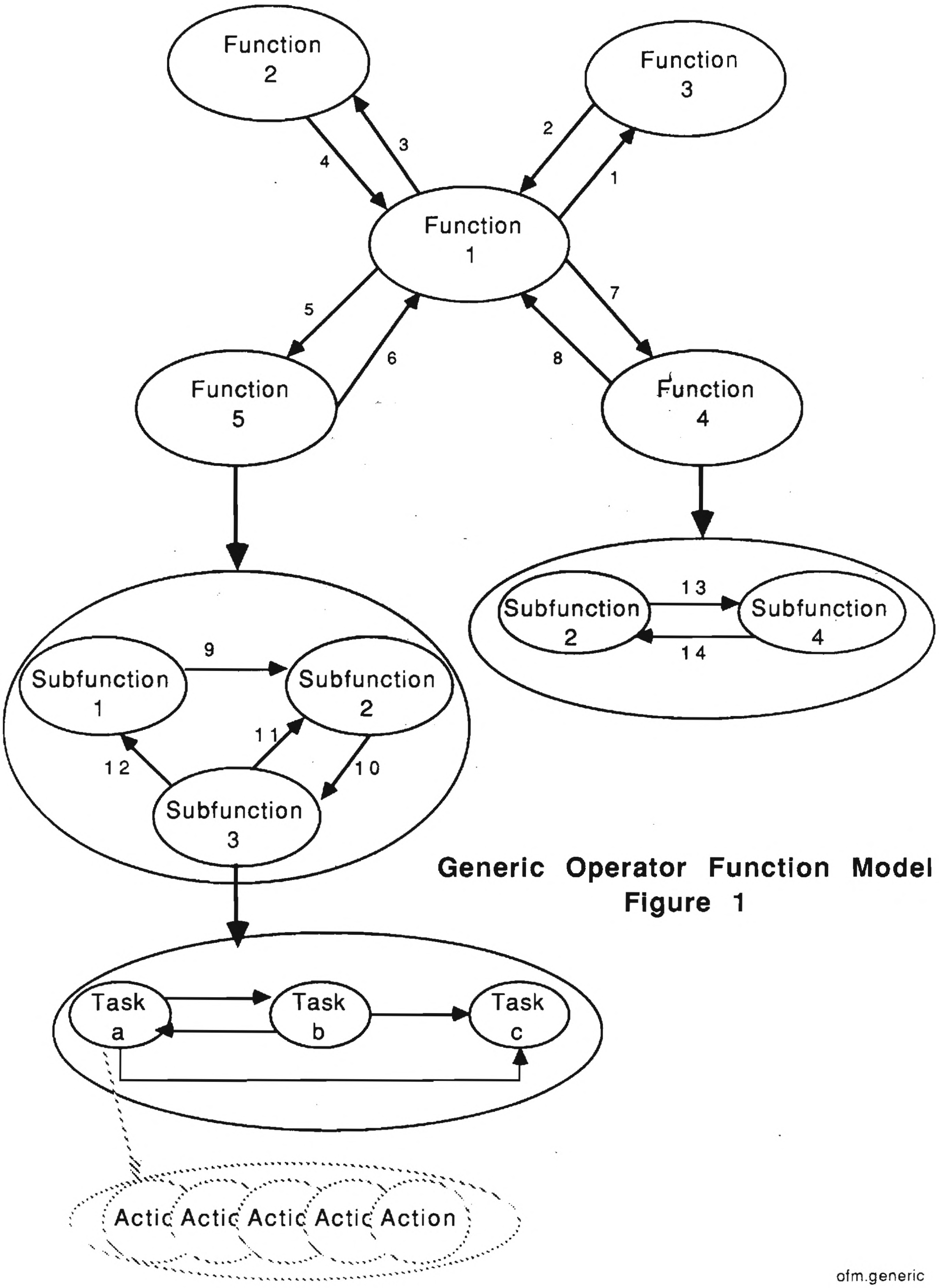
Rubin, K. S., Jones, P. M., and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC. 18, No. 4, 618-637.

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

Woods, D.D. (1986). Paradigms for intelligent decision support. In E. Hollnagel, G. Mancini, and D.D. Woods (Eds.), *Intelligent Decision Support in Process Environments*, Berlin: Springer-Verlag, 255-269.

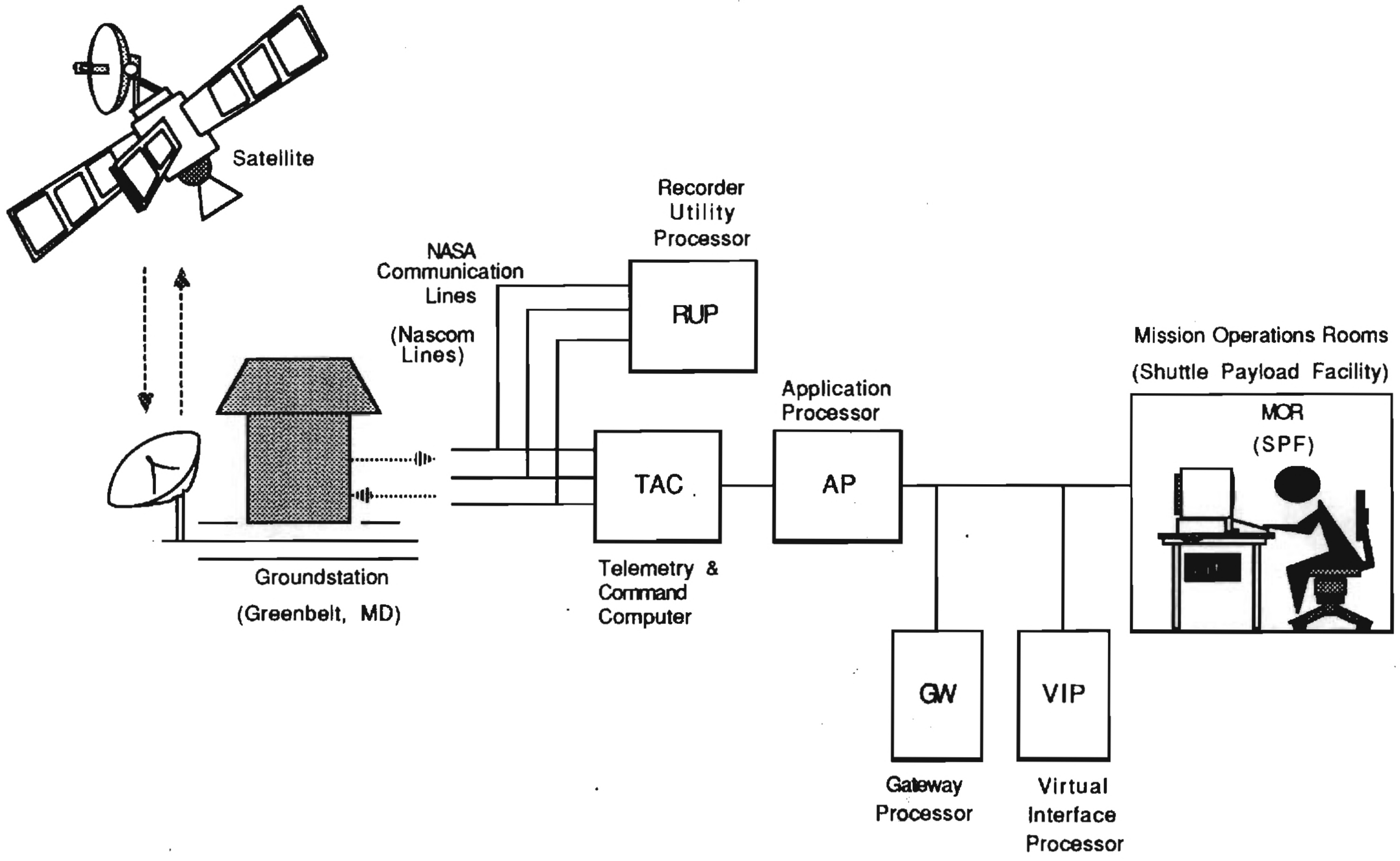
Appendix A

Summary of Papers, Reports, Theses, and Degrees Sponsored in Part by this Grant

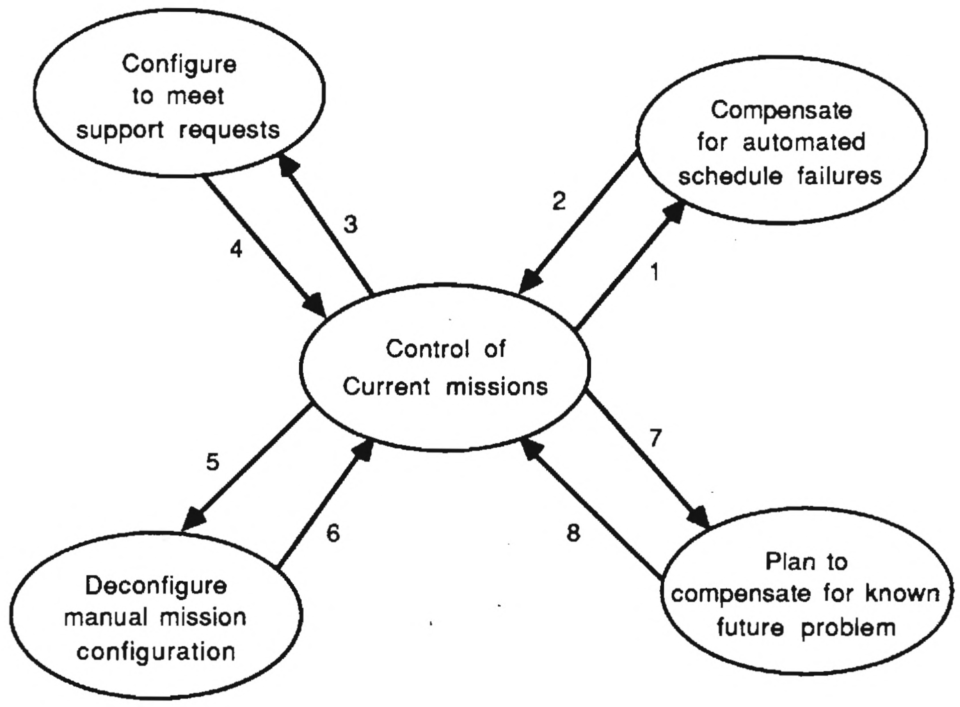


Generic Operator Function Model
Figure 1

ofm.generic



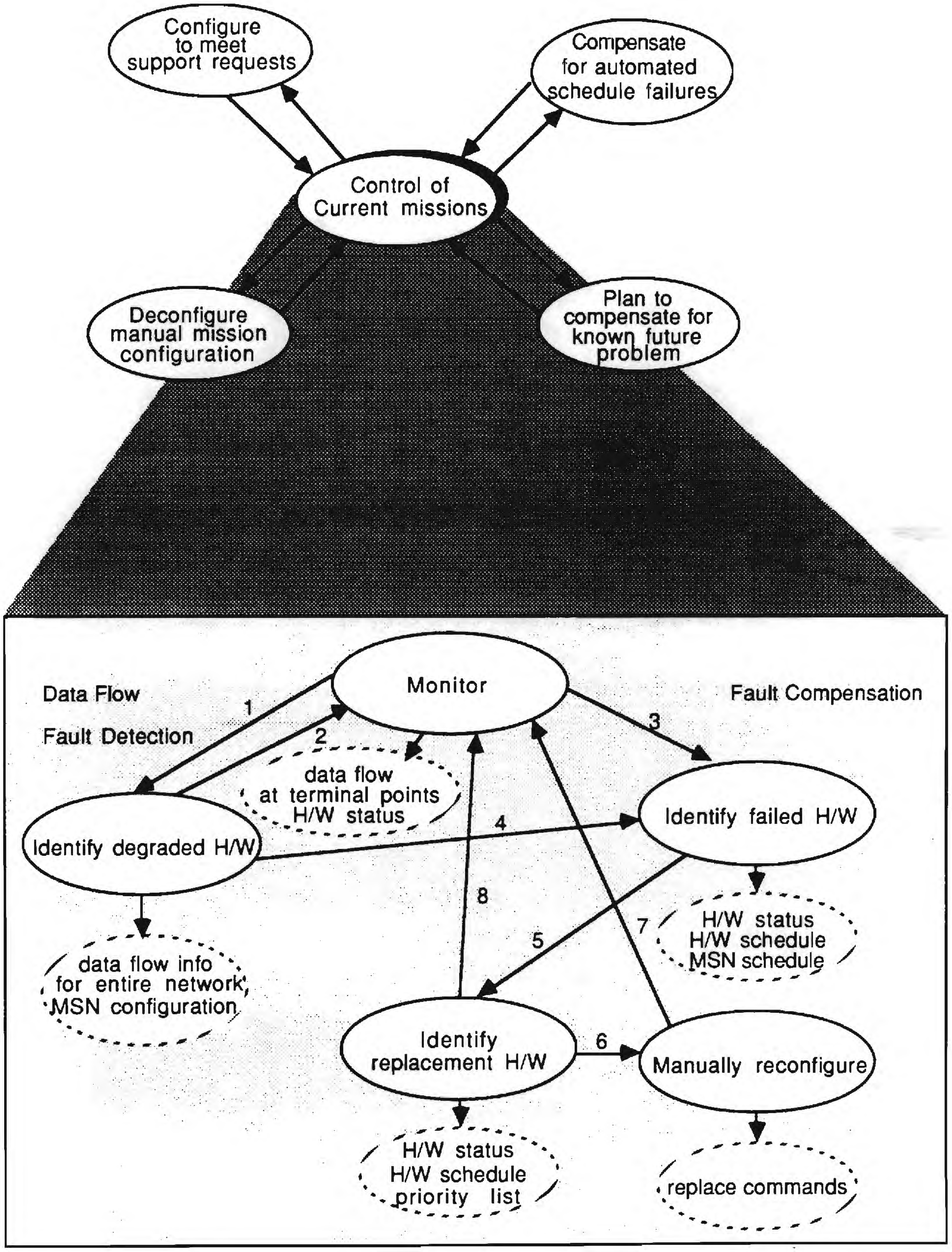
Multisatellite Operations Control Center
Figure 2


Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
1
2
3
4
5
6
7
8

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

Major GT-MSOCC Supervisory Control Functions
Figure 3

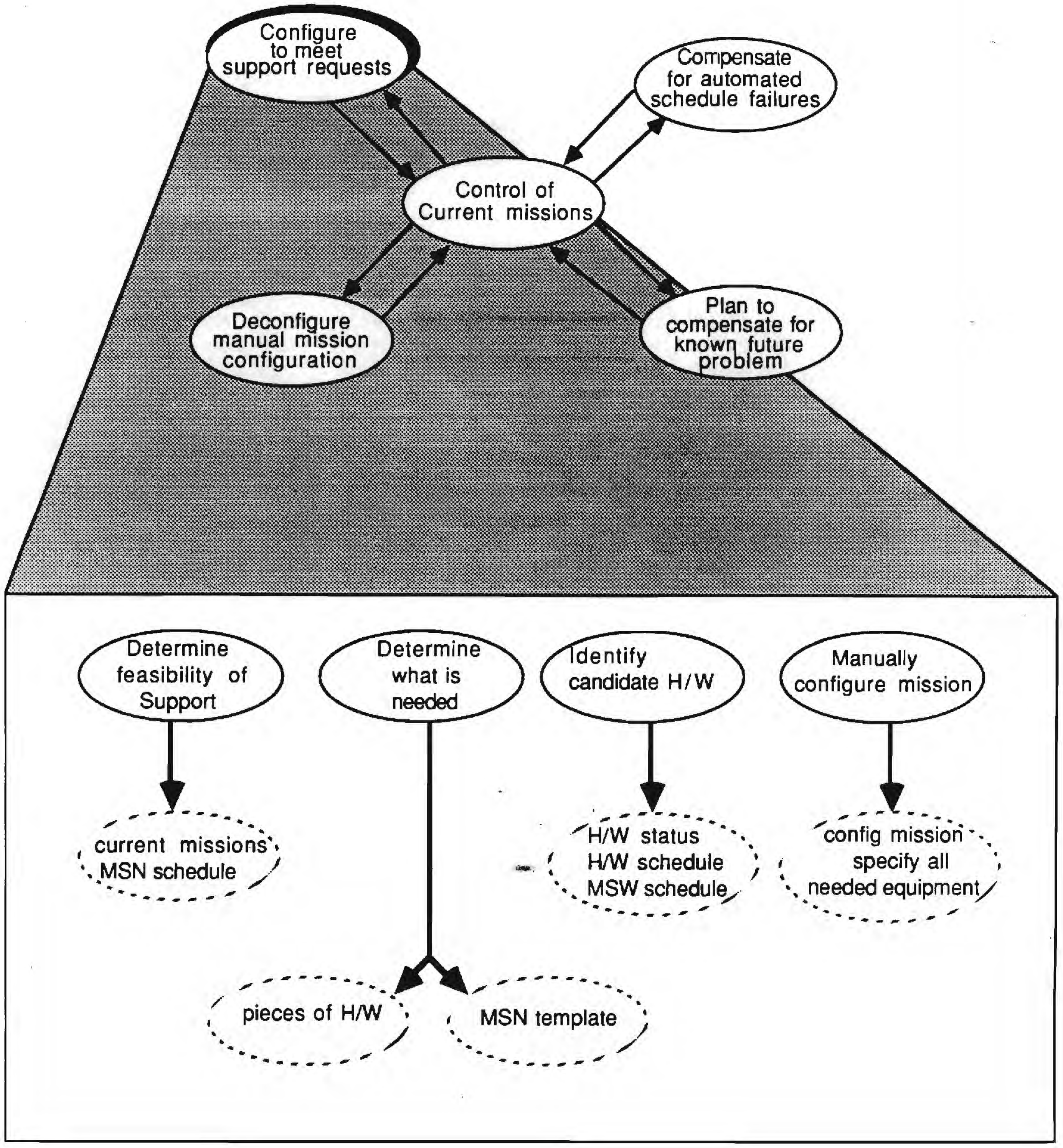
ofm.toplevel



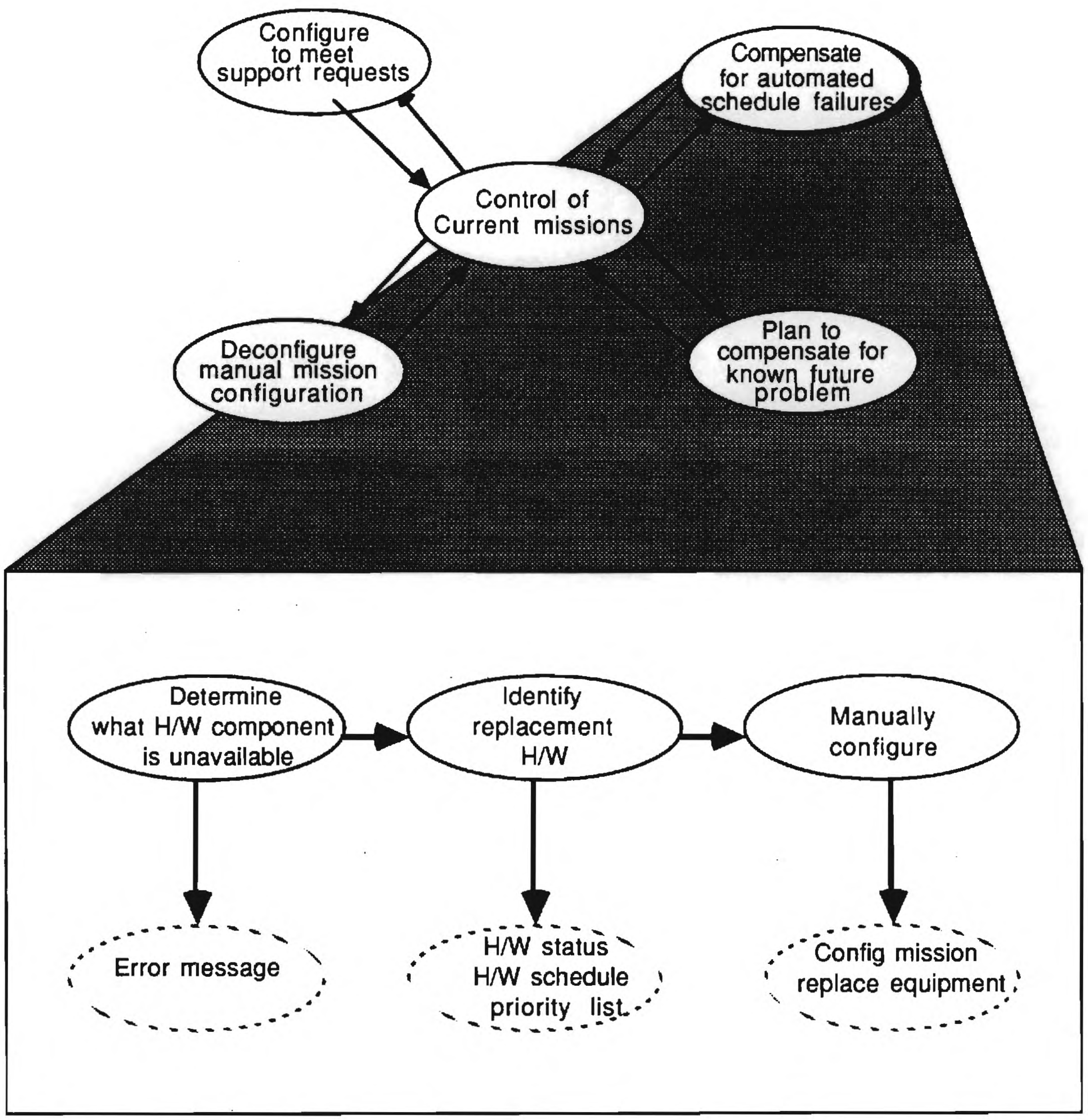
Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
Data Flow
Fault Detection
Monitor
Fault Compensation
1
2
3
data flow at terminal points H/W status
Identify degraded H/W
4
Identify failed H/W
8
7
5
H/W status H/W schedule MSN schedule
data flow info for entire network MSN configuration
Identify replacement H/W
6
Manually reconfigure
H/W status H/W schedule priority list
replace commands

Control of Current Missions
Figure 4

ally.ofm.ccr

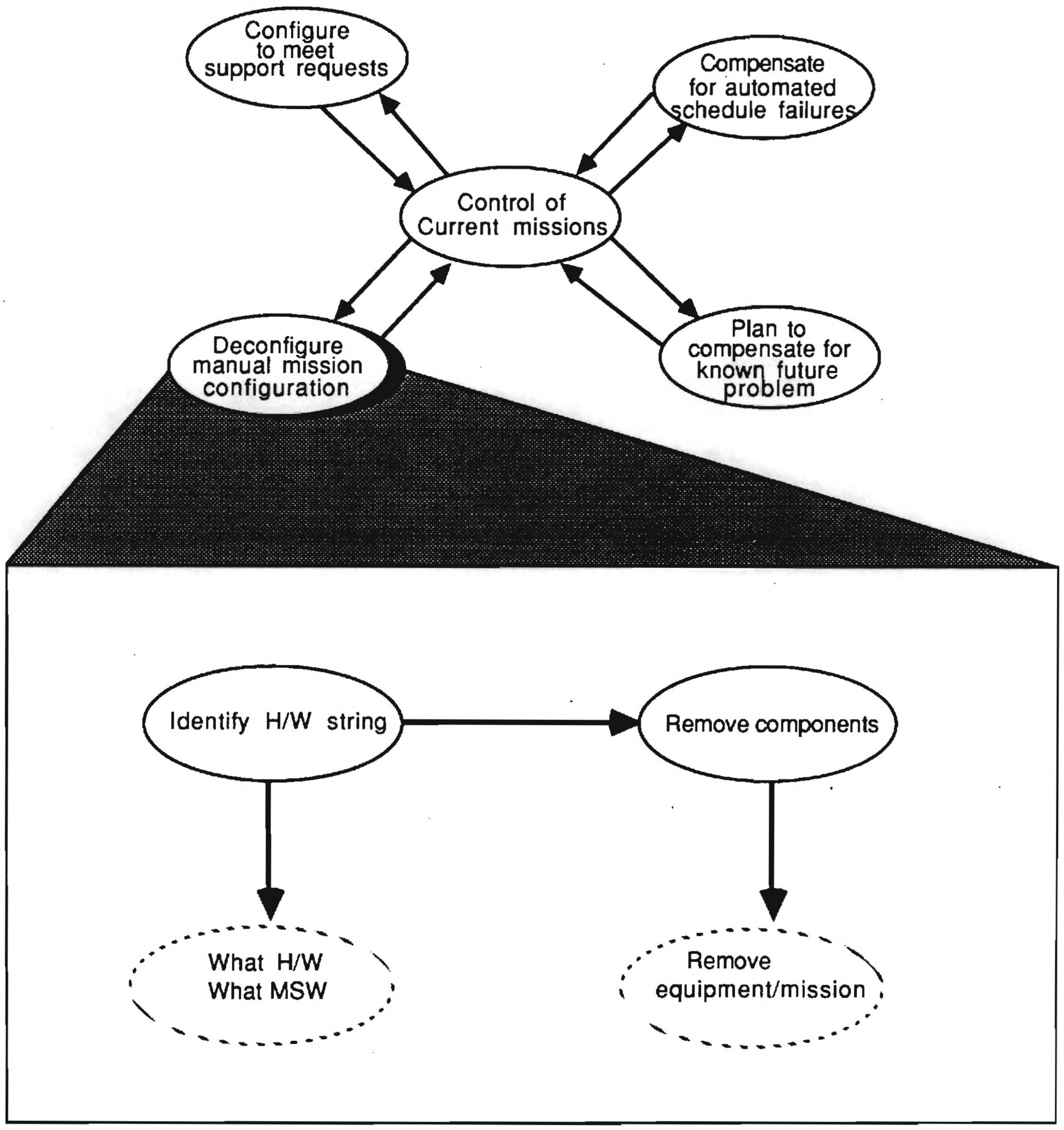

Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
Determine feasibility of Support
Determine what is needed
Identify candidate H/W
Manually configure mission
current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

Unscheduled Support Request
Figure 5

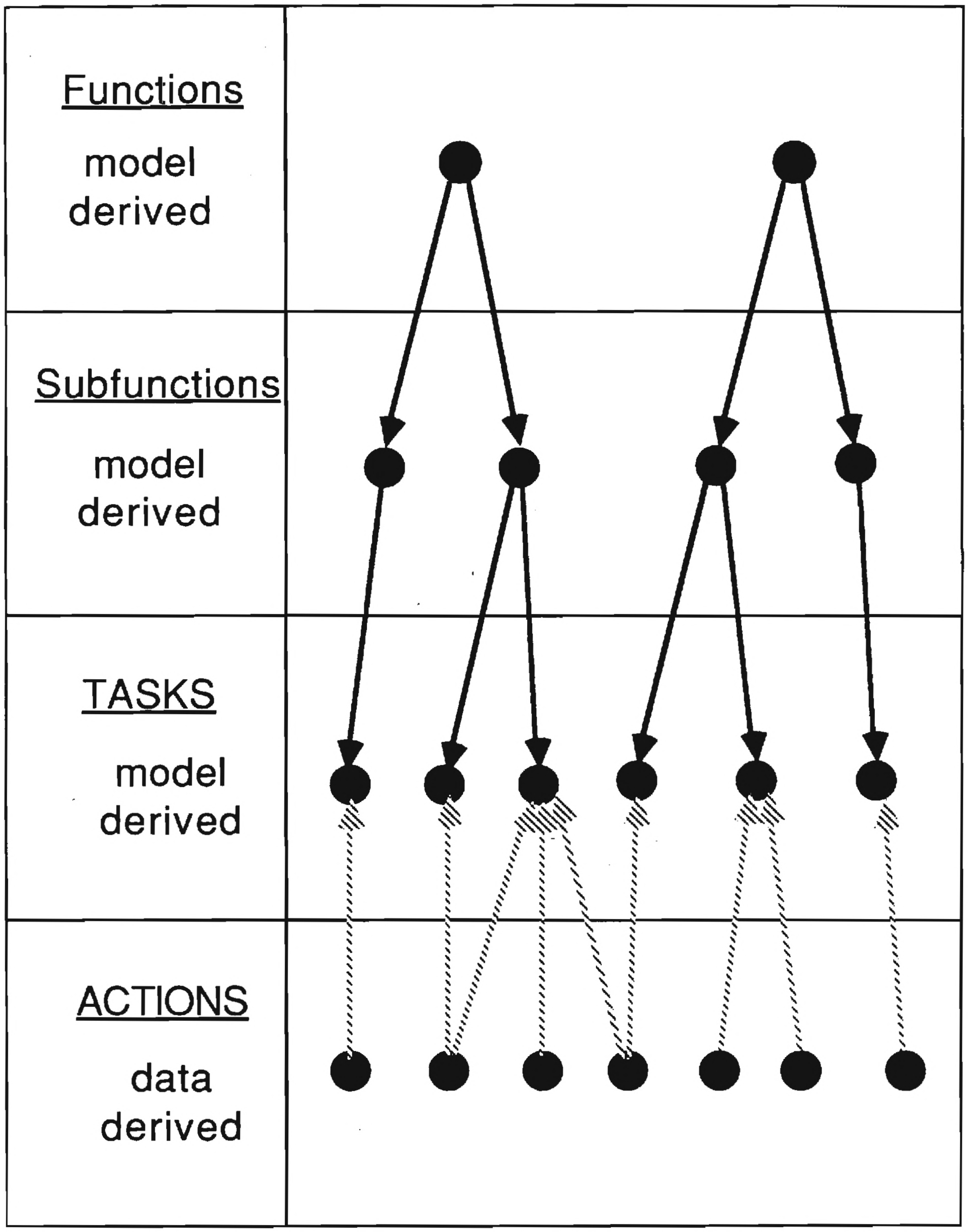


Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
Determine what H/W component is unavailable
Identify replacement H/W
Manually configure
Error message
H/W status
H/W schedule
priority list
Config mission
replace equipment

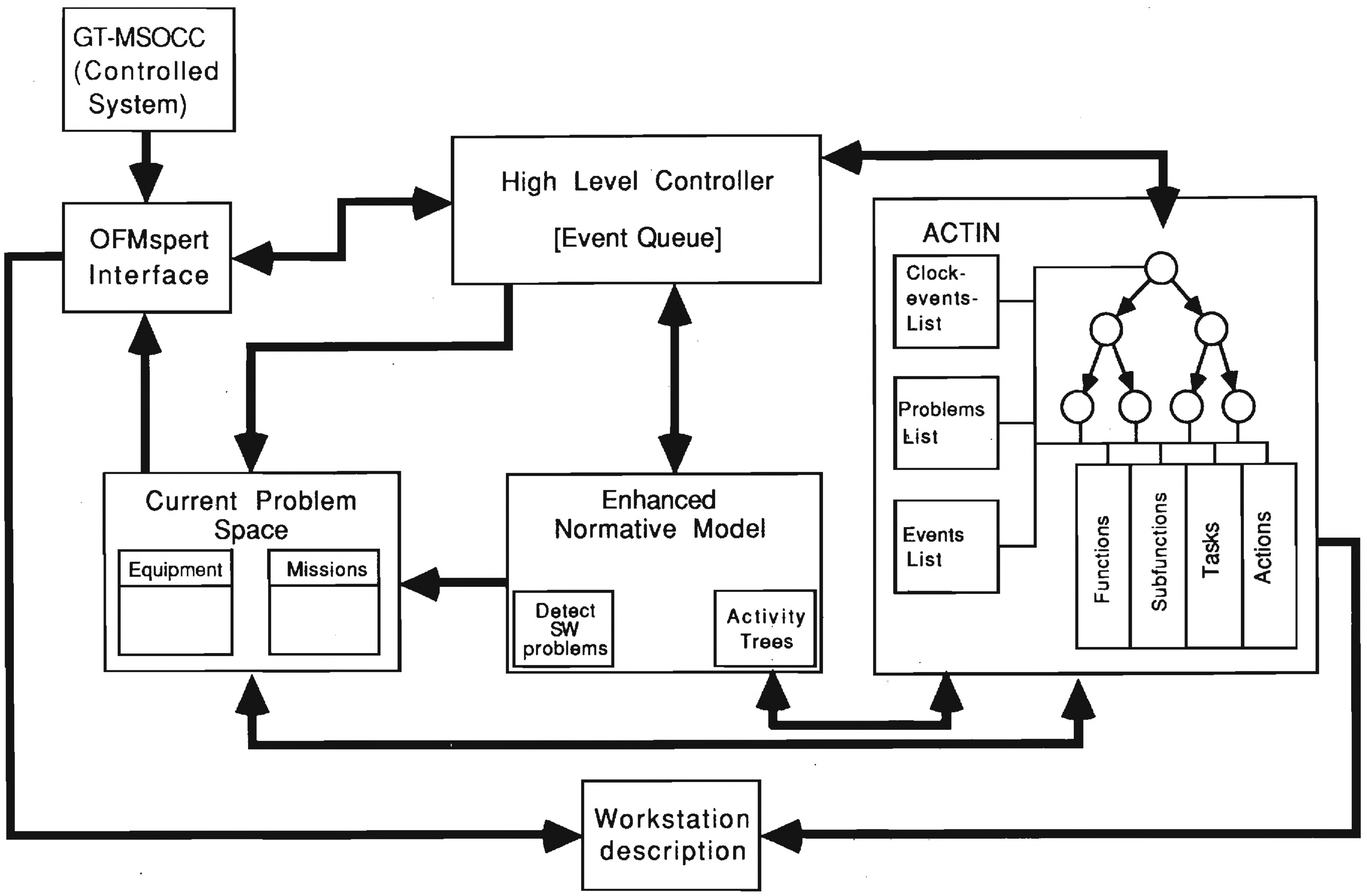
Compensate for Schedule Failures
Figure 6


Configure to meet support requests
Compensate for automated schedule failures
Control of Current missions
Deconfigure manual mission configuration
Plan to compensate for known future problem
Identify H/W string
Remove components
What H/W
What MSW
Remove equipment/mission

Deconfigure Request
Figure 7

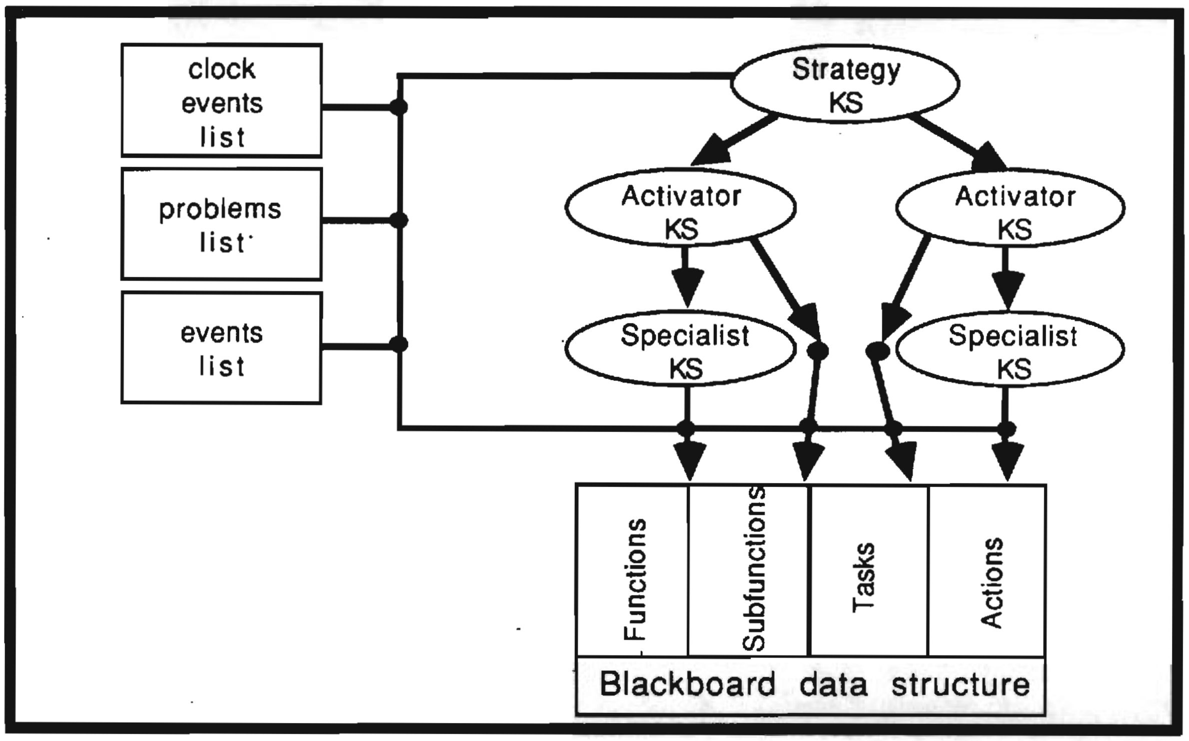


ACTIN's Intent Inferencing Structure
Figure 8



OFMspert Architecture
Figure 9

ofmspert.msocc.architecture



ACTIN: OFMspert's Blackboard Model

Figure 10

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

Experiment 1: Average Percentage of Equivalent Interpretations Between ACTIN and a Human Domain Expert. (Ordered by Rank).

Figure 11

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

Experiment 2: Average Percentage Of Equivalent Interpretations Between ACTIN And Verbal Reports. (Ordered By Rank).

Figure 12

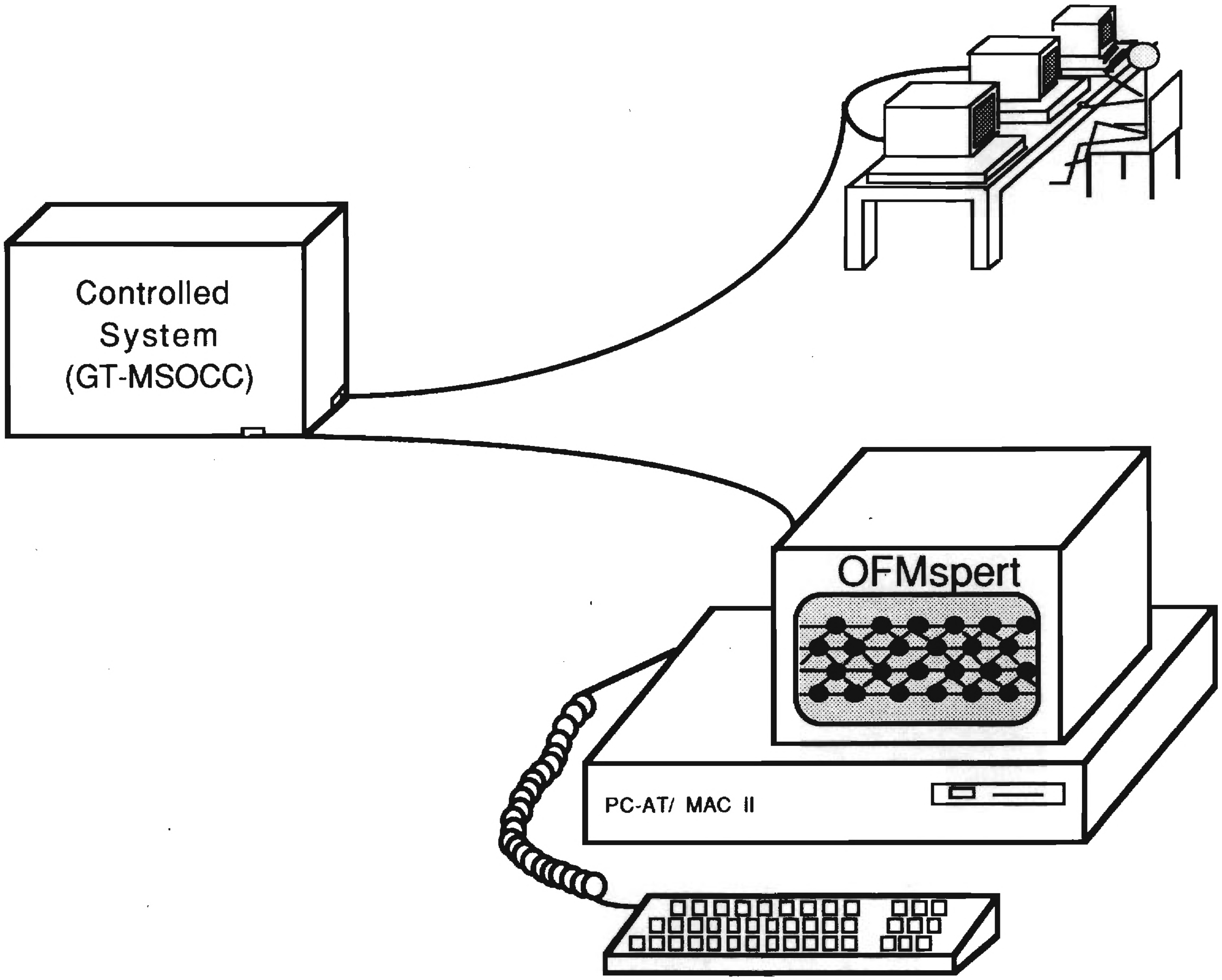
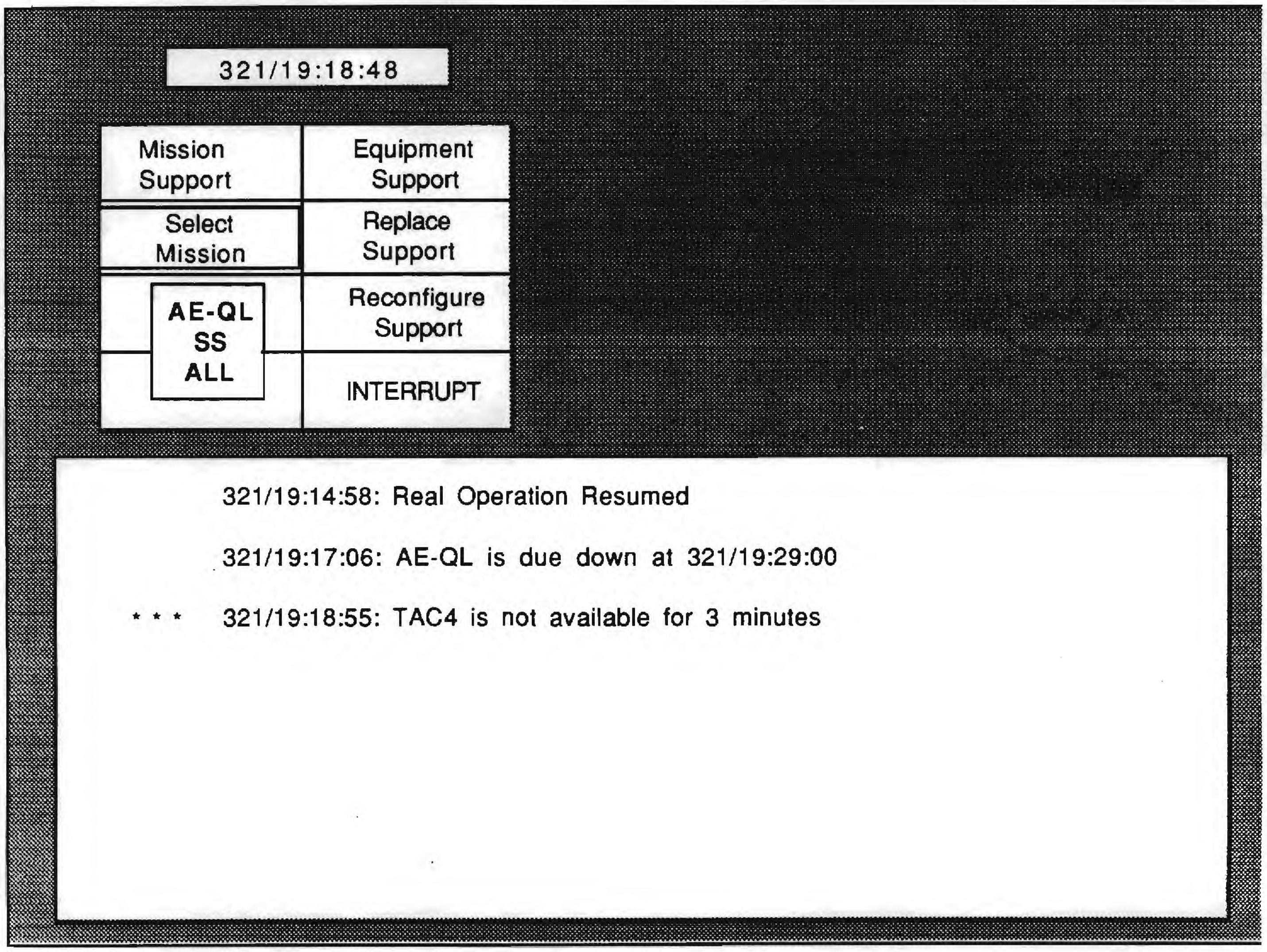


Figure 13

ally.overview1

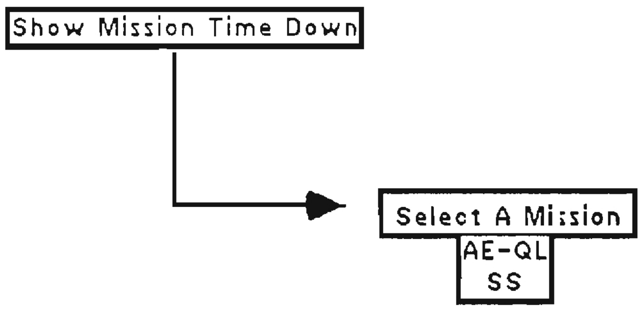


```
      321/19:14:58: Real Operation Resumed

      321/19:17:06: AE-QL is due down at 321/19:29:00

* * * 321/19:18:55: TAC4 is not available for 3 minutes
```

Example of Ally's User Interface

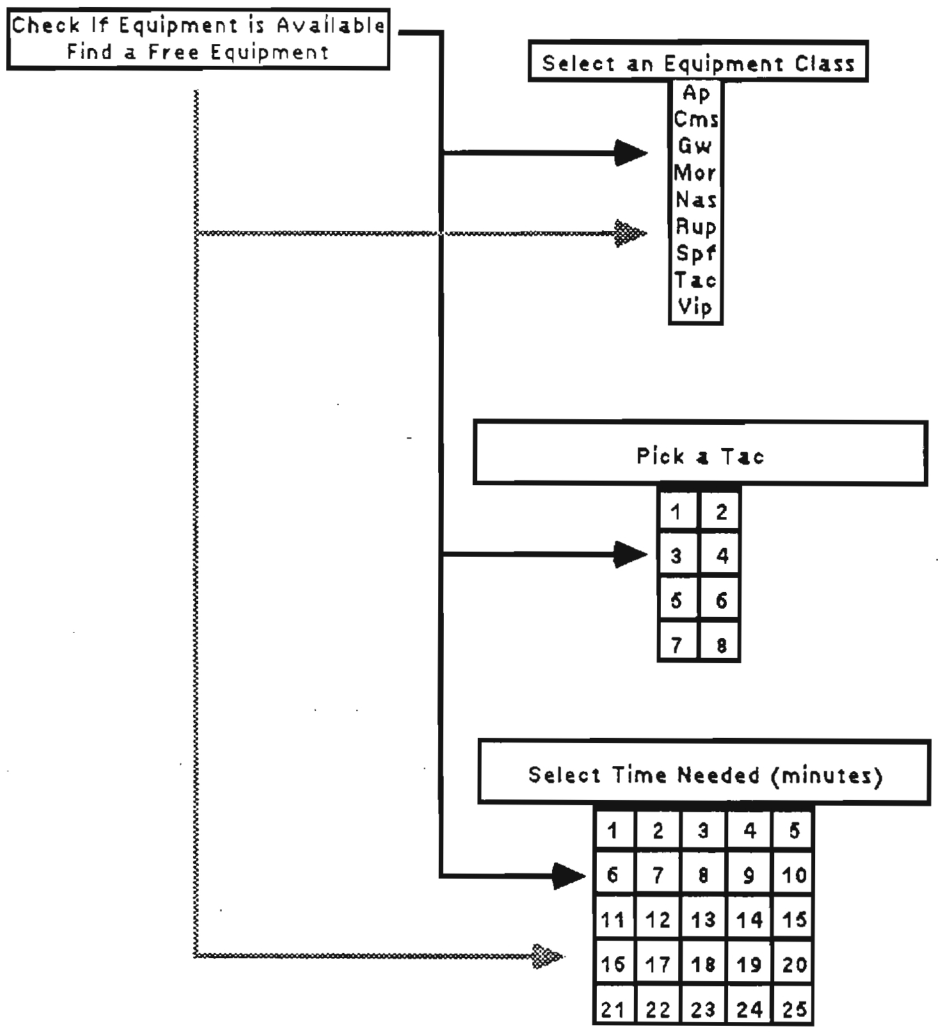
Figure 14

ally.interface.user

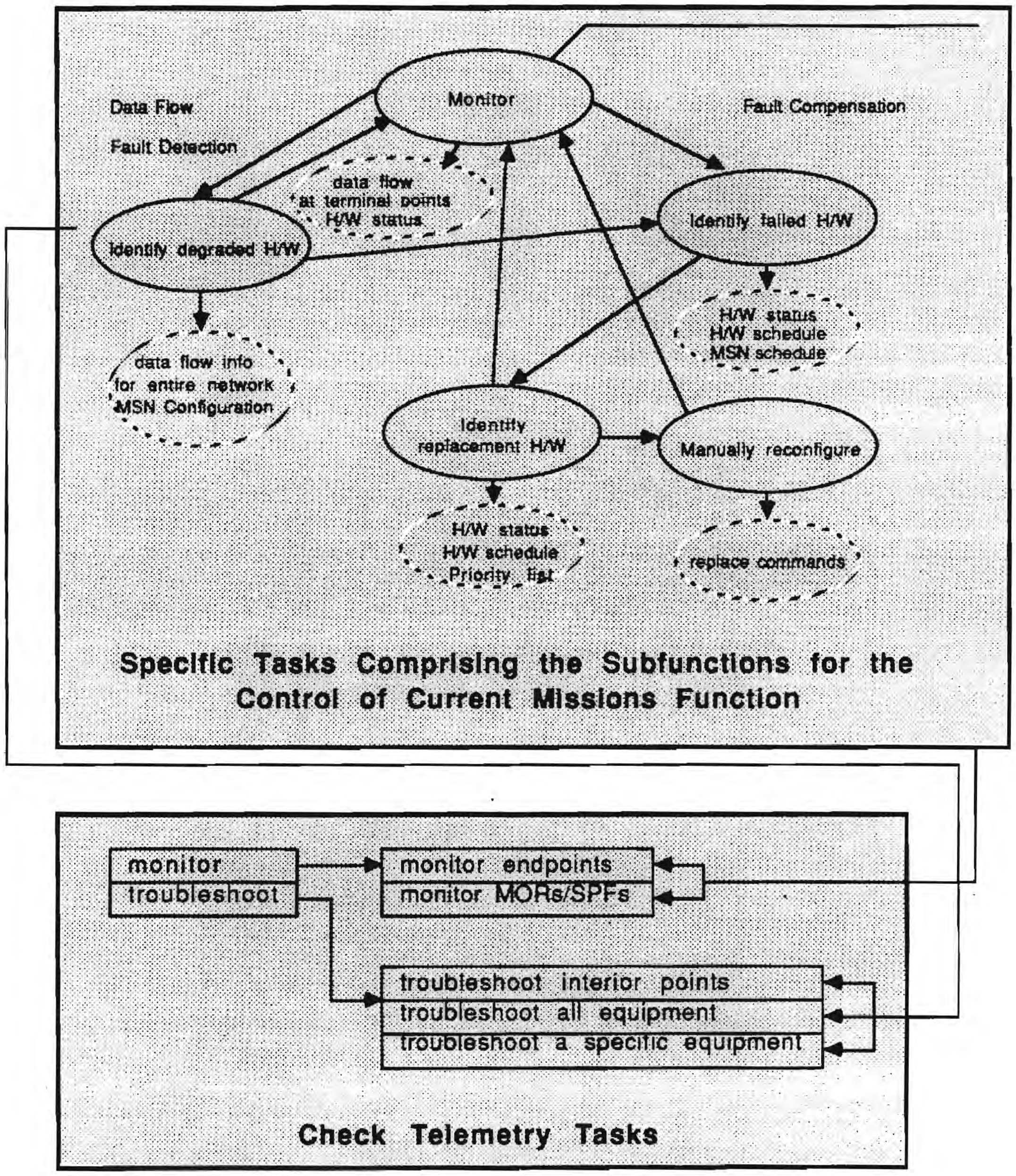


Mission Support Menus

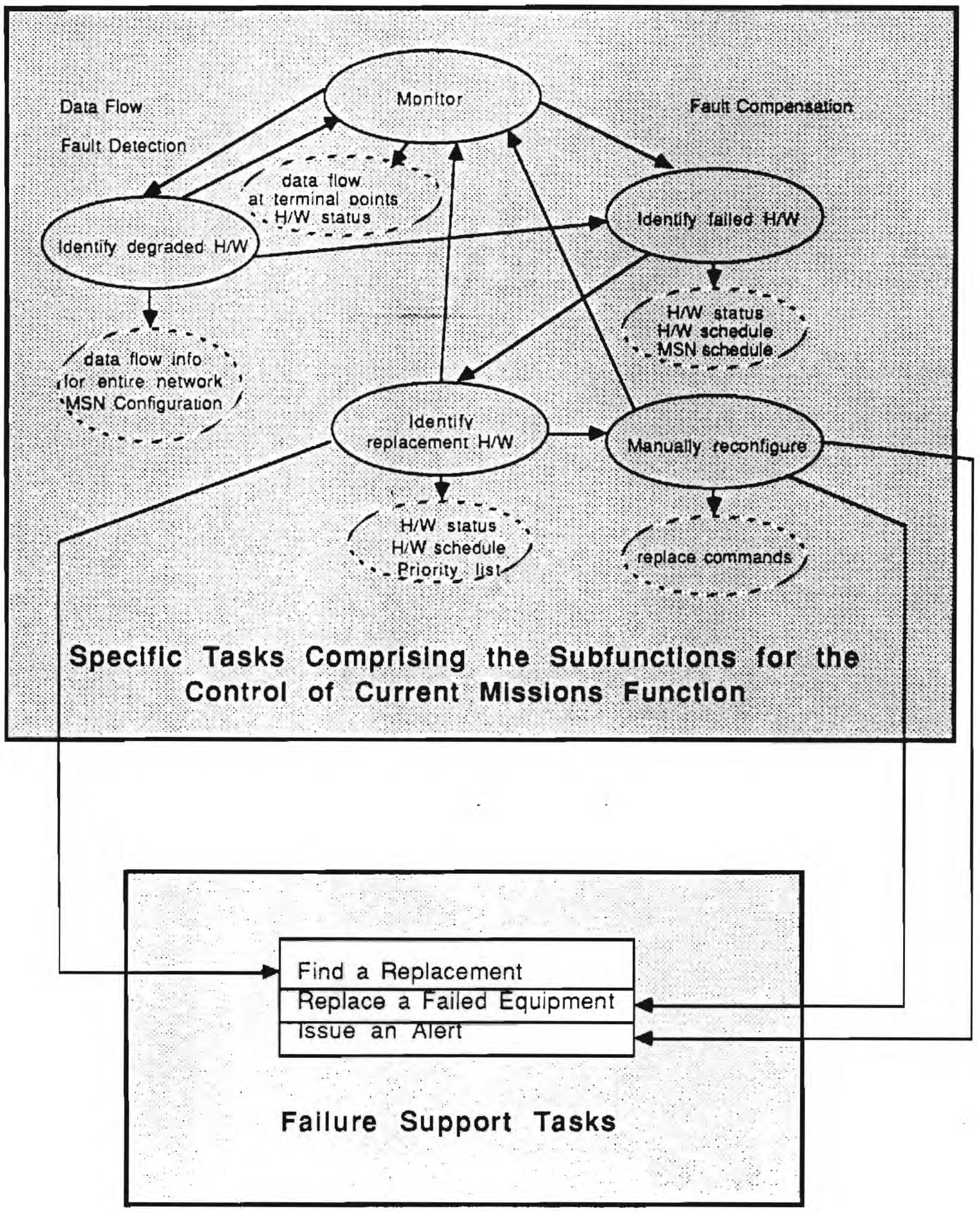
Figure 15



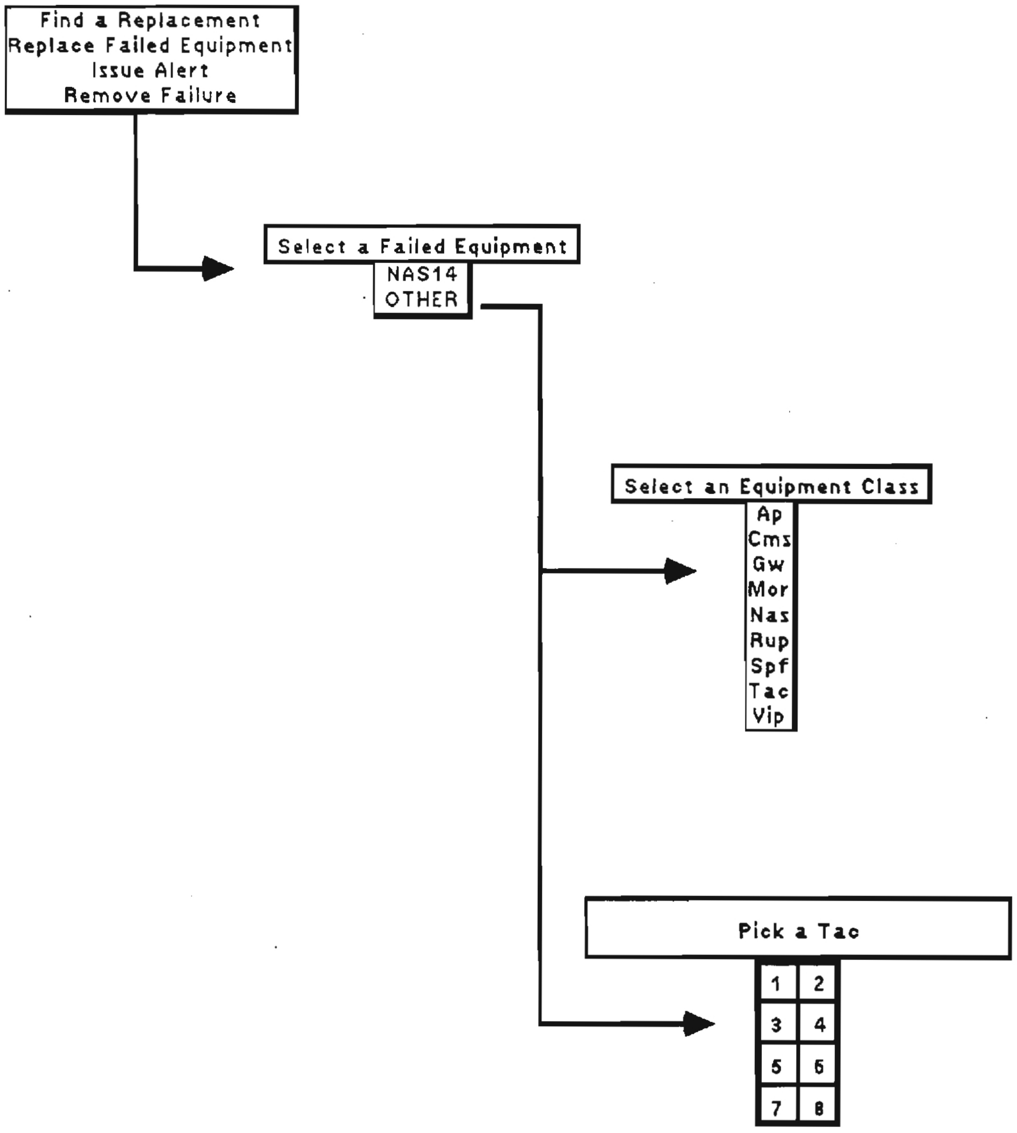
Equipment Support Menus
Figure 16


Monitor
Data Flow
Fault Detection
Fault Compensation
data flow at terminal points H/W status
Identify degraded H/W
Identify failed H/W
data flow info for entire network MSN Configuration
H/W status H/W schedule MSN schedule
Identify replacement H/W
Manually reconfigure
H/W status H/W schedule Priority list
replace commands
Specific Tasks Comprising the Subfunctions for the Control of Current Missions Function
monitor
troubleshoot
monitor endpoints
monitor MORs/SPFs
troubleshoot interior points
troubleshoot all equipment
troubleshoot a specific equipment
Check Telemetry Tasks

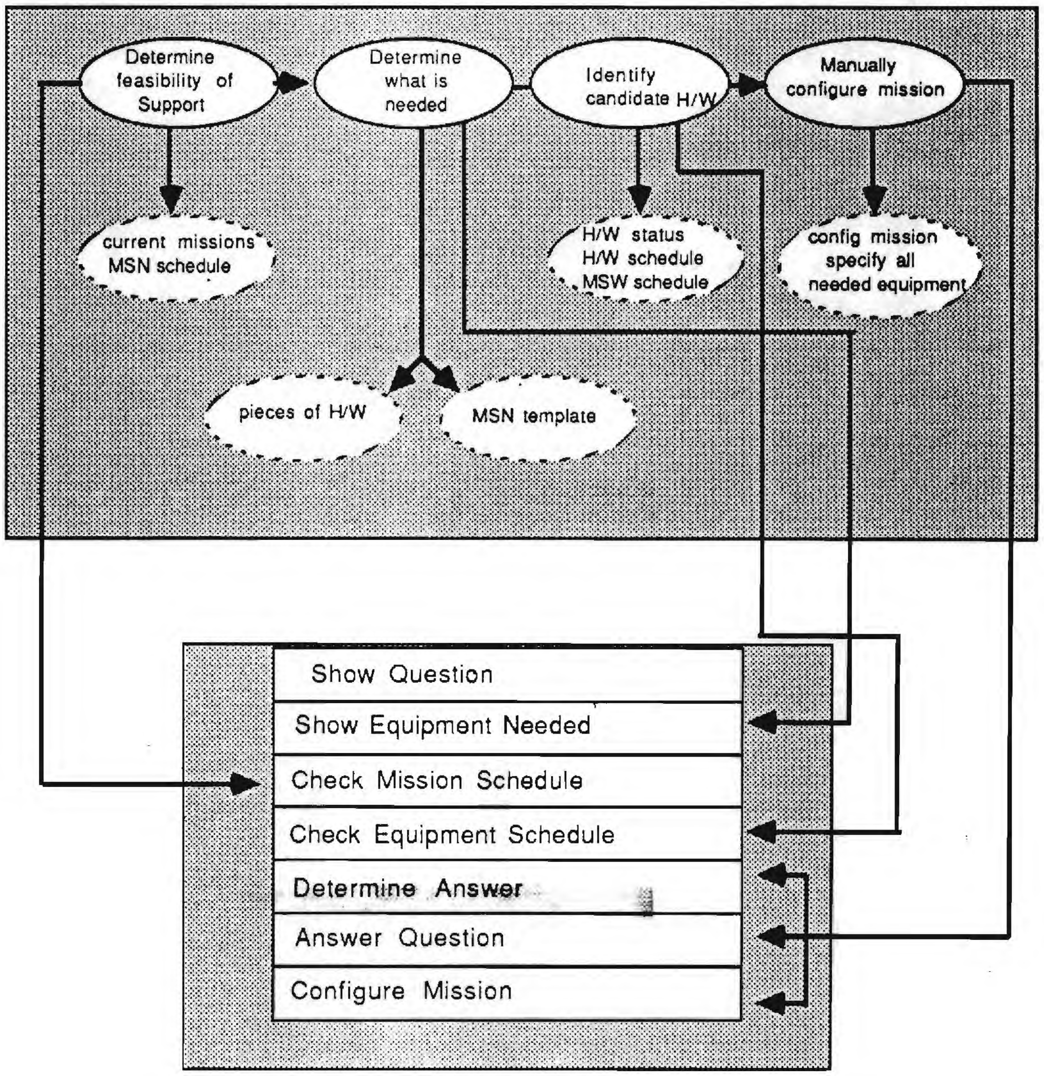
Check Telemetry Tasks and OFM
Figure 17


Data Flow
Fault Detection
Monitor
Fault Compensation
data flow at terminal points H/W status
Identify degraded H/W
Identify failed H/W
H/W status H/W schedule MSN schedule
data flow info for entire network MSN Configuration
Identify replacement H/W
Manually reconfigure
H/W status H/W schedule Priority list
replace commands
Specific Tasks Comprising the Subfunctions for the Control of Current Missions Function
Find a Replacement
Replace a Failed Equipment
Issue an Alert
Failure Support Tasks

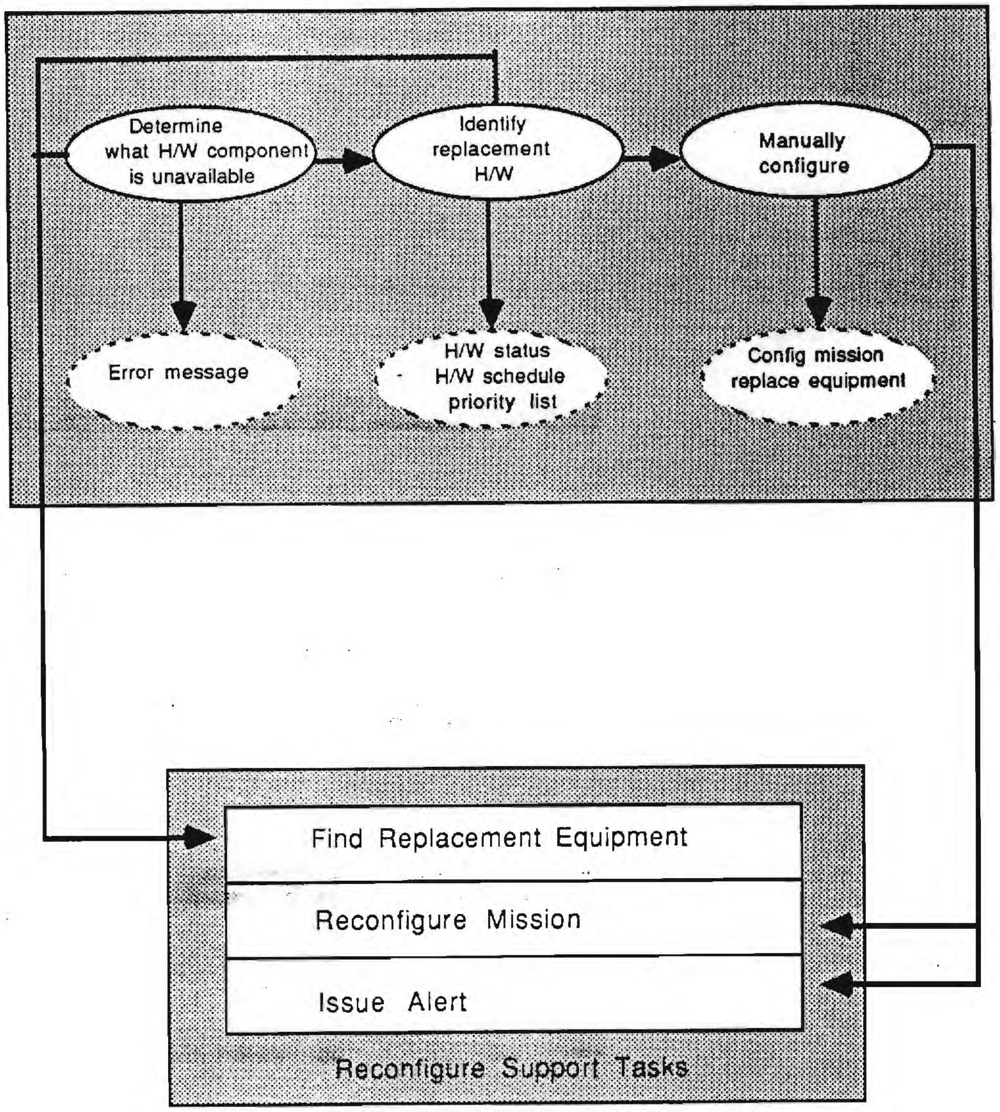
Failure Support Tasks and OFM
Figure 18



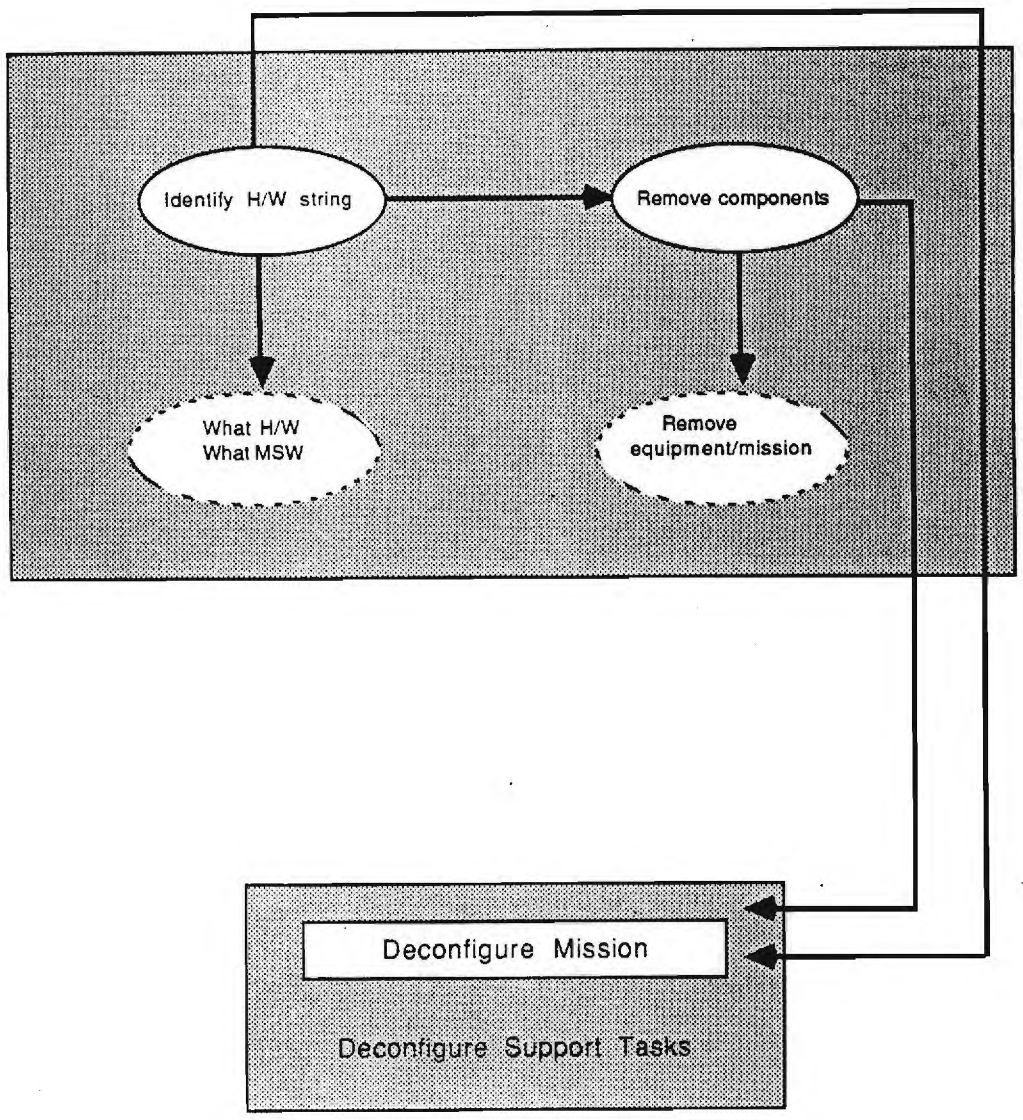
Failure Support Menus
Figure 19



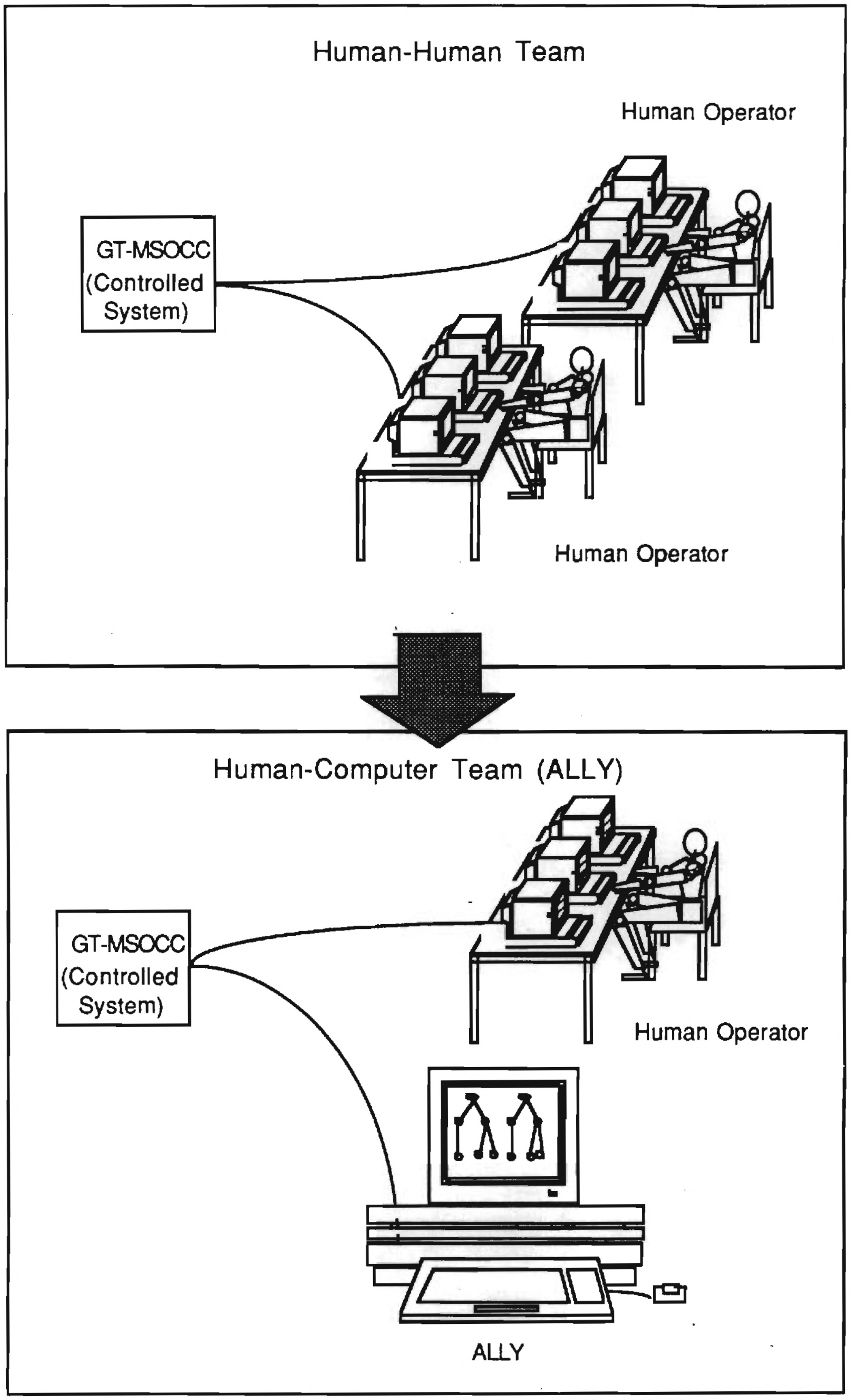
Question Support Tasks and OFM
Figure 20


Determine what H/W component is unavailable
Identify replacement H/W
Manually configure
Error message
H/W status H/W schedule priority list
Config mission replace equipment
Find Replacement Equipment
Reconfigure Mission
Issue Alert
Reconfigure Support Tasks

Reconfigure Support Tasks and OFM
Figure 21

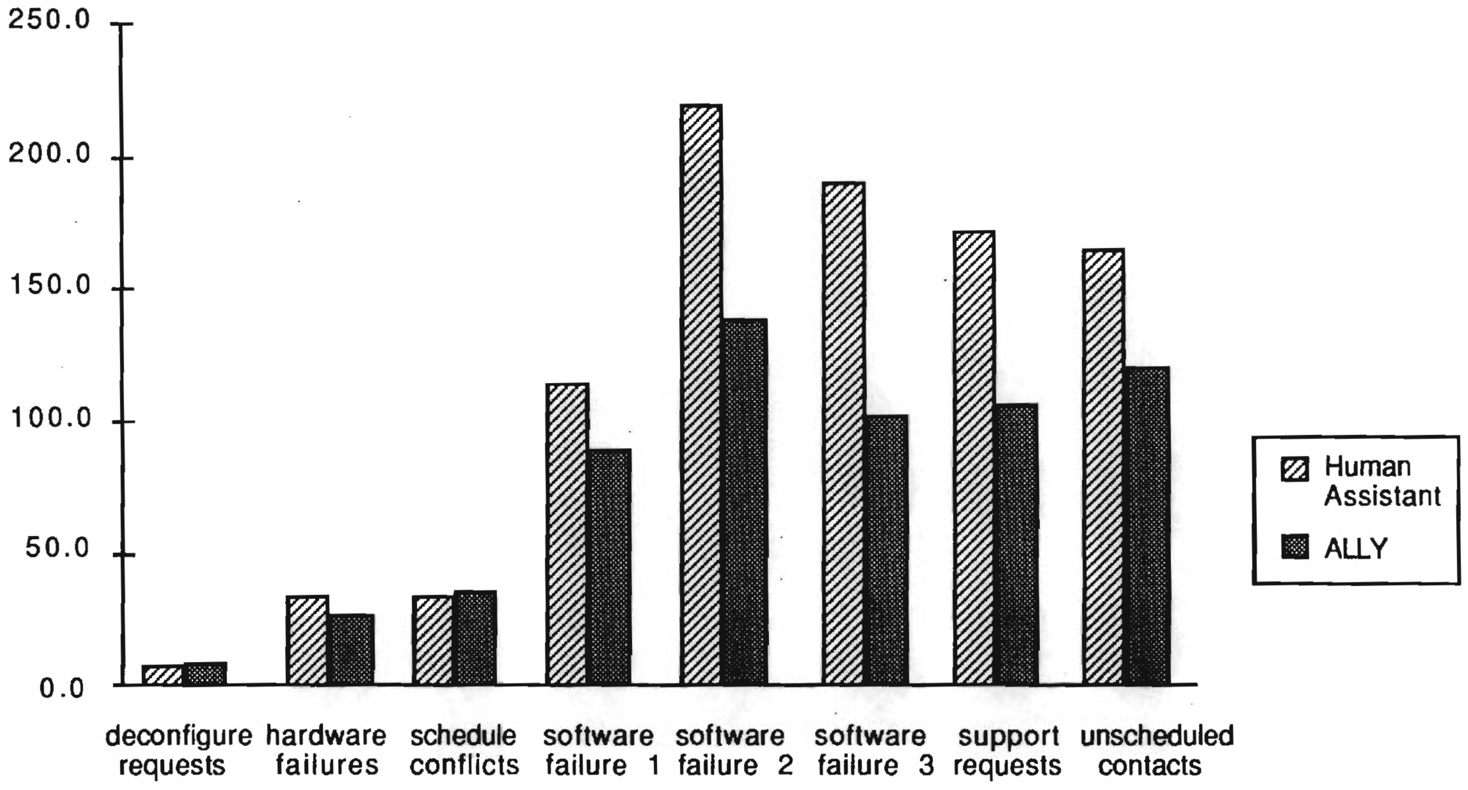


Deconfigure Support and OFM
Figure 22


Human-Human Team
Human Operator
GT-MSOCC (Controlled System)
Human Operator
Human-Computer Team (ALLY)
GT-MSOCC (Controlled System)
Human Operator
ALLY

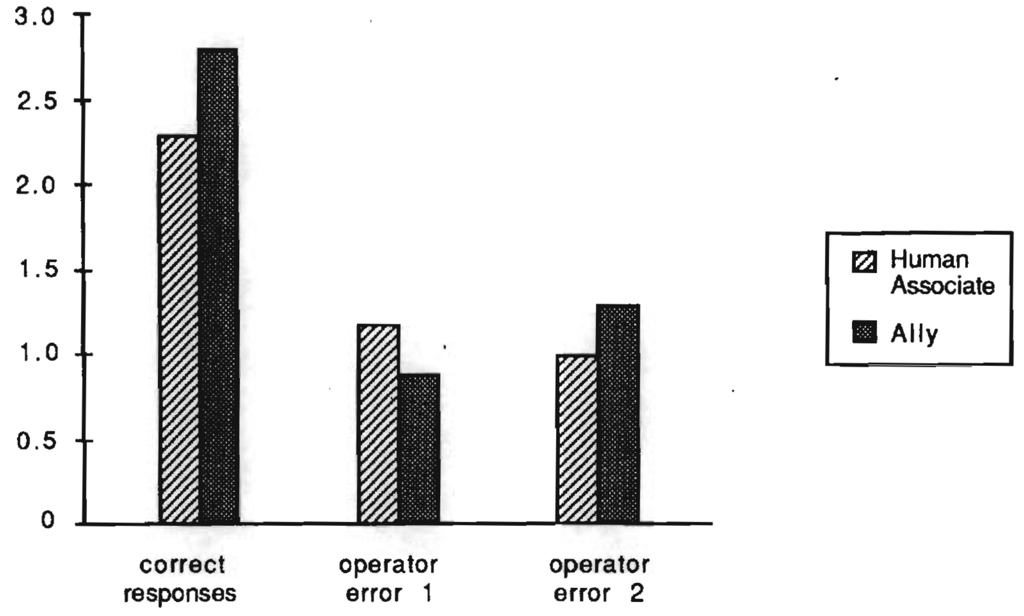
ALLY: An Operator's Assistant
Figure 23

ally.experimentalovervie



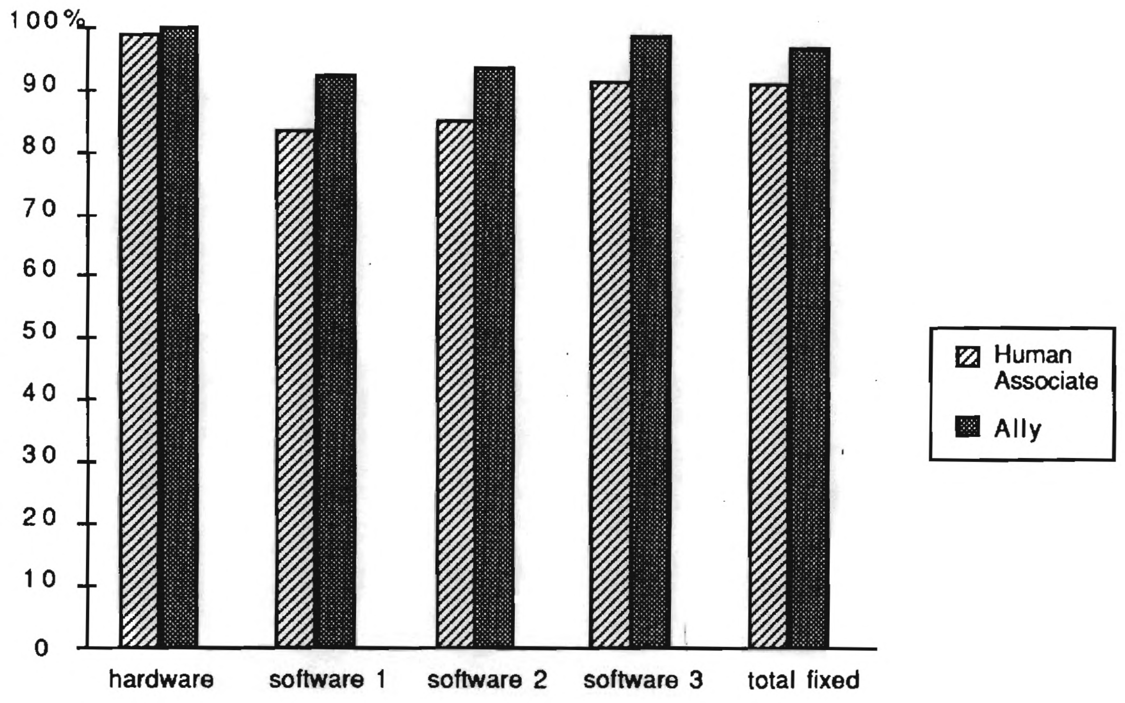
Comparison of Human Assistant and ALLY
Figure 24a

Ally.seconds.graph



Per Session Comparison of Human Associate and Ally
Figure 24b

Ally.perSession.graph



Percent Fixed Comparison of Human Associate and Ally
Figure 24c

NASA-SPONSORED RESEARCH (1985-1990)*
Christine M. Mitchell
Principal Investigator

Published Journal Papers (refereed)

Mitchell, C. M. (1987). GT-MSOCC: A research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, No. 4, 553-570.

Mitchell, C. M. and Saisi, D. S. (1987). Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, No. 4, 573-593.

Mitchell, C. M. and Forren, M. G. (1987). Effectiveness of multi-modal operator interfaces in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17, No. 4, 594-607.

Rubin, K. S., Jones, P. M., and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC 18, No. 4, 618-637.

Rubin, K. S., Jones, P., and Mitchell, C. M. (1988). A smalltalk implementation of an intelligent operator's associate. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '88)*, 234-247.

Jones, P. M. , Mitchell, C. M., and Rubin, K. S. (1990). Validation of intent inferencing by a model-based operator's associate. *International Journal of Man-Machine Studies*, to appear.

Bushman, J. B., Mitchell, C. M. Jones, P. M. and Rubin, K. S. *ALLY: An operator's associate for cooperative supervisory control.* Submitted for publication.

Conference Presentations with Proceedings (refereed)

Forren, M. C. and Mitchell, C. M. (1986) Multi-model interfaces in supervisory control. *Proceedings of the Human Factors Society 30th Annual Meeting*. Dayton, 317-321.

Fath, J. L., Govindaraj, T., and Mitchell, C. M. (1986). A methodology for development of a computer aided instruction program in complex, dynamic systems. *Proceedings of the 1986 International Conference on Systems, Man, and Cybernetics*, 1216-1221.

Mitchell, C. M., Rubin, K. S., and Govindaraj, T. (1986). OFMspert: An operator function model expert system. *Proceedings of the 1986 International Conference on Systems, Man, and Cybernetics*, 282-285.

Bushman, J. B. and Mitchell, C. M. (1986). Modeling the supervisory control hierarchy in a command and control system. *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, 875-878.

* Note: This research was also sponsored in part by NASA Goddard Space Flight Center (NAS5-28575 11/84-3/88), Walt Truszkowski, Technical Monitor.

Forren, M. G. and Mitchell, C. M. (1986). Voice input in real time decision making. *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, 879-884.

Saisi, D. L. and Mitchell, C. M. (1986). Model-based window display interfaces in real time supervisory control. *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, 885-889.

Rubin, K. S., Mitchell, C. M., and Jones, P. M. (1987). OFMspert: An operator function model expert system. *Proceedings of the 4th Mexican National Artificial Intelligence Conference.*

Fath, J. L., Mitchell, C. M., and Govindaraj, T. (1987). Evaluation of an architecture for ICAI programs for troubleshooting in complex dynamic systems. *Proceedings of the 1987 International Conference on Systems, Man, and Cybernetics*, 1145-1148.

Rubin, K. S., Mitchell, C. M., and Jones, P. M. (1987). Using a blackboard architecture for dynamic intent inferencing. *Proceedings of the 1987 International Conference on Systems, Man, and Cybernetics*, 1150-1153.

Jones, P. M., and Mitchell, C. M. (1987). Operator modeling: Conceptual distinctions and an expert system application. *Proceedings of the 4th Annual Mid-Central Human Factors/Ergonomics Conference*, 81-87.

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

Jones, P. M., Mitchell, C. M., and Rubin, K. S. (1988). Intent inferencing with a model-based operator's associate. *Proceedings of 6th Symposium on Empirical Foundations of Information and Software Sciences.*

Jones, P. M. (1988). Intent inferencing by an intelligent operator's associate: A validation study. *Proceedings of the Human Factors Society 32nd Annual Meeting*, 409-413.

Rubin, K. S., Jones, P. M., Mitchell, C. M., and Goldstein, T. C. (1988). A smalltalk implementation of an intelligent operator's associate. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '88)*, 234-247.

Mitchell, C. M. (1989). Decision support for real-time decisions in complex dynamic systems: Some principles and empirical results. *ACM SIGCHI '89 Conference on Human Factors in Computer Systems, Workshop on Real-Time, Decision Support Human-Computer Interaction.*

Jones, P.M. and Mitchell, C. M. (1989). Operator models for supervisory control systems. *Proceedings of the Human Factors Society 33nd Annual Meeting*, 291-295.

Jones, P.M. and Mitchell, C. M. (1989). Validation of a blackboard architecture for dynamic intent inferencing. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, 35-39.

Bushman, J. B., Mitchell, C. M., Jones, P. M. and Rubin, K. S. (1989). ALLY: An operator's associate model for cooperative supervisory control situations. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, 40-45.

Mitchell, C. M. and Govindaraj, T. (1989). A tutor for satellite systems operators. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, 766-771.

Vasandani, V., Govindaraj, T. and Mitchell, C. M. (1989). An intelligent tutor for diagnostic problem solving in complex dynamic systems. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, 772-777.

Chu, R.W., Mitchell, C. M. and Govindaraj, T. (1989). Characteristics of an ITS that evolves from tutor to assistant. *Proceedings of the 1989 IEEE International Conference on Systems, Man, and Cybernetics*, 778-783.

Degrees Awarded

M.S.	(with thesis)	Donna S. Saisi, 1986.
M.S.	(with thesis)	Michelle G. Forren, 1986.
Ph.D.		Janet L. Fath, 1987.
B.S.	(computer science senior design)	Kenneth S. Rubin, 1987.
M.S.	(with thesis)	Patricia M. Jones, 1988.
Ph.D.		James B. Bushman, 1989.

Degrees in Progress

Ph.D.		Patricia M. Jones, 1990
Ph.D.		Rose Chu, 1990
Ph.D.		Thomas Pawlowski, 1990
M.S.	(with thesis)	Serena Smith, 1990
M.S.	(with thesis)	David Resnick, 1990.
M.S.	(with thesis)	Julie Chronister, 1990

Technical Reports

Mitchell, C.M. (1986). *GT-MSOCC: A research domain for modeling human-computer interaction and aiding decision making in supervisory control system*, Report No. 86-1, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Saisi, D.L. (1986). *The use of model-based, window display interfaces in real time supervisory control systems.*, Report No. 86-2, M.S. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Forren, M.G. (1986). *A comparison of voice-augmented and keyboard control in a supervisory control task*, Report No. 86-3, M.S. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Fath, J. L. (1987) *An architecture for adaptive computer-assisted instruction programs for complex dynamic systems*, Report No. 87-3, Ph.D. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Rubin, K.S., Jones, P.M., and Mitchell, C.M. (1987). *OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture,* Report No. 87-6, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

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

Jones, P.M., Mitchell, C.M., and Rubin, K.S. (1988). *Intent inferencing with a model-based operator's associate,* Report No. 88-2, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Bushman, J. B. (1989). *Identification of an operator's associate model for cooperative supervisory control situations,* Report No. 89-1, Ph.D. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Mitchell, C.M. and Govindaraj, T. (1989). *ITSSO: Intelligent tutoring system for satellite operators,* Report No. 89-2, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Newsletters

Rubin, K.S., Jones, P.M., and Mitchell, C.M. (1988). *The OFMspert Project. HOOPLA Newsletter.*

Research Awards

Outstanding M.S. Thesis: School of Industrial and Systems Engineering, Georgia Institute of Technology
Saisi, D.L., (1986). *The use of model-based, window display interfaces in real time supervisory control systems.* Report No. 86-2, M.S. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Best Student Paper Award: The Human Factors Society.
Jones, P.M. (1988). Intent inferencing by an intelligent operator's associate: A validation study. *Proceedings of The Human Factors Society 32nd Annual Meeting,* 409-413.

Runner-Up Outstanding M.S. Thesis: School of Industrial and Systems Engineering, Georgia Institute of Technology
Jones, P.M. (1988). *Constructing and validating a model-based operator's associate,for supervisory control,* Report No. 88-1, M.S. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Outstanding Ph.D. Thesis: School of Industrial and Systems Engineering, Georgia Institute of Technology
Bushman, J. B. (1989). *Identification of an operator's associate model for cooperative supervisory control situations,* Report No. 89-1, Ph.D. Thesis, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Seminar Participation

Mitchell, C. M. (1987). *GT-MSOCC: A domain for research on advanced workstations for supervisory control systems.* Software Psychology. Washington, DC.

Mitchell, C. M. (1987). *OFMspert: An operator function model expert system.* Software Engineering Research Center. Georgia Institute of Technology, Atlanta, GA.

Mitchell, C. M. (1987). *The operator function model expert system.* NASA Ames Research Center, Moffet Field, CA.

Mitchell, C. M. (1988). *The operator function model expert system.* PARC Place Systems, Palo Alto, CA.

Mitchell, C. M. (1988). *The operator function model expert system.* Advanced Decision Systems, Palo Alto, CA.

Mitchell, C. M. and Govindaraj, T. (1988). *Decision of intelligent tutoring systems for complex Domains.* ONR Contractors' Meeting on Cognitive Science and Instructional Theory, Pittsburgh, PA.

Mitchell, C. M., (1988). *A model to infer operator intentions in a complex dynamic system.* Cognitive Science Group, Georgia Institute of Technology, Atlanta, GA.

Mitchell, C. M. and Jones, P.M. (1988). *A model of operator intentions in the control of dynamic systems.* Engineering Psychology Symposium, American Psychological Association 96th Annual Convention, Atlanta, GA.

Mitchell, C. M. and Govindaraj, T. (1988). *Intelligent tutors for supervisory control and fault diagnosis in complex dynamic systems.* Engineering Psychology Symposium, American Psychological Association 96th Annual Convention, Atlanta, GA.

Bushman, J. B., Chu, R., Jones, P. M., and Mitchell, C. M., and Rubin, K. S., (1988). *ALLY: A computer-based operator's associate.* Sixth Symposium on the Empirical Foundations of Information and Software Sciences, Atlanta, GA.

Mitchell, C. M. (1988). *Man-machine systems architecture.* Workshop on man-machine symbiotic systems, Oak Ridge Associated Universities, Oak Ridge, TN.

Mitchell, C. M. (1989). *Decision support for real-time decisions in complex dynamic systems: Some principles and empirical results.* ACM CHI '89 Workshop on Real-Time Decision Support Computer Human Interaction, Austin, TX.

Mitchell, C. M. (1989). *Cognitive engineering: Operator models for supervisory control.* AAS/NASA Human Factors Workshop, NASA Johnson Space Flight Center, Houston, TX.

Govindaraj, T. and Mitchell, C. M. (1989). *Supervisory control in distributed decision making: computer-based operator's associates that evolve from tutor to assistant. Special session on supervisory control.* Supervisory control: 30 Years and counting (a panel hosted by Neville Moray). 1989 IEEE International Conference on Systems, Man, and Cybernetics, Boston, MA.

Mitchell, C. M. (1989). *Operator models as design tools for intelligent displays and aids in complex dynamic systems.* Shell Oil Research and Development Center, Houston, TX.

Mitchell, C. M. (1990). *Supervisory Control, Panel on Real-Time Decision Making,* ACM CHI '90, Seattle, WA.

Appendix B

Major Papers and Technical Reports Sponsored in Part by this Grant