N94- 35062

Intelligent Resources for Satellite Ground Control Operations

Patricia M. Jones
University of Illinois at Urbana-Champaign
Department of Mechanical and Industrial Engineering
1206 W. Green St., Urbana IL 61801
217-333-3938 (Voice)
217-244-6534 (Fax)
pmj@ux1.cso.uiuc.edu (email)

Keywords: knowledge-based spacecraft command and control; intelligent user interfaces

ABSTRACT

This paper describes a cooperative approach to the design of intelligent automation and describes the Mission Operations Cooperative Assistant for NASA Goddard flight operations. The cooperative problem solving approach is being explored currently in the context of providing support for human operator teams and also in the definition of future advanced automation in ground control systems.

INTRODUCTION

The increasing sophistication and complexity of satellite ground control operations requires new approaches to the design of ground control systems. One approach to providing intelligent assistance to operators in real-time control tasks is via an intelligent cooperative problem solving system. Unlike the traditional expert system that queries the user, detects problems, and offers advice, the cooperative problem solving approach advocates providing a variety of flexible intelligent resources to the human operator (Woods, 1986; Jones, 1991; Jones and Mitchell, 1993).

A theory of human-computer cooperative problem solving proposed by Jones (Jones, 1991; Jones and Mitchell, 1993) provides high-level design guidelines for the development of intelligent cooperative automation. These principles advocate human authority, mutual intelligibility, openness and honesty, multiple perspectives, and the management of trouble. In short, the human operator(s) should retain locus of control in interaction and decision making, and the intelligent automation should be obvious, inspectable, unambiguous, provide multiple views of the situation, and provide support for varying levels of help and expertise.

The rest of this paper will focus on the modeling, representation, and architecture of an existing prototype cooperative problem solver system and current research on defining new requirements and architectures for intelligent cooperative support.

THE MISSION OPERATIONS COOPERATIVE ASSISTANT

A prototype cooperative problem solver system, the Georgia Tech Mission Operations Cooperative Assistant (GT-MOCA, hereafter referred to as MOCA), was developed for the context of NASA Goddard real-time flight operations and was experimentally evaluated with ERBS and COBE flight analysts in the context of a high-fidelity real-time interactive simulation of MOCA provides an interactive normative operator model, messages, and interactive graphics to be utilized in conjunction with the operator's existing work environment. In particular, the interactive normative model provides a visualization of the content and structure of current expected activities and allows operators to query this representation and also delegate activities to MOCA.

MOCA is based on the operator function model (OFM) (Mitchell, 1987) and the OFMspert architecture (Rubin, Jones, and Mitchell, 1988). The operator function model is a heterarchic-hierarchic network model that specifies activities at various levels of abstraction and the actions needed to successfully accomplish those activities. Activities and actions are nodes in the OFM network, and arcs represent system triggering events or the successful completion of activity. The OFM is implemented as a blackboard architecture that forms the basis of intent inferencing (Rubin, Jones, and Mitchell, 1988). Here, the basic OFMspert components as implemented in MOCA and the MOCA-specific Cooperative Problem Solver component are described.

Figure 1 illustrates the overall software architecture. The Controlled System Interface class parses information from the controlled system (in this case, a high-fidelity interactive simulation) and sends appropriate messages to other OFMspert components.

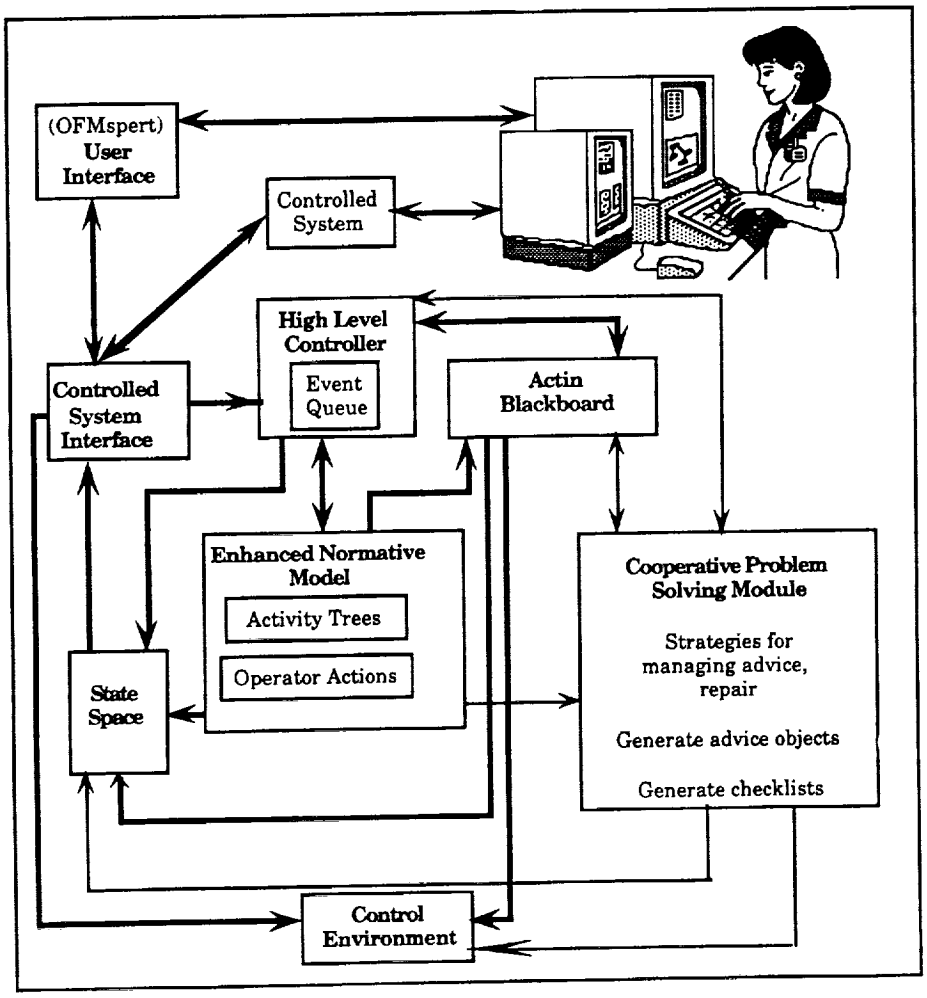

Figure 1. The MOCA architecture.

The State Space class represents the current state of the controlled system. Thus, many of the classes that are fully implemented in the simulation itself (e.g., classes to represent spacecraft components) are partially replicated here to provide a modular, efficient representation of the current system state. The Control Environment class represents the simulation user interface as a collection of DisplayPanel objects, each of which has a name,

a character string denoting what information that panel contains, and an integer denoting whether or not the panel is currently displayed.

The EnhancedNormativeModel class encapsulates the knowledge of the operator function model for a particular domain (here, real-time flight operations), where the model is represented as activity trees (organized functions, subfunctions, and tasks) and actions. In general, the Enhanced Normative Model contains tables of all these structures and member functions that are used to instanciate and schedule the removal of activity trees and actions on the blackboard. Most of this information is file-driven; at the beginning of the program, the Enhanced Normative Model reads in files that give the structure of the activity trees (i.e., function-subfunction-task relationships), their supporting actions, and the structures of all the nodes in the operator function model. The different information and formats in each file correspond to four classes to represent that information. The class ActivityTreeStruct represents the model-driven information (i.e., function-subfunction-task activity trees) of the operator function model. The ActivityTreeStruct class also encapsulates the alert message associated with each tree. The class ModelTreeStruct is a subclass of ActivityTreeStruct that also includes the actions associated with each task. The class ConnectionStruct encapsulates intent inferencing information; i.e., the names of actions are associated with the names of the tasks which those actions support. Also, the ConnectionStruct class contains members to represent the "what is" and "how to" information associated with each action. Finally, the NodeStruct class is used to represent ActivityNodes. This class represents a node's level, name, type, purpose, and enabling event as character arrays. This information is used by Enhanced Normative Model methods for "instanciating" a particular tree.

The Blackboard class of MOCA is essentially the same as previous implementations of the OFMspert blackboard (Rubin, Jones, and Mitchell, 1988; Chronister, 1990). The blackboard data structure has four levels: functions, subfunctions, tasks, and actions. The control structures consist of three lists of events: the clock events list, the problems list, and the events list. The clock events list is a time-sorted list of events to be done at prespecified times. The problems list is a list of unconnected actions. The events list is a list of current events to be processed. The knowledge sources are member functions for processing events to maintain and update the blackboard. Some extensions were made to the representation of blackboard nodes. The abstract superclass ActivityNode now also contains a member denoting the names of the supernodes that this node can connect to, because subfunctions and tasks as well as actions can be posted and connected to nodes at a higher level. ActivityNode also contains a new member that describes the enabling event for its posting (e.g., the Monitor function's enabling event is that telemetry data have begun to arrive). The subclasses of ActivityNode are FunctionNode, SubfunctionNode, TaskNode, and ActionNode. The ActionNode class includes "what is" and "how to" information.

Part of the structure of the Cooperative Problem Solver class is given in Table 1. It represents message-sending paths with other OFMspert components, the history and current focus of allocated activities and communicative acts, the current checklist of actions to be displayed or performed, and an organized collection of declarative information used in the performance of allocated bookkeeping activities. The Cooperative Problem Solver class has a number of member functions for the creation and maintenance of these structures. Besides the behaviors of creating, adding, deleting, and finding items on the various lists, the Cooperative Problem Solver has a number of key functions that are summarized in Table 2.

As noted in these tables, significant classes associated with the Cooperative Problem Solver represent communicative knowledge in the form of *communication knowledge objects* (in a similar spirit to context spaces (Reichman, 1985). Class CommunicationKO represents the communicative act's name, the time it was initiated, the time it was ended, its initiating and terminating conditions (as described in the main thesis text), its history of status changes, purpose, priority, and content. Class CommunicationKO has three

subclasses that represent specialized types of communicative acts. Class AdviceReminderKO has an additional member that represents the feedback structure generated by blackboard assessments. Class LimitViolationKO has additional members to represent the status and current value of the associated spacecraft parameter (whose name is

Table 1. Partial structure of Class Cooperative Problem Solver

Member	Description
giveAdvice	Boolean variable Denotes whether or not to present advice/reminder messages to the user. This is the variable that is set when the user clicks on the "Give Advice" checkbox on the MOCA main menu.
allocatedFunctions allocatedSubfunctions allocatedTasks	Pointers to an AllocatedActivity object These members of class CPS denote the heads of linked lists of AllocatedActivity objects that correspond to the functions, subfunctions, and tasks allocated to MOCA.
currentConversational-Context	Pointer to a CommunicationKO object Denotes the communicative act which is the current focus of processing by MOCA.
currentActivity	Pointer to an ActivityKO object Denotes the activity which is the current focus of processing by MOCA.
currentAdvice	Pointer to an AdviceReminderKO object Denotes the advice which is the current focus of processing by MOCA.
currentChecklist	Array of 10 character pointers (character strings) Denotes the current dynamically-generated checklist to be displayed to the user and/or to be performed by MOCA.

Table 2. Significant member functions of class CooperativeProblemSolver

Member function	Description
propagateResponsibility	When the user allocates an activity node to MOCA, its responsibility is set to "MOCA". This effect propagates downward such that the responsibility of subnodes is also set to "MOCA". Propagation also occurs upward such that supernodes' responsibility is either "shared" (if other subnodes of the supernode are still allocated to the human) or "MOCA" (if all other subnodes have also been allocated to MOCA).
makeRespCommAct	Given an activity node allocated to MOCA, this function returns an associated CommunicationKO that represents the communicative act of "echoing" that act of delegation.
manageCommunication	This function is the heart of MOCA's management of communicative acts. Every CommunicationKO that is created and added to the appropriate list is passed as an argument to this function. Currently, manageCommunication sets the CommunicationKO as the currentConversationalContext, and decides, based on the purpose of this act, which function to call to generate a message to the user as shown in the next box.
generateAdvice generateAlert generateConfirmation generateExplanation generateDetails generateAcknowledgment generateLimitViolation-Message generateDataDropout-Message generateLimitSummary-Message	The functions called from manageCommunication if the currentConversationalContext's purpose is, respectively, to provide ADVICE on missing, out of order, or late actions; ALERT the user that certain functions, subfunctions, and/or tasks were posted on the blackboard; CONFIRM that MOCA has performed an action in response to an activity delegation; EXPLAIN how to do an action on the dynamic checklist display; ELABORATE on "what is" an action on the dynamic checklist display; acknowledge that MOCA received the request to perform an activity (ACTIVITY_MANAGEMENT); or provide domain-specific alert messages. This function places the currentConversationalContext's content in the appropriate textlist display.
generateDynamicChecklist	Given an activity node for which the user requests delegation or a checklist, this function queries the EnhancedNormativeModel to find the actions that constitute the successful fulfillment of that activity.
generateSpecialChecklist	Given an activity node for which the user requests delegation or a checklist, this function examines context-specific information to generate the appropriate checklist of actions. For example, if the user requests a checklist for the TRPBK subfunction, this function examines the current support characteristics to decide if the appropriate control actions involve Tape Recorder 1 or Tape Recorder 2 (i.e., the list may be TR1STBY and TR1PBK, or TR2STBY and TR2PBK).

assigned as the name of the LimitViolationKO). Class DataDropoutKO has the same structure of a CommunicationKO, but is treated differently; its name represents the type of data loss (e.g., "fwd link") and its start and end times represent the start and end times of the data loss.

INTELLIGENT SUPPORT FOR ACTIVITY MANAGEMENT

MOCA provides the basis for further architectures to support human-computer cooperative problem solving. MOCA's limitations,however, included its lack of integration between the resources for cooperative problem solving and its lack of support for planning (Jones and Mitchell, 1993). Currently we are building the Intelligent Support for Activity Management (ISAM) architecture which addresses these issues (Jones, 1993a, 1993b, Jones and Goyle, 1993; Jones, Patterson, and Goyle, 1993). In particular, activities are represented more completely by Activity Objects that explicate knowlege of priorities, resources, constraints, and temporal relationships between activities. Furthermore, the context of activity -- including the current state of the user's "information space" (e.g., displays), current state of the controlled system, and evolving status of artifacts that both guide activity and are the result of activity -- is explicitly captured and represented by objects as well. This architecture is currently under development; a high-level conceptual overview is provided in Figure 2 below.

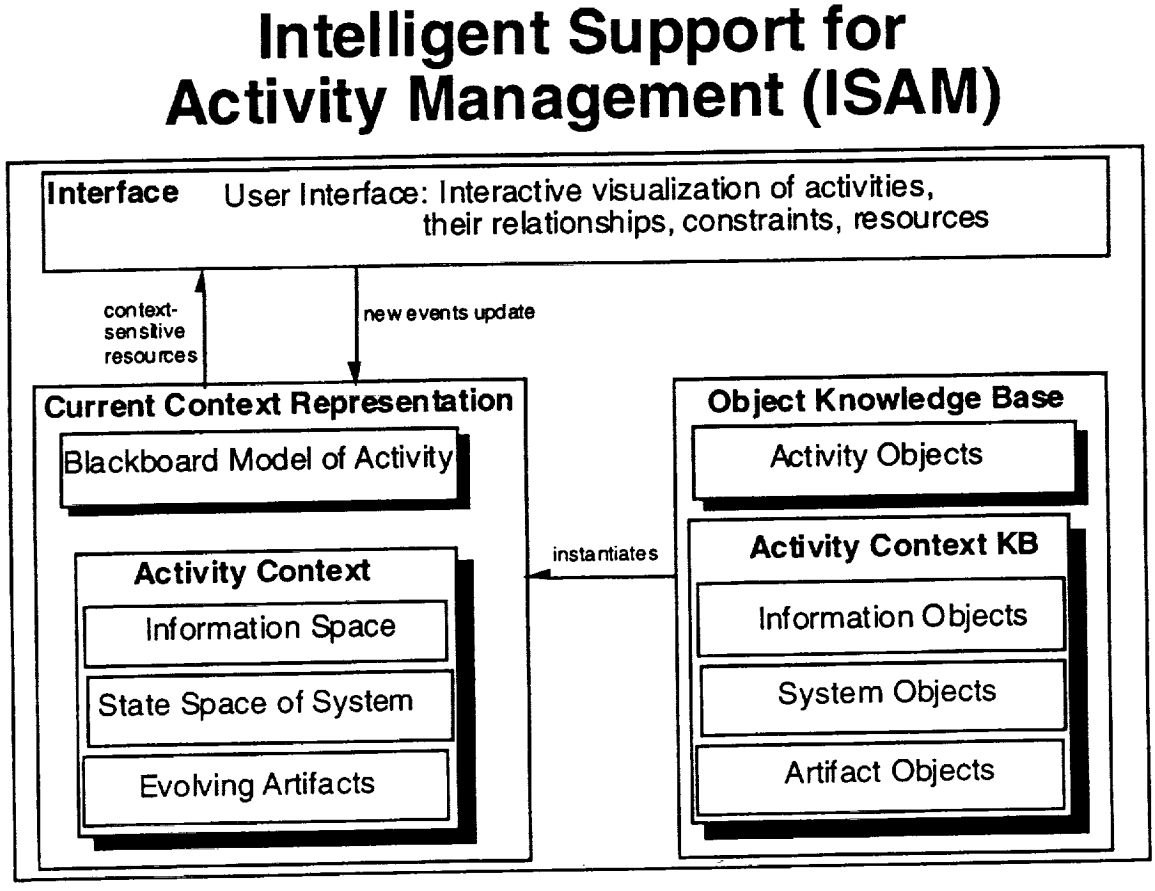

Figure 2. ISAM Architecture.

ACKNOWLEDGMENTS

This research is supported by grants from NSF (IRI92-10918) and NASA Goddard Space Flight Center (NAG5-244).

REFERENCES

Chronister, J. A. (1990). MS thesis, School of ISYE, Georgia Institute of Technology.

Jones, P. M. (1993a) Cooperative support for distributed supervisory control: Issues, requirements, and an example from mission operations. *Proceedings of the ACM Intenational Workshop on Intelligent User Interfaces*, 239-242, Orlando, FL January 1993.

Jones, P. M. (1993b). Cooperative work in mission operations: Analysis and implications for computer support. Manuscript in preparation.

Jones, P. M. (1991). Human-computer cooperative problem solving in supervisory control. PhD dissertation, School of ISYE, Georgia Institute of Technology.

Jones, P. M. and Goyle, V. (1993). A field study of TPOCC mission operations: Knowledge requirements and cooperative work. EPRL-93-05, Engineering Psychology Research Laboratory, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign.

Jones, P. M. and Mitchell, C. M. (1993). Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent operator's associate. Manuscript accepted for publication to *IEEE Transactions on Systems, Man, and Cybernetics.*

Jones, P. M. and Mitchell, C. M. (1991a). A mechanism for knowledge-based reminding and advice-giving in the supervisory control of complex dynamic systems. *Proc. of the 1991 IEEE International Conference on Systems, Man, and Cybernetics.*

Jones, P. M. and Mitchell, C. M. (1991b). Cooperative interaction in the supervisory control of complex dynamic systems. *Proc. of the 1991 IEEE International Conference on Systems, Man, and Cybernetics.*

Jones, P. M., Mitchell, C. M., and Rubin, K. S. (1988). Intent inferencing with a model-based operator's associate. *Proceedings of the Sixth Symposium on Empirical Foundations of Information and Software Sciences* (249-258). Atlanta,GA.

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, 33*, 177-202.

Jones, P. M., Patterson, E. S. and Goyle, V. (1993). Modeling and intelligent aiding for cooperative work in mission operations. *Proc. of the 1993 IEEE International Conference on Systems, Man, and Cybernetics*, Le Touquet, France.

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.

Reichman, R. (1985). *Getting computers to talk like you and me.* Cambridge, MA: MIT Press.

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, 18*, 618-637.

Woods, D. D. (1986a). Cognitive technologies: The design of joint human-machine cognitive systems. *The AI Magazine, 6*, 86-92.