Expert System Applications to Telecommunications

Edited by

Jay Liebowitz

Department of Management Science
George Washington University
Washington, D.C.

WILEY
INTERSCIENCE

A Wiley-Interscience Publication
JOHN WILEY & SONS
New York • Chichester • Brisbane • Toronto • Singapore

Library of Congress Cataloging-in-Publication Data

Expert system applications to telecommunications/edited by Jay Liebowitz.

p. cm.

"A Wiley-Interscience publication."

Includes bibliographies and index.

ISBN 0-471-62459-4

1. Telecommunication systems—Data processing. 2. Expert systems (Computer science) I. Liebowitz, Jay.

TK5102.5.E97 1988

384—dc 19

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

DISTRIBUTED EXPERT SYSTEMS: FACILITY ADVISOR

Barry G. Silverman
Institute for Artificial Intelligence, George Washington University, Washington, DC

Henry H. Hexmoor, Ravi Rastogi, and Joseph Chang
IntelliTek, Inc., Rockville, Maryland

INTRODUCTION

chapter describes design considerations, progress to date, and next steps for loping a set of cooperating expert systems called the FACILITY ADVISOR can support real-time operations, off-line planning, and adaptive behavior at ities in general and at spacecraft ground facilities in particular. The focus of FACILITY ADVISOR is to support the distributed problem protocols and ıvior that traditional expert systems technology tends to ignore. The goal is to d a FACILITY ADVISOR shell that (1) consists of a number of validated, etal elements that can be tailored to the individual supervisory positions at any ıity, (2) can support the distributed problem-solving aspects of these jobs, and ɔan successfully be extended (or integrated with any previously existing stand-ıe supervisory expert system) to support the stand-alone nondistributed super-ıry control problems encountered at that position.

Because of the relative newness of distributed expert system (DES) technology the no-risk requirement of facility operations, a DES testbed activity for the CILITY ADVISOR is described. Whether or not all the goals of the FACILITY VISOR can be met, the testbed is also expected to provide, at a minimum, a S training ground, lessons learned, how-to-do-it manuals, and insights into grating DESs into facility operations and planning activities.

An increasing number of facilities exist where a semiautomated process is op-ɛd and controlled by a team of human supervisory operators. For example, in ypical telecommunications network, three main facilities include (1) an orbit

facility that has orbit equations concerning all orbiting platforms and that is the source of raw orbit information needed to support each spacecraft acquisition event; (2) a telecommunications control center that receives spacecraft acquisition schedule requests from various users (customers), which it in turn merges with raw orbit data to generate a message that the ground terminal can use to "acquire" the spacecraft at the appropriate time, location, and frequency; and (3) a telecommunications ground terminal that error checks incoming acquisition messages and, if they pass acceptability tests, proceeds to acquire the appropriate spacecraft at the scheduled interval. In each of these facilities, there is a hierarchy of human-staffed positions, each responsible for a subset of the overall task of running the process.

This chapter describes a set of expert systems (ESs) to be researched, built, and connected as a DES. The intent is to show a generic expert system approach (1) to certain classes of human supervisory positions commonly encountered in many facilities (e.g., facility schedulers, workstation operators, and facility hardware/software fault detection positions) and (2) to create a cooperating set of expert systems designed to operate in a loosely coupled hierarchy (i.e., a DES) that can serve as a FACILITY ADVISOR. The FACILITY ADVISOR is a set of ES kernels that potentially can be tailored to any facility. An overview of this chapter can be offered in terms of three principal areas of activity:

1. *Cognition and Protocol Analysis Studies.* The extensive results to date in studying human operators' protocols in ground facilities are being incorporated and extended, particularly in terms of adaptive problem-solving protocols that support the "telescience" concept.
2. *Logical Model of the DES.* The protocol and cognitive results have been formulated into an idealized facility level blackboard ES. The logical model as described in this chapter provides design guidelines for (1) each of the over one dozen components of that ES, (2) how these components interact on both an intra- and interfacility level, and (3) an idealized dual-calculus approach wherein real-time operations are supported by fast ESs and planning functions are off-loaded to more powerful plausible reasoning expert systems.
3. *Physical Implementation: DES Prototype Testbed.* Several of the key elements of the logical model have already been implemented on the VAX and LISP machine. These are explained as are the set of tasks anticipated to research and extend the prototype fully into a generic FACILITY ADVISOR that holds the potential to be used at actual facilities.

Because of the research nature of the FACILITY ADVISOR, it is hoped that it can serve as a training ground for new knowledge engineers and a testbed for demonstrating and teaching how ESs and DESs may be created. For each of the three areas of research just mentioned, a set of items are being generated to provide (1) how-to-do-it handbooks, (2) good design examples that can be emulated, (3) discussion of tools that have proved useful in the various knowledge engineering steps, and (4) lessons learned in the DES arena.

1.1. Hierarchical Framework Definition

Numerous metaphors exist in which *decentralized, loosely coupled* collections of entities *cooperate* in task execution and result syntheses. The entities *cooperate* in the sense that none of them has sufficient information to solve the entire problem and mutual sharing of information is necessary to allow the groups as a whole to produce an answer or result. By *decentralized*, it is meant that both control and data are logically and often spatially distributed; there is neither global control nor global data storage. There is often a *hierarchy* in which control is delegated along with subtasks responsibility. Finally, *loosely coupled* means that individual entities are free to spend more time in computation than in communication.

For example, the human brain is organized into a hierarchy, each level of which performs higher and higher mental functions:

1. The human brain is hypothesized to be a composite structure consisting of at least three layers: (a) a reptilian brain that provides basic reflexes and instinctive responses; (b) an old mammalian brain that is more sophisticated and capable of emotions and delayed responses; and (c) a new mammalian brain that can imagine, plan, and manipulate abstract symbols. The outer layers inhibit and modulate the more primitive tendencies of the inner layers.
2. Within each layer a hierarchy of distributed intelligent nodes is connected to each separate sensory–motor system. These become increasingly interrelated at the higher levels and eventually merge into a unified command and control structure. This enables a complex organism to coordinate its actions in pursuit of high-level goals.

It seems clear that intelligent human behavior is not achieved by one large "routine" but by numerous, small, distributed, and independent centers of intelligence working in harmony with each other.

By the same token, one could envision a FACILITY ADVISOR as a hierarchy of several levels with multiple, stand-alone nodes of intelligence distributed throughout each level. Figure 1 illustrates how the FACILITY ADVISOR will capture this distributed characteristic. The three lowest layers are what exist at present-day automated facilities, while the two higher layers are staffed by humans employed in these facilities. While many facilities are currently sponsoring stand-alone ES projects to support one or the other of the "example station positions," there is no attempt at present to approach the facility as a complete unit, which in fact it is.

As shown in Figure 1, the bottom three layers of a facility (i.e., existing automation, existing sensors/actuators, and actual facility hardware/software) can be viewed as corresponding to the reptilian brain, sensory system, and motor parts (body), respectively. These three levels already exist at various facilities (such as NASA Goddard Space Flight Center) as the automatic control execution and feedback and system parts.

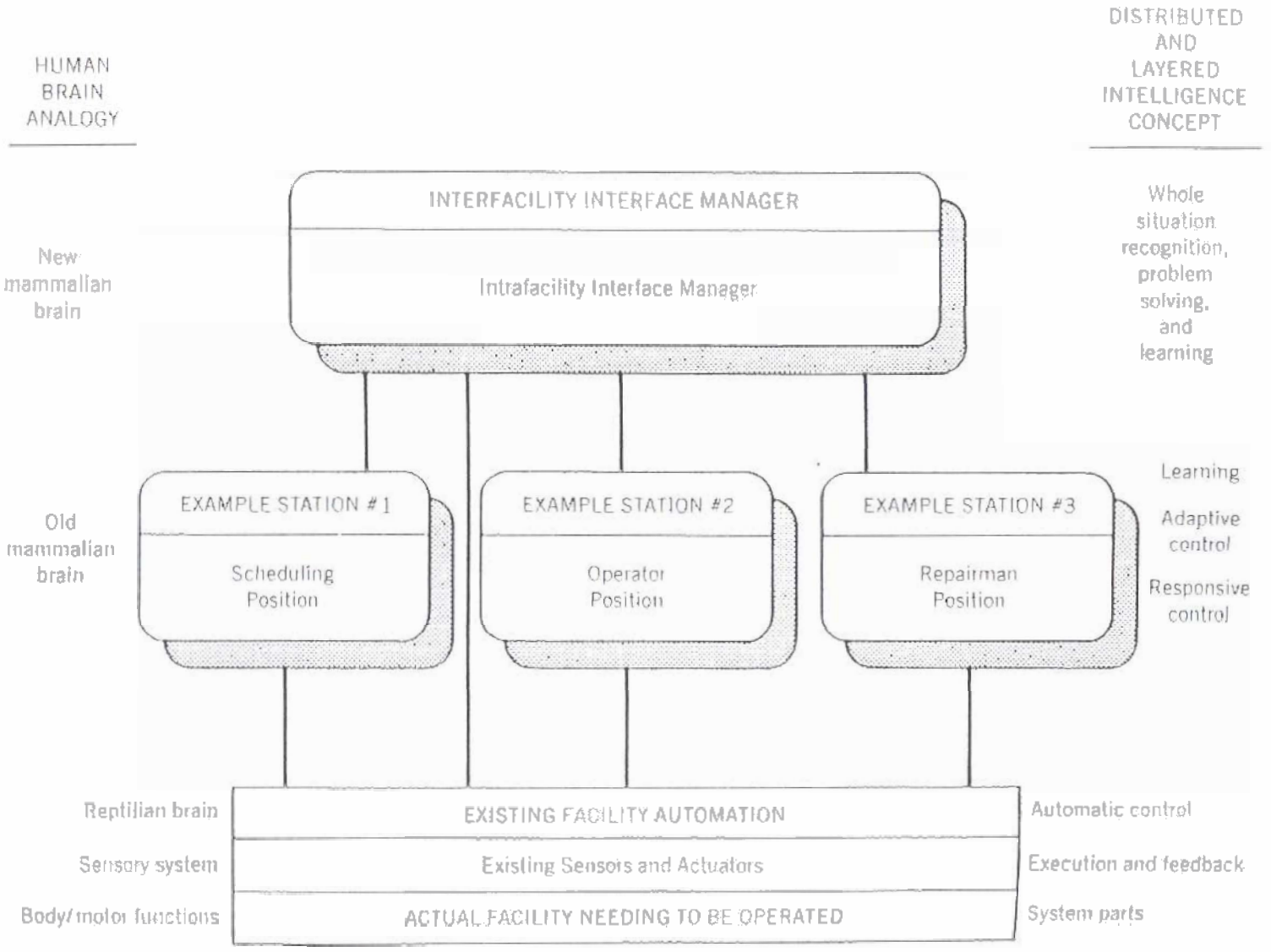


Figure 1. Relevance of DES principles to a FACILITY ADVISOR.

The goal of the FACILITY ADVISOR is to implement DES technology corresponding to the two upper layers. The two higher levels are not simply a collection of three or four stand-alone ESs. Like brain level counterparts, each of the four boxes (MANAGER plus three example stations) in fact contain a *set* of specialized but cooperating ESs each capable of performing a single, intelligent function.

1.2. Distributed Expert Systems

At this point it is useful to extend the DES capabilities discussion to a general view of DES advantages and disadvantages. This will hopefully set the stage for the reader to understand some of the research issues and motivations involved with any DES or hierarchy of ESs that might be attempted. To begin with, several of the potential advantages of a DES approach are summarized in the top half of Table 1. In short, DESs are achieved by relaxing central control, eliminating global knowledge, and permitting entities to select the tasks on which they will work. If this is carried too far, the individual entities could begin to encounter any of several pitfalls listed in the bottom half of Table 1 along with selected solution techniques.

The items in Table 1 have served to highlight a few of the reasons for choosing a DES approach and a few of the research questions that need to be considered carefully in order to build an effective DES such as a FACILITY ADVISOR. More will be said on each of these items in ensuing sections. At this point, it seems fruitful to introduce one final issue: the role of blackboards in a DES.

Traditionally, well-structured programs exhibit high degrees of modularity where each module performs a specific task. An important distinction, however, is the manner in which control is passed to perform a given task. Decentralized control is an essential feature of a system able to decompose a problem before it solves it.

To understand this distinction, it is useful to envision a trend from traditionally designed distributed software programs to DESs. For the sake of clarity, DESs will be discussed as a range from blackboard ESs to full DESs as displayed across the top of Table 2. The ESs focused on here are the more sophisticated because they use blackboards (e.g., HEARSAY II) for communicating knowledge between individual specialist ESs. ES nodes of fully decentralized DESs are generalists rather than specialists and all entities can perform any tasks.

Briefly, it can be seen that blackboard ESs impose less certainty in the control structure than traditional designs. One of the reasons to build blackboard ESs is to allow different points of view from different specialists to be aired, hypotheses to be tested, and conclusions to be reached on the basis of the best available (and uncertain) information. Despite the power of this approach, experiments have shown that it leads to potentially redundant and unnecessary processing and the formulation of answers that are ultimately rejected. This is a redundancy that, if not carefully controlled, can be deleterious to the speed and ability of real-time applications.

In the same context, fully decentralized DESs allow entities at different levels

full freedom of choice in not only who to task but for whom they will work. The result is entities receiving commands from higher-level entities and deciding which task assignment to pursue. This permits busy nodes to shed workload and idle ones to pick up the slack. Finally, since an entity possesses internal metaknowledge, it will not process a task assignment it "knows" it cannot do well.

The lesson of this discussion is to permit multiple types of control ranging from centralized to fully decentralized DESs as appropriate to the application. It is possible also to consider an adaptive (learning) mechanism that can dynamically determine the appropriate combination: for example, an entity might start out fully decentralized and through experience recognize either a few specialists or possibly one it should always return to with a given problem. Empirical studies for each application will be necessary to determine which forms of control (three from Table 2 plus the adaptive) are most appropriate. An adaptive technique is elaborated in Section 1.3.3.

1.3. Goals for the FACILITY ADVISOR

One of the goals for the FACILITY ADVISOR* is to explore how three worthy design guidelines can successfully be integrated.

1.3.1. Supervisory Controller Position (SCP) Expert Systems. Many of the ground operations personnel are employed at automated workstations. The person plus the workstation comprise three levels of intelligence at the position: one human supervisor plus two computerized lower levels of intelligence (see Figure 1). For convenience of discussion, all three level positions shall be referred to as SCPs: the reader can mentally replace the term SCP with any other word (e.g., scheduler, operator, analyst, or command controller). ESs built for facilities, be they stand-alone ESs or part of a DES, must carefully be integrated into the SCP tripartite if they are to be useful as suggested by the communication control lines in Figure 2.

1.3.2. Cooperative Interdependent Behavior. While most SCPs have relatively well-defined areas of responsiblity, in order to do their job they must interface and cooperate with other SCPs. For example, within a given facility there are generally at least three (and often many more) interacting positions: (1) the scheduler, who decides when equipment and other resources may be allocated to support each user, (2) the operator, who uses the resources to perform a user-requested service and who detects and corrects quality problems of the end product (e.g., message code errors and data set noises) by obtaining inputs from the equipment monitor to assist in problem isolation tasks, and (3) an equipment monitor, who detects and isolates equipment and resource problems and either corrects them

*It should be noted that not all elements of the FACILITY ADVISOR DES will be ESs. Some elements will use non-ES forms and algorithms of artificial intelligence (AI), and other elements will be objects, frames, and/or databases. The acronym DES, however, will be used in this chapter to encompass all the elements.

Benefits of DES

ed Reliability and Flexibility. Achieved through redundancy in communication , and DES entities and through the modularity of design (which permits incremental ion of new entities and communication paths).

ed Real-Time Response. Achieved through parallelism and through the placement of ES entities near sensing devices and devices to be controlled.

Communication Costs. Achieved by abstracting (preprocessing) data for transmission ring communication bandwidth requirements) and by placing DES entities near the (reducing the distance over which the data must be transmitted).

Processing Costs. Achieved through the use of cheaper, less-complex processors, h can be mass produced, and through load sharing (allowing relatively idle DES ies to handle some of the work of a busy processing node).

ed Software Complexity. Achieved by decomposing the problem-solving task into asks, each more specialized than the overall task; the result of this decomposition is ced software complexity at each DES entity (which performs a small number of asks) as compared to one piece of structured software performing the complete task.

ed Search Space. The decomposition minimizes potentials for combinatorial plosion." Each DES entity only "worries about" a portion of the search space. This ices solution time by an order of magnitude and increases near real-time capability.

System Specialization. Complex behavior is achieved by the coordination of less plex, narrowly focused entities of "specialists." The nature of living entities can be lated more accurately, a feature that at a minimum facilitates acquisition and tagement of the knowledge base.

ing Incomplete, Uncertain, Conflicting Information. Like ESs, DES are adept at ching the best possible conclusion in the presence of partial and uncertain rmation. One of their purposes, however, is to be better than ESs for handling flicting information from alternate knowledge sources or sensors.

DES Limitations to Avoid

Top-Level Control. If top-level control is extremely weak (or nonexistent), the ordinate specialists may speak up in different, possibly conflicting voices. Multiple ntrol must not be allowed to substitute for unitary control and the decomposition ction must be carefully and thoroughly tested.

cessary Processing. Entities will accept and perform tasks that other entities are eady doing unless a careful scheme to prevent this is incorporated. It is necessary to lude some form of metalevel status exchange.

of Synchronization. Without proper guidance, entities can get out of synch (timewise functionwise) and be unable to understand each other. Again, it is necessary to change a metalevel view of their local problem-solving task.

rated Communication Channels. Interactions (communications) among entities in a stributed architecture are generally slower than computations. The framework for operation must minimize communication to avoid channel saturation and to avoid tities sitting idle while messages are being transmitted.

ptimal Results. Reduced search spaces of each specialist can lead them to less than timal conclusions unless the decomposition and results synthesis functions provide bust and possibly parallel subtask allocations.

eating Mistakes/No Learning. The same problems, if always handled the same way, will ad to repetition of mistakes. Learning should be accomplished either by incorporating a nemory" or introducing it manually as in the Japanese quality control approach.[a]

[a] ality is introduced into Japanese products by identifying each product defect as it occurs and by ually correcting the product line so it cannot occur again. Eventually, products are virtually defect-free.

TABLE 2 Transfer of Control (Selection)

	Traditional, Centralized Structured Program	Expert System (Early Blackboard)	Fully Distributed
1. Character of choice	None, predefined by the programmer (master, slave)	Caller (interpreter) chooses among respondents; respondents have no freedom of choice	Each node is aware of and able to choose between those nodes it might respond to and make requests of
2. Type of information for choosing	Subroutine name calls, preset	Patterns to be matched	Complex, dynamic
3. Information transfer and transmission	Caller to respondent transfer "directcast" to a single respondent	Caller to respondent and respondent to caller queries all with fixed format; broadcast to all potential respondents, each of whom must reply	Full opportunity for interactive conversation general transmission mechanism with freedom to reply or not
4. Evaluation of which respondent to choose	Centralized, predefined	Global evaluation function; metarule set is used	Localized and context-specific evaluations; can even decide that no one is currently suitable and reinitiate process later
5. Selection process	None except in the master or executive	Conflict resolution schemes: for example, evaluate priorities embedded in each rule. All selection is centralized in the rule interpreter. Respondents (rules) have no say in it	Mutual selection (symmetry on both sides based on evaluation of info from all requestors and respondents)

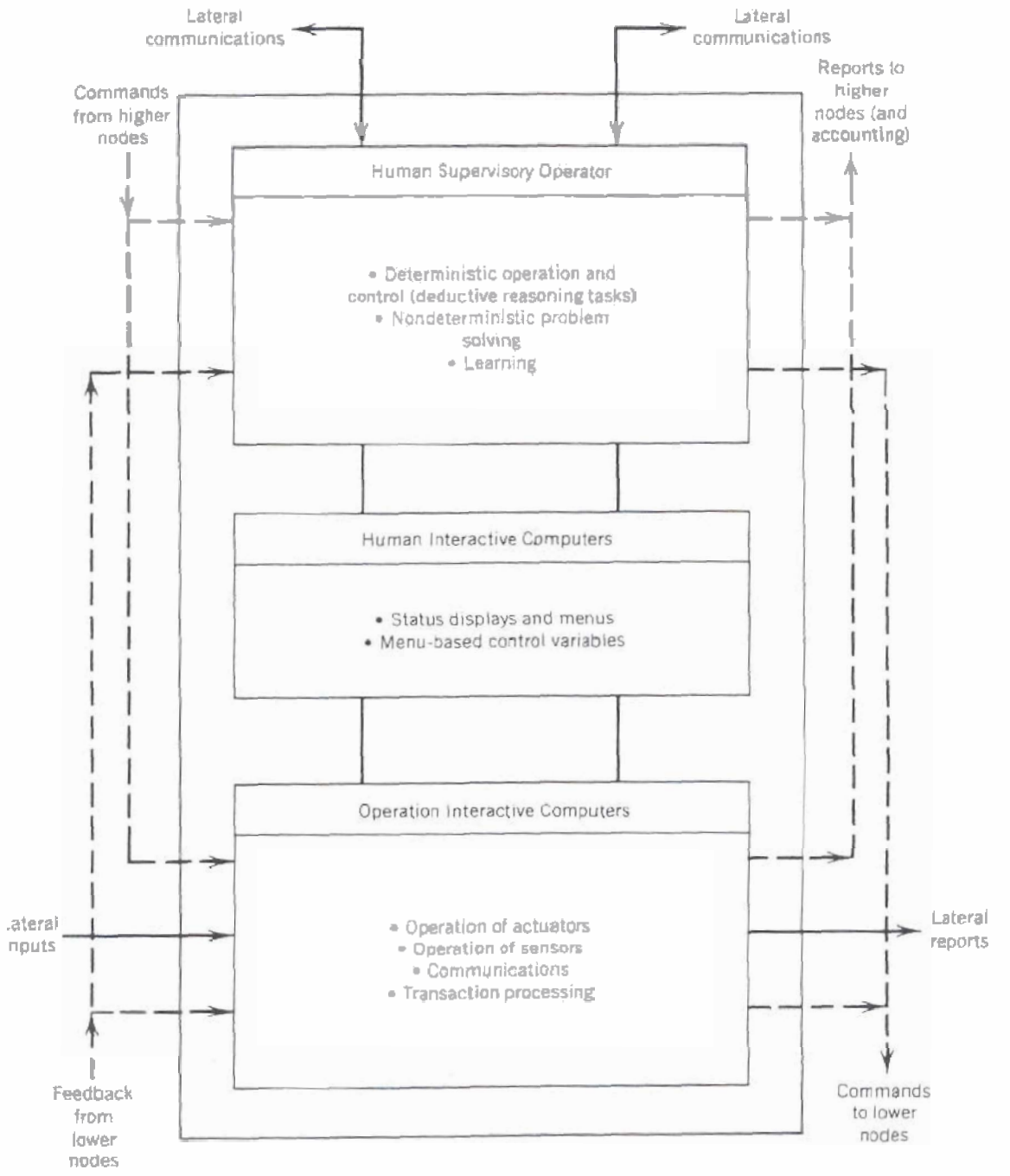


Figure 2. Model of local intelligence.

in time for a given service to be completed or suggests an alternative equipment pathway for the scheduler's consideration. In addition, each of these three SCPs individually must interact and cooperate with their counterparts at other control centers and facilities to solve problems and to perform their jobs.

1.3.3. Adaptive, Flexible Reasoning. The users of facilities desire and continuously make unusual requests for services: for example, these might include, among other requirements, (1) a large degree of flexibility in event scheduling, (2)

dynamically varying spacecraft instrument commanding and controlling, and (3) alternative telemetry formats, frequencies, and durations. In short, SCPs must exhibit a great deal of adaptive behavior within their "well-defined" responsibility areas. The Space Station, as one example, is intended to be able to handle currently unforeseen activities, targets of opportunity, synergistic experiment operations, and adaptive instrument behavior and to allow user interactive use of the overall system in a flexible, transparent manner. (This has at times been called *telescience*: telescience is well documented in the CODMAC Report [1] and in Silverman et al., [2].) The implications for SCP ESs are in terms of two radically divergent modes of reasoning: at times SCP's must be capable of elaborate planning and adaptive behavior while at other times they must be prepared simply to execute the plans with great speed.

2. DISTRIBUTED PROBLEM-SOLVING PROTOCOLS

The FACILITY ADVISOR is being designed to manage a generic version of a facility consisting of the three representative positions: Operator, Repairperson, and Scheduler. This generic facility is based on data collected over several years of knowledge engineering studies by the author at Goddard Space Flight Center facilities (i.e., NCC, POCCs, MORs, DCRs, NASCOM, WSGT, and FDF). In all these studies, the SCP has been described in terms of the three levels of intelligence portrayed earlier in Figure 1 and as further elaborated in Table 3 [3, 4].

As can be seen from Table 3, Supervisory Control Nodes tend to leave the most intellectually arduous cognitive functions up to the *human* supervisory operator: for example, (1) the anomaly troubleshooting of the Real-Time Controller, (2) the stochastic, plausible reasoning of the Problem Solver and Planner, or (3) the learning side of the Metaknowledge and Learning. Even so, with current day artificial intelligence (AI) and ES techniques it is becoming increasingly possible to replace many functions of the human at the third and highest level of local intelligence. For the sake of illustration, the human will be called a Specialist and the human tasks will not be explained as if his or her cognitive functions were separate, quasi-intelligent elements (as indeed they are) of his or her brain.

2.1. Cognitive Functioning of a Specialist

Once a task is sent to the External Blackboard, any supervisory Specialist that takes responsibility for that task must be structured with the nine generic elements of Table 3. The task enters the Specialist as a message decoded by the Communicator/Network Listener and is placed on the Internal Blackboard in queue with other tasks needing to be handled by that node. First, the Negotiator references the Metaknowledge and Learner to determine if the task can indeed be done internally and then Chair decides the sequence of tasks to be processed with the aid of the Fusion/Situation Recognizer. Anomaly-free tasks are generally handled by the Real-Time Controller, who takes the necesssary steps to complete the task, places the

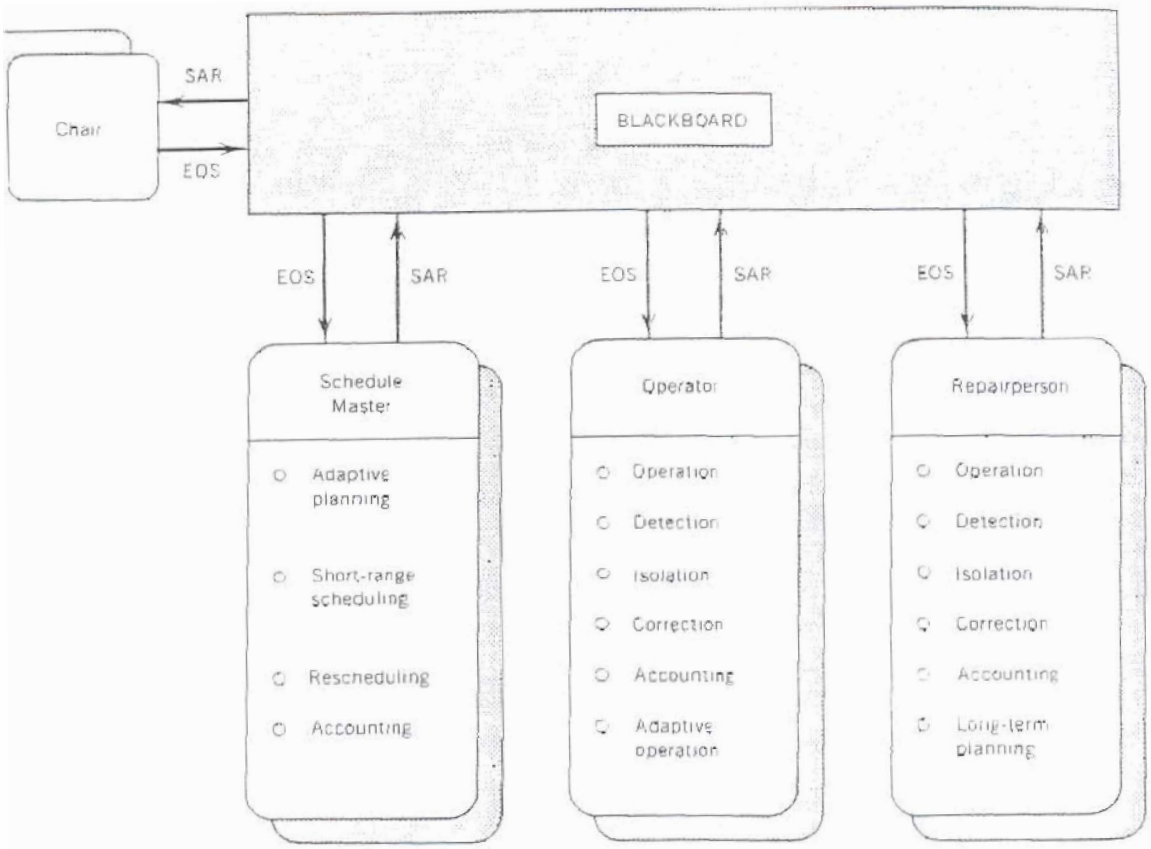
TABLE 3 Handling of Cognitive Functions at Each Local SCP By Three Levels of Intelligence

Levels of Intelligence	COGNITIVE FUNCTIONS							
	Communicating & Negotiating	Blackboard Writing/Reading (Internal)	Chair/Task Scheduling	Fusion/Situation Recognition	Problem Solving & Planning	Controlling	Accounting & Logging	Meta-Knowledge & Learning
Task Level Computer	• Network Listener • Front End Processor • Code up Transmissions/ Requests • Decode Responses	• General Display (Info Alerts Status, Action, Etc.) • Read Tasks	• Service Tasks In Priority Order Subject to On-going Constraint	• Automatic Sensing • Automatic Anomaly & Status Detection & Reporting	• Automatic Deduction (Check Meta-Knowledge & Pass Solutions to Controlling Problem to Human	• Semi Automatic Servo-Control	• Automatic Storage/Retrieval In Long Term Memory	• Execute Meta-Knowledge About Who to Call for Anomalies
Human Inter-Active Computer	• Display Organizer (Translate Machine to Menu Parameter) • Translate Human to Machine Language	• Hold/Display Info & Organize • Translate Human Input	• Fifo Run Scheduling Aids For Human	• Organizing Anomaly & Status Info Hierarchically	• Place Problem Alerts at Top of Information Hierarchy	Not Applicable	• Execute Retrieval Queries/ Storage Requests	Not Applicable
Human Supervisory Operator	• Negotiate • Voice -In Room -over Phone -Keyboard	• Read Display • Write Cases • Make Mental Notes	• Reassign Task Priorities at Will	• React to Anomaly Alerts and Multiple Inputs • Remain Cognizant of System Status	• Full Range of Plausible Deductive Reasoning	• Implement Solutions & Exercise Interrupt Ability Adjust. Takeover)	• Summarize, Account & Log	• Learn System Strengths & Weaknesses • Reprogram on Line as Needed

result along with tasks for other Specialists on the Internal Blackboard, and generates task results information useful to the Accounter/Logger. Anomalous tasks, on the other hand, if accepted, are preprocessed by the Problem Solver and Planner, who attempts to evaluate alternative solutions and to identify the proper sequence for the Real-Time Controller to then execute. Metaknowledge and Learning serves as the repository of information about what alternatives the Problem Solver and Planner should consider, including the possibility of referring the task to a completely separate Specialist for help, for result sharing, or for subtasks assignment. Negotiator is again used in subtask assignment interactions (i.e., subcontracting). Metaknowledge and Learning also "learns," as time progresses, which alternatives, which solutions, and which other Specialists are most useful and which are most counterproductive for each type of exchange.

2.2. The Multiple SCP Protocol for a Single Facility

The multiple SCP protocol is typified for one illustrative facility as shown in Figure 3, where there are three SCPs that specialize, respectively, in facility scheduling, facility operation, and facility repair. The Schedule Master is responsible for allocating the facility's equipment and other resources along a timeline that will



ıure 3. Multiple SCP blackboarding at an illustrative facility. EOS—execution order for a ecialist; SAR—specialist activation request.

assure response to user service requests. The Operator in turn receives the schedule (authorization) and uses his or her local intelligence to assure the service is performed in a satisfactory fashion. If any problems arise that the Operator cannot correct, the Repairperson is contacted for assistance with the equipment or other resources and/or the Operator requests new schedules and/or resource allocations of the Schedule Master as needed. The Repairperson is constantly monitoring equipment operations and will alert the Operator and Schedule Master of new allocation requirements as needed. Other interactions exist as well. The Repairperson can learn more about the source of an equipment problem by asking the Operator about the symptoms. The Schedule Master in turn needs the Repairperson's recommendations for alternative (backup) equipment pathways that can be scheduled so as to minimize interruptions to other previously planned activities.

The intrafacility interactions are facilitated only *in part* by the existing automation at each Specialist's workstation. For example, the Operator can generally call up displays that show equipment sensor feedback data and thereby attempt to infer whether the source of a problem is equipment related (rather than operational). If so, the Operator would then compose a repair message to be sent to the Repairperson. However, the information is not always on the computer and voice interaction is often necessary. While current levels of automation can at times assist the Specialist, at other times it can actually contribute to increased workload and deteriorated performance.

2.3. Interfacility Cooperation and Protocols

In addition to the interactions within a given facility, each Specialist interacts with his or her counterpart at one or more other facilities. For example, the Schedule Master must exchange messages with a hierarchy of other Scheduler nodes in order to perform many of the planning responsibilities. The Operator may need to communicate with Operators at other facilities, for example, to isolate sources of message errors or, as another example, to obtain authority to assist in correcting an incorrect ID. The Repairperson in turn helps in the process of isolating which facility is the problem origination point in the event of a hard to isolate equipment problem.

While much interfacility exchange is facilitated by currently planned workstation automation, the human-to-human exchange is common and will probably remain so for the foreseeable future. These exchanges tend to be voice-based, although some of them seem to be migrating to teletype. For these reasons, it is essential to explore how much of the interfacility problem solving can be assumed by the DES when there are humans at the other facilities. To what extent will these individuals be able to migrate voice communications to teletype exchanges? Even more importantly, will they be willing to have natural language dialogues with an expert system and how can the effectiveness of these exchanges be maximized? These and other human factor engineering issues comprise some of the research tasks needed.

2.4. Real-Time Execution Protocols

Real-time controlling is a continuous function, 24 hours a day, of monitoring and controlling activities, scanning a selection of critical parameters, seeking cautionary or out of limit conditions (or alarms), and low-level inferencing along cause–effect lines. For example, in terms of operator screen settings consider a *condition X* for which the logical and necessary *action* is *Y*. Here *X* might be an input that occurs in the presence of an initial or current system state while *Y* might be the outputs that lead to a final or next system state. In this fashion, chains of such deductive sequences can be envisioned where *Y*, for example, becomes a condition forcing yet another state transition action called *Z*. For the supervisory operator in a real-time setting, the deductive sequence just elaborated occupies the majority of the operator's time and there are a multitude of control variables to be managed in exactly this fashion, with rarely a need to call the Problem Solver and Planner.

Real-time controller functions require the human (or ES) to respond to each new "situation" and (1) to deduce that a set of rules or sequence of steps must now be attempted, (2) to recall those steps from memory (or from a look-up notebook), and (3) to execute those steps rapidly. A "situational" calculus can readily be conceptualized (although tedious to develop) in which human knowledge is organized into and replaced by one of five categories of (modus ponens) rules: transition, initialization, status, control, and accounting. Such a calculus has been demonstrated with the use of our Hierarchical Control System Environment (HCSE) [5].

2.5. Alternative Planning Protocols

Planning and adaptive reasoning require repeated construction and testing of alternative plans and subplans, the weighing of different pieces of evidence, and much "milling over" the pros and cons of choices that can be made. In short, there is no valid reason to try and constrain SCPs to a single uniform calculus and the proposed approach involves two distinctly separate reasoning approaches (calculi) for the real-time and off-line functions, respectively. Furthermore, there is no uniform off-line calculus for all SCPs. However, a relatively limited number of planning calculi can be formulated that occur repeatedly across all facilities. In particular, each Specialist engages in three interrelated types of planning:

1. Routine operations planning that includes the preparation of ordered timelines for equipment allocations and services performed.
2. Active period planning (or replanning) that includes problem solving needed to continue operations when changes and/or anomalies occur.
3. Long-term adaptive planning that is responsible to relatively major shifts in user patterns and/or user service requests.

The first two of these levels currently are partially automated in many existing SCPs and one can envision relatively off-the-shelf ES technology, if creatively

applied, offering substantial inroads into the human's domain (e.g., "mulling over" of pieces of evidence can be shown to include blackboarding, opportunistic reasoning by multiple modules, combinations of backward and forward chaining, and progressive deepening aided by heuristic search functions and dependency-directed backtracking techniques).

The adaptive planning is usually relegated to a system engineering team. However, there is no fundamental reason why this cannot be handled by the FACILITY ADVISOR via the use of intelligent assistant (IA) modules in each relevant Specialist, one for each desired adaptive planning capability. As an example, an IA might be created as an on-line software or data set assistant under the Operator SCP. This software (data set) assistant, in grossly simplified terms, could facilitate the facility's ability to adapt to new user message (or telemetry) frame formats and content meaning by (1) containing rules that query a user for the new formats and content meanings, (2) suggesting to a Human SCP how to alter the facility's software (or data set) to accommodate the new formats and content meanings *safely*, and (3) inserting those changes if given the authorization to do so.

The human would retain interactive as well as override capabilities in the event the IA's suggestions are naive or nonresponsive.

3. DES TECHNOLOGY FOR THE FACILITY ADVISOR

The FACILITY ADVISOR implies constructing a system with an architecture somewhat like that shown in Figure 4. The reader can see at this point that (1) the ES includes three representative Specialists plus a manager, (2) each Specialist includes nine generic elements that need be built only once and that handle all the cooperation and coordination needs, and (3) each Specialist includes a set of "modules" (unnumbered boxes) that are unique to its operation. There are thus certain generic elements to be added to the FACILITY ADVISOR and for any facility to which the FACILITY ADVISOR might ultimately be applied. Furthermore, the Specialist-unique module sets contain elements that can and should ideally be standardized so that any facility that wants a Specialist of that type (as part of its FACILITY ADVISOR or as a stand-alone entity) will receive a validated structure and certain validated knowledge base elements: the Specialist can thus be tailored to a given facility primarily by extending and tailoring its rule base.

One other factor critical to the FACILITY ADVISOR is that, because a number of stand-alone ESs are already under construction at existing facilities, the FACILITY ADVISOR must provide an open environment that maximizes the ease with which these third-party ESs can be incorporated into the FACILITY ADVISOR. It must provide application developers with utilities that support reuse of previously constructed components, integration of diverse components and modules, and interfaces to their favorite ES development shells (e.g., ART, KEE, or IntelliShell).

A DES and a testbed that supports cooperation yet more-or-less autonomous reasoning components are needed. A prototype of the FACILITY ADVISOR is

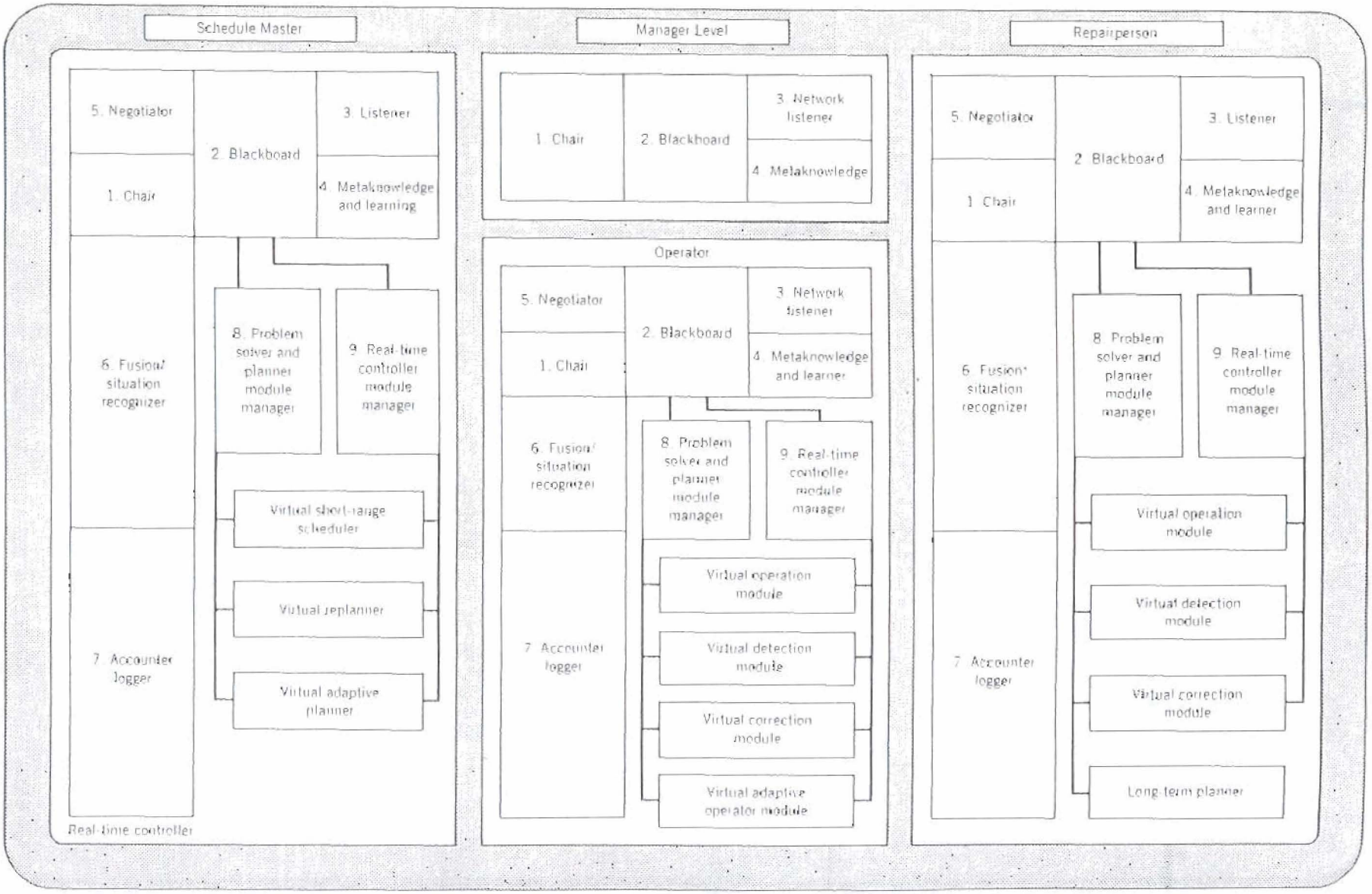


Figure 4. Nine generic elements and three module sets to be developed for the *FACILITY ADVISOR*.

already in place (see Section 4) and a number of the elements of the model have been described fully at a logical level in Section 2.0: for example, it must support a wide range of control schemes (slavelike to fully distributed), several levels of communication interfaces (intermodule, interprocess, intrafacility, interfacility), multiple reasoning schemes (situational calculus, belief-based fusion, and several planning calculi), and synchronization between numerous semiautonomous components not all of which use AI techniques. Since the FACILITY ADVISOR is ultimately targeted for transfer to a variety of facilities, the knowledge engineering effort must also provide a "virtual machine" that can be mapped into a wide range of underlying hardware/OS environments: primary targets are parallel and distributed environments.

In particular, the work to be completed and the technology to be used can be discussed in terms of the tailored modules, generic elements, and interentity coordination:

1. *Tailored Modules.* If the three representative specialists are considered equivalent to three stand-alone ESs, industrywide experience would suggest 10–20 work years of effort would be required to develop each of them fully. Clearly, that would be an enormous effort and in addition the DES aspects of the job would be ignored. To this end, the author and his staff have already invested close to 10 work years of effort on these modules and have already completed a substantial fraction of the knowledge elicitation effort required for the representative specialists' modules (see Section 4). Because of this preparation, the author believes they can generate and code rules for the Operator, Repairperson, and Schedule Master at the rate of about 200+ rules per month for several months.* In a short period of time, an extremely "interesting" FACILITY ADVISOR could be generated where "interesting" means it could handle a relatively large number of higher-level cognitive functions and protocols that are important to most SCPs. This would be a practical limit to place on effort expended in this direction as concerns real-time control and repetitive planning modules. At least an equal effort is contemplated, however, for collection, analysis, and coding of more complex protocols such as those encountered in intra- and interfacility problem-solving sessions, as well as for adaptive planning situations.
2. *Generic Elements.* The discussion thus far also suggests a substantial investment of effort for defining the generic elements and precisely how they would work. In addition, effort is needed for coding and validation testing

*For those modules in which the knowledge collection obstacles have already been overcome, the coding of rules should be roughly equivalent to normal programming rates. Two hundred rules per month is about 10 rules per work day. This is a fairly conservative estimate as most programming projects can generally generate code at the rate of 10–40 lines per work day.

of these generic elements so that they can be offered as customizable elements. The authors have had a chance to design, prototype, and test most of these elements by virtue of several DES and ES jobs they are doing. In particular, the off-line blackboard and chair have been developed for other work [6] while the real-time blackboard, chair, and module manager exist in a software environment called HCSE that already is part of the FACILITY ADVISOR prototype (see Section 4). Schedule Master planner and replanner techniques will be recycled from our prototype and from our Space Station Customer Scheduling Expert System currently being developed for McDonnell–Douglas Astronautics Company. All remaining generic elements exist at the detailed design level as summarized in earlier reports [3, 4].

3. *Interentity Coordination.* Finally, the fifth level of knowledge engineering is intended to assure communications between diverse modules and elements, cooperation between operating systems of different machines on which the parts of the FACILITY ADVISOR will reside, and transportability for applications purposes. To leverage the effort, it would be fruitful to use techniques being developed elsewhere, to the extent possible, such as in the COP project at George Washington University. COP is a software effort to develop a cooperation between operating systems package for a separate DES project that the author is involved in at George Washington University, which will develop capabilities to be phased in over the next several years. COP presently offers very modest capabilities for affecting communications between parallel physical processes and resource management abilities. In summary, COP is a set of functions that an application (such as a FACILITY ADVISOR, specialists, blackboards, processes, or modules) can call on to establish logical networks to other applications and to select ways to send mail (e.g., point to point or broadcast).

   COP also currently includes certain network listener and metaknowledge functions. That is, each local COP Controller monitors the "internal network" of the computer it's installed on for messages of interest to applications on its own board. It also monitors the interboard network(s) for messages of interest to applications on its own board. Upon finding such messages, it places them on the internal network for routing. In these regards, COP acts as an interboard daemon capability.

   COP is thus a far less ambitious project than many other distributed operating system efforts. However, it is felt that not all DES applications need such power. In numerous DES applications (like the FACILITY ADVISOR), once design begins it is generally clear to programmers that certain subsystems deserve "privately held boards" with some capability for picking up added power during peak workload periods. Any application on a private board should be able to carry out operations without intervention by COP. This is thus taken as a currently planned extent of COP's capability: a goal that would seem to be sufficient as an entire goal for a DES such as the FACILITY ADVISOR.

4. CURRENT PROTOTYPE OF THE FACILITY ADVISOR

If the FACILITY ADVISOR is ultimately to be used, it must be capable of handling realistic workload levels with a minimum of computer investment. The fundamental problem common to both the interfacility situation and the across-user service request situation addressed here is that of managing processor resources to maintain the pace of the workload despite delays encountered while responding to a given user service request. For several reasons, it is necessary to be able to create and manage dynamically the SCP cognitive processes, modules, or module elements (subprocesses or IAs). Each of these entities must be a "virtual entity," that is, an entity that can create a clone capable of doing a task in the same fashion as the busy entity could have. A virtual entity occupies no memory when it is not needed.

It is inefficient to keep all these processes and their clones active in main memory simultaneously; however, for some problems there may not be a unique decomposition apparent to the higher levels of the hierarchy. The point of making all entities in the DES intelligent (they have metaknowledge) is to facilitate the ability of some entity to ask another whether it can offer a meaningful solution (or part of a solution) to the problem. That is, most entities must (1) keep some "listener" portion of themselves active and (2) in addition, an entity manager must exist that knows which additional entities to "awaken" for a response. This is a fully adaptive control formation and one that is supportable within the COP approach.

Progress already made toward the first version of the FACILITY ADVISOR is summarized in Figure 5 and described below:

The Virtual Machine Hardware. The virtual machine includes seven parallel boards plus a host. These boards are four Xerox LISP machines, one VAX 11/780, and an IBM PC. One of the LISP machines also contains a PC emulation board. The boards are physically connected via an EtherNet except the VAX, which is connected separately to the host.

The Virtual Machine Operating System (COP). The virtual machine straddles three distinct types of operating system (MESA, DOS, VMS) with the aid of the COP capability. A COP station exists on the host board and a local COP has been assigned a "beat" on each of the parallel boards.

FACILITY ADVISOR: Off-line Manager. The Off-line Manager position has been prototyped in LISP on a separate machine using an in-house blackboard technology (developed for the ARIEL and EPMS Shells). In brief, the Manager places a planning problem or goal on the blackboard, collects and evaluates Specialist Activation Requests (SARs) from the various position specialists who have offered to solve part of the problem, and issues Execution Orders for the Specialists (EOSs) that it feels can make the best contributions at the present time. These specialists, in the prototype, are on other boards of the virtual machine.

Schedule Master. The Off-line Master is prototyped on LISP machine #2 primarily in LISP. Objects are created for each piece of hardware, each user service request, service pathways, and for each of several types of constraint (e.g., priority, window, and service type). The LISP functions create a set of system states,

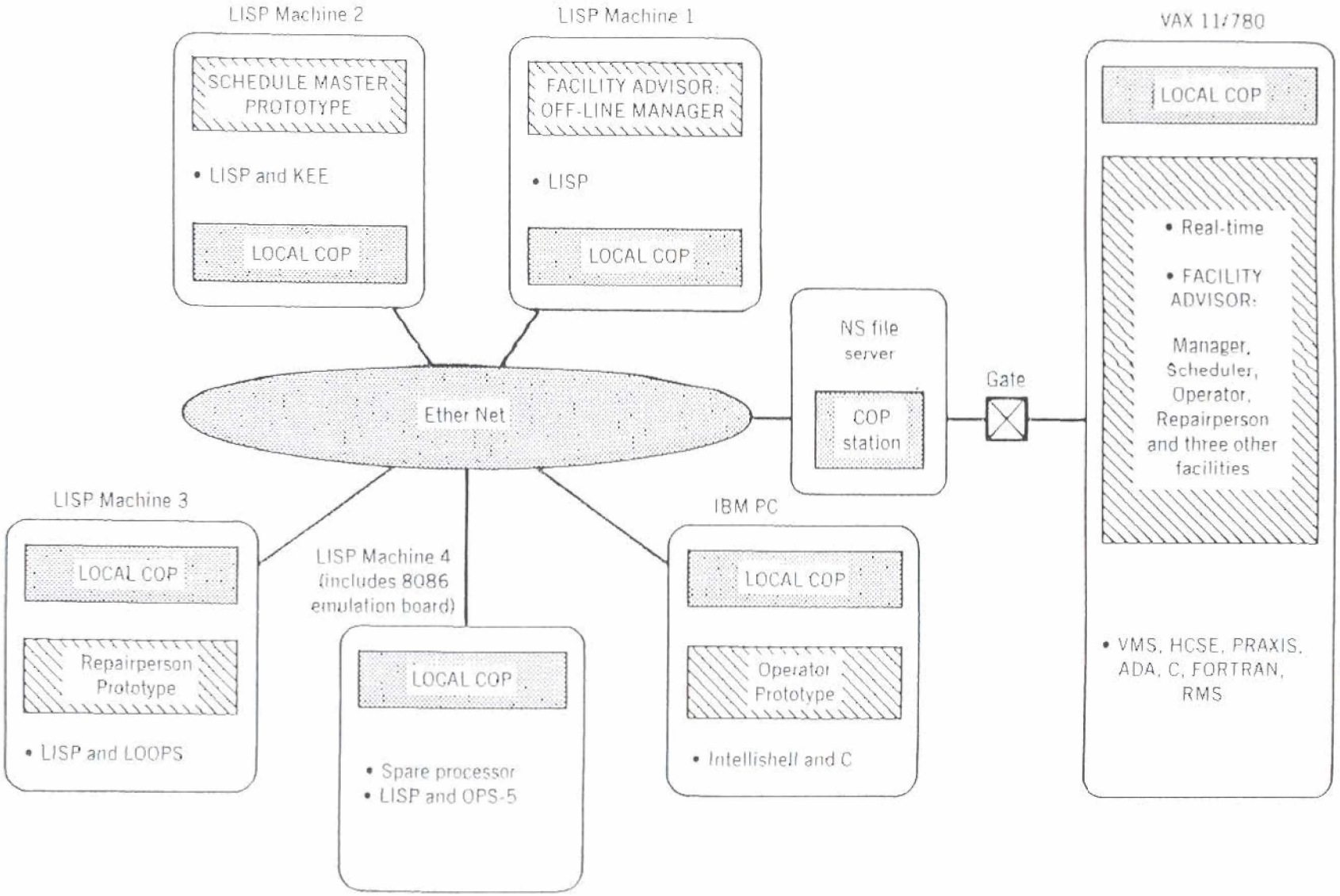


Figure 5. The FACILITY ADVISOR prototype. (*Indicates local COP not fully implemented.)

heuristically search the states, relax various constraints, and recommend feasible schedule alternatives to which the Repairperson, Operator, and User are asked to respond. The final selection is sent to the Real-Time FACILITY ADVISOR for execution (at present the Real-Time FACILITY ADVISOR only executes one schedule alternative).

Repairperson. The investigators have collected over 1000 rules used in the NASA Goddard Space Flight Center (GSFC) Repairperson positions; however, only a very simple prototype has yet been implemented. This Repairperson uses seven LOOPS rules and 21 LISP functions to operate and monitor one piece of equipment (currently an object). When it detects a failure, it uses a second set of rules to isolate and correct (or abort) the problem and it reports the result to the Off-line Manager.

Operator. Here too, the investigators have collected over 1000 rules used by GSFC Operators yet only about 15 or 20 rules have been implemented. The Operator is not presently integrated into the FACILITY ADVISOR; however, it demonstrates the detection, isolation, and correction of user request message errors as well as one of the explanation and accounting features. The Operator was originally written in M.1 but is currently being ported to IntelliShell, an in-house ES written in C and ultimately targeted to contain both forward and backward reasoning (it only offers backward at present).

The Real-Time FACILITY ADVISOR. A situational calculus capable of supporting the DES real-time elements has been tested on the VAX with the aid of a DES generator called Hierarchical Control System Environment (HCSE). HCSE provides a language in which to create fast, deterministic production-oriented specialists that communicate with each other via a blackboard mechanism. Specialists can be encoded with sets of condition–action rules and/or with more traditional procedures (in FORTRAN, C, PRAXIS, or ADA). The HCSE also permits hierarchies of specialists to be established and executed so that lower-level experts execute commands from and send results and feedback to higher-level experts. Specialists communicate through and are synchronized via the blackboard and chair. HCSE is thus a viable alternative for testbedding the FACILITY ADVISOR real-time elements. It may not be the ultimately chosen tool; however, skeletal elements can be prototyped rapidly within HCSE. To that end, seven ESs were built in HCSE corresponding to (1) a Facility Manager, (2) a Schedule Master (active period), (3) an Operator, (4) a Repairperson, and (5) three external facilities to emulate interfacility protocols. A busy entity cloning capability was also tested. The entire real-time DES is fully documented in an earlier report [5].

In summary, Figure 5 and its explanatory notes explain (1) the feasibility of off-line planning elements being constructed in different shells, languages, and machines, (2) the role of the HCSE for testbedding of real-time elements and modules, and (3) the cloning and virtual entity framework explained above. Albeit, each item in Figure 5 is in its relative infancy and most of the dedicated machine (private board) allocations have been made only for proof-of-concept purposes. In fact, only two boards are truly needed at this point: (1) the entire off-line FACIL-

ITY ADVISOR could easily have been implemented on a single LISP machine, and (2) the real-time FACILITY ADVISOR deserves a dedicated board. Nevertheless, the early prototype does demonstrate the fundamental precept of this chapter: the FACILITY ADVISOR concept works.

As a final note, the prototype of the FACILITY ADVISOR testbed has been devised in a fairly flexible, portable manner owing to a variety of techniques:

1. Common LISP is being used to the extent possible on individual machines.
2. Relatively portable shells are also being used (e.g., KEE, HCSE, COMMON LOOPS, and IntelliShell).
3. The C language is being integrated where feasible.
4. COP is open to many new hardware configurations (including a MIMD machine).
5. The distributed components and modules are being designed with parallel implementation in mind, thereby increasing the possibility of incorporating a massively parallel processor of some form.

5. NEXT STEPS

This chapter has described a DES approach called the FACILITY ADVISOR for supporting distributed problem-solving protocols commonly encountered in real-time control and operations. The FACILITY ADVISOR is suggested as an attempt to prevent individual facilities from reinventing the wheel and to create a way to share new, useful ES technology, techniques, lessons learned, and insights. If all FACILITY ADVISOR goals can be met, the results will be a set of validated skeletal elements that any facility can readily tailor to its domain and that can be extended "simply" by adding appropriate rules to individual specialist modules. To reach this goal, a testbed is needed to evaluate the viability of what is described herein. Specifically, this testbed for the FACILITY ADVISOR is depicted in Figure 6 in terms of an illustrative facility consisting of N supervisory controller positions existing in conjunction with other facilities. Figure 6 also shows how the various SCP expert systems (i.e., Specialists) will ultimately be integrated into the facility as advisors to the humans who currently perform those tasks and cognitive functions. In this advisory role the Specialist (1) watches what the task-level computer sends to each human, (2) formulates recommendations, (3) advises the humans if queried, and (4) alerts the humans if the Specialist "thinks" a given human's response is in error or if the human has failed to respond. Enormous amounts of validation testing are needed to reach this level of implementation to ensure that the various Specialists will help more than hurt the humans. Even more validation testing would be needed (e.g., several years of actual facility operation) to determine if the FACILITY ADVISOR's recommendations and performance are continually good enough to warrant collapsing the individual SCP humans into a single human position as shown at the very base of Figure 6.

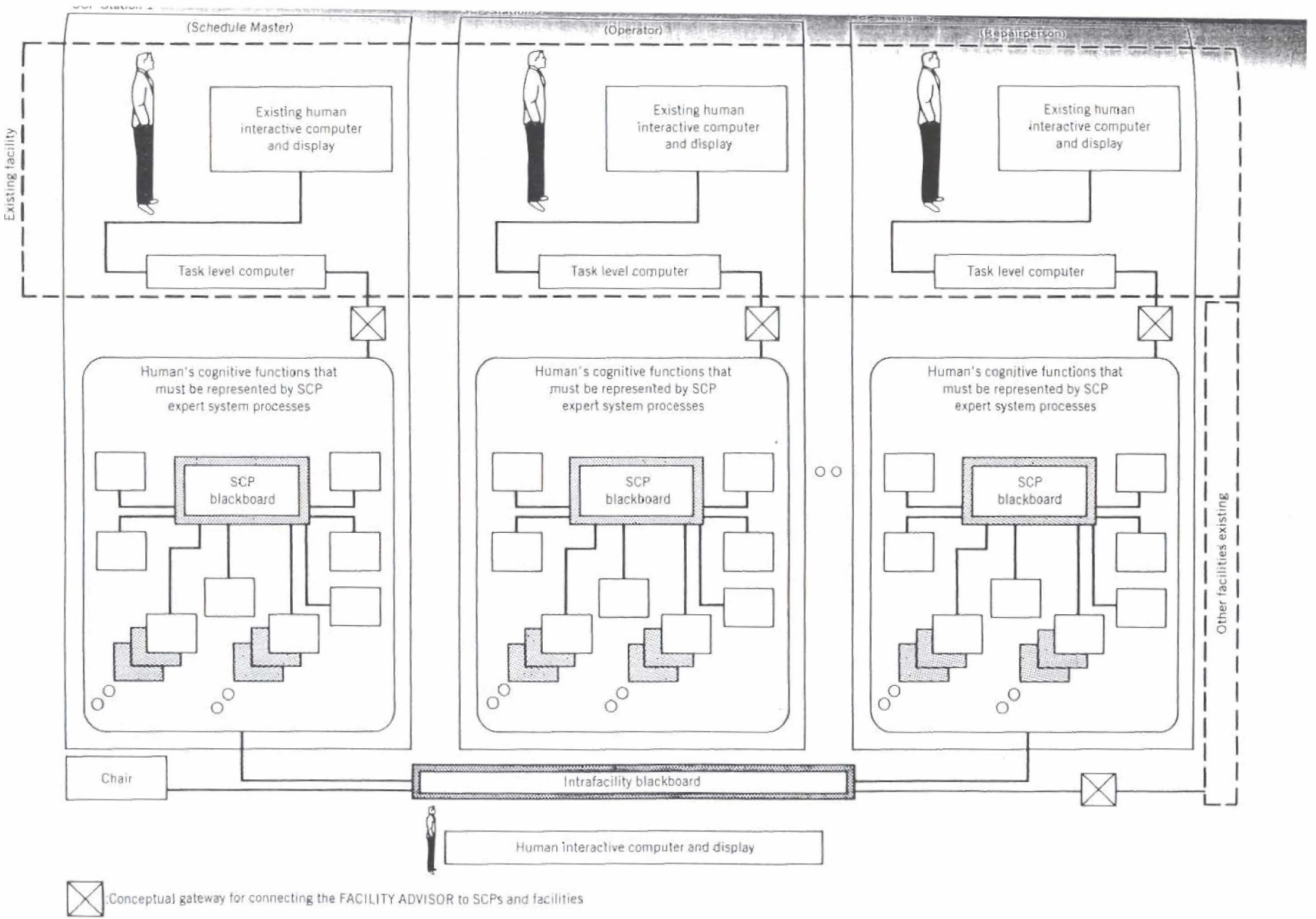


Figure 6. Overview of the FACILITY ADVISOR logical model and how it fits into existing SCPs and

Of more immediate concern are four testing aspects of the FACILITY ADVISOR shown in Figure 6:

1. The logical model of a single generic SCP ES is shown to include nine cognitive functions (boxes) each of which requires separate and possibly "parallel" ES process. All but two of the processes (real-time execution and off-line planning) as explained already are to be identical across all SCPs and probably will only require one version to be built and tested. The two remaining processes, however, capture the dual nature of each SCP: real-time execution versus off-line problem solving and planning. A dual calculus approach was discussed earlier: (a) deterministic situational calculus for real-time execution and (b) nondeterministic, belief-maintenance-oriented reasoning for the data fusion, problem-solving, and adaptive planning aspects of each position. How each Specialist's modules may best use these two types of reasoning must be tested extensively.
2. In Figure 6, FACILITY ADVISOR is shown to encompass three representative SCPs cooperating within a single facility. The intrafacility blackboard model and chair are felt to be identical from facility to facility; however, the specific modules of each of the three specialized SCPs need to be built and tested for SCP and facility-unique aspects. The same applies for modules of new SCP ESs.
3. Up to this point discussion of testing has concentrated on how the real-time execution and repetitive planning needs of facilities can be satisfied with ES technology. Earlier discussion has also indicated a need for a truly adaptive planning capability: that is the ability to handle previously unplanned activities such as (but not limited to) new targets of opportunity, adaptive instrument behavior, and mission redefinitions. While adaptive behavior is a design goal of the DES, the extent to which it can be achieved is a relatively open-ended research question; testing must also encompass validation of unplanned adaptive behavior.
4. Finally, Figure 6 intimates that testing must account for cases of interfacility cooperation as well as management of multiple user requests simultaneously. This is achieved via the dynamic creation and management of multiple copies of each SCP process and module on an as-needed basis. ESs are assigned to specific problems which they follow to their completion. For example, the Operator is assigned to a single user request. If the Operator fails to process the request completely before another user request arrives, a "copy" of the Operator is created to handle the next request. All ES operations and working memory elements are recorded on disk as they occur. This facilitates both the accounting function and the ability to manage processor resources by putting copies to sleep (out of virtual memory) and waking them up as needed—for example, while awaiting other SCP or other facility inputs. All such features must be considered in the validation testing plan.

In summary, DES technology holds much promise as this chapter has attempted to elaborate. On the other hand, facility operations need to be completed reliably and a no-risk approach is mandated. Since DES technology is relatively new, several years of exploratory development and testbedding are warranted to prove its capabilities. The next steps toward the integration of DESs into facility operations are to improve what we know and to determine what can in fact be delivered reliably.

ACKNOWLEDGMENT

The support of NAS5-28604, NAS5-30037, and NASA/GSFC/Code 520 are gratefully acknowledged.

REFERENCES

1. CODMAC Report, *Data Management and Computation*, National Academy Press, Washington, DC, 1982.
2. B. G. Silverman, V. S. Moustakis, and R. L. Robless, Ground Systems Autonomy Study for the Space Station Era, Technical Report of NAS5-28604, October 1984. This report is partially summarized in Expert Systems and Robotics for the Space Station Era, *Expert Systems in Government Conference Proceedings*, IEEE Computer Society, Washington, DC, 1985.
3. B. G. Silverman, *Distributed Expert Systems*, Technical Report prepared for NAS5-28604, NASA/GSFC/Code 520, August 1985.
4. B. G. Silverman, Fusion and Plausible Reasoning for Real Time Supervisory Controller Positions, presented at the *AI in Engineering Conference*, October 1985; reprinted in *AI in Manufacturing: Special Issue of the IEEE Transactions on Systems, Man and Cybernetics*, March/April 1987, IEEE, New York.
5. B. G. Silverman and Q. Ali, Blackboard and Distributed Expert Systems for Real Time Control Applications, prepared for NAS5-28604, GSFC/Code 520, 1986.
6. B. G. Silverman and C. Diakite, The Expert Project Management System. *In Proceedings of 1986 Conference on Artificial Intelligence Applications*, NASA/GSFC, Greenbelt, MD, Code 514, May 15, 1986.