THE VIRTUAL MISSION OPERATIONS CENTER

N94- 33615

Mike Moore
NASA Goddard Space Flight Center, Code 522.1
Greenbelt, MD 20771
moore@kong.gsfc.nasa.gov

Jeffrey Fox
Pacific Northwest Laboratory*
370 L'Enfant Promenade, S. W., Suite 900
Washington, D. C. 20024-2115
jfox@kong.gsfc.nasa.gov

Abstract

Spacecraft management is becoming more human intensive as spacecraft become more complex and as operators are asked to perform additional functions and interact more with external groups. Operations costs are growing accordingly. Several automation approaches have been proposed to lower these costs. However, most of these approaches are not flexible enough in the operations processes and levels of automation that they support. This paper presents a concept called the *Virtual Mission Operations Center (VMOC)* that provides highly flexible support for dynamic spacecraft management processes and automation. In a VMOC, operations personnel can be shared among missions, the operations team can change personnel and their locations, and automation can be added and removed as appropriate. The VMOC employs a form of *on-demand supervisory control* called *management by exception* to free operators from having to actively monitor their system. The VMOC extends management by exception, however, so that distributed, dynamic teams can work together. The VMOC uses *work-group computing* concepts and *groupware* tools to provide a team infrastructure, and it employs *user agents* to allow operators to define and control system automation.

1 INTRODUCTION

Several trends suggest the need for a new approach to spacecraft management [1]. First, operations complexity is on the rise because the number of telemetry points and spacecraft commands is increasing rapidly and because missions are requiring more real-time, dynamic science processes. It is becoming difficult for operators to actively monitor all telemetry points and issue appropriate low-level commands while trying to respond to dynamic user needs. Second, opportunities for direct spacecraft interaction are decreasing as the ground-spacecraft relationship moves from a master-slave to a peer-to-peer relationship, as more ground system functions are automated, as passes become shorter due to use of ground networks, and as the number of mission events decrease. This decreased interaction combined with increasing complexity suggests that operators will require more information from less interaction. Third, operator work loads are increasing as operations moves from the control center concept to the Mission Operations Center (MOC) concept; previously separated functions are being added to the operators' tasks. Operators therefore must do more under tighter time constraints. And finally, operators are having to interact more with external groups as missions become increasingly linked to each other, as scientists are given more control access, and as the number and distribution of science users increases. Operators will have to spend more time coordinating actions with other groups. All of these trends have tended to drive up the operator skill levels and the number of operations personnel needed. Unfortunately, smaller missions have less money to spend on operations support.

* Pacific Northwest Laboratory is operated for the US Department of Energy by Battelle Memorial Institute under contract DE-AC06-76RLD 1830.

Any solutions that address these trends must do so without violating several spacecraft management tenets. First, the solution must allow people to take charge of the system. The solution therefore must be semi-autonomous, and it must allow human operators to lower, and possibly remove, any level of autonomy as desired. Spacecraft behavior simply is not sufficiently predictable to allow complete autonomy. Second, the solution must allow operators to move freely from distant, high-level spacecraft interaction to immediate and detailed interaction. Special events, such as launch and maneuvers, require immediate, detailed monitoring and control. The solution also should allow the number of operations personnel to increase and decrease as needed. Special events and problems require engineers and additional operators.

This paper proposes a unified approach to addressing the above trends that does not violate the tenets of spacecraft management. The approach, called the Virtual Mission Operations Center (VMOC), is based on three technologies that are currently being applied in commercial applications: management by exception, work-group computing, and user agents. Figure 1 illustrates how these technologies form an integrated approach to spacecraft management. The *management by exception* model [2] extends the concept of supervisory control [3] to be more asynchronous or on-demand. In the model, the system detects deviations from normal system behavior (i.e., exceptions) and contacts the operator to address the exception before a problem develops. Management by exception frees the operators from having to actively monitor the system and allows them to act more proactively. In this paper, the model has been extended to support teams of operations personnel working together. A team approach is needed because spacecraft complexity requires a specialist for each subsystem. The extended model is called *team-based management by exception*.

Work-group computing [4] is an automated environment in which distributed, dynamic teams perform dynamically defined processes. Work-group computing uses groupware tools and technology to provide the infrastructure needed to support team-based management by exception.

User agents are semi-autonomous programs that act as personal assistants to provide indirect interaction between the user and the system. Such agents provide a user-defined, user-controlled way to gradually introduce automation. The technology also provides a model of how agent authority can develop over time based on proven competence.

The remainder of this paper explores how each of these three technologies can provide a unified approach to the spacecraft management trends identified above. The next three sections discuss each of the technologies in more detail. Section 5 illustrates how the combination of technologies addresses the trends through a series of operations scenarios. The scenarios also show how the approach adheres to the previously identified tenets.

2 MANAGEMENT BY EXCEPTION

In traditional system management, action is taken in response to an existing problem, i.e., when an error has already occurred. The system is constantly monitored and, when an error is detected, problem resolution is typically crisis driven. In the basic *management by exception* model [2], action is taken when there is a deviation from normal behavior, i.e., an exception occurs. Once an exception is detected, there is normally time to address it before a problem develops. Monitoring therefore can be less constant, and response can be less of a crisis. This model currently is used in several enterprise (i.e., distributed computing resource) management systems, for example Hewlett-Packard's PerfView and Operations Center.

What is an Exception? There are several ways to define the conditions under which an exception should be flagged. Most *exception conditions* are of the form: "X must [must not] be the case Y% of the time." For example, the exception "Response times must be less than 10 seconds

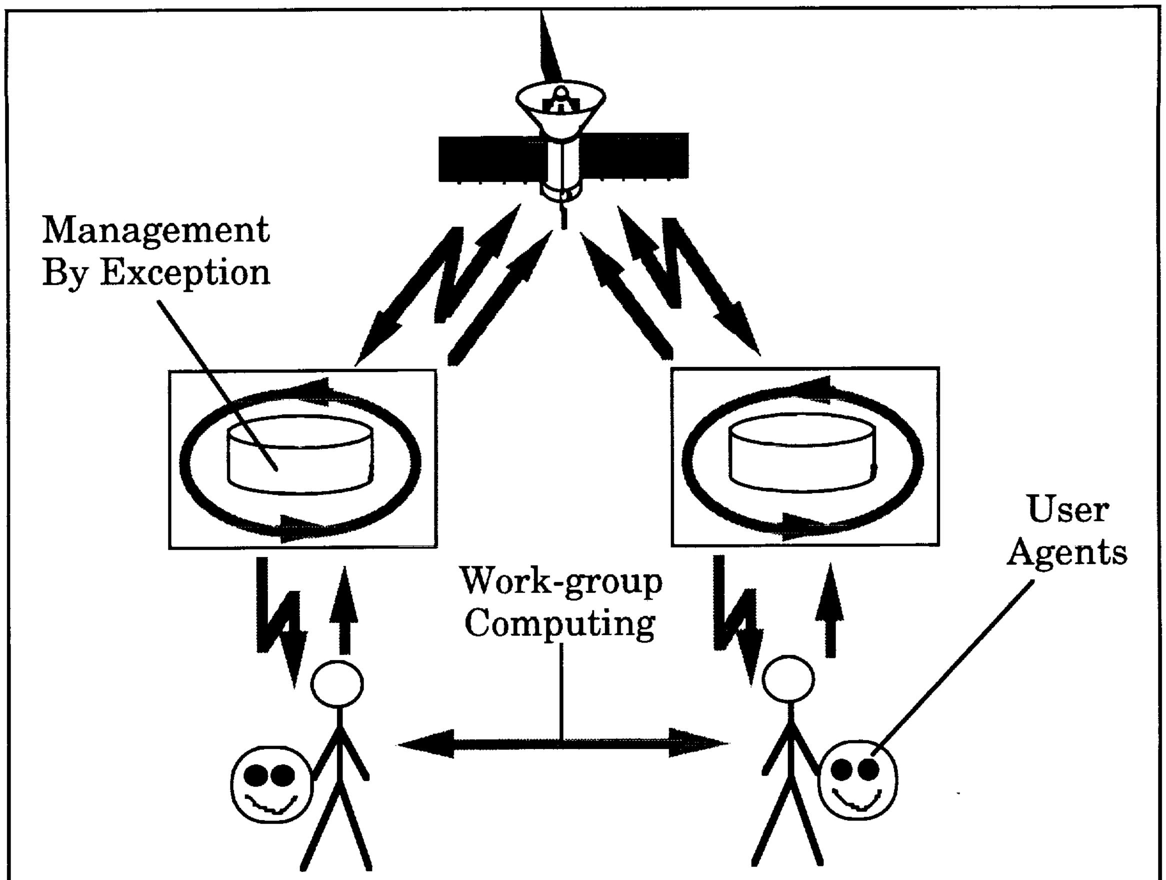

Figure 1: Elements of a Virtual Mission Operations Center

90% of the time" might be used to describe a computing system's normal behavior. The condition (i.e., X) usually contains a system value threshold. Exception conditions also can be defined using trend analysis and knowledge-based techniques. Using these techniques enables more proactive system management but is more difficult to implement.

An exception's definition includes an exception condition, the time interval over which the exception should be checked, and the exception's priority and severity levels. Exception conditions should be tested against a sufficiently long time period to avoid responding to transient behavior. Priorities and severity levels may be assigned to exceptions to assure that bad problems are resolved quickly. Severity may have to be determined dynamically based on the extent of the deviation and the services affected.

The Basic Process. Figure 2a illustrates the basic management by exception model. A centrally located operator defines a set of exceptions for the various system elements and downloads these to the elements. The elements periodically check for exception violations. Data relevant to the exceptions are collected so that they can be included in a possible exception report. While the system is performing its checks, the operator is free to perform other tasks. The operator therefore does not have to actively monitor the system. When an exception occurs, the exception and supporting data are sent to the operator. The operator is notified that an exception has been detected, either directly through the monitoring interface or indirectly through email, FAX, pager, or phone. The operator then may establish a close monitoring or real-time connection to the element.

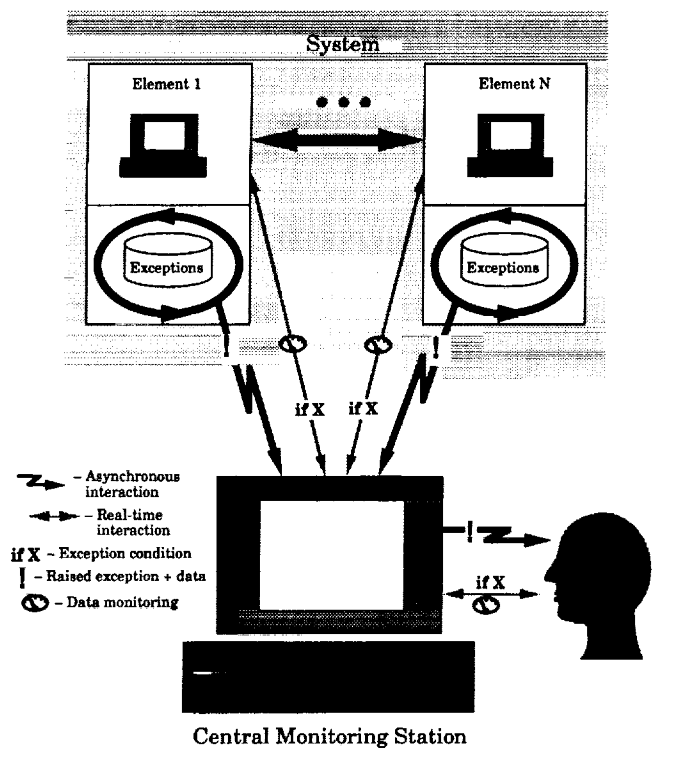
System
Element 1
Element N
Exceptions
Exceptions
if X
if X
– Asynchronous interaction
– Real-time interaction
if X – Exception condition
! – Raised exception + data
– Data monitoring
if X
Central Monitoring Station

Figure 2a: The Basic Management-By-Exception Model

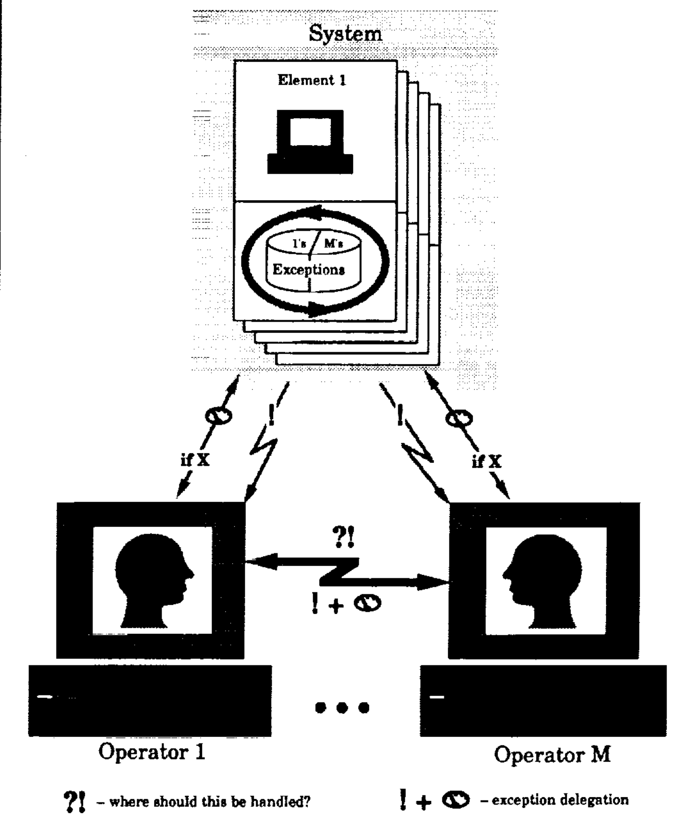

Figure 2b: Distributed Management By Exception

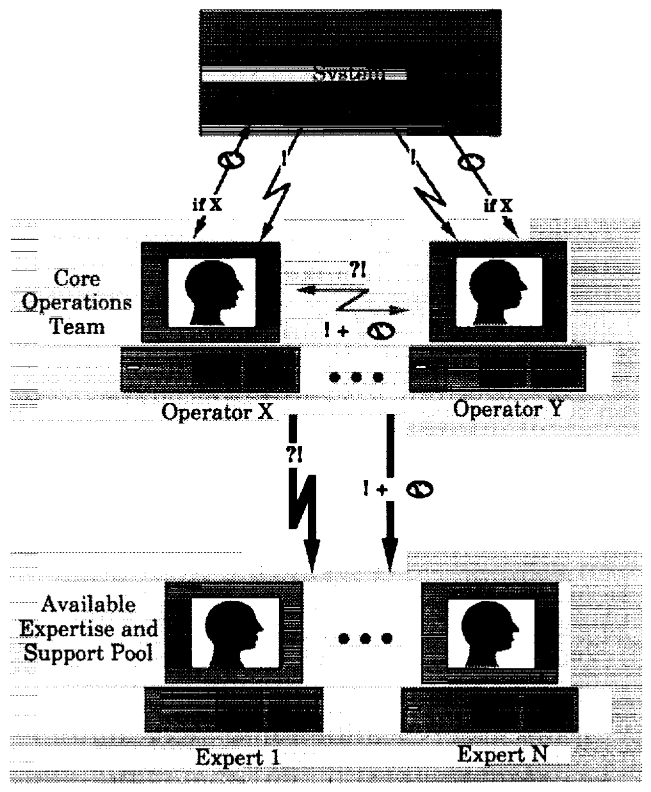

Figure 2c: Management By Exception With Dynamic Groups

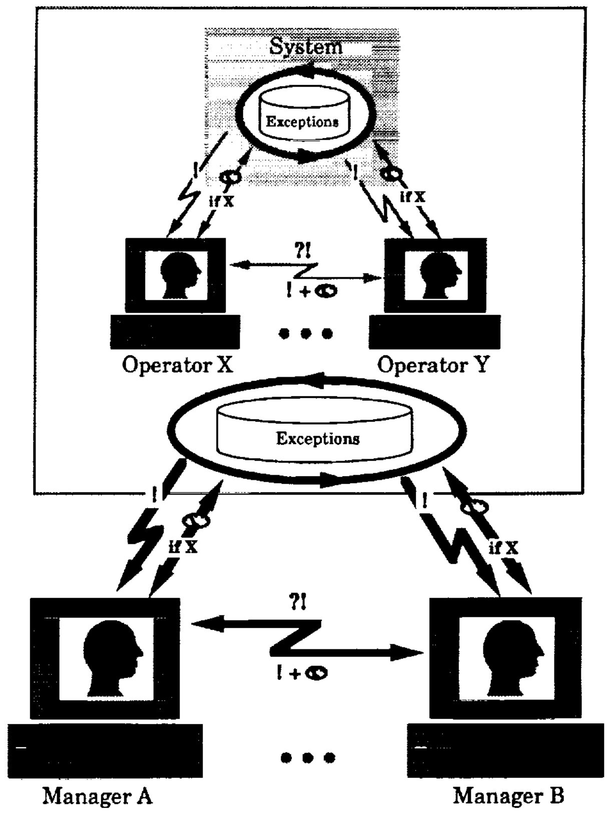

Figure 2d: Hierarchical Management By Exception

Figure 2: Toward Team-based Management By Exception

Team-based Management by Exception. When the system being monitored is sufficiently complex, it may not be possible for a single operator to monitor and resolve all exceptions. Instead, a group of operators may be needed. The basic model can be extended in several ways to support group-based management by exception. Each of these extensions would require specific tools to support the group interactions.

Distributed Management by Exception. First, the model can be extended to *support several monitoring sites and operators* instead of a single central location. This first extension is illustrated in Figure 2b. Operators allocate existing exceptions among themselves. Operators also can define their own exceptions. Data relevant to each exception is collected as in the basic model. When an exception occurs, the exception is sent to the appropriate operator as described in the basic model. However, the receiving operators can decide among themselves who should resolve the exception. Or, they can decide to delegate the exception to another operator. To make this decision, the operators may need to establish close monitoring relationships with the relevant element. These relationships and the exception would be sent to the delegated operator automatically.

Management by Exception with Dynamic Groups. A major assumption of the first extension is that the group of operators assigned to a system is predefined and static. In deciding where to delegate the exception, for example, the available candidates are all assumed to be immediately available. In the second extension to the model, this assumption is removed; the *group of operators managing a system can change dynamically* based on system needs. The second extension is illustrated in Figure 2c. When an exception occurs, the operators may resolve it directly, delegate to another available operator, or they may decide to seek help. The operators then must issue a request for assistance. This request is sent to candidate specialists via a combination of mechanisms. The request includes the exception, data that was collected in determining the exception, and any close monitoring relationships that have been established. If a candidate accepts responsibility for the exception, they notify the appropriate operator. The operator then delegates the exception to the specialist as in the previous extension. The operator must periodically check, however, to make sure that the exception is being resolved.

Hierarchical Management by Exception. In any mission critical environment, it is not enough to simply monitor system elements. There needs to be a way of monitoring the monitoring process to assess its effectiveness. The third extension to the basic model supports the *definition, monitoring, and resolution of exceptions related to the system monitoring process.* For example, a manager might want to be alerted if "The number of unresolved system exceptions is greater than 10 for 20% of the time in any shift." The third extension is illustrated in Figure 2d. It applies the tools of the basic model and extensions to the monitoring process itself.

Benefits of the Extensions. A system that combines the basic model and the three extensions would allow a system to be supervised at several levels of abstraction. It would eliminate the need for active system monitoring during nominal system operations. The operators therefore would be free to perform other functions (e.g., trend analyses and planning for future support) between exception events. The approach would not preclude, however, close monitoring and interaction when needed. The system also would allow managers and senior staff to monitor the monitoring process and system.

A system based on the combined model would be highly flexible. The list of exceptions could be modified easily to reflect changing system behavior and understanding. The number of persons involved in managing the system could be changed quickly and easily.

3 WORK-GROUP COMPUTING

To achieve the benefits of team-based management by exception while ensuring desired system performance, teams employing the model will require tools to help them work together. In this paper, we use the term *work-group computing* [4] for the computational resources needed to support these teams. We use this term because it currently is being used in management science to describe tools that support distributed, dynamic teams performing dynamically-defined processes.

Before we can define tools to support the model, we must understand the nature of interactions implied by the model. Schmidt [5] identifies three *interaction types* that people in teams engage in: augmentation, integration, and debate. We have added two interaction types to describe interactions in dynamic and non-peer relationships: negotiation and direction. *Augmentation* is used when one person cannot do a job alone because of its size. Job size is the main reason for moving to a distributed management-by-exception model. *Integration* is used when specialists are added to the team to support complex problem resolution, which is the motivation for the dynamic group model of management by exception. *Debate* is used for brainstorming and refinement. Operators and specialists will debate among themselves to resolve exceptions in both the distributed and dynamic group models. *Negotiation* is used when one person must locate another person to perform a task, and the parties must reach agreement on the nature of the task, assignment of the task, and task execution constraints. This category describes the process of building a team within the dynamic group model. *Direction* is used when one person tells another person what to do, and social circumstances compel the other person to perform the work. Managers in the hierarchical model will send guiding directions to system operators.

Team-based management by exception supports and encourages fluidity in the system management process. This fluidity extends to the spatial and temporal distance of team members, team member autonomy, directness of interaction between team members, and team composition [5]. Tools that support these dynamics are called *any place/any time* tools. These tools emphasize group communications, memory, and process support [6]. The group memory tools allow team members to share and extend evolving group goals to support cooperation, while the process support tools provide team coordination.

Tool Support. Our search for tools to support team-based management by exception was governed by several principles [7]. First, each tool must allow a single user to address as much of the problem as possible without having to interact with other users. Second, users (however many) deciding to use a tool to interact should receive immediate benefits over those that do not. And finally, the number of tools should be kept to a minimum. This reduces training costs and the user's mental burden.

Figure 3 illustrates the work-group computing tools needed for team-based management by exception. They are explained below.

Tools for Distributed Management by Exception. To support distributed management by exception, operators will need tools that support the augmentation and debate interaction types. Augmentation tools should allow operators to send each other exceptions, associated data, and close monitoring relationships, and allow operators to track exception resolution. A combination of threaded email, conversation-based tools, and Active Mail can provide these capabilities. Threaded email and conversation-based tools [8] provide the necessary group communication and memory capabilities. The Active Mail tool [9] unites email with screen (i.e., real-time connection) sharing capabilities. Another tool that could support augmentation is the shared task blackboard from the ALLY system [10]. The ALLY blackboard appears especially promising as a group coordination and memory tool in situations where operators interaction frequently. Each team member can post exceptions and resolutions to the blackboard, and identify who was responsible for each posting.

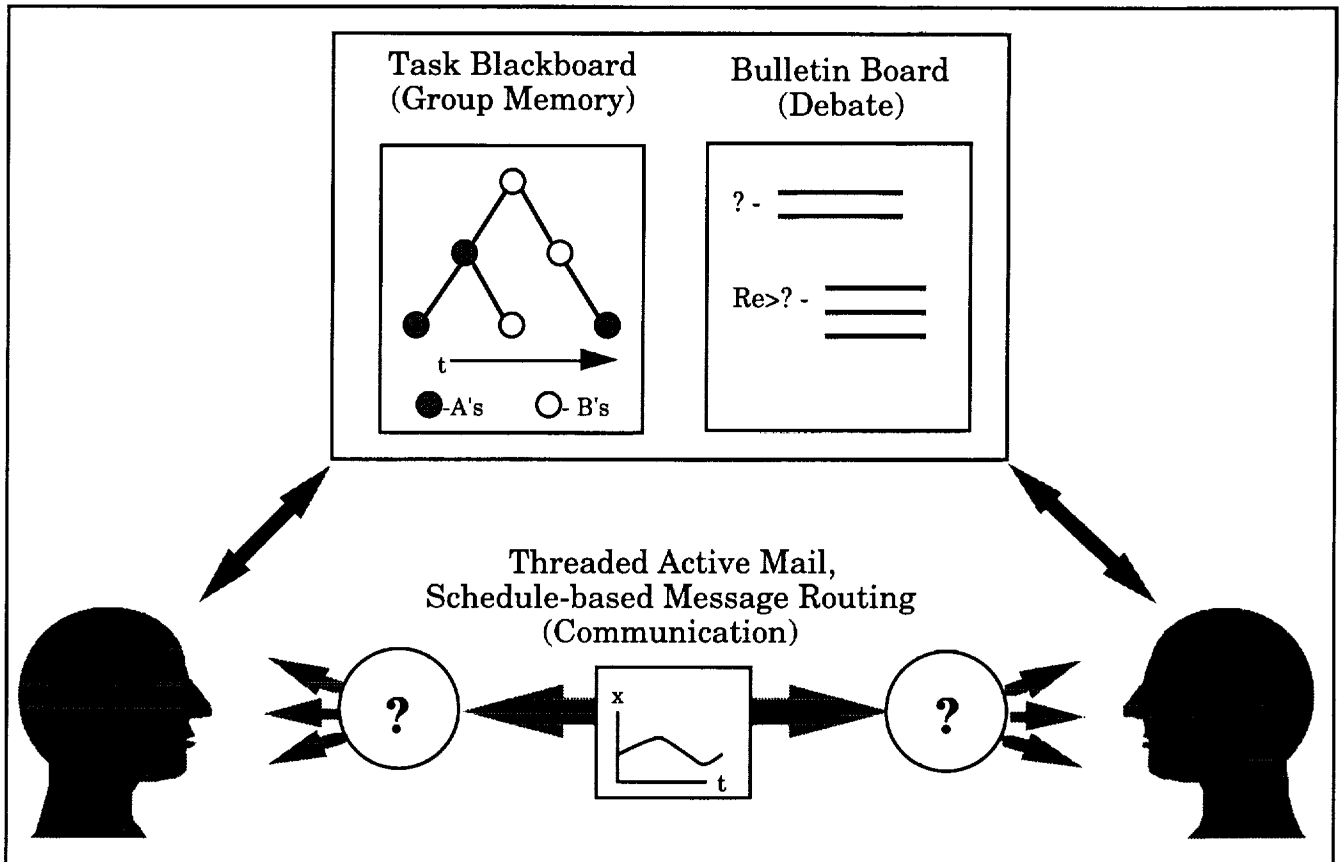
Task Blackboard (Group Memory)
Bulletin Board (Debate)
t
●-A's
○- B's
? -
Re>? -
Threaded Active Mail, Schedule-based Message Routing (Communication)
?
x
t
?

Figure 3: Tools for Team-based Management by Exception

Debate tools should allow operators to raise questions about exceptions and their resolution and receive answers from team members. Ideally, this knowledge should be recorded for future reference. Bulletin board tools, such as the Internet News services [11], support posting information, requests and responses to a common, hierarchical bulletin board, and tracking relationships between these postings.

Tools for Management by Exception with Dynamic Groups. To support management by exception with dynamic groups, operators and engineers will need tools that support the integration and negotiation interaction types. Integration tools should help identify available expertise and help contact these individuals. One of the key aspects of the dynamic group model is that expertise should be located based on availability. Schedule management tools can help in locating available support. Communications support like that provided in Wang's Freestyle system could contact operations staff using the most appropriate method (e.g., email, FAX, pager, phone) based on their scheduled location and the importance of the problem.

Negotiation tools should help operators and engineers broadcast work to be performed, responses to announced work, and work assignments. This interaction could be achieved through extensions of the augmentation tools described earlier. Conversation tools have already been used to support negotiation in other systems. The shared task blackboard also could be extended to provide some support for negotiation, particularly work announcements. We would expect, however, that most of the work allocation process would take place out of public view (i.e., off of the shared blackboard) via the conversation tools because the process would add too much distracting clutter to the shared view.

Tools for Hierarchical Management by Exception. To support hierarchical management by exception, managers and senior staff will need tools that support the direction interaction type. Direction tools should help managers monitor the spacecraft management process, and they should help managers give direction to the process. Support tools for the other models could be extended to allow managers to raise exceptions to operators and to adjust exception priority.

4 USER AGENTS

Team-based management by exception through work-group computing lowers operator workload by not requiring the operators to actively monitor the system. However, it does require that they issue all commands needed to respond to any exceptions. In this section, *user agents* [12] are discussed as a way of automating the action part of operator activity. *User agents* are semi-autonomous programs that act as personal assistants to provide indirect interaction between the user and the system. Agents are *semi-autonomous* in that they can act in the background without direction from the user. Agent support is called *indirect interaction* [13] because once the user teaches the agent how to do a task the agent performs future task instances automatically.

Flexibility is the main advantage user agents have over other automation approaches. User agents can be defined on a per exception basis. Automation therefore can be introduced gradually. Each agent's authority to act can be adjusted by its user. The internal behavior of an agent may be viewed and controlled by the user. And all information, actions, and tools available to the agent are available to the user. In fact, agents can be viewed as a self-invoking macro system. The user can perform all or part of the agent's actions manually.

Figure 4 illustrates the interactions between a user and an agent. Before an agent can perform a task, the agent must learn how and when to perform the task. A user can teach her agent how to perform a task in two ways. The user may instruct the agent to watch her perform the task. The agent then can *imitate* the user's actions. This teaching method has proven powerful in macro systems. Second, the user *can program* the agent to perform the task. It is strictly up to the user what tasks will be taught to her agent and how these tasks will be performed.

Agents can be taught when to perform a task through user feedback, and by generalizing from situations in which they originally learned the task. In the *user feedback* approach, the user reinforces the agent when it tries to perform the task in an appropriate situation and discourages the agent when the situation is inappropriate. The agent uses the feedback to weight features in the situation as indicating an appropriate or inappropriate situation. In the *generalization* approach, the agent uses heuristics to weight features. The user feedback approach is generally preferred since it gives the user more control.

An agent is not allowed to perform a task until it has *demonstrated competence* for the task. When an agent initially learns a task, the agent *watches* the user perform the task to assess if it would have performed the task in the same way and under the same circumstances. The agent asks for corrections when it would have performed the task incorrectly. Once the agent can successfully predict its user's actions, the user may allow the agent to begin *suggesting* when and how to perform the task. This type of interaction can serve as a reminder for the user. After the agent has proven itself in a suggesting role, the user then may allow the agent to *perform* the task .

Users drive all aspects of agents. This addresses some issues that other forms of automation ignore. First, because the progression from watching to performing is controlled by the user and because users teach agents how and when to perform tasks, users tend to trust agents. Second, agents perform tasks in user-specific ways. This makes agents more accepted and understandable. Third, agents easily can adapt to changing operational processes. This helps avoid obsolescence and costly maintenance.

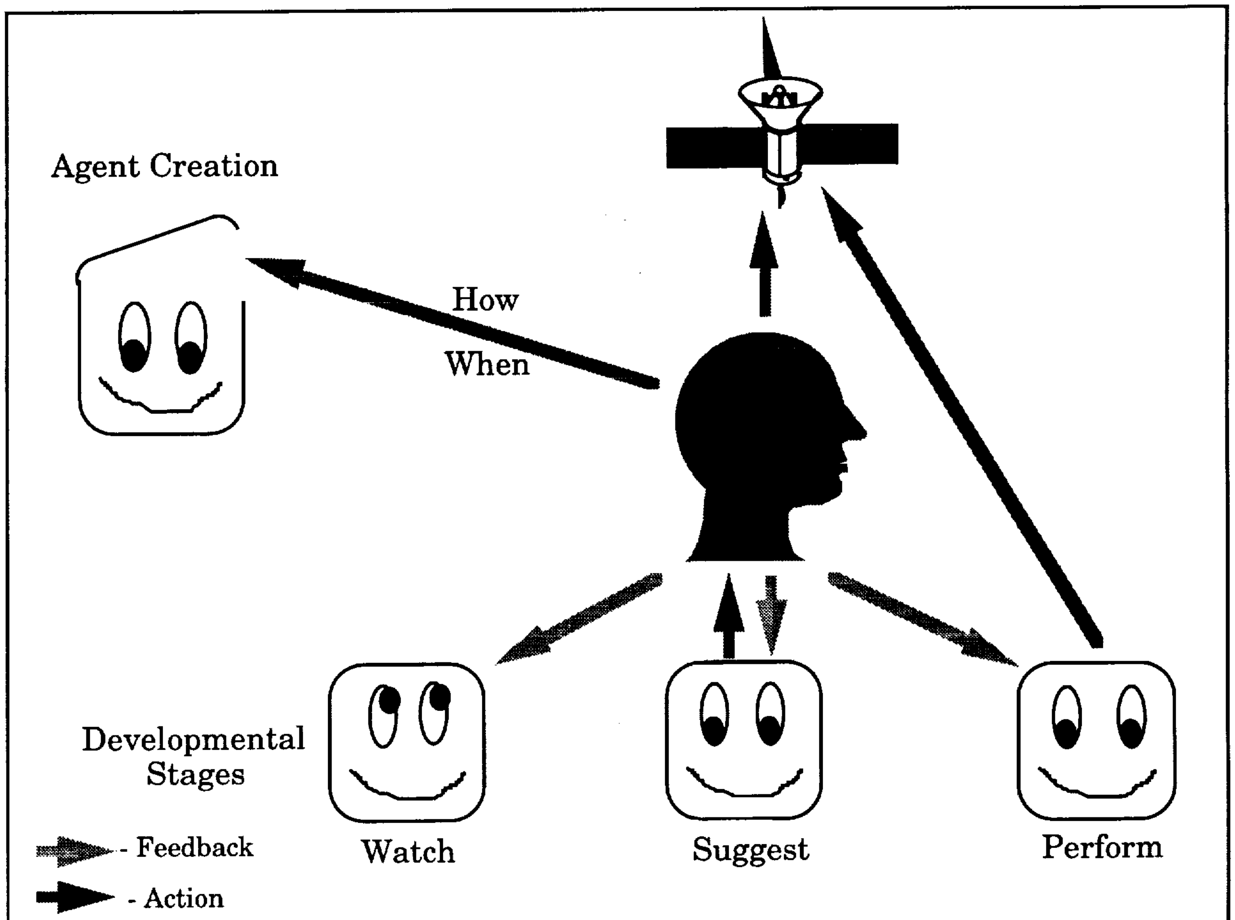

Figure 4: Interactions with a User Agent

User agents would fit well in a mission operations environment, especially one using team-based management by exception. An agent could be trained to automatically respond to selected exceptions. Exceptions provide a well-defined situation in which a task should be performed. The commands and scripts used in most operations centers easily could be taught to an agent using macro recorder-like capabilities. And the work-group computing tools of a VMOC would make it easy for an agent to act as another team member. For example, agents could post tasks to the shared task blackboard described earlier.

5 PUTTING IT ALL TOGETHER

A VMOC is the combined application of the previously presented technologies to spacecraft management. This section shows, through a series of common operations situations, how the combination addresses the spacecraft management trends and tenets presented in Section 1.

Routine Monitoring. In a VMOC, an operator would no longer have to spend time actively monitoring large numbers of changing data values. The system could monitor these values for her, even if the pass length were dramatically shorter. The operator could perform other tasks, possibly for other missions, until the system alerted her to an exception. If the exception's resolution were simple enough, she could delegate responsibility for responding to her personalized user agent. If the user later wished to respond, she could. If she wished to monitor the system in real-time, she could. The VMOC therefore would allow the operator to adjust her workload as she saw fit.

The Night Shift. When a mission's operations become sufficiently predictable, the use of management by exception could be extended both in the time and distance between the operator and the system. In the night shift, for example, the operator might stay on-call at home. When a sufficiently serious exception is detected, the system would call or page the operator at home. The system might even FAX the operator the data needed to determine if the situation warranted her returning to the operations center. Minor exceptions could be resolved automatically by the operator's user agent.

Expert Assistance. If an operator in a VMOC determines that an exception cannot be resolved without additional expertise, the operator can send a request for assistance to the engineering team. The request could include any data sent with the exception, and any real-time displays. The appropriate engineer would be located automatically using the engineers' schedules. The engineer would be notified using whatever means was appropriate (e.g., a pager). The engineer then could access the same capabilities available to the operator to determine how to respond.

Special Events and Remote Users. The capabilities described in the previous scenarios could be made available to other people associated with the mission to allow them to participate in the monitoring process and to give them a view of the system's behavior. During special events, additional personnel could be added to the monitoring team simply by giving them access to these capabilities, possibly from their offices. Scientists and engineers could use the capabilities to monitor system resources related to their work. They also could use the capabilities to generate personalized reports. This could dramatically lower operator workload, and allow external groups to interact easily with the operations process.

Managing a MOC. In a VMOC, managers or senior operations staff could define, monitor, and respond to exceptions about the process. For example, a manager might request that the system notify her if too many exceptions are occurring in some time period. Managers also could use the exception definition and notification capabilities to introduce exception violations into the process or raise the importance of an exception as a way of guiding the process.

Training a New Person. In a VMOC, new operations team members could be given access only to exception monitoring capabilities for a few subsystems. They also might be given the ability to watch the step-by-step response of other operators. In this way, new operators could learn by watching others, possibly from a remote site. Their access could be increased as they learned the system. The managing capabilities from the previous scenario could be used to monitor and assess a trainee's progress.

These scenarios suggest that a VMOC-based approach to mission operations could reduce staff workloads dramatically, possibly to the point of allowing operations personnel to be shared among missions and possibly reducing staff levels for some shifts. VMOCs would enable external groups to easily join or access the spacecraft management process. The automated nature of the support could allow operators to manage more complex systems with even less access to the system. The fluidity provided by the VMOC approach would achieve these benefits without eliminating existing capabilities. Operators still could perform active spacecraft monitoring and control as needed, including low-level control. Large operations teams still could be co-located in the same room around the clock. In a VMOC, however, these scenarios would no longer be the only options.

REFERENCES

1. Fatig, Michael, "Mission Trends: FOT Workload Definition," PRESENTED TO THE AUTONOMOUS SPACECRAFT CONTROL SYSTEMS WORKING GROUP, NASA Goddard Space Flight Center, Greenbelt, MD, June 1993.

2. Morris, Wayne, "Performance Management of Open System Environments," PROCEEDINGS OF THE 18TH INTERNATIONAL CONFERENCE FOR THE MANAGEMENT AND PERFORMANCE EVALUATION OF ENTERPRISE COMPUTING SYSTEMS, December, 1992.

3. Sheridan, Thomas, TELEROBOTICS, AUTOMATION, AND SUPERVISORY CONTROL, The MIT Press, Cambridge, Massachusetts, 1992.

4. Tapscott, Donald, and A. Caston, PARADIGM SHIFT: THE NEW PROMISE OF INFORMATION TECHNOLOGY, McGraw-Hill Inc., New York, New York, 1993.

5. Schmidt, Karl, "Cooperative Work: A Conceptual Framework," WORKSHOP ON NEW TECHNOLOGY, DISTRIBUTED DECISION MAKING, AND RESPONSIBILITY, Bad Homburg, May 1988.

6. Johansen, Robert, "The Future of Open Groupware," GROUPWARE AND OPEN SYSTEMS: AN NTU SPECIAL SERIES, National Technology University, April, 1993.

7. Malone, Thomas, K. Grant, F. Turbak, S. Brobst, and M. Cohen, "Intelligent Information-Sharing Systems," COMMUNICATIONS OF THE ACM, Vol. 30, No. 5, May 1987.

8. Winograd, Terry, and F. Flores, UNDERSTANDING COMPUTERS AND COGNITION: A NEW FOUNDATION FOR DESIGN, Ablex, Norwood, New Jersey, 1986.

9. Goldberg, Yaron, M. Saffron, and E. Shapiro, "Active Mail – A Framework for Implementing Groupware," PROCEEDINGS OF THE 1992 ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, Toronto, Canada, October, 1992.

10. Bushman, James, C. Mitchell, P. Jones, and K. Rubin, "ALLY: An Operator's Associate for Cooperative Supervisory Control Systems," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Vol. 23, No. 1, January 1993.

11. Krol, Ed, THE WHOLE INTERNET, O'Reilly and Associates, Inc., Sebastopol, California, 1992.

12. Maes, Pattie, "Intelligent User Interfaces," PROCEEDINGS OF THE ELEVENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, Washington, D.C., July, 1993.

13. Kay, Alan, "Computer Software," SCIENTIFIC AMERICAN, Vol. 251, No. 3, September, 1984.