Intelligent Control of Life Support for Space Missions

Debra Schreckenghost, Carroll Thronesbery, Peter Bonasso, David Kortenkamp, and Cheryl Martin, *NASA Johnson Space Center*

NASA's Johnson Space Center developed and field-tested intelligent control systems for crew air regeneration and water recovery. These systems support human intervention when needed but operate autonomously most of the time.

Future manned space operations will include a greater use of automation than we currently see.[1] For example, semiautonomous robots and software agents will perform difficult tasks while operating unattended most of the time. As these automated agents become more prevalent, human contact with them will occur more often and become more routine, so designing these automated agents according to the principles of human-centered computing is important.

In this article, we describe two cases of semiautonomous control software developed and fielded in test environments at the NASA Johnson Space Center. This software operated continuously at the JSC and interacted closely with humans for months at a time.

Our approach

For the past seven years, we've worked on developing intelligent software for the control of advanced life support systems. We fielded this control software in an operational environment in which test engineers manually controlled and continuously monitored all life support systems from a console in a test control room. Such operations required the engineers to spend considerable time on routine data monitoring and low-level commanding. Our biggest challenges initially were to prove that automated control software was reliable enough to be useful and that automating routine control tasks would be worthwhile.

Thus, from the beginning, we had the goal of using automation to reduce the engineer's workload. However, our objective was not to replace humans in operations but to free them from routine tasks (such as vigilant monitoring), thereby enabling them to concentrate on activities that capitalize on human strengths (such as supervisory monitoring). To perform these new tasks, humans still must interact with the control automation. In fact, human interaction becomes more challenging because the human is less involved in routine day-to-day operations and, as a result, might be less aware of the ongoing control situation and could lose anomaly response skills through lack of practice. This is a critical consideration for the human centering of semi-autonomous control systems.

We also recognized that the change in test operations resulting from the use of automated control would be fundamental. The human role changes to one of supervisory monitoring with occasional intervention when operations cannot be automated or when exceptional situations occur. During normal operations, engineers supporting these tests will spend most of their time doing activities unrelated to control but will need to be on call should the automation or life support hardware experience problems. In addition, humans will need to supervise and command these continuously operating systems from remote locations (such as their offices) with only infrequent (and possibly asynchronous) interaction. For such operations, human supervisors must be able to quickly form an integrated view of distributed control without having to continuously monitor control data.

This concept of test operations, however, represented too radical a change to be quickly accepted, so we took a gradual approach—first demonstrating the viability of automating routine control in freeing up test engineers before trying to support remote operations from their offices. This slower approach also gave us the opportunity to learn from our experiences and incorporate these lessons in subsequent systems and revised operations concepts.

Case studies in life support control

The Crew and Thermal Systems Division (CTSD) of JSC's Engineering Directorate develops advanced technology for regenerative life support systems. Regenerative life support removes the waste products that biological systems generate (for example, carbon dioxide or waste water) and recovers any consumables from those products (for example, converting carbon dioxide to oxygen or recovering potable water from waste water). In 1997, we began developing a product gas transfer (PGT) system for regenerating crew cabin atmosphere as our first case study. We built intelligent software that could control the injection of the carbon dioxide the crew produced into a plant growth chamber and control the extraction of the oxygen these plants produced for crew use and waste incineration. We operated this control software 24 hours a day, seven days a week during a 90-day manned ground test.

In 1999, as our second case study, we began developing the advanced water recovery system (AWRS) to recover potable water for crew consumption. Twenty-four hours a day for 12 months, we controlled a biological water processor in the CTSD Advanced Water Lab. In 2000, we began developing control software for the remaining water subsystems, working toward an integrated water recovery test to start in Spring 2001 (which continued through Spring 2002).

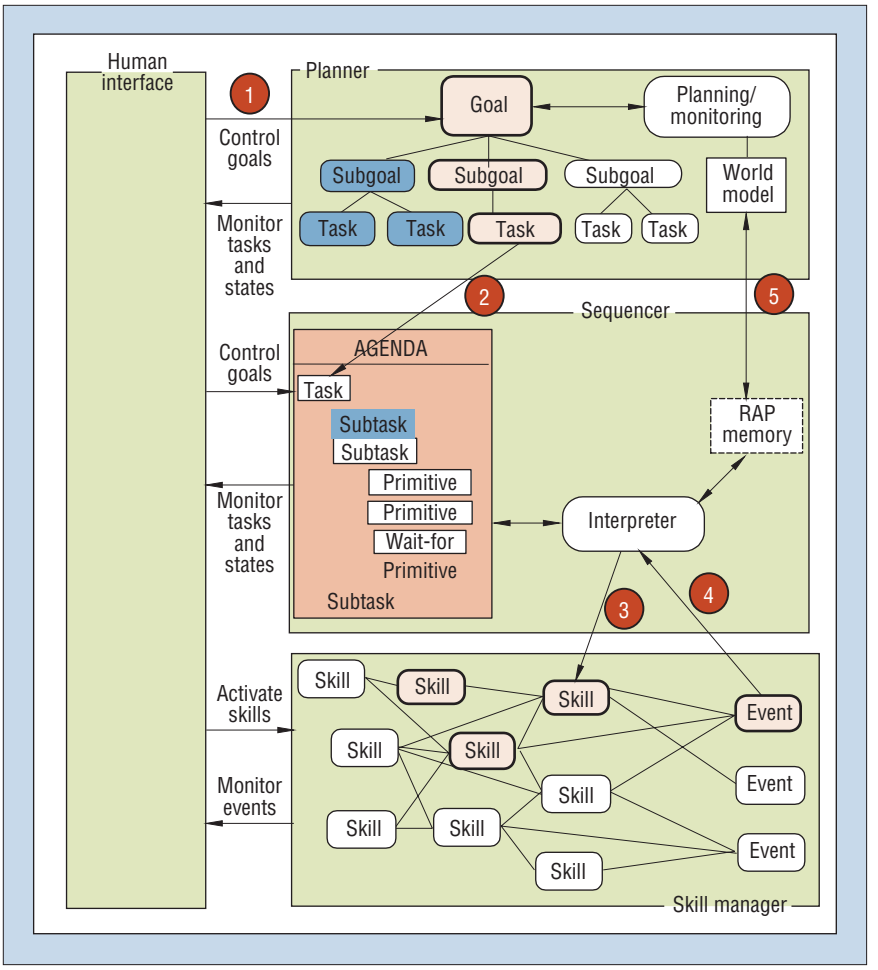


Figure 1. The three-tier control architecture.

The autonomous control architecture

We developed the autonomous control software for PGT[2] and water recovery[3] using the three-tier (3T) control architecture,[4] which consists of three concurrently operating tiers of control processing (see Figure 1). We implemented the top, or *planning*, by using a hierarchical task net planner—called the Adversarial Planner (AP)—developed at Mitre. This planner predicts the control tasks required to achieve control objectives, given an initial control situation. We implemented the middle, or sequencing, tier by using I/Net's Reactive Action Package System (RAPS). It reactively selects and orders procedures to accomplish the planned tasks passed to it by the top tier. It accomplishes this by decomposing high-level planned tasks into low-level procedure steps (called *primitive actions*) that are appropriate for the control situation. These primitive actions are passed to the bottom tier—the skill manager—for execution. Procedure steps consist of primitive control actions that issue commands (called *skills*) and monitoring actions that confirm the effects of these commands (called *events*). Control processing is closed loop at all tiers, permitting execution monitoring and reactive replanning and reconfiguration in response to environmental changes.

The flow of control among these tiers proceeds as follows. Humans specify the control goals that initiate the building of a control plan, which consists of time-ordered control tasks (Step 1 in Figure 1). Control tasks in this plan are passed from the planning tier to the sequencing tier when it's time to execute them (Step 2). Each control task from the planner maps to a top-level task in the sequencer. When the sequencer receives a high-level control task, it decomposes the task into a sequence of primitive actions. These actions then pass to the skill manager for execution (Step 3). Each action from the sequencer maps to a skill in the skill manager. Because control processing is closed loop at all tiers, the effects of actions pass from skill manager events into states in the sequencer memory (Step 4), and the effects of control tasks pass from states in the sequencer memory into states in the planner memory (Step 5).

Case 1: The product gas transfer system

The PGT control software semiautonomously controls life support systems designed for regenerating the air in space habitats. Researchers used it during the Phase III test of NASA's Lunar–Mars Life Support Test Program (see Figure 2). For this test, four crew members lived in a closed habitat for 91 days. Wheat was grown in a separate closed chamber for air regeneration and food and was planted and harvested in four stages separated by 16 to 24 days. For the available crop space in the plant chamber, this crop staging resulted in enough oxygen (O_2) for one of the four crew members. Oxygen for the remaining three crew members was regenerated using more traditional physical–chemical techniques. Additionally, the crew's solid waste was incinerated every four days, and effluent from this incinera-

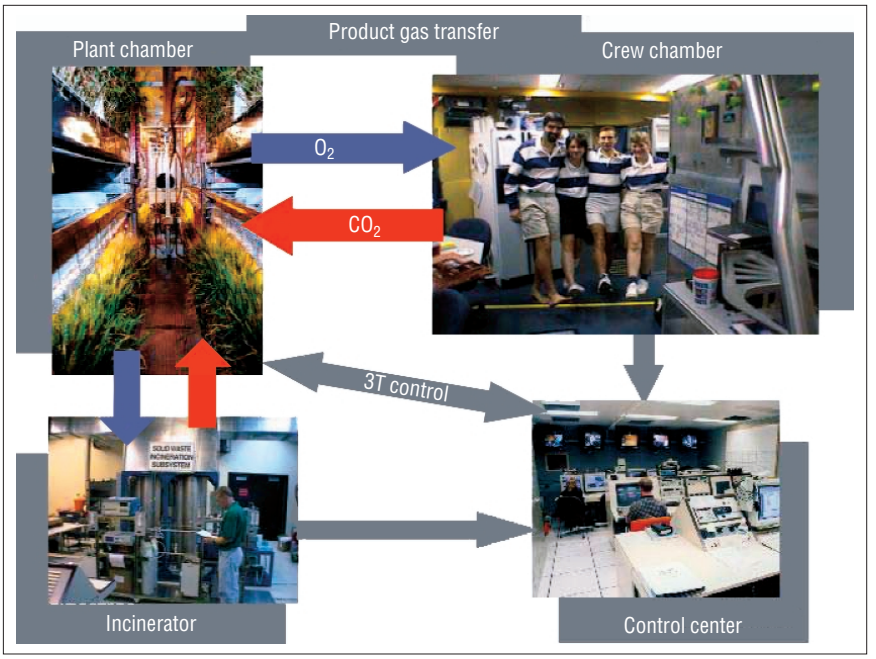


Figure 2. The control of product gases for the product gas transfer system's Phase III test.

tion contained carbon dioxide (CO_2) that the plants regenerated into O_2. Other sources of CO_2 included gas removed from the air in the crew chamber and a pressurized bottle.

The PGT system managed the transfer of O_2 and CO_2 (called product gases) among gas reservoirs, ensuring crew and crop health and recycling gases produced during waste incineration. These reservoirs included a crew habitat, a plant chamber, an airlock, and several pressurized tanks. The PGT system performed the following tasks autonomously during the test:

- Removal of O_2 and injection of CO_2 in the plant chamber to maintain the setpoints needed for plant growth
- Storage and transfer of O_2 from the plant chamber to the crew habitat for crew use
- Storage and transfer of O_2 from the plant chamber for use during solid waste incineration
- Selection and configuration of the best available source of CO_2 for the plants
- Configuration of PGT for periodic activities—specifically, harvesting and replanting crops every 16 to 24 days and incinerating waste every four days

The PGT control software operated continuously from 6 October through 19 December 1997. In terms of human centering, we changed the role of the human from vigilant monitoring and control to supervisory monitoring, thereby reducing the amount of time test engineers spent on the console. Operation of the control software in previous tests was manually intensive, requiring vigilant monitoring, frequent manual adjustment of control parameters, and two eight-hour shifts manned daily. In contrast, operating the PGT control software typically required six to eight hours per week of console support, with an additional six hours for each waste incineration and three hours for each planting and harvest.

Although we recognized that providing software for test engineers to remotely monitor and control these systems (for example, from their offices) would be useful, the sanctioned test operations only permitted monitoring and controlling from the test control room. Even so, we made several inroads into supervisory monitoring by having the PGT software detect off-nominal conditions requiring human intervention and sound alarms in the control room when these conditions occurred.

Case 2: The advanced water recovery system

The AWRS control software semiautonomously controls the recovery of potable water from wastewater. Researchers at JSC used it continuously for more than a year in the Advanced Water Lab. During this period, they conducted a series of unmanned tests to evaluate the AWRS hardware operating at different waste loads and hours of operation. The AWRS comprises four subsystems that operate in tandem, each one designed to remove a type of impurity from the water:

- *Biological water processor*. The BWP removes organic compounds and ammonia from incoming wastewater, using microbes to remove both the bulk of the total organic carbons as well as the ammonium. The resulting N_2 gas is vented, and the resulting CO_2 gas is sent to a CO_2 reduction system.
- *Reverse osmosis system*. The RO system removes inorganic compounds from the BWP's effluent, forcing the water to flow at high pressure through a molecular sieve that rejects inorganic compounds and concentrates them into brine.
- *Air evaporation system*. The AES recovers additional water from the brine the RO produces. It removes the concentrated salts from the brine by depositing it on a wick, evaporating the water by blowing heated air through that wick, and then condensing the water by cooling the air.
- *Post-processing system*. The PPS brings the water product from the RO and the AES to within potable limits. This "water polisher" removes trace organics and ammonium by a series of small beds of ion exchange resins and the trace organic carbons through a series of ultraviolet lamps. It also oxygenates the water with an O_2 concentrator.

Figure 3 shows the flow of water products through water recovery systems in the Advanced Water Lab.

The AWRS control software operated continuously for 98.75 percent of the time between Spring 2001 and Spring 2002. This 12-month test demonstrated our control approach's reliability for long-duration operations and scalability to multiple subsystems. We also demonstrated the viability of remotely monitoring control operations from engineers' offices. As expected, this capability changed the human's role in control operations to supervisory monitoring with occasional human intervention. Each week, one control engineer served as prime engineer to supervise the control autonomy. The prime engineer used the remote monitoring capability to check the health and status of both the life support hardware and the control software three to four times daily, including weekends. He or she was also on call to go to the Advanced Water Lab and handle occasional control problems (such as network performance or data acquisition anomalies).

Because of funding constraints, developing remote monitoring capability was a low-

level effort. However, the software resulting from this limited effort proved quite popular with the engineers, who quickly requested more remote interaction capability than we had the resources to create.

Case study results

In both the air regeneration and water recovery cases, we reduced human workload by automating routine control tasks. In both cases, human participation was still necessary, but as intended, the human's role in control changed from vigilant monitoring with frequent command intervention in previous life support tests to supervisory monitoring with infrequent intervention.

Some tasks are not easy to automate and still require human intervention:

- *Calibrating sensors.* Because calibration is a manual task and determining when it should occur is not easy, a test engineer must reconfigure sensors during calibration.
- *Reconfiguring data interfaces.* Waste incineration occurred every four days during portions of the Phase III test, with the waste incineration control software shut down between occurrences. To receive data during an incineration, we had to manually start incineration software and reconfigure data interfaces.
- *Transferring data logs to a centralized computer.* Custom interfaces between control computers with data logs and the central archive computer required the logs to be manually transferred three times a week.
- *Tuning the control strategies.* Because these case studies were the first long-term tests using biological air regeneration and water recovery, we had to tune the control strategies throughout the test. To do so, we would temporarily suspend the automated control strategy and manually enter revised control actions.

Both case studies had infrequent anomalous situations occur that required human intervention. Some of the problems encountered during these tests included computer hardware failures, network performance problems, and data acquisition anomalies. We also discovered that biological regenerative systems are more sensitive to their environment than physical–chemical systems and were thus more likely to exhibit surprising or unexpected behaviors requiring intervention. For example, microbial clogs in the BWP required a manual procedure to slough a layer of the microbe colony; loss of wheat crops due to nutrient leaching required planting extra crops and manually adjusting nutrient balance.

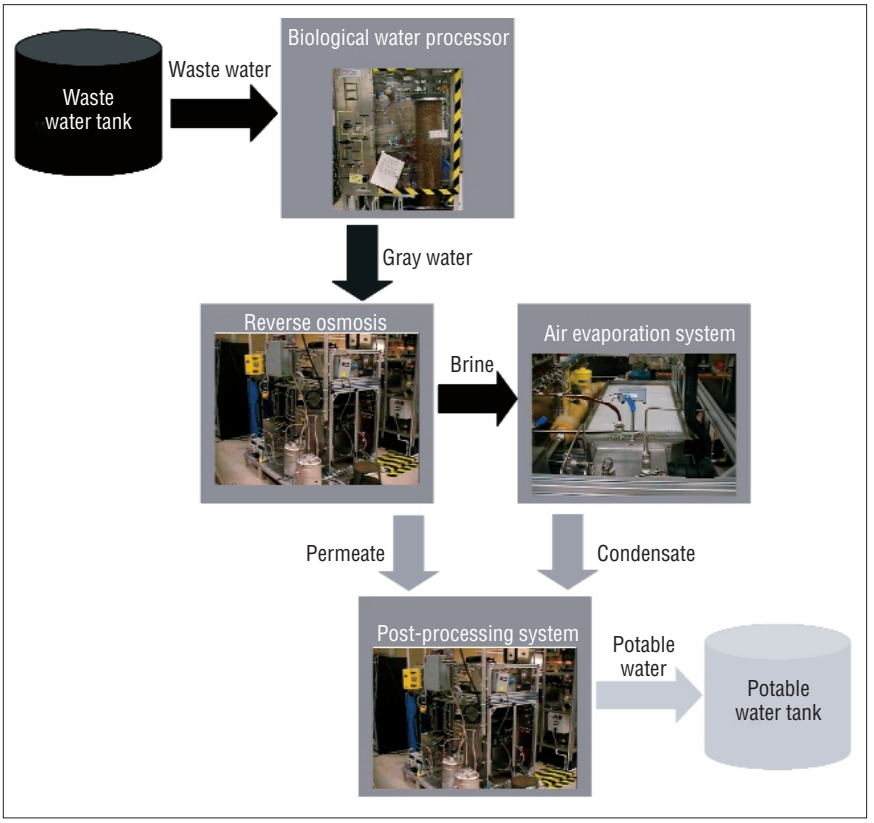


Figure 3. The control of water recovery in the Johnson Space Center's Advanced Water Lab.

Our approach to design automation for these anomalous situations was to provide capabilities for adjustable control autonomy.[5,6]

Temporary suspension of automated control activities that interfere with manual activities. Test engineers suspended autonomous control while issuing manual commands to prevent conflicting or inconsistent commanding. The scope of this suspension ranged from inhibiting the execution of single procedures to preventing autonomous commanding of entire subsystems. Examples of activities requiring temporary suspension of control automation include maintenance activities (for example, sensor calibration) and activities with human safety risks (for example, not injecting CO_2 when people are in the plant chamber).

Monitoring control states autonomously regardless of the level of autonomy. The autonomous system actively monitored state and control information about the life support system even during manual control. This continuous monitoring permitted detection of caution and warning states for all levels of control autonomy and ensured the control software was "cognizant" of state changes caused by manual action.

Reconfiguring sensors and controllers for maintenance. Either a human or the automation could take a sensor offline for calibration or repair (both of which we needed during the course of these tests) without impeding autonomous control.

Adjusting control setpoints and warning thresholds without taking the control software offline. Either the human or the automation could modify these control parameters (for example, the planner changed the gas setpoints in the airlock when incinerating waste).

For supervisory monitoring, we also found it essential to provide data summarization software to capture and present information that could assist a human in quickly understanding the control situation.[8] Because the human was not monitoring most of the time, this information had to be collected and stored in data structures that retained all transient information relevant to describing the situation. Information needed to describe the control situation includes statistics and trends in production and consumption of water and

air (life support system performance), the relationship of control actions with their intended and actual effects on system state, and system anomalies diagnosed with the actions taken to resolve them.

Applying human-centered computing to control automation

We found that effective human interaction with control automation requires defining a new concept of operations that affects the behavior of both humans and automated control agents. To do this, we had to develop new protocols for interaction between humans and automated agents. Existing protocols for human–human interaction in space operations likewise must be adapted to account for the change in human roles. To implement this new concept of operations, we found it necessary to modify the intelligent control software as well as develop new interface software. We added the ability to export information about automated control actions and control situation assessments to the control software. We also adapted autonomous control strategies to permit adjustments to the level of control autonomy for coordinated human and automated commanding.

The new interface entails more than just display software: it requires software that implements the organizational policies and etiquette protocols arising from this new concept of operations. These policies ensure that the right people are notified of significant events regarding operations, including events indicating that they might need to take manual action. These policies also address consistent, reliable commanding among groups of human and software agents. They can help determine if an agent is authorized to issue commands and determine how to allocate control authority to avoid conflicts when more than one agent is commanding. Both the specification of event notification and the allocation of control authority must consider the policies associated with organizational roles as well as individual agent preferences. Therefore, preferences must be overlaid on organizational policy requirements to permit customization without compromising these policies.

This new type of interface software must also assist the human in relating control information as represented in the automation to human mental models of control operations. This assistance ensures that autonomous actions are more predictable and human error is less likely. It can include

- Associating automated control actions with the environmental consequences the human expects (both intended control effects and side effects, such as that turning on a fan noticeably changes the crew's ambient noise level)
- Labeling a configuration of system states as a human-recognizable operating mode or phase
- Mapping from programmatic device references to colloquial device names (for example, v0102 is the same as the blower valve)

These information relationships should be captured in the interface, not the control software, because this information is needed to support human collaboration, not automated control. Using this information, we can build user interfaces that communicate the beliefs and intentions of the automated control software more accurately, thus enabling a shared "cognitive wavelength" between the software and the human.[7] This shared understanding aids the human in conducting manual actions or in issuing instructions to the software more effectively.

Supervisory monitoring

Even when control automation operates on its own, the human still must be able to maintain or rapidly reacquire situational awareness of control activities and their effects. Locating humans remotely from the control system can isolate them from information acquired when working in close proximity and can limit the information bandwidth available to them. This constraint on situational awareness is exacerbated by the fact that humans can be temporarily out of communication. Yet, the ability to maintain ongoing situational awareness is beneficial to inform humans even of "typical" performance because this ability provides a shared operational context that is essential when responding to anomalies.[7] During both the Phase III test and the water test, control engineers maintained situational awareness by checking on the system three to four times a day. Thus, automation supervision requires the human to occasionally monitor the system remotely and to be notified when a need for manual action arises or when interesting or unusual events occur.

Data summarization and visualization techniques assist humans in assessing the control situation status at a glance. This becomes particularly important when controlling multiple, distributed life support systems whose functions are coupled. Summarization and visualization software should provide an operational overview that integrates control information along multiple dimensions.[8] From the temporal perspective, it should integrate information about predicted control activities with information about control activities as executed. From the state perspective, it should associate control activities with the intended consequences of these activities on the environment and the system configuration. From the data perspective, it should provide an integrated view of a heterogeneous set of information characterizing system control, such as planned control activities, activity histories, system performance, anomaly summaries, and archived data and statistics. Taken as a whole, these different perspectives afford insight into both the current operational situation as well as typical system performance.

Implementing these data summarization designs required us to build a model in the interface that associates control knowledge from multiple components in the autonomous software into an understandable, unified model of control tasks and their intended consequences. It also required detecting and labeling sets of state changes as operating modes that had meaning to the test engineers but no use in automated control.

Event detection and notification are needed to inform humans who are working remotely about important operational events and control actions. In lieu of such notification software, humans during both the Phase III test and the water lab support had to interrupt their regular tasks and periodically check on the system. In structured environments such as space operations, event notification must be implemented to guarantee that the right information gets to the right people in a timely manner. Thus, the implementation of event notification includes defining the conditions that indicate an event has occurred as well as encoding the organizational policies for determining who to inform of an event and how. Determining who to inform about events should be based on assigned roles. Determining how to inform will depend on the event's importance and urgency and the accessibility and availability of receivers.

An important aspect of event notification is knowing when and how intrusively to interrupt the human. When possible, human task priorities and notification preferences should be considered to prevent annoying or dis-

tracting autonomous actions.[9] When situations arise requiring human intervention, humans need support not only in coping with the new tasks but also in handling interruptions to their ongoing tasks that are unrelated to the arising control situation. This includes assistance in managing multiple ongoing tasks, notification of task deadlines, aid in returning to interrupted tasks, and assistance in revising their schedules when interruptions impede their ability to complete assigned tasks.

Remote commanding and group situation awareness

When situations arise that automated control cannot address, it is necessary to support some level of human intervention in system control. During the life support tests, circumstances requiring human intervention included both nominal manual activities such as calibrating sensors or replacing filters and off-nominal manual activities such as system repair. In such circumstances, other members of the control team must maintain awareness of manual actions taken. During the water test, it was common for members of the control team to share such information using phone calls and email. Thus, notification services should include detecting and communicating information about human intervention in control.

Our approach for human intervention in control is to provide for the interactive adjustment of autonomy. Techniques for such adjustment include reallocating tasks from the automated control software to humans, temporarily adapting automated control procedures for unique situations, and manually overriding automated control actions. Applying these techniques when people are located away from the life support hardware requires the ability to remotely command life support systems. This command capability includes reconfiguring automation to support manual actions, manual activation of automated procedures, and, less frequently, low-level manual control of life support hardware (such as turning on a pump). It also requires the capability to coordinate commands when multiple agents (human and software) might be commanding at the same time.

The interface between the human and the automation can implement policies for coordinating distributed, remote commanding. Such interface software must also support dynamic allocation and authentication of control authority according to organizational roles and situational needs. It must detect and prevent command authorization conflicts when more than one agent is commanding at a time. Such prevention includes reconfiguring the automated control system to temporarily suspend automated tasks that conflict with manual tasks. When such commanding is mediated through the interface software, it's possible to track manual activities and coordinate them with autonomous activities (a form of notifying the automation of manual actions).

A new architecture

We have designed an architecture for human interaction with control automation that supports human supervisory monitoring, group situational awareness, and remote, distributed commanding. The basis of this architecture is personalized software agents that assist humans in remotely interacting with automated control agents such as the AWRS and PGT systems described earlier. The Electric Elves system is one successful example of using software agents to aid human interaction.[10] In this system, proxy agents for each person perform tasks for their users including (among other things) monitoring each user's location, keeping other users in the organization informed, and rescheduling meetings if one or more users is absent or unable to arrive on time. Although the Electric Elves system incorporates multiple humans and software agents, each human interacts primarily with the capabilities of his or her own proxy (or with nonautomated software accessed through the proxy) to aid human–human interaction. Thus, the Electric Elves architecture does not address the need for support agents (proxies) to act as mediators or enablers for interaction between humans and a third class of agents: semiautonomous software agents that perform complex tasks such as control.

We are developing proxy agents for humans that mediate and assist interaction among humans and automated control agents. They can fulfill several roles by acting as a representative or stand-in for the human, an aid or assistant for the human, or a regulator or critic of human actions. Each human in the operational group has a proxy agent to represent his or her interests and concerns. We are implementing a proxy as the following set of customizable coordinated services for individual humans:

- *Notification service*. Uses information about human presence (location combined with whether the human is online), human roles, and user-specified preferences to determine if an operational event is of interest to a human and, if so, how to inform him or her.
- *Task status service*. Provides activity tracking and plan management capabilities for use by both the humans and the autonomous control agents affected by human activities.
- *Location service*. Provides human location information for the task status service to use in tracking the completion status of human activities and by the notification service in determining how to notify the human of events. We're also investigating the usefulness of customizing information presentation using location information.
- *Command and authorization service*. Supports the human in remotely interacting with and controlling the life support systems by determining if the human is authorized to command, resolving authorization conflicts when more than one human is interacting with the life support systems, and reconfiguring both the automation and user interface when making a transition between automated and manual commanding.
- *Interactive event service*. Assists the human in interactively defining temporary, new operational events and controlling automated monitoring conditions indicating these events have occurred.
- *Interactive procedure service*. Assists the human in temporarily modifying standard operating procedures executed by the automated control software.
- *Interruption handling*. Provides a set of capabilities including an extension to the notification service to determine if the crew should be interrupted (and how intrusive the interruption modality should be), an extension to the task status service to mark the completion status of interrupted activities and add new activities in response to the interruption, and new interruption handling services that aid a human in managing concurrent activities.

Figure 4 illustrates the interaction among two proxy agents and two control software agents. Although each proxy agent in this diagram appears to be identical, the human associated with each agent can customize and configure the agent according to his or her preferences and organizational roles.

Our architecture for human interaction with

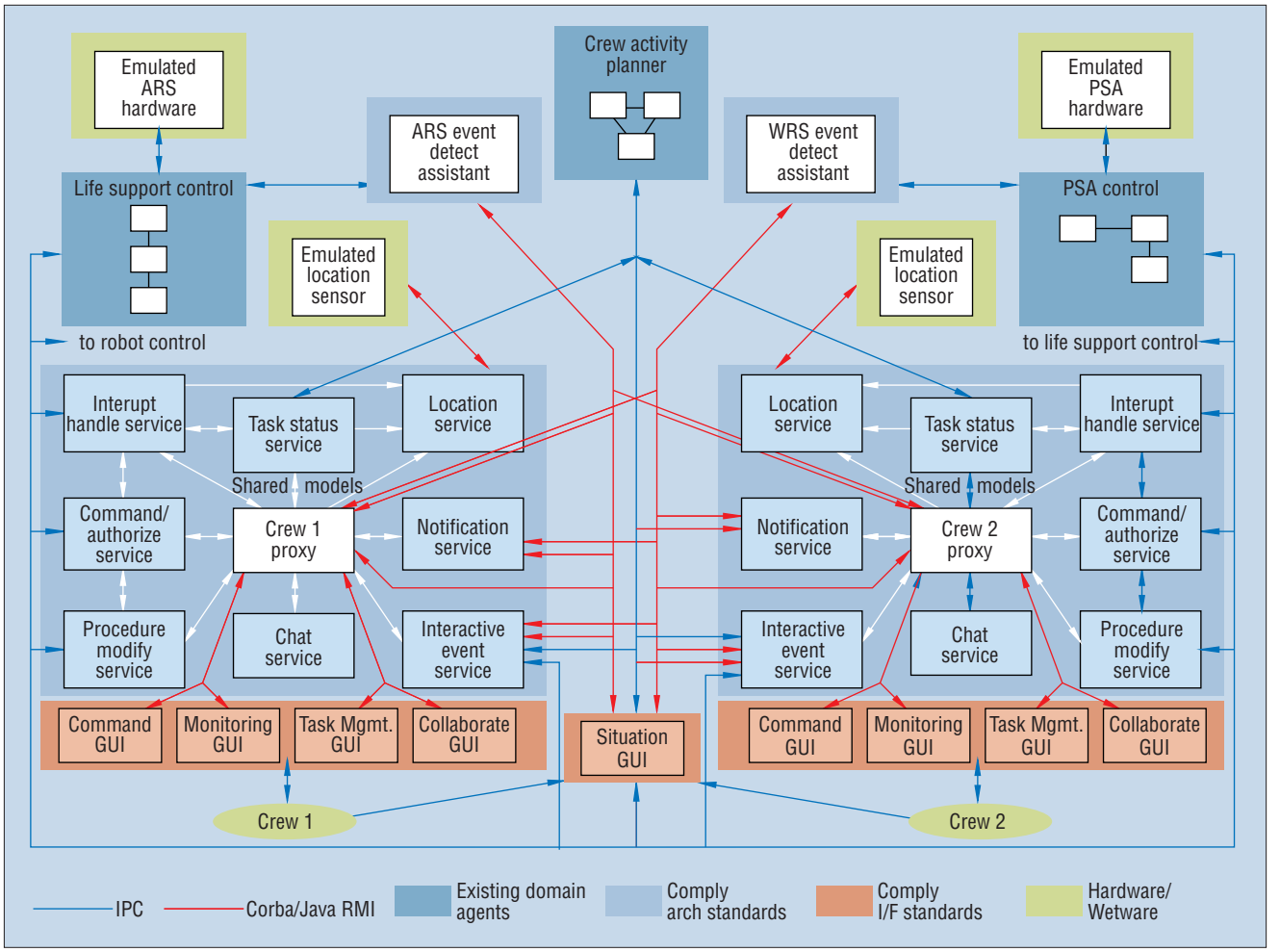


Figure 4. An architecture for human interaction with control automation.

control automation also includes control assistants that aid the human in interacting with automated control agents. These assistants permit integration with agent software developed outside of our architecture, where our architecture is associated with a specific set of knowledge models and communication protocols. To date, we have defined two types of control assistants that are needed for human interaction with automated control agents. The event detection assistant detects and broadcasts operational events that are significant to humans interacting with automated life support control, including anomalies that occur in the life support systems. The human error detection assistant detects conditions indicating the human has taken an action with potentially adverse effects on the life support environment.

We have developed an initial prototype of this architecture and have demonstrated its utility in supporting control engineers for the Advanced Water Lab. This prototype implements the proxy with three services (notification, task status, and location) and a control assistant for detecting events in the AWRS.

We are currently implementing a design for the specification and enforcement of both organizational policies and individual preferences for notification interaction. Preliminary results from our investigation of notification interaction indicate that developing and maintaining domain ontologies are essential when specifying which roles require a notice and what presentation modalities are most effective for issuing that notice. We have also done preliminary work on designing software to implement policies for coordinating distributed commanding. Such software must support dynamic allocation and authentication of control authority according to organizational roles and situational needs. It must also detect and prevent command authorization conflicts when more than one agent is commanding at a time. Once we have implemented both designs, we will define candidate models of human–software interaction for remote space operations and use the proxy agent prototype to evaluate how well these policies support these different models.

The use of automated control software changes the nature of the human's role in control operations. New protocols for interaction between humans and control agents must still be developed, and existing protocols for human–human interaction must be adapted, to account for these changed human roles. The resulting new tasks and changed protocols require new types of software to assist humans in performing them. Based on our experience with advanced life support control, we believe that such new software requires more than just good display design. It involves substantial challenges in the development of human-centered interaction support software to aid people in interacting and cooperating with automation as well as challenges in the design of the control systems themselves. Such challenges include

- Supporting humans in maintaining awareness of control situations
- Defining the circumstances and methods by which the control system notifies humans of

environmental events (nominal or off-nominal) and accepts task inputs from humans
- Considering human task priorities and preferences when determining how intrusive an interruption can be
- Aiding humans in resuming tasks interrupted by the demands of supervisory monitoring and control
- Developing techniques for tracking the completion of human activities for the purposes of planning for human–software agent teams
- Assisting the human in relating control information as represented in the automation to human mental models of control operations, to make autonomous actions more predictable and to reduce the potential for human error
- Defining the circumstances and methods for a human to adjust the level of autonomy or change the allocation of roles and responsibilities among the control agents and humans
- Developing strategies for preventing conflicting or confounding actions when multiple humans interact simultaneously with distributed, autonomous control agents
- Building the automation to distinguish between expected control situations requiring no action and unexpected control situations where no action has been anticipated (software that recognizes when it is outside the scope of its capabilities)

Although these lessons are based on our experience in manned space operations, this work is relevant to other complex, dynamic domains where humans must work with semiautonomous software agents. Other domains where these lessons might apply include robotic assistants, automation for the care of the elderly, and automated process control (including nuclear power). ■

Acknowledgments

We thank Michael Shafto, manager of the human-centered computing topic in NASA's Intelligent Systems Program, under which the work on proxy agents is being performed. We also recognize a long-term collaboration with David Woods at Ohio State University and Jane Malin at the JSC on the subject of human interaction with intelligent systems, which has influenced the work described in this article.

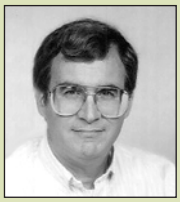
References

1. D. Cooke and B. Hine, "NASA's New Era in Space Exploration," *IEEE Intelligent Systems*, vol. 17, no. 2, Mar./Apr. 2002, pp. 63–69.
2. D. Schreckenghost et al., "Intelligent Control of Life Support Systems for Space Habitats," *AAAI Innovative Applications of AI*, AAAI Press, Menlo Park, Calif., 1998, pp. 1140–1145.
3. P. Bonasso, "Intelligent Control of a NASA Advanced Water Recovery System," *Proc. Int'l Symp. Artificial Intelligence Robotics and Automation in Space*, Canadian Space Agency, Montreal, 2001.
4. P. Bonasso et al., "Experiences with an Architecture for Intelligent, Reactive Agents," *J. Experimental Theory of AI*, vol. 9, no. 2, 1997, pp. 237–256.
5. D. Kortenkamp, D. Keirn-Schreckenghost, and R.P. Bonasso, "Adjustable Control Autonomy for Manned Space Flight," *Proc. IEEE Aerospace Conf.*, IEEE CS Press, Los Alamitos, Calif., 2000.
6. P. Scerri, D.V. Pynadath, and M. Tambe, "Adjustable Autonomy in Real-World Multi-Agent Environments," *Proc. Autonomous Agents*, ACM Press, New York, 2001, pp. 300–307.
7. G. Klein et al., *Cognitive Wavelength: The Role of Common Ground in Distributed Replanning*, tech. report, Air Force Research Laboratory, Ohio, 2000.
8. D. Schreckenghost and C. Thronesbery, "Integrated Display for Supervisory Control of Space Operations," *Human Factors and Ergonomics Society 42nd Ann. Meeting*, Optical Archives, 1998, pp. 481–485.
9. D.D. Woods and E.S. Patterson, "How Unexpected Events Produce an Escalation of Cognitive and Coordinative Demands," *Stress Workload and Fatigue*, P.A. Hancock and P. Desmond, eds., Lawrence Erlbaum, Hillsdale, N.J., 2002.
10. H. Chalupsky et al., "Electric Elves: Applying Agent Technology to Support Human Organizations," *Proc. Innovative Applications of Artificial Intelligence*, AAAI Press, Menlo Park, Calif., 2001, pp. 51–58.

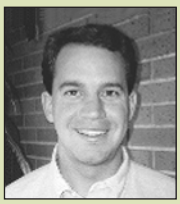
The Authors

Debra Schreckenghost is a senior scientist with Metrica, supporting the Automation, Robotics and Simulation Division at NASA Johnson Space Center in Houston. Her research interests include automated monitoring and control, architectures for intelligent software agents, and human-computer interaction. She received a BS in electrical engineering from the University of Houston and an MS in electrical engineering from Rice University. She is a member of the IEEE, AAAI, and ACM. Contact her at NASA JSC Mail Code ER2, Houston, TX 77058; d.schreckenghost@jsc.nasa.gov.

Carroll Thronesbery is a senior scientist at S&K Technologies at Johnson Space Center/NASA. His research interests include human-computer interaction with intelligent systems issues such as situation awareness, nonvigilant system monitoring, system safety, and design methodology. He received a PhD in cognitive psychology from the University of Houston. He is a member of the Human Factors and Ergonomics Society. Contact him at 1300 Hercules, Ste. 140, Houston, TX 77058; c.thronesbery@jsc.nasa.gov.



Peter Bonasso is a senior staff consultant for AI & Robotics at Metrica, based at NASA's Johnson Space Center. He currently supports the Automation, Robotics, and Simulation Division investigations of intelligent monitoring and control using layered architectures. He received his BS in engineering from the US Military Academy at West Point and two MSs in operation research and computer utilization from Stanford. He is a member of the American Association for Artificial Intelligence. Contact him at NASA JSC Mail Code ER2, Houston TX 77058; r.p.bonasso@jsc.nasa.gov.



David Kortenkamp is a senior scientist with Metrica, supporting the Automation, Robotics, and Simulation Division at NASA Johnson Space Center in Houston. His research interests include software architectures for intelligent agents, human-computer and human-robot interaction and integration of perception and action. He received his BS from the University of Minnesota and his MS and PhD in computer science and engineering from the University of Michigan. He is a member of the IEEE and AAAI. Contact him at NASA JSC Mail Code ER2, Houston TX 77058; kortenkamp@jsc.nasa.gov.

Cheryl Martin is a research scientist in the TRACLabs division of Metrica. Her research focuses on multiagent systems and human-computer interaction. She received her PhD and MS in software engineering from the University of Texas at Austin and her BS in electrical engineering from Virginia Tech. She is a member of the IEEE and AAAI. Contact her at TRACLabs, Metrica, 1012 Hercules, Houston, TX 77058; cmartin@traclabs.com.