Intelligent Control of a Water-Recovery System
Three Years in the Trenches

Pete Bonasso, David Kortenkamp, and Carroll Thronesbery

■ This article discusses our experience building and running an intelligent control system during a three-year period for a National Aeronautics and Space Administration advanced life support (ALS) system. The system under test was known as the Integrated Water-Recovery System (IWRS). We used the 3T intelligent control architecture to produce software that operated autonomously, 24 hours a day, 7 days a week, for 16 months. The article details our development approach, the successes and failures of the system, and our lessons learned. We conclude with a summary of spin-off benefits to the AI community and areas of AI research that can be useful for future ALS systems.

Pete Bonasso dedicates this article to his father, Russell P. Bonasso, 1920 – 2003

"We'll have to go with 4 two-head pumps for the nitrifier."

The AI controls engineer frowned at the speaker, a young mechanical engineer in charge of the physical design of a state-of-the-art biological water processor (BWP). "That pump doesn't give me any feedback for speed, so we can't be sure it's responding to commands."

"It'll have to do," said a woman at the far end of the conference table. As the manager for the Integrated Water-Recovery System (IWRS), she made the final calls. "The eight-head pump won't function at the required pressures, and the four heads are just too expensive. Can't you use the tube pressures to know if the pumps are working?"

The controls engineer shrugged, spreading his hands. "Sure, but with the single transducer to monitor eight tubes, we won't know for three to five minutes after the pump command is sent."

"Can we live with that?" asked the manager, glancing around the table at each member of the assembled group of microbiologists and chemical engineers.

One of the engineers tapped at his personal digital assistant, then spoke up, "Even at 32 mils a minute, the pressure buildup from the recirculation pump won't be enough to trigger the relief valve. I think it's in the noise."

"Okay," said the manager. "We go with the two heads."

The time frame was the winter of 1999, and this exchange was typical of many conversations that the AI controls team from the Automation, Robotics, and Simulation Division (AR&SD) at Johnson Space Center (JSC) would have with the advanced water-recovery personnel as the two groups prepared for a year-long test of a new Integrated Water-Recovery System (iWRS), slated to begin in January 2001. We were building an AI control system for this test that had to handle upward of 200 sensors and actuators grouped among 4 water processing subsystems. The control system would run 24 hours a day, 7 days a week, and be completely autonomous. It was an applied AI engineer's dream, and in the end, we were extremely successful. However, there were events that happened for which we were ill prepared, and we would come away with a much better appreciation for the difficulties involved in controlling long-duration life-support systems.

This article is the story of our experiences developing and running the iWRS AI control system.

The Early Years

By 1995, the AI controls team had been working with several groups in the Crew and Thermal Systems Division (CTSD), building AI control systems in support of CTSD's investigations in advanced life support (ALS). In 1995, they put a man in an airlock linked to a

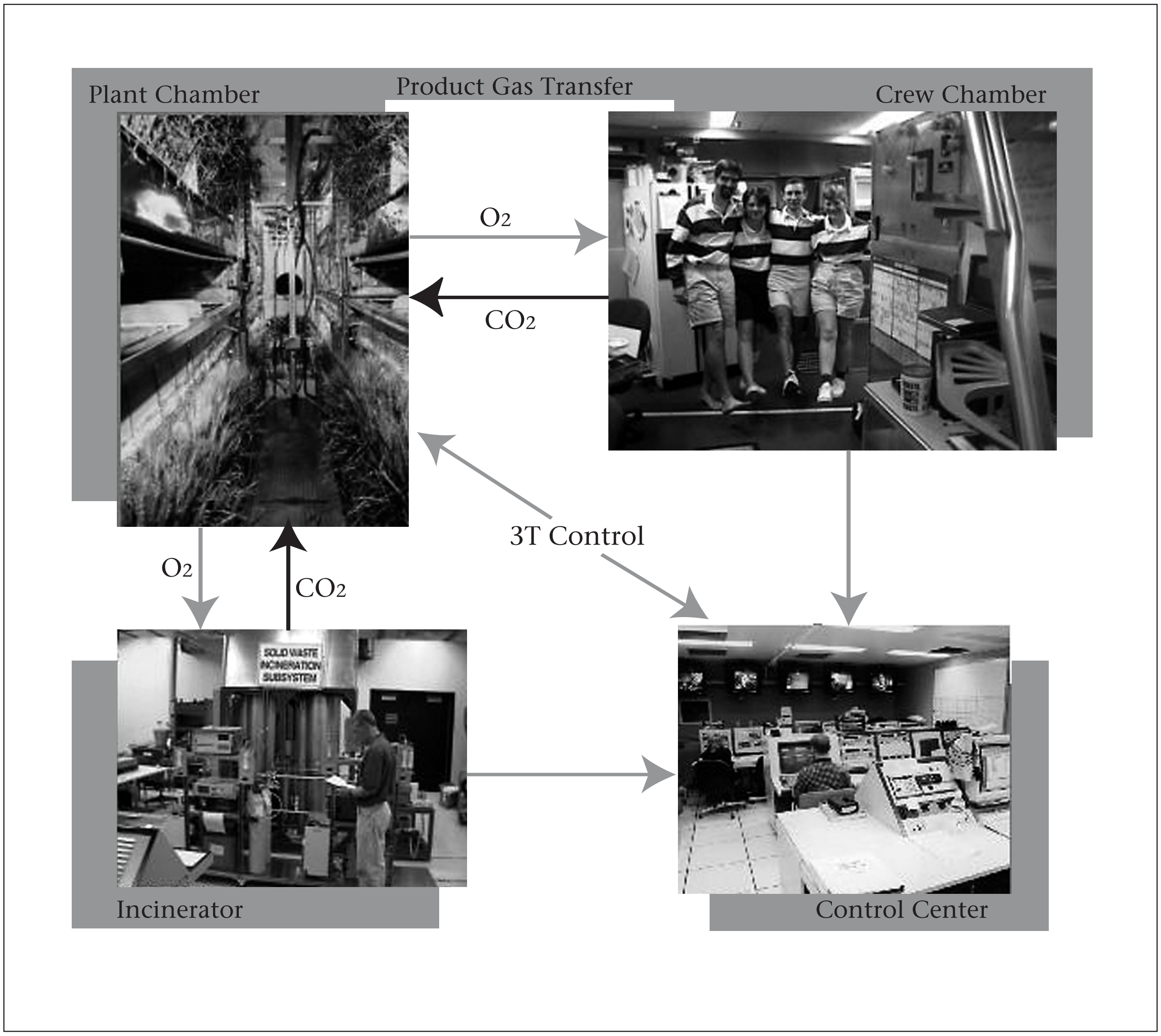


Figure 1. The Product Gas Transfer Environment.

10-foot-diameter chamber full of wheat (Laifook and Ambrose 1997). For 15 days, the man lived, worked, and exercised in the chamber as the wheat crop took in his carbon dioxide and produced oxygen for him. The control system—our first for ALS—monitored and provided caution and warnings for the climate and nutrient environment of the wheat crop.

In 1997, they put 2 men and 2 women in a 30-foot chamber for 91 days (Schreckenghost et al. 1998b). A physical-chemical air revitalization system recycled the air for 3 of the 4 people, and a wheat crop in the 10-foot chamber did the same for the fourth. The ALS team also experimented with a solid-waste incinerator. Our second ALS AI control system managed the transfer of O2 and CO2 among the gas reservoirs for this test to ensure crew and crop health and to recycle gases produced by waste incineration. These reservoirs included a crew habitat, a plant chamber, an airlock, and a number of pressurized tanks (figure 1). Operating 24 hours a day, 7 days a week, the AI system also used a generative planner that scheduled waste incinerations and crop planting and harvesting, coordinating these tasks with the day-to-day product gas transfer.

For both these projects, we used a three-layer architecture (Gat 1998) to design, organize, and develop the control software. AR&SD had

The 3T Intelligent Control System

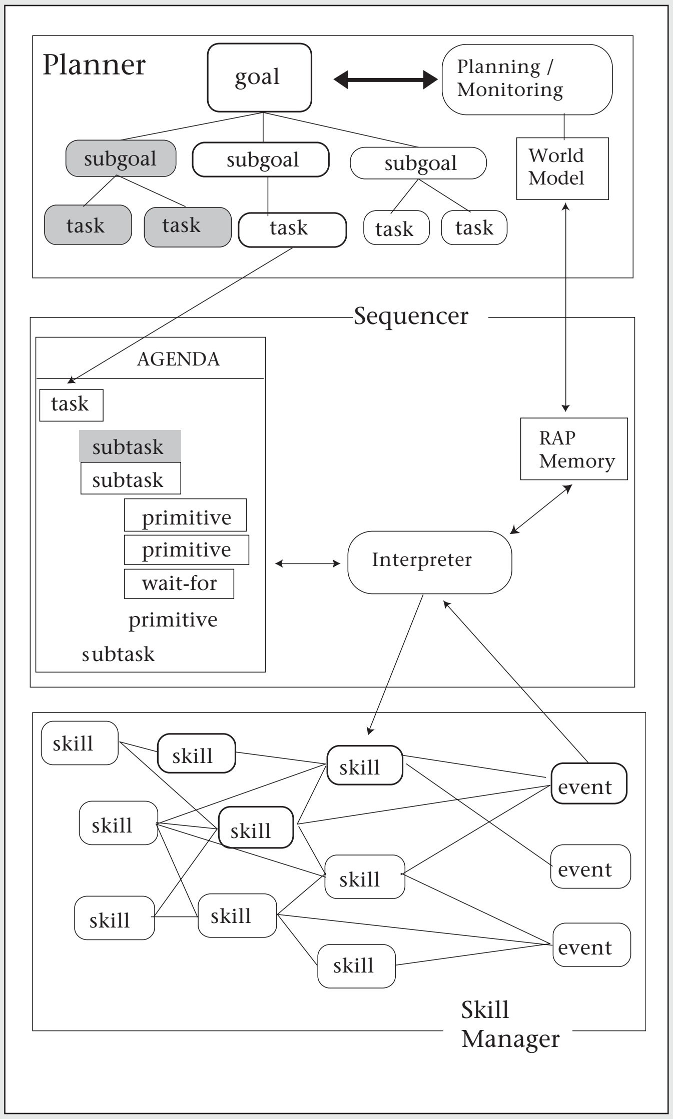


Figure A. The 3T AI Control Architecture.

The ALS control system uses the intelligent control software for autonomous systems known as 3T (Bonasso et al. 1997), which separates the general robot intelligence problem into three interacting tiers: (1) the planner, (2) the sequencer, and (3) the skill manager (figure A):

A set of robot-specific, situated skills (or behaviors) represent the architecture's connection with the world through the sensors and actuators. The term *situated skills* is intended to denote a capability that, if placed in the proper context, will achieve or maintain a particular state in the world. The 3T implementation includes primitive actions, queries, and monitoring events that can be combined to form autonomous behaviors. The 3T skill layer is a distributed set of skill groups coordinated by a skill manager for each ALS subsystem. For the IWRS system, control signals and sensor data for the skills are obtained from a suite of analog to digital (A/D) conversion cards in the controls rack (figure 6) colocated with the central processing units (CPUs) (we are using a VERSA-MODULEEUROCARD (VME) bus, with VXWORKS running on Power PCs).

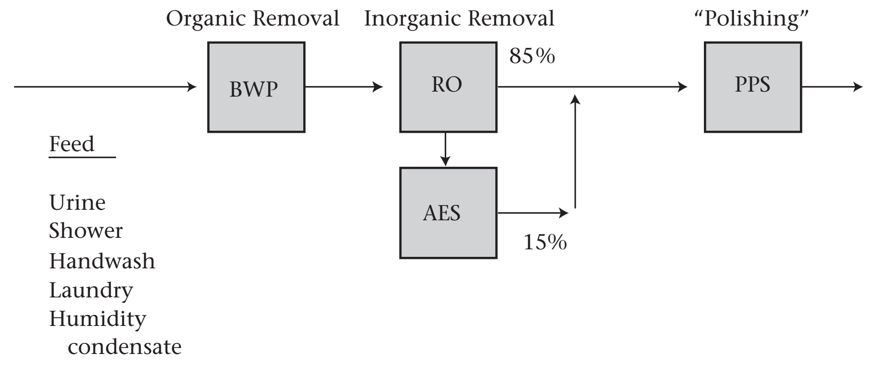
A sequencing capability can differentially activate the situated skills to direct changes in the state of the world and accomplish specific tasks. The 3T architecture uses the reactive action packages (RAPS) system (Firby 1999) for this portion of the architecture. The RAPS engine is an interpreter, indexing RAPs (essentially sets of linear plans) from a library based on the changing world situation. Thus, one can change a RAP or add new RAPs while the sequencer is executing.

A deliberative planning capability reasons in depth about goals, resources, and timing constraints. The 3T hierarchical task net planner known as AP (Elsaesser and Sanborn 1990) uses the highest-level RAPs as its primitive plan operators and can replan both spatially and temporally. The planner is efficient, but it becomes even more potent when its level of detail is abstracted to the RAPs of the sequencing layer below it. It is important to note that once the planner generates a plan, it executes the plan by placing primitive plan actions on the sequencer's agenda and monitoring the results of the sequencer's actions.

Communication among the layers and between skill managers uses the interprocess communication (IPC) message-passing protocol (Simmons and Dale 1997). With this communications infrastructure, data from any part of the system can be monitored by any other part of the system.

A key aspect of 3T is that it gives developers the ability to integrate the continuous, near–real-time control algorithms in the bottom layer with advanced AI algorithms in the top layer—that is, automated planners and schedulers—that are event driven but more computationally expensive. The 3T architecture provides this integration through the middle or sequencing layer. Essentially, the middle layer translates the goal states computed by a planning and scheduling system into a sequence of continuous activities carried out by the skills layer and interprets sensor information from the skills layer as events of interest to the upper layers.

The 3T applications run autonomously, in large part because of the principle of cognizant failure (Gat 1998) embodied in each level of the architecture. The skills level notifies the sequencer when it loses any of the states it must achieve; the sequencer uses alternative sequences when the primary methods fail, ultimately putting the control system in a safe state; and the planner can synthesize alternative plans in light of the failures of the lower two tiers.



		BWP Effluent	RO/AES Effluent	PPS Effluent
Ammonium	800 mg/L	300 mg/L	75 mg/L	<0.5 (n/a)mg/L
Total Carbons	600 mg/L	20 mg/L	3 mg/L	<0.5 (1.5) mg/L
Inorganics	10 mS	10 mS	0.3 mS	<0.01 (0.15)mS
pH	9	8	8	7
HCO3	0 mg/L	500 mg/L	50 mg/L	3 mg/L (n/a)

Figure B. The Water-Flow Paths and the Target Quality Values in Milligrams and Millisemens (an Indirect Measure of Water Quality) for Each Liter for the Integrated Water-Recovery System.

The numbers in parentheses for the postprocessing system effluent are those for typical residential tap water.

Advanced Water-Recovery System

The advanced Water-Recovery System (WRS) is a set of next-generation WRS components that promise to provide potable water using fewer consumables (filters, resins, and so on) and much less power than the components currently planned for use on the International Space Station (ISS) (figure B). Figures 2 and 3 show the four subsystems used in the IWRS test. The IWRS comprises (1) a biological water processor (BWP) to remove organic compounds and ammonia, (2) a reverse-osmosis subsystem to remove inorganic compounds from the effluent of the biological water processor, (3) an air evaporation system (AES) to recover additional water from the brine produced by reverse osmosis, and (4) a postprocessing system (PPS) to bring the water within potable limits.

The WRS planned for use on the ISS is a physical-chemical system that requires a yearly resupply of roughly 3000 pounds of consumables (filters, membranes, and so on). In contrast, the advanced WRS developed and tested at JSC is projected to require only 250 pounds of consumables a year and use 50 percent less power.

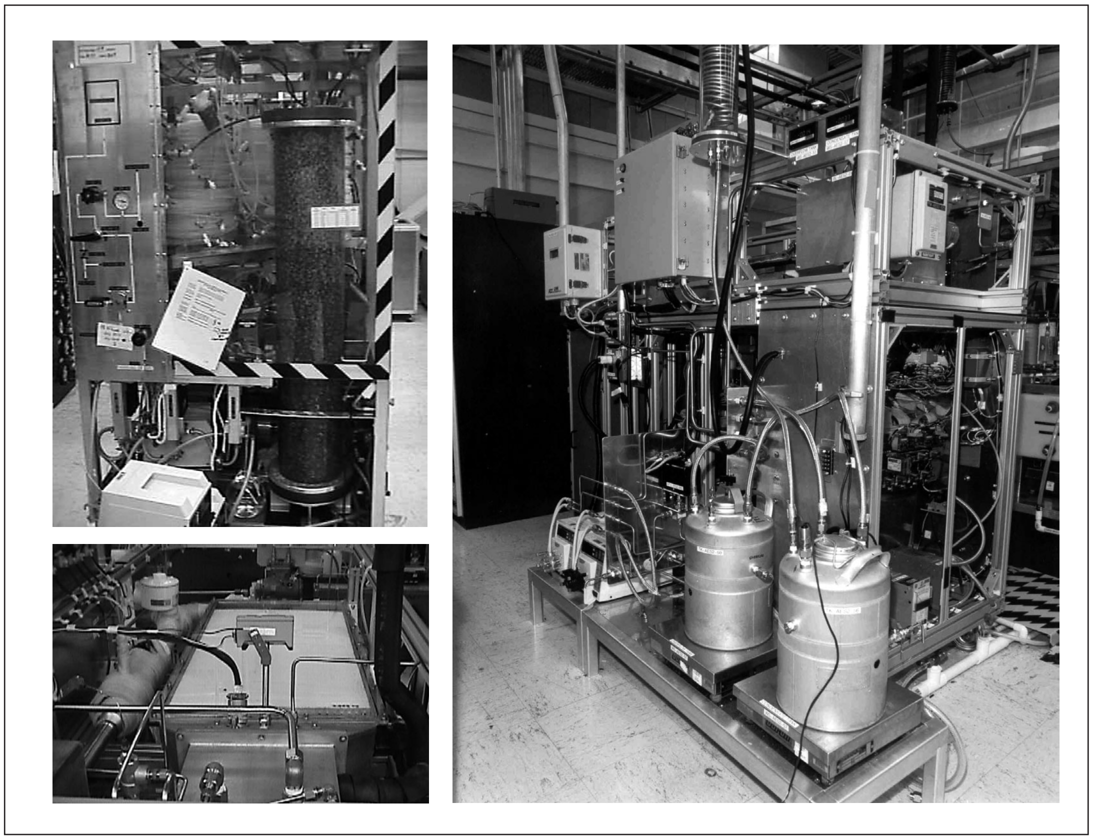


Figure 2. The Advanced Water-Recovery System Subsystems.

Upper left is the biological water processor. At right is the rack containing the reverse-osmosis subsystem in the rack bottom, the air evaporation subsystem (AES) at the top of the rack, and the postprocessing system in the rack's left rear. The lower left picture is a close up of the wick in the AES.

used a particular implementation of this architecture known as 3T (see sidebar 1) in a number of robot projects prior to 1995 (Bonasso et al. 1997), and because life support systems are a form of immobots (Williams and Nayak 1996), its application to ALS projects was straightforward.

In each of the previous efforts, the 3T team from AR&SD was required to interface the AI architecture to existing legacy software and hardware systems (Schreckenghost et al. 1998a). In 1999, however, we began to support advanced water-recovery projects that were being built from the ground up. As a charter member of the water research group in CTSD, the AR&SD AI team was influential in the selection of hardware components and the design of the overall control of these systems. For the first time, we were able to build the full 3T system from the A/D converter boards used by the sensors and devices to the top tier of the architecture.

In the summer of 1999, we had used the bottom two layers of 3T to provide autonomous control for a single subsystem—a second-generation biological water processor—during a 450-day, 24-hour-a-day, 7-day-a week test. Then in January 2000, the Advanced Water Research Group received ALS funding for the year-long IWRS test, involving four advanced water-recovery subsystems (Bonasso 2001) (see Advanced Water Recovery System sidebar).

Buildup

Using 3T allowed us to develop the control for the IWRS in a modular fashion in two ways. First, moving from bottom to top (figure 4), each layer has its own data structures, timing constraints, and development tools that allow for parallel development of the software. Thus, we were able to develop skills sets based on the evolving hardware specifications and simultaneously develop the sequencer procedures. Early on, as the water research team developed the design for each subsystem, one part of the 3T team wrote the sequencer procedures for each subsystem in the RAPS language (which, in turn, is written in Lisp)[1] using *virtual skills*, that is, Lisp skills connected to a Lisp simulation of the

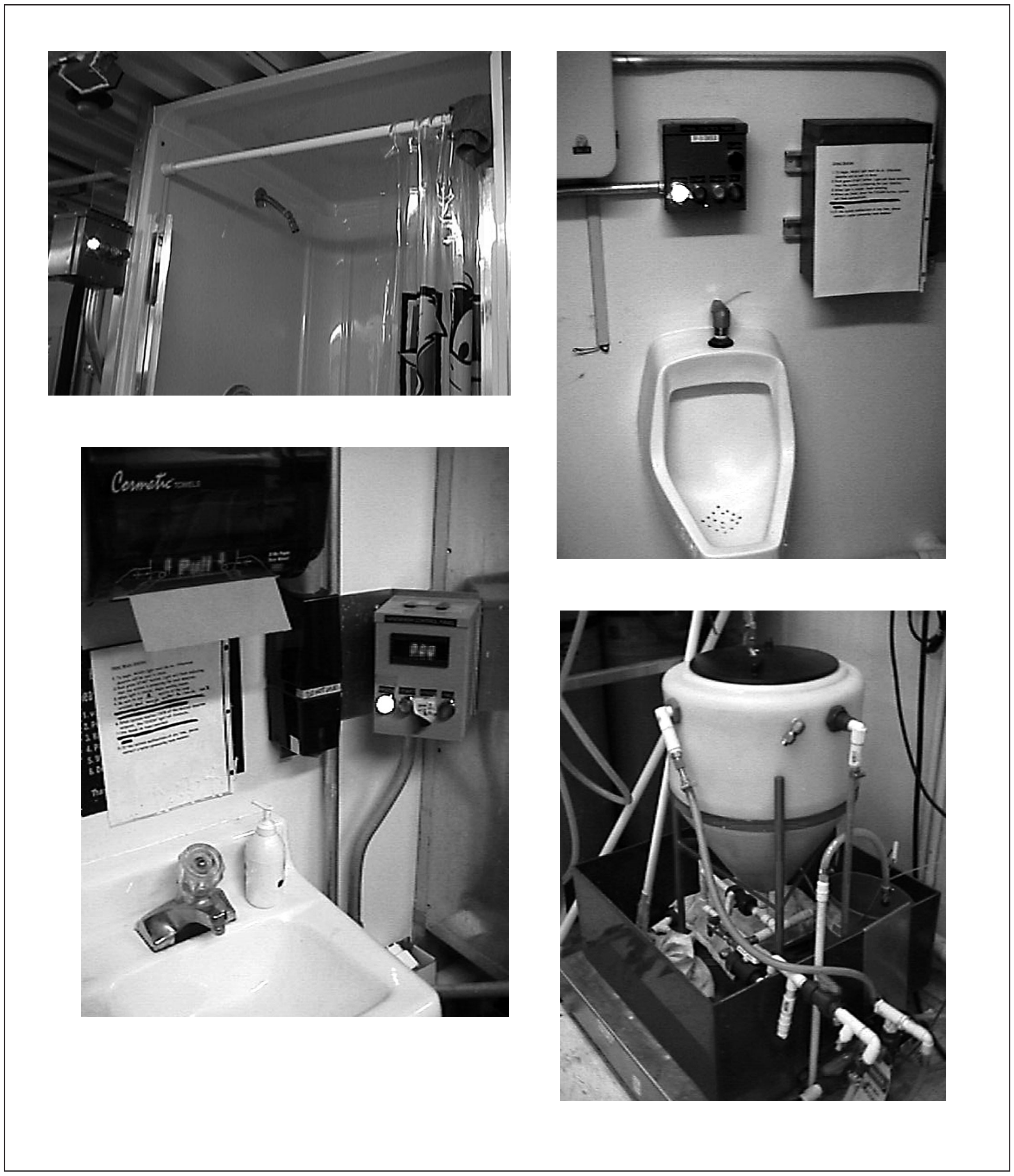


Figure 3. The Integrated Water-Recovery System Waste-Water Collection System.

Human volunteers donate urine, showers, and hand washes, using liquid soap with the chemical composition of that to be used on the space station. A computer system responds to the push buttons at each donation site to weigh and record each type of donation before sending the donation to the main feed tank for the INTEGRATED WATER-RECOVERY SYSTEM. Prepared solutions representing respiration water are added to the feed tank to complete a composition representative of that expected on the space station or planetary outposts.

expected hardware. A virtual simulation of, say, the reverse-osmosis subsystem could then be shown on a laptop to the WRS engineers and the control design refined in an iterative process even before the actual hardware was available. The primary result of this process was a set of skill specifications for each subsystem (figure 5).

As the hardware specifications became more firm (figure 6), another part of the 3T team wrote the skills for the subsystems in C on a VXWORKS rack in the AR&SD laboratories, using the skill specifications and testing them with rudimentary C simulations of the expected hardware. When the hardware for a given subsystem came online, the skills for this sub-

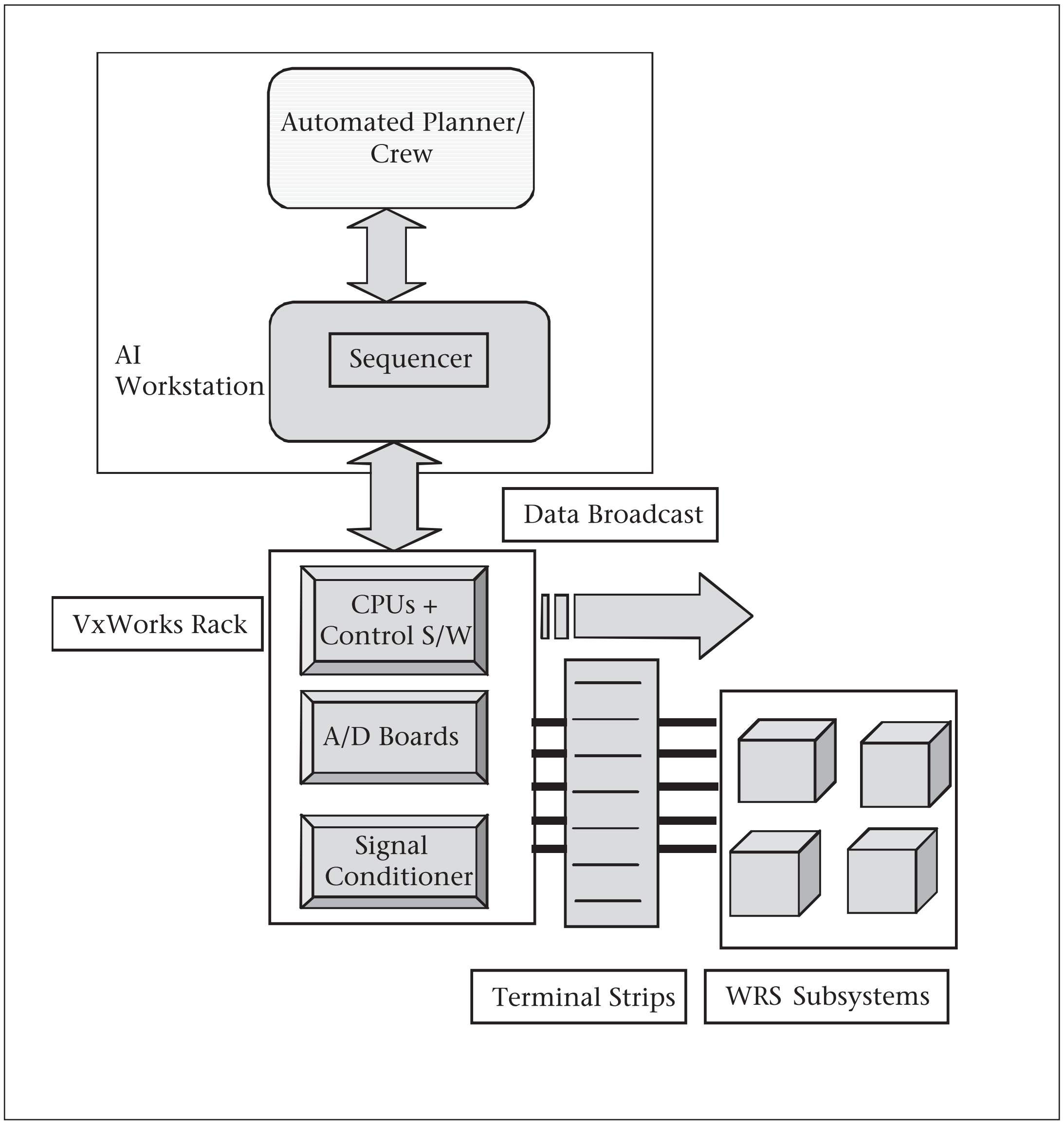


Figure 4. The 3T Implementation for the IWRS Test.

We developed one skill manager for each subsystem, which ran on its own CPU. Skill managers broadcasted data at specified intervals for use by extant clients for analysis and review. The sequencer level managed task control, and the top-level control was provided for the most part by the engineers running the test.

system were installed in the test rack in the water research laboratory. After testing the individual data channels, the skills developers used a skill-level command graphic user interface (GUI) to activate and deactivate individual skills. This development approach enabled the 3T team to deliver the low-level control for each subsystem within two weeks of the subsystem hardware installation.

Next, the sequencer procedures for the subsystem (RAPS) were installed on the AI workstation and tested with the validated skills. An example of the resulting RAPs is shown in figure 7. There are several methods associated with the processing-start RAP, each indexed by a context clause. The method shown is valid when the required reverse-osmosis stage is purged, and the pps-select valve is open to the postprocessing systems (PPSs). In this case, the RAP starts the reverse-osmosis main feed pump, stops the recirculation pump, and turns the pps-select valve to reject (relieving downstream pressure). Then, the RAP turns the reverse-osmosis process valve to the purge position and turns the pps-select valve to the tank position.

The skills level remained relatively stable once the sensors and actuators were in place.

```
------------------------------------
Skills -- for the RO agent
----------------------------------

Name      RO
Type      device
Params    interval
Outs      none
Function: A device skill that gets all the sensor values and provides them to
the other skills. Also sends commands to the pumps and valves. Also every
interval seconds, this skill broadcasts a data message with the values of all
the channels listed above to the IPC server so that clients (e.g., a logging
facility) can access them (see the IPC structure at the end of this document).

Name      valve_position
Type      query
Params    valve (process/pps_select)
Outs      value (for process:primary/secondary/purge/off/unknown;
          for pps_select:pps/tank/reject/off/unknown), and result (okay or Err)
Function: Checks V02 or V03. One of lines V02_i1 through  V02_i3 or V03_i1
through V03_i3 will be hi, and the rest will be low. If all are low, the result
is off. Any other pattern is unknown.

Name      valve_at
Type      event
Params    valve (process/pps_select), value (for
          process:primary/secondary/purge/off; for
          pps_select:pps/tank/reject/off)
Outs      result (okay/ERR)
Function: Waits for V02_i1 through V02_i3 or V03_i1 through V03_i3 to indicate
value (see the valve_position skill). When the condition is achieved the event
returns result.

Name      turn_valve
Type      block
Params    valve (process/pps_select), value (for
          process:primary/secondary/purge/off;
          for pps_select:pps/tank/reject/off)
Outs      none
Function: Sets one of V02_o1 through V02_03 to hi the rest to low, except for
off when all lines will be set lo.
```

Figure 5. Excerpts from the Reverse-Osmosis Skill Specification.

We repeated the process for each subsystem and then developed and tested additional sequences to integrate the subsystems. The total initial software development took on the order of four and a half months, using roughly one month for each subsystem and two weeks for integration testing.

The second manner in which the modularity of the 3T system sped our development is that the architecture allows the independent development and testing of groups of ALS subsystems and a subsequent incremental integration of these subsystems (figure 8). This aspect of the control development became important for the WRS team in dealing with the startup time of the BWP. The microbes in the BWP take one to two months to form viable colonies to process feed water. This inoculation period meant that bringing the other subsystems into test would be delayed by at least one to two months, and even longer if the inoculations became problematic.

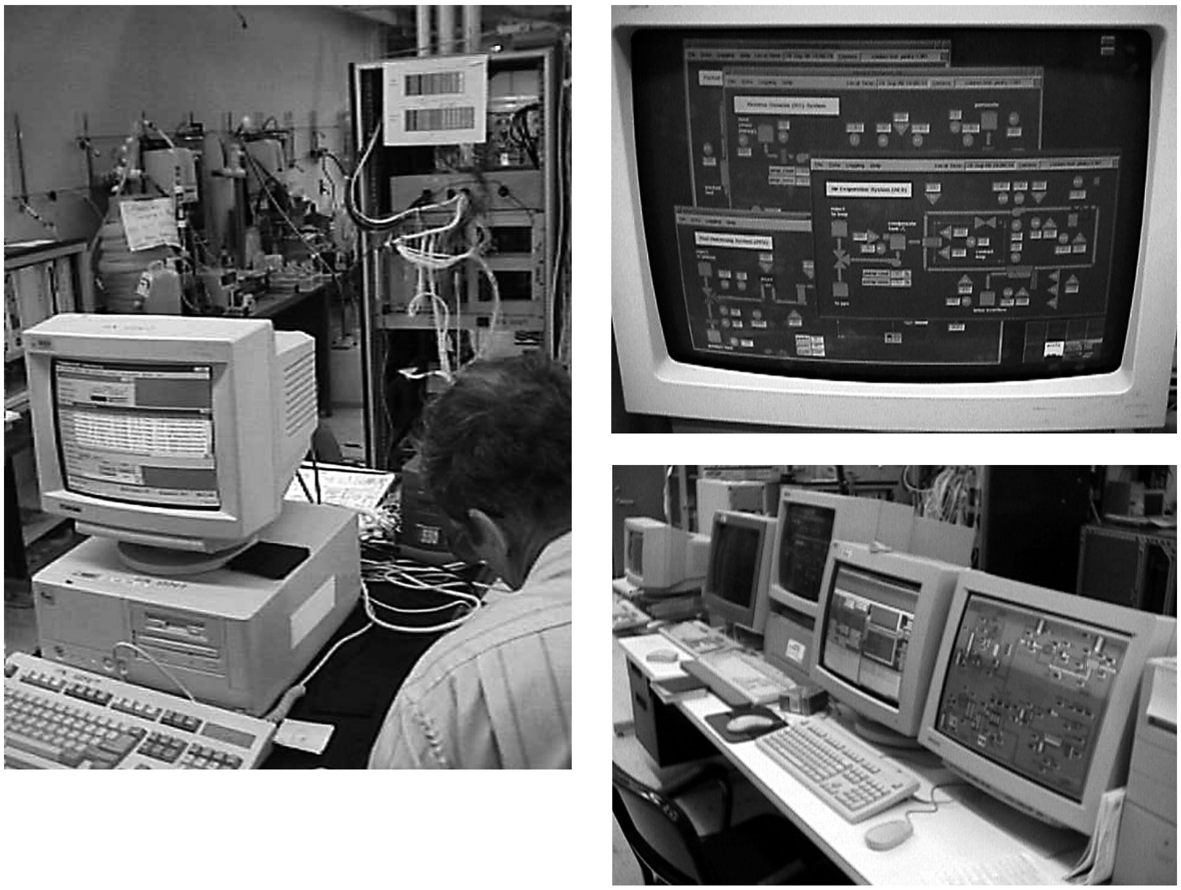


Figure 6. 3T Control Computers for the INTEGRATED WATER-RECOVERY SYSTEM.

On the left is a view of the 3T VERSAModuleEurocard (VME) rack behind the computer that is used as a secondary interface to the reverse-osmosis high-pressure pump. On the right is a view of the (unattended) 3T control table. From foreground to back, the displays are two sequencer-planner displays, the IPC–skill manager display, the display of the broadcast server, and a display associated with the high-pressure pump used in the reverse-osmosis subsystem. In the upper right is the display showing the graphic user interfaces (GUIs) for each subsystem generated by the data broadcast from each skill manager.

To give the water team more breathing room, the 3T group suggested that the water team divide the official start of the test into two components: (1) the BWP and the reverse osmosis and (2) the reverse osmosis and the other two subsystems (AES and PPS). In the iWRS system, the pivotal subsystem is the reverse osmosis. This system receives BWP effluent, processes it, and provides product water for the two downstream systems. In effect, the BWP is independent of the downstream systems, so it could conceivably be started early when the downstream systems were still being built. Because of the modularity of 3T, the initial iWRS could consist of the first two subsystems, with the output going to drain while the inoculation proceeded, and the second iWRS could include all four subsystems. In this manner, the water team started the test with only the first two subsystems in April 2000 and brought the other two systems online in December 2000 in time to make a January 2001 full start.

Controlling the Integrated Water-Recovery System

In this section, we describe the control tasks for the final IWRS system that went into test in

```
(define-primitive-event (valve-at ?agent ?valve ?open-closed ?error)
  (event-definition (:valve_at (:valve . ?valve) (:value . ?open-closed)))
  (event-values :bound :bound :bound :unbound))

(define-rap (turn-valve-p ?agent ?valve ?open-closed ?timeout)

  (succeed (and (valve-position ?agent ?valve ?value ?error)
                (= ?value ?open-closed)))
  (timeout ?timeout)
  (method
    (primitive
     (enable (:turn_valve (:valve . ?valve) (:value . ?open-closed))
             (wait-for (valve-at ?agent ?valve ?open-closed ?result)
                 :succeed (?result))
     (disable :above)
     ))
  )

(define-rap (processing-start ?stage ?adjust-time)

...

(method purge
          (context (and (= ?stage purge)
                        (valve-position roskm pps_select ?old-pos ?error)
                        (= ?old-pos pps)
                        (nominal-pump-speed roskm feed ?wwsp)
                        (default-timeout ?dto)))
    (task-net
     (sequence
      (t1 (syringe-pump-p roskm start feed ?wwsp 30))
      (t2 (water-flowing-p roskm stop recirc 0 ?dto))
      (t3 (turn-valve-p roskm pps_select reject ?dto))
      (t4 (turn-valve-p roskm process purge ?dto))
      (t5 (turn-valve-p roskm pps_select tank ?dto)))))
```

Figure 7. A Primitive Event, a Primitive Reaction Action Package (RAP), and a High-Level RAP That Use the Skills from the Skill Specification Shown in Figure 5.

The event definition and the primitive enable clause invoke the C-code skills, whose name and arguments are delineated by colons. The primitive RAP succeeds when the valve position matches the commanded value.

2001, first by subsystem and then for the IWRS as a whole (figure 8).

The Biological Water Processor

Feed water from the waste-water collection system (figure 3) first passes through the BWP. The main control task for the BWP is to keep the water in the gas-liquid-separator (GLS) (see the lozenge icon with the three-level switches in the upper left of figure 8) at mid-level, which is accomplished by varying the speed of the feed pump, and the draw from the reverse-osmosis main pump remains constant. The other requirement is to monitor the pressures in the recycle loop, as well as in the nitrifier tubes, and carry out automatic shutdown procedures in the case of off-nominal values. For example, if one of the nitrifier tubes shows too high a pressure, the water and air pumps associated with this tube are shut down, and a warning is issued.

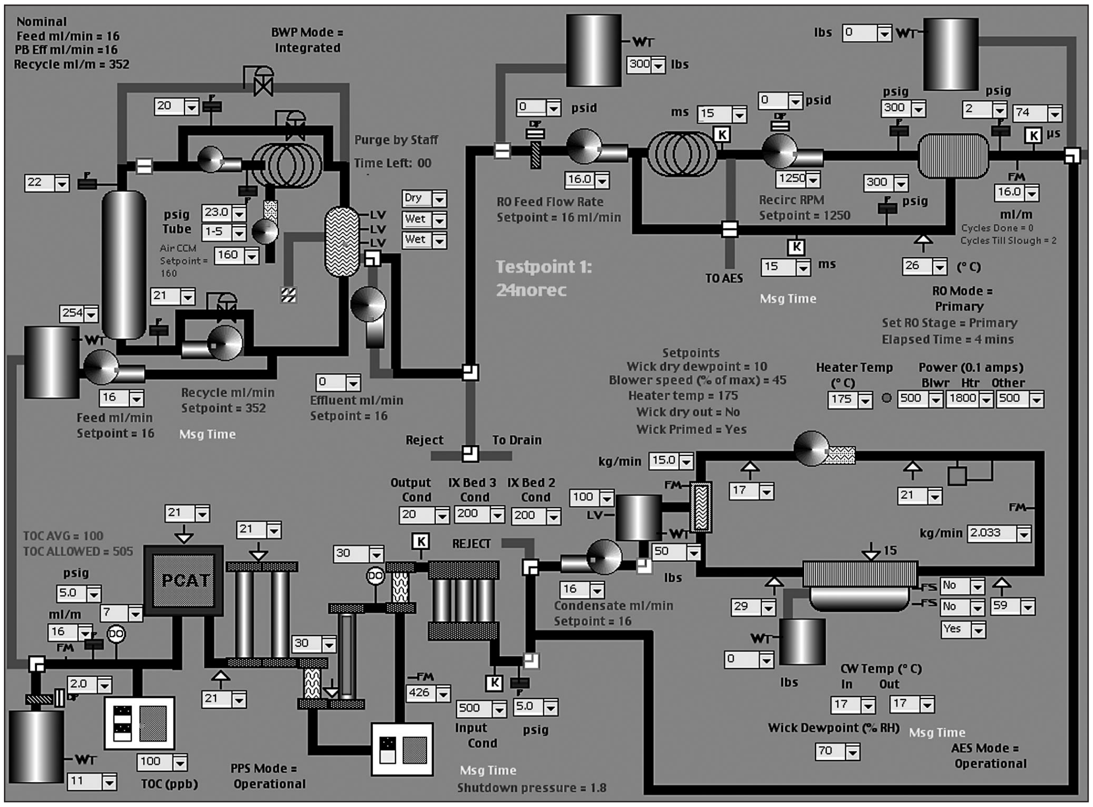


Figure 8. A Schematic Display of the INTEGRATED WATER-RECOVERY SYSTEM *as Seen by 3T.*

The biological water processor is in the upper left, the reverse-osmosis subsystem in the upper right, the air evaporation system in the lower right, and the postprocessing system in the lower left. Gray lines indicate flow pipes; black lines indicate pipes with water or air currently flowing. Small boxes with lines at junctures indicate motorized valves.

The Reverse-Osmosis Subsystem

The reverse osmosis (upper right portion of figure 8) is the lynchpin subsystem because it pulls water from the GLS of the BWP and delivers its output water, called *permeate,* to the PPS and brine to the AES. It removes inorganic compounds by pushing the input water at high pressure through tubular membranes that act like molecular sieves. The reverse-osmosis subsystem must go through as many as four distinct phases in each cycle. The primary phase draws water into a coiled section of pipe that acts like a reservoir while it processes permeate in the outer loop of pipes. In the secondary phase, the rejected water is concentrated into brine in the inner loop of pipes. The usual third phase is to purge the brine to the AES. However, periodically the membrane needs to be cleaned of particulates that collect on its surface by running the water counterclockwise in the inner loop during what is known as the *slough phase*.

Additionally, there are a number of ASDs associated with back pressure on the membranes, permeate conductivity, and loss of pressure in the recirculation loops.

The Air Evaporation System

The AES wick absorbs reverse-osmosis brine as it fills the AES reservoir (lower right of figure 8) during the reverse-osmosis purge cycle. During operation, hot air blows across the wick, taking up evaporated water and leaving solid waste on the wick. The moisture-laden air then passes through a heat exchanger where water is condensed into an output tank. The AES processes the brine in batches. When the brine fills the reservoir to the second-level switch, the AES starts up, processing the brine until the lowest switch reads dry, at which point the AES goes into standby, awaiting another load. ASDs concern overheating and loss of coolant fluid in the heat exchanger.

Additionally, the AES pumps condensate to the PPSs when the condensate tank reaches a certain level or when the reverse-osmosis subsystem is not sending its condensate to the PPS (to keep a steady flow of water to the PPS). When the wick is spent, as indicated by the conductivity of the condensate, the AES engineers initiate a dry-out procedure prior to replacing it. A typical wick lasts 45 days.

The Postprocessing System (PPS)

The PPS (lower left of figure 10) "polishes" the water from the reverse-osmosis subsystem and the AES by removing trace inorganic material by way of ion-exchange beds and trace organics by oxidizing them with ultraviolet (UV) radiation. The PPS controls monitor the input water pressure. When the pressure goes above a threshold indicating water flow from either the AES or the reverse osmosis, the O2 concentrator is started, and a number of UV lamps are turned on commensurate with the measured total organic carbons (TOCs). When the pressure falls below the threshold, the concentrator and lamps are turned off. An average TOC is calculated based on the instantaneous TOC and the water accumulated in the product tank to determine whether the PPS output should be rejected to the BWP feed tank. ASDs concern overheating of the lamps and high output conductivity, indicating that the resin in the ion-exchange beds has been used up.

Integrated Control

The modularity of the hardware systems is such that these subsystems are considered four loosely coupled agents, which mainly react to their input and water quality and only rarely respond to the operation of the other subsystems. One such response is when the AES pumps condensate to the PPS in the reverse osmosis's stead, as previously discussed. Another response is when the reverse osmosis monitors the level of the GLS in the BWP to ensure that there are sufficient resources for it to draw on.

Additionally, PPS pressure changes are corroborated by sensing the state of the pumps and the valve configurations of the reverse osmosis and the AES. For example, if the inlet pressure is not high enough to indicate water flow, but the AES or reverse-osmosis pump speeds and valve configurations indicate water is flowing, then the PPS will begin operations.

Finally, when there is a complete ion exchange bed breakthrough in the PPS, the reverse osmosis and the AES sense the high PPS conductivity and recycle their effluent to the BWP feed tank.

The Test

The IWRS test consisted of a series of test points, each representing a different configuration, and each slated to last until the IWRS product water could no longer be maintained at the required potable standard (see the target quality values in the Advanced Water-Recovery System sidebar). This nonpotable end point occurred when the quality of the water from the last of the three ion-exchange beds rose above a predefined level of TOC concentration. The test configurations were 2-person, 24-hour operation; 2-person, 24-hour operation with condensate rejected to the feed tank to reduce the loading on the ion-exchange beds; 4-person, 24-hour operation with condensate reject; 2-person, 18-hour operation (allowing 6 hours for maintenance); and 2-person, 18-hour operation with condensate reject.

Each test point called for either different flow rates or full or partial reject of internal flows or both. Besides rejecting AES condensate, 4-person or 18-hour operations required the reverse-osmosis subsystem and BWP to process water at an increased rate, with some of the reverse-osmosis permeate being returned to the BWP feed tank during the highest-conductivity periods in the cycle.

The test team began the first test point in January 2001. Soon they found that the ion-exchange beds were performing so well that instead of 30 to 40 days, a test point might take 3 months. To reduce the length of the overall test to a manageable level, the water team resized the ion-exchange beds to one-third of their original size and then restarted the test beginning with the first test point in March 2001.

On 25 December, the third ion-exchange bed "broke through" for the last test point, marking the end of the test proper. From January through mid-April 2002, the team maintained the IWRS running in the first test-point configuration to support an unrelated antibiotic study by Texas Technical University.

Software Engineering Lessons

A fundamental purpose of an article such as this one is the recounting of lessons learned from the experience. Some of these lessons fall into the category of software engineering of slow-running, long-duration systems. The 3T system was designed for the intelligent control of autonomous robots, fast-running robots that never had to function for longer than a few hours at a time. Although we have been successful in using 3T to control ALS systems that have much longer response times and operate continuously, our experience in applying this architecture to these systems and to the WRS in particular has given us insights into key characteristics of long-duration systems and the implications of these characteristics for intelligent control of similar systems in the future.

Advantages of Reaction Action Packages and Lisp

That RAPS uses a plan interpreter and that the top two layers of 3T are written in Lisp allowed us to make changes in subsystem operation on the fly. In addition to changing set points and warning levels interactively, RAPs could be modified while the subsystems were in operation. RAPs are stored in a plan library, and instances are created and put on the task agenda as other tasks are removed. Thus, we could store modified RAPs in the library, which would then be picked up the next time the RAPs processing called for them.

Often, new RAPs were required that were unanticipated at the beginning of the test, or existing RAPs had to be modified as we learned more about the actual online system performance. New or modified RAPs were tested with virtual skills in the AR&SD laboratories and then installed in the running system in the water laboratory.

An example of a new RAP is the TOC calculation RAP described in the Advantages of Distributed, Layered Control section later. An example of modifying a RAP concerned the operation of the AES condensate pump. To maintain constant operation of the PPS for as long as possible, the AES condensate was pumped to the PPS whenever the reverse-osmosis subsystem was not sending its effluent to the PPS, for example, when the reverse-osmosis system is in purge mode. Over the course of the test, this simple control scheme was expanded to include sending condensate to the PPS whenever the tank was full to prevent overflow, inhibiting condensate flow whenever the PPS output conductivity was too high and modifying the full condensate pumping scheme whenever the test point called for rejecting the AES output to the feed tank.

Of course the incremental nature of the Lisp compiler allowed us to change other code without stopping operations. A representative example of this kind of on-the-fly code changing involved the central WRS data display. Frequently, in the early months of the test, the test engineers would desire additional information to be shown on the main WRS monitor. Examples of additional data output not called for in the original design include the reverse-osmosis stage elapsed time (upper right of figure 8) and the allowed and average TOC (lower left of figure 8). Because the entire interface was written in Lisp (we used Macintosh Common Lisp [Digitool 1996] running on a Power Mac G4), a control engineer could build, debug, and install such changes to the displays online without disturbing the main control code.

After the first month, the control code was placed under configuration control, so the types of changes described earlier were always discussed with the water team in a weekly tag-up meeting before being implemented. Nonetheless, once the changes were approved, the water team appreciated the rapidity with which they were implemented.

Managing Systems with Long Response Times

A key aspect of ALS systems is the slow event times associated with them. In our WRS system, turning a valve took 3 or 4 seconds, the PPS oxygen concentrator took a minute and a half to come up to speed and several minutes to turn off, and the AES heaters took 5 minutes to heat the air circulating in the AES and then upwards of 10 minutes to cool down. The key control insight here is that the final responses were more important than the initiating events. Thus, instead of building procedures as sequences of enable and wait-for clauses (as was normally the case with our robot systems), we wrote procedures as a sequence of activation steps followed by the initiation of a series of separate monitoring procedures that waited for the final responses and took appropriate action if they failed. When one of these long-term events—for example, waiting for the oxygen concentrator to become operational—failed, it was often because over the length of the test, the device was just taking several seconds longer to activate. With our coding approach, however, we only needed to adjust the time-out parameters in the monitoring procedures (online in real time—see Advantages of Reaction Action Packages and Lisp).

Related to the long activation response times is the fact that the WRS system-level events occurred on the order of hours or days. To find out if a system-level change was having the desired effect, we often waited for days or weeks. An example of dealing with long activation response times had to do with determining the optimum number of reverse-osmosis cycles before having the controls perform a membrane slough. A reverse-osmosis cycle typically completed every four and half hours. At the beginning of the test, the system was directed to slough the membranes every 8 cycles, or every 36 hours of processing. As the test continued, the quality of the reverse-osmosis permeate tended to be worse toward the end of the last two cycles, suggesting that a slough was required more often. It took the team over a week and a half of experimenting to determine that the number of cycles to slough should be set to four to keep the permeate quality consistently high. Additionally, a new cycles-to-slough value had to be determined as often as every month, either because a new membrane was used or because the amounts of its permeate, being recycled back to the feed tank varied with different test points.

To help engineers manage these types of problems, we needed to "catch" and log key events whose duration was one or two orders of magnitude smaller than the rest of the system events. For example, in determining the optimal number of cycles between sloughs, the reverse-osmosis engineer needed to correlate the permeate water quality with a count of how many sloughs had taken place. A slough took no more than 4 minutes to execute, but although the skills broadcast the data every 15 seconds, the logging rate for data used for analysis was every 5 minutes (of the WRS system events, only the reverse-osmosis slough event took less than 5 minutes to occur). Thus, the slough indicator—the reverse-osmosis recirculation pump running in reverse at one third the normal revolutions per minute (rpm)—was often missing from the logs; so, we built an event detector into the reverse-osmosis GUI that detected the slough indicator using the 15-second data and cached it as a yes or no flag in the 5-minute log.

Simple Systems Become Complex When Integrated

With the possible exception of the BWP, we have found that individual ALS subsystems are relatively straightforward to control. They normally require a startup procedure, several actuator check monitors (such as one to ensure that the reverse-osmosis recirculation pump doesn't start before the feed pump), an ASD monitor, and a shutdown and/or standby procedure. When several subsystems are integrated, however, the complexity increases, and the need for look-ahead reasoning, such as the crop rotation scheduler for the 91-day human test discussed earlier, becomes evident.

Without a sufficient domain theory for WRS, we could not bring generative planning techniques to bear, but our loosely coupled agent approach allowed us to isolate the increased complexity to the interagent interfaces. This isolation gave rise to more complex code to handle the larger number of contexts (system states) that arose from the interfaces. For example, the procedure that managed the level in the AES condensate pump discussed earlier required only two methods (the number of methods roughly equates to the number of system states of concern for the procedure). However, integrating the AES with the rest of the WRS required an additional five methods and a rewrite of the original two.

We were successful in this endeavor because the RAPs system forces a modular coding approach (multiple contextual methods to achieve a goal) and because Lisp allowed us to both bring the new methods online and to test the old ones while the control system was operating.

Long-Duration Systems Have Their Own Problems

By their very nature, ALS systems are long running, carrying out their prescribed processing for weeks or months. When anomalies occur, they are rare but must be detected and processed to prevent often catastrophic results. In developing and maintaining the IWRS 3T system, we have come to understand several control implications of this long-duration characteristic of ALS systems.

Equipment Degradation During the 12 months of IWRS operation, we witnessed the slow degradation of pressure transducers, flow meters, a dew-point sensor, the AES blower, and the main reverse-osmosis feed pump. These were not failures as such but a slow deterioration of performance to the point that the equipment could not perform its function. Sometimes the ultimate failure brings the test to a halt, such as in the case of the reverse-osmosis feed pump. With the other equipment, the degradation is gradual and difficult to detect because the symptoms are often intermittent. The point is that often the degradation takes place over several months, and neither the water team nor the controls engineers had the experience to determine if the various problems stemmed from software or hardware. We had few utilities in place to help capture the intermittent events and spent much time in each instance adding trace code and studying the results. After about six months, we became familiar with the character of each of the subsystems and were able to more easily ascertain the cause of these types of anomaly. In the future, use of specialized event-detection algorithms will be required (see Spin-Off Benefits for the AI Community).

Autonomy versus Hardware Besides the deterioration of WRS hardware, we had to replace all but one computer used in the control system as well as the power supply in one of the VME racks. Disk failure and memory problems were easy to detect and repair, but the power-supply problem taught us a fundamental rule about user acceptance of automation in long-duration systems: The automation must last longer than the hardware. What we mean here is best described by the situation surrounding the loss of the power supply.

The microbes in the BWP could not go longer than five or six hours without being "fed," that is, having feed water circulating around the colonies. The power supply to the rack controlling the BWP began to fail during the BWP–reverse-osmosis phase of testing in the spring of 2000. Early on, the only indication of a problem was that the rack CPUs would reset, zeroing the pump speeds, thus halting feed water to the BWP. When this reset happened after the last human check at 11 PM, the water laboratory personnel would arrive the next morning to find the colonies destroyed. The first failure required a two-week reinoculation of the BWP; as a result, the WRS manager assigned humans to monitor the control system around the clock. It was not clear why the CPUs had reset, and once the software was restarted the system ran for days before another reset occurred.

After experiencing more frequent resets over a weekend, the water team decided to take both subsystems "off controls" and run them manually; that is, all actuators were run open loop. The team decided that the chance of a BWP or reverse-osmosis hardware failure was far less likely than a catastrophic control failure. Even after the control team replaced the power supply—which is still operational as of this writing—the water team did not put the subsystems back "on controls" for two weeks and did not cancel the around-the-clock personnel shifts until the control system "caught" a hardware failure—a failed BWP nitrifier pump—two weeks later.

Memory Leaks Most software developers delivering an application will write their code carefully enough to make efficient use of memory resources. However, there can be inefficiencies in the resulting code that will not appear with the normal amount of debug testing. Such inefficiencies have a cumulative effect and make themselves felt only after weeks of operation. We discovered over the course of the first several months of operation that all the software we developed and installed in the water laboratory "leaked" memory; that is, the code was using small amounts of memory resources daily without releasing these resources. Memory leaks were discovered in the skill managers, the IPC clients, and the sequencer. The lesson here is that wherever possible, the code should be run for several days (as opposed to several hours as is usually the case with robot systems) with memory metering before delivery.

Safety Shutdowns No matter the number of precautions taken to prevent system failure, there was always a set of variables outside our control. Some of these can be corrected by smart shutdown schemes; we discuss an example in this subsection. Others cannot be prevented, but check-pointing states can speed up the recovery process; we discuss an example of this type of problem in the next subsection.

Every six weeks or so, over the course of a 12-month test, we experienced random faulty data packets. These would produce a data set that would cause the sequencer to break and thus stop reading IPC messages. This event inevitably occurred after the last check by the control engineers (typically about 11 PM) and before the laboratory personnel arrived in the water lab six hours later. With the sequencer down, messages sent to the sequencer would build up in the IPC server, and after about an hour, the server would crash, bringing down all clients connected to it, including the logging GUIs and the skill managers.

When the skill managers died, they left the last settings for the pumps and valves on the A/D boards. In one instance, the failure occurred while the condensate pump was on; in another, the failure occurred when the BWP controls were in the middle of adjusting the GLS level, and the feed pump was running lower than usual. In the former case, the condensate pump pumped the tank dry and started pushing air into the ion-exchange beds of the PPS, requiring a shutdown and a manual repacking of the beds. In the latter case, the GLS was pumped dry by the reverse-osmosis action, and the reverse osmosis drew air from the BWP GLS at high pressure into its membranes, rendering them useless.

Our solution to these network failures was to make the skill managers aware of the loss of communications with the sequencer and execute procedures for putting their respective subsystem in a safe state. We developed the idea of a watchdog timer in each skill manager. If the elapsed time since the last sequencer communication was greater than a predetermined time (we used five minutes), the skill manager would put its subsystem in a protected state; for example, the AES skill manager would turn off its heaters and the condensate pump, and the BWP would reconfigure itself to operate in a recycle mode, with both the feed and effluent pumps off. We also made the loss-of-RAPs communications an IPC message broadcast by the BWP skill manager to the user GUIs (see Unattended Operations: Supporting

Intermittent Monitoring with a Data Management and Distribution System). The watchdog timers, instituted soon after the restart of the first test point, protected the IWRS from network failures through-out the remainder of the test.

Logging-State Memory We experienced loss of power to the water facility five or six times during the course of the test. In these instances, the valves would remain in their last commanded state, but all pumps would be turned off. The primary dilatory effect was the loss of the bacteria colonies in the BWP if the feed water was not restored to the BWP in a timely manner. Because the staff members had learned to resuscitate the bacteria after the worst-case time lapse (a power failure after the last check at 11 PM with the lab staff members not discovering it until 5 AM), a power failure posed little problem.

However, with the loss of power, as well as the numerous times the IWRS had to be halted because of hardware failures—about 25 times during the course of the test—it became important to be able to restart the system quickly, without searching the logs manually to determine the state the system was in before the malfunction. Thus, we wrote a RAP to log the internal state of the IWRS every 30 seconds. When the staff members restarted the system, the sequencer would read in the last state of each subsystem and then determine how to resume operations. This determination was possible because (1) each primitive checks the condition of the valve or pump before com-manding the device and (2) the RAP executive will skip steps in a procedure that have been obviated by outside or serendipitous events. To restart the reverse-osmosis subsytem, for example, the sequencer might determine from the logged state file that reverse osmosis was last stopped 20 minutes into its secondary phase, set the valve configurations for secondary if they were not left in that state, check to see that all the pumps are on for secondary operation, advance the phase timer to +20, and resume monitoring secondary-phase processing.

Lessons Learned about Intelligent, Autonomous Control

The other more important category of lessons learned is that of using autonomous AI systems in an arena that heretofore used little automation and no AI.

We consider the use of the 3T control system a resounding success of applied AI. The resulting software ran unattended for 98.75 percent of the test period (6684 of 6768 hours), averaging on the order of only 6 hours down time a month. In an environment where the experimental hardware is being tested, this achievement is especially notable and is directly attributable to the distributed, layered approach of our control design. Additionally, it is important to understand that although autonomy was the order of the day, there was a need to adjust this autonomy at various times during the test. Finally, on a more fundamental level, the shift from human managed control to full autonomy had both cultural and practical implications. The role of the human in managing these systems changed from that of vigilant hands-on managers at the control site to part-time supervisors at remote sites, reducing the human support from eight-hour shifts of as many as five persons each to a few key staff members who would intermittently check the system every six hours. However, the change in the humans' role generated several new requirements to support the staff members in their data management and analysis, giving rise to the need for a distributed data management system.

Advantages of Distributed, Layered Control

Calibrating instruments is an example situation of how our layered design limited system down time. For each sensor and variable command output, for example, pressure or pump speed, the skills used a linear equation to convert the A/D counts to the appropriate device value, for example, pounds per square inch (psi) or rpm. Over time, the instrument output drifted from these calibrated values and had to be recalibrated, resulting in a new equation to be coded in the device skill, which then had to be recompiled. Because the instruments were grouped by subsystem, we only had to interrupt the given subsystem to restart the newly compiled skill and then only for the few seconds required for the skills to reconnect to the IPC server.

Another situation that exploited the distributed nature of our design concerned the shutdown of a given subsystem, for example, when the reverse-osmosis subsystem experienced a high-pressure event, triggering an ASD. With the reverse-osmosis subsystem down, there was no effluent being sent to the PPS and no brine being produced for the AES to process. As described in the control section earlier, whenever the reverse-osmosis subsystem is not providing water to the PPS, the AES would send its condensate to the PPS. Eventually, though, the condensate tank would empty, and the AES would stop sending water to the PPS. Without input water, the PPS would put itself in standby mode; without brine to process, the AES also put itself in standby mode. Finally, without the reverse osmosis drawing water from the BWP, the level in the GLS of the BWP began to rise, causing the feed pump to slow down to compensate. This compensation continued until the feed pump was at 0 rpm, effectively putting the BWP in standby mode. Thus, all the subsystems responded to the failed reverse osmosis by eventually achieving a standby mode of operation.

A similar situation took place when a subsystem was taken offline by the staff, such as when the AES wick was being changed. Each subsystem could be informed through the user interface about the availability of the upstream and downstream subsystems and would reconfigure itself accordingly. For example, if the AES were down, the reverse-osmosis brine would be directed to an overflow tank, which would subsequently be pumped back to the AES reservoir when the AES was operational. If the PPS were down, the AES and the reverse-osmosis subsystem would redirect their effluent back to the feed tank or the drain, depending on the needs of the test.

An advantage of the layered nature of 3T was that it allowed us to shift computations between layers as the need arose. An example of this shifting concerns TOC calculations. The average and allowed TOC are computed based on the TOC for increments of water volume deposited in the product tank, integrated over time. They require both a measure of the instantaneous TOC reading, obtained from the TOC analyzer in the PPS (the leftmost white box icon in the lower left of figure 8), and the volume of water in the product tank, measured by a weight scale in the PPS. Whenever the quality of the water from either the reverse-osmosis subsystem or the AES was low enough to trigger a high-instantaneous TOC value, the PPS product water was redirected to the feed tank until such time as the quality dropped below this threshold. During this time, because no water was being deposited to the product tank, the average and allowed TOC were not updated.

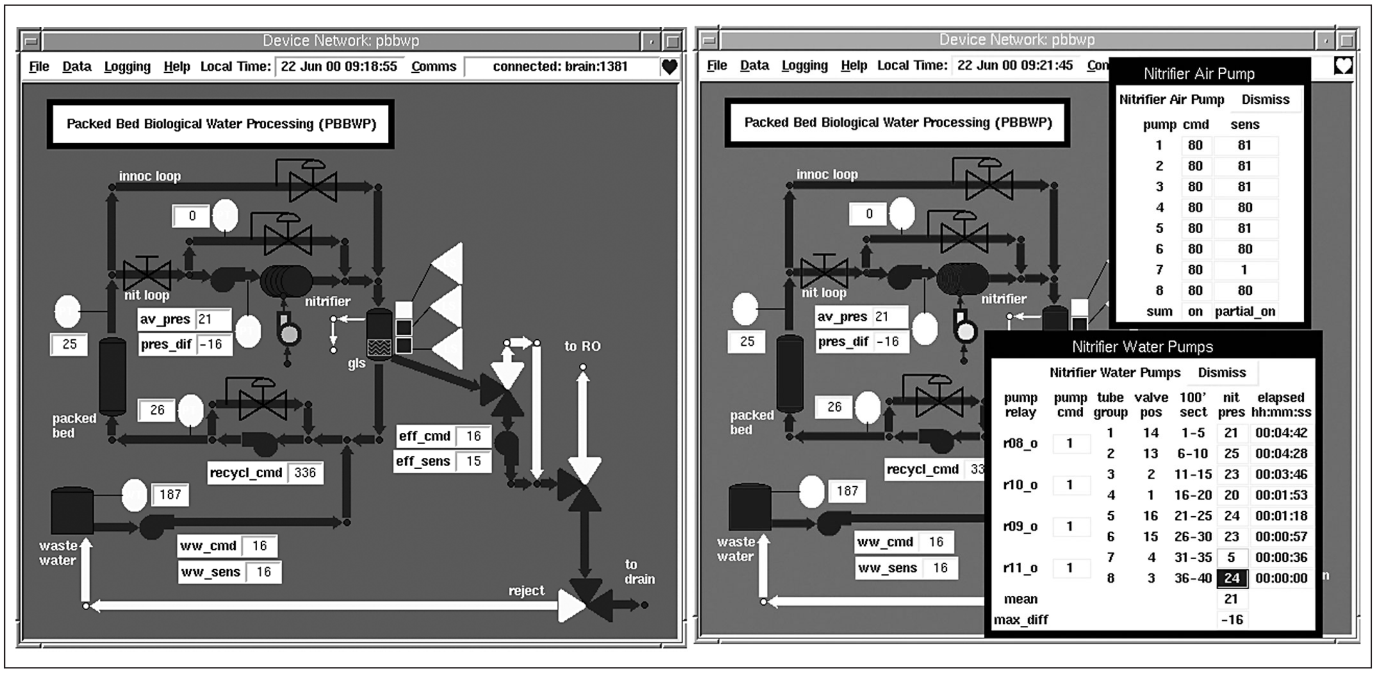
Early in the test, the PPS experienced few high-TOC spikes, which were of relatively short duration. As the test wore on, however, the AES wicks and reverse-osmosis membranes began to degrade, the high-TOC incidents became more frequent and lasted longer, and as a result, the average and allowed TOC calculations became less and less accurate. We needed a way to calculate the volume of water that would have flowed into the product tank to update the TOC calculations. Such a volume could be computed from the flow rates of the water coming from the reverse-osmosis subsystem and the AES, but because the PPS skills and the skills for reverse osmosis operated on two different computer racks, the PPS skills could not access the required flow rates. The obvious 3T solution was to have the TOC calculations performed by the sequencing layer—which had access to data from the AES and the reverse-osmosis skills as well as the instantaneous TOC readings from the PPS skills—during the time the product water was being rejected. Once the average TOC dropped below the allowed TOC, the sequencer would redirect the PPS output to the product tank and reseed the PPS TOC calculations at the skill level with the values computed during the low–water-quality time.

Staff Member Acceptance of Autonomous Systems

As mentioned earlier, the 3T control approach is designed for autonomous systems. However, in all previous tests, the ALS teams maintained 24-hour vigilance using 3 eight-hour shifts daily. The 3T system ran autonomously in these tests, but because human presence was required for the nonautomated ALS subsystems, the ALS teams never had to relinquish total control to the software. Early in the IWRS work, however, as the hardware groups were setting their test goals, the controls group put forth a control goal of 95-percent unattended operations as a way of getting the water team to start thinking about autonomous operations. Except for the power supply incident discussed earlier, the water team allowed 3T to run autonomously from February 2001 through the end of IWRS operations on 15 April 2002. A fundamental indicator of the staff member acceptance of this autonomy was that at any time day or night, one could walk into the control room and see all the staff member chairs empty.

The Need for Adjustable Autonomy

Although the 3T controls for iWRS were able to run with full autonomy during hardware build up, functional testing, and the first three months of operation (January through March 2001), it was important that the test engineers be able to command the system or its subsystems at all levels of operation. Thus, we provided the test team with interactive interfaces that the control engineers used for code testing.


Device Network: pbbwp
File Data Logging Help Local Time: 22 Jun 00 09:18:55 Comms connected: brain:1381
Packed Bed Biological Water Processing (PBBWP)
innoc loop
nit loop
nitrifier
gls
to RO
packed bed
waste water
reject
to drain
Device Network: pbbwp
File Data Logging Help Local Time: 22 Jun 00 09:21:45

av_pres	21
pres_dif	-16
eff_cmd	16
eff_sens	15
recycl_cmd	336
ww_cmd	16
ww_sens	16

Nitrifier Air Pump

Nitrifier Air Pump Dismiss

pump	cmd	sens
1	80	81
2	80	81
3	80	81
4	80	80
5	80	81
6	80	80
7	80	1
8	80	80
sum	on	partial_on

Nitrifier Water Pumps

Nitrifier Water Pumps Dismiss

pump relay	pump cmd	tube group	valve pos	100' sect	nit pres	elapsed hh:mm:ss
r08_o	1	1	14	1-5	21	00:04:42
		2	13	6-10	25	00:04:28
r10_o	1	3	2	11-15	23	00:03:46
		4	1	16-20	20	00:01:53
r09_o	1	5	16	21-25	24	00:01:18
		6	15	26-30	23	00:00:57
r11_o	1	7	4	31-35	5	00:00:36
		8	3	36-40	24	00:00:00
mean					21	
max_diff					-16	

Figure 9. The Graphic User Interface Client for the Biological Water Processor (BWP), Showing the Data Broadcast from the BWP Skill Manager Displayed on an Analog of the BWP Hardware.

The right-hand picture shows additional data that could optionally be displayed.

These interfaces included commands for individual pumps, valves, and relays; commands to execute primitive RAPs (see figure 7 for an example of a primitive RAP for turning a valve); commands to execute mid-level RAPs, such as executing a reverse-osmosis purge; and commands to start or stop the autonomous operation of any subsystem, such as running the BWP in a stand-alone mode.

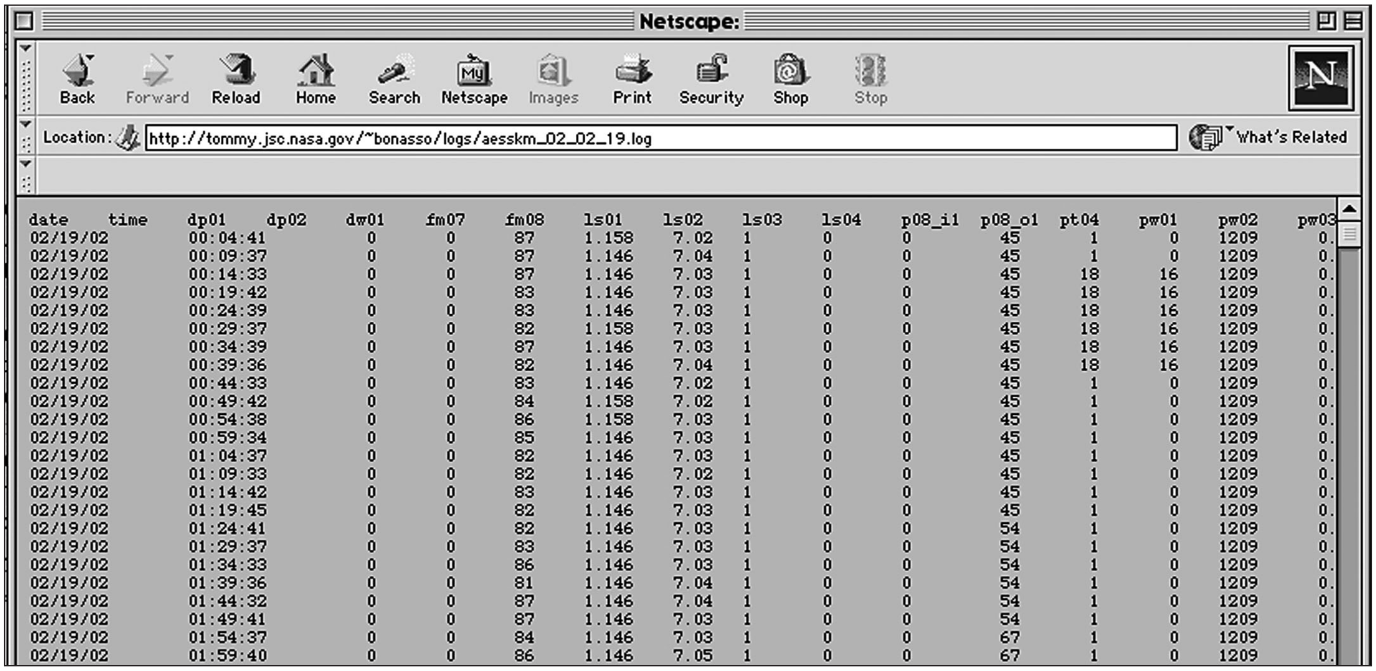
Being able to suspend parts of the control system's autonomy was important as well. For example, midway through the test, it became necessary for the BWP engineers to manually purge the individual tubes in the nitrifier portion of the BWP. This purging often resulted in a low-pressure condition that would trigger a low-pressure ASD in the BWP control code, which would stop the nitrifier pumps and air controllers for the problem tubes. To prevent the ASD during staff member purge operations, we modified the ASD RAP to check the state of a RAP memory flag for staff member purging. When the flag was present, the ASD would put out the ASD warning but would take no action. We then added interactive text to the WRS display (see the "Purge by Staff" text in the upper left of figure 8) that could be triggered to set the staff member purge flag in memory and start a 20-minute timer. At the end of the 20 minutes, the timer code would remove the flag.

Unattended Operations: Supporting Intermittent Monitoring with a Data Management and Distribution System

Logging the data broadcast from the skills—the sensed values and the commands sent to the devices—was required to support data analysis by the staff members both during and after the test. We developed a GUI for each subsystem in the control room (figure 6) to display in analog form the data broadcast from the skills (figure 9) and to set the logging rate for each subsystem. Menu options on these displays allowed a user to view logged data, set up strip charts, and plot any data item being logged.

However, as described earlier, after the first two months, the staff members were no longer in the control room, but the engineers still wanted to view these displays on their workstations in their offices. Additionally, to check the system for network or power failures, we needed to distribute the data broadcast from the skills for remote, intermittent monitoring.

To support the engineers, we ported the GUIs to the WINDOWS environments used by the staff members and installed the code on their workstations. Using these GUIs, the staff members could log data from any or all subsystems to their computers as they desired, but the logs of record were maintained in the water lab. Throughout the test, new logging requirements from the water team changed the for-



Location: http://tommy.jsc.nasa.gov/~bonasso/logs/aesskm_02_02_19.log

```
date     time     dp01     dp02    dw01    fm07    fm08    ls01    ls02    ls03    ls04    p08_i1  p08_o1  pt04    pw01    pw02    pw03
02/19/02          00:04:41         0       0       87      1.158   7.02    1       0       0       45      1       0       1209    0.
02/19/02          00:09:37         0       0       87      1.146   7.04    1       0       0       45      1       0       1209    0.
02/19/02          00:14:33         0       0       87      1.146   7.03    1       0       0       45      18      16      1209    0.
02/19/02          00:19:42         0       0       83      1.146   7.03    1       0       0       45      18      16      1209    0.
02/19/02          00:24:39         0       0       83      1.146   7.03    1       0       0       45      18      16      1209    0.
02/19/02          00:29:37         0       0       82      1.158   7.03    1       0       0       45      18      16      1209    0.
02/19/02          00:34:39         0       0       87      1.146   7.03    1       0       0       45      18      16      1209    0.
02/19/02          00:39:36         0       0       82      1.146   7.04    1       0       0       45      18      16      1209    0.
02/19/02          00:44:33         0       0       83      1.146   7.02    1       0       0       45      1       0       1209    0.
02/19/02          00:49:42         0       0       84      1.158   7.02    1       0       0       45      1       0       1209    0.
02/19/02          00:54:38         0       0       86      1.158   7.03    1       0       0       45      1       0       1209    0.
02/19/02          00:59:34         0       0       85      1.146   7.03    1       0       0       45      1       0       1209    0.
02/19/02          01:04:37         0       0       82      1.146   7.03    1       0       0       45      1       0       1209    0.
02/19/02          01:09:33         0       0       82      1.146   7.02    1       0       0       45      1       0       1209    0.
02/19/02          01:14:42         0       0       83      1.146   7.03    1       0       0       45      1       0       1209    0.
02/19/02          01:19:45         0       0       82      1.146   7.03    1       0       0       45      1       0       1209    0.
02/19/02          01:24:41         0       0       82      1.146   7.03    1       0       0       54      1       0       1209    0.
02/19/02          01:29:37         0       0       83      1.146   7.03    1       0       0       54      1       0       1209    0.
02/19/02          01:34:33         0       0       86      1.146   7.03    1       0       0       54      1       0       1209    0.
02/19/02          01:39:36         0       0       81      1.146   7.04    1       0       0       54      1       0       1209    0.
02/19/02          01:44:32         0       0       87      1.146   7.04    1       0       0       54      1       0       1209    0.
02/19/02          01:49:41         0       0       87      1.146   7.03    1       0       0       54      1       0       1209    0.
02/19/02          01:54:37         0       0       84      1.146   7.03    1       0       0       67      1       0       1209    0.
02/19/02          01:59:40         0       0       86      1.146   7.05    1       0       0       67      1       0       1209    0.
```

Figure 10. A Browser View of the Broadcast Data from the Air Evaporation System.
Data were normally recorded at five-minute intervals.

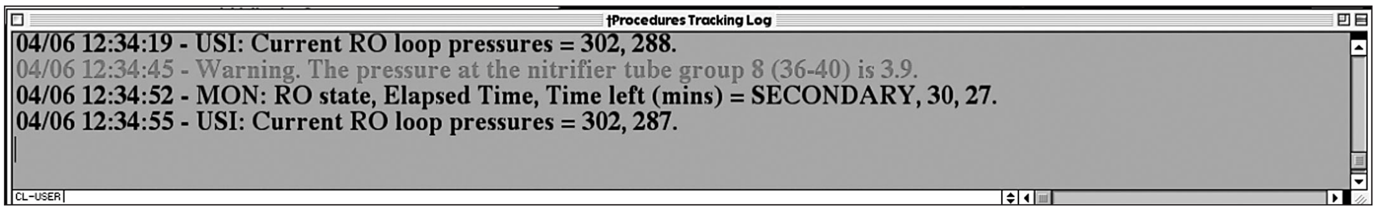
mat of the data and also the variables displayed in the GUIs. The format of the logs allowed viewing from the GUIs; a browser (figure 10); and Microsoft EXCEL, a favorite analysis tool of ALS engineers. To give the staff members access to the latest GUI code, we made the changes available through a National Aeronautics and Space Administration (NASA) web page. A prompt when a GUI started up would allow the staff members to download the new code and update their GUI display accordingly.

To watch the system on nights and weekends, we set up a duty roster of 3T control engineers to monitor the system. Every six hours from 8 AM until midnight, the control engineer on duty would "look in on the system" (lab personnel taking samples in the water lab each day checked on the system at 6 AM). The duty engineer could start up the GUI clients on his/her remote computer at home and use a NASA dial-up connection to receive and view the data broadcast from the water lab. We also made the logged data available in columnar format at a NASA-JSC URL (figure 10) so that the on-duty control engineer could monitor the system from computers around the world without the GUI software.

The previous 91-day test marked our first experiences in designing displays to provide users of 3T systems with a view into the state of the monitored system and the 3T software. At this time, we began developing an understanding of how to support intermittent monitoring of ALS systems (Thronesbery and Schreckenghost 1998). With the IWRS, we, as monitors of the software controls system, shared many of the same concerns as our intended users, the engineers monitoring the ALS hardware. Our experiences with these 24-hour-a-day, 7-day-a-week operations allowed us to expand our understanding of how to support intermittent monitoring.

Maintaining System Awareness The subsystem GUIs (for example, see figure 9) helped the user maintain system awareness by providing a quick overview, the subsystem schematic, and additional details on demand. The more commonly desired data (tank levels, pump speeds) were displayed directly on the schematic, and additional information was available (units, human-readable device names, component values for a computed value) by clicking on the schematic component in question.

Reviewing Performance History Although only monitoring data intermittently, the IWRS engineers needed to be able to review performance history to detect system anomalies or indicators that an anomaly was developing. The IWRS engineers were also in the process of determining efficient configurations of equipment that were also effective at recovering water for space operations. To evaluate the performance characteristics of each test



```
04/06 12:34:19 - USI: Current RO loop pressures = 302, 288.
04/06 12:34:45 - Warning. The pressure at the nitrifier tube group 8 (36-40) is 3.9.
04/06 12:34:52 - MON: RO state, Elapsed Time, Time left (mins) = SECONDARY, 30, 27.
04/06 12:34:55 - USI: Current RO loop pressures = 302, 287.
```

Figure 11. A Snapshot of the Procedures Tracking Log.

configuration, the water team made extensive use of the data logs to support anomaly detection and performance analysis. The logs could be displayed in a table from the displays, and variables could be plotted for viewing performance over time.

Responding to Problems Intermittent monitoring requires not only that the intelligent system function autonomously most of the time but also that it is able to recognize when failures occur and notify the human expert in a timely fashion. Initially, the GUIs subscribed only to device-level data from the skill manager layer of 3T. Consequently, the primary anomaly that could be detected was a loss of data connection between the skill manager and the GUIs, signaled by both audible and visual alarms. Later, the watchdog timers were added, which indicated the health of the communications between the skills and RAPs layers of 3T. The user could then use an information pop-up to see how long the skills and RAPs had been out of contact with one another. This information would also go to an error log, with time stamps allowing the user to examine performance history just prior to the anomaly.

Accessing Reference Information From interactive parts of the GUIs, the user could display the skills specifications (figure 5), allowing operators to refresh their understanding of how the controls work and providing access to device nomenclature from standard IWRS drawings used in the hardware specifications. In addition, the lead controls engineer prepared operations notes that were used by the remainder of the controls team to assess the health of the software controls systems. These operations notes were also available from the GUI schematics displays.

Advanced Techniques Because the software development resources were limited, we were unable to try a number of advanced automation techniques to support intermittent monitoring of IWRS and software controls systems. Although the combination of flow paths, alarms, and data values in the GUIs were helpful in gaining a quick overview of the system, it would have been more valuable to have information from the upper tiers of 3T, so that the GUI information could integrate high-level data with the observed device performance data (Schreckenghost and Thronesbery 1998). We had two opportunities to explore this idea, both involving the BWP sloughing operations.

When IWRS engineers wanted to know when automated sloughing took place so they could monitor it during initial deployment, we had RAPs broadcast a sloughing message and then integrated device-level data to accompany this communication. This integrated information was not only available in the GUI and the log, but it was also sent to subscribing e-mail addresses.

In an experiment conducted toward the end of the test, we had RAPs broadcast to a web-accessible data base information from the RAPs procedure tracking log, a file used in the water laboratory for controls debugging (figure 11). The experiment lasted two weeks, at a time when we experienced no important anomalies. Nonetheless, our experience showed that just as the procedure tracking log was used regularly during the course of the IWRS tests in the water laboratory to determine the cause of anomalies, the web-based tracking log information made it possible to conduct similar activities remotely.

Ultimately, fault diagnosis using AI techniques such as model-based reasoning will alleviate much of the need to distribute the details of problems that occur. Resources were not available to provide automated fault diagnosis, so when major problems occurred, the team could take the IWRS "out of test" to spend several days analyzing and solving the problem. For an actual flight system, automated techniques will be required (see Spin-Off Benefits for the AI Community).

The Best AI Win

The most rewarding experience in the two-year IWRS test came when one of the biological engineers asked the control team to use the top tier of 3T to help them with a particularly troublesome aspect of the BWP. The nitrifying portion of the BWP (figure 12) consisted of bacte-

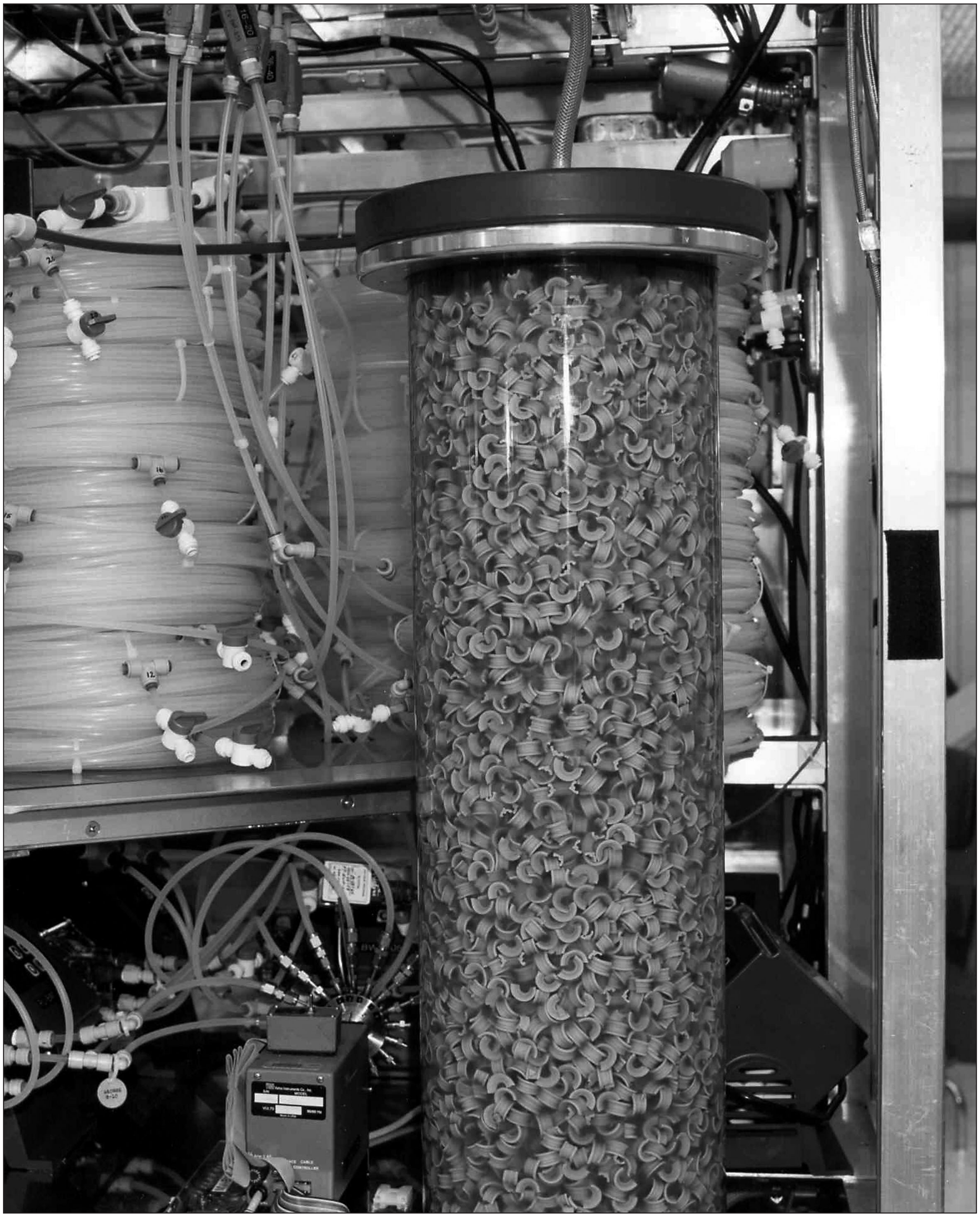
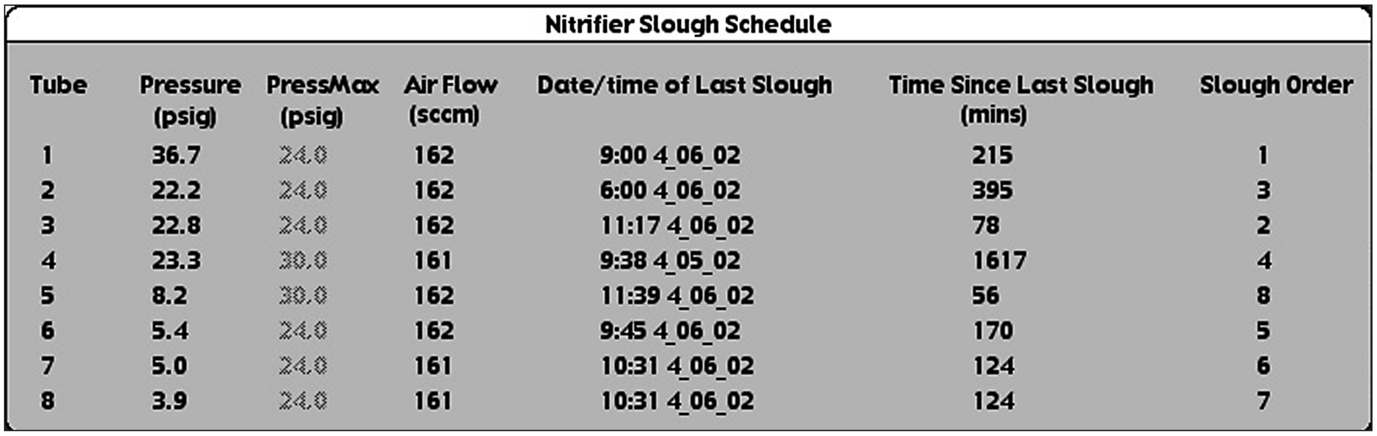


Figure 12. Symbiotic Biological Reactors.

Feed water flows from the packed bed on the right through the nitrifier (coiled tubes on the left and to the rear), through the gas-liquid separator (GLS) (not shown), and back to the packed bed. The nitrifier bacteria uses oxygen (O_2) to reduce feed water ammonia (NH_4) to nitrates (NO_3) and nitrites (NO_2), which the bacteria on the ceramic saddles in the packed bed use to remove organic carbons. Nitrogen and carbon dioxide gases are released in the GLS.



Nitrifier Slough Schedule

Tube	Pressure (psig)	PressMax (psig)	Air Flow (sccm)	Date/time of Last Slough	Time Since Last Slough (mins)	Slough Order
1	36.7	24.0	162	9:00 4_06_02	215	1
2	22.2	24.0	162	6:00 4_06_02	395	3
3	22.8	24.0	162	11:17 4_06_02	78	2
4	23.3	30.0	161	9:38 4_05_02	1617	4
5	8.2	30.0	162	11:39 4_06_02	56	8
6	5.4	24.0	162	9:45 4_06_02	170	5
7	5.0	24.0	161	10:31 4_06_02	124	6
8	3.9	24.0	161	10:31 4_06_02	124	7

Table 1. The Nitrifier Slough Interface.

The water staff members could set the maximum pressure (PressMax column) for each tube. The scheduler obtained the instantaneous pressures from the Biological Water Processor.

ria that grew on the insides of eight sets of coiled tubes through which feed water flowed. The microbial biofilm in the tubes would grow over time, constricting the passage of water and air and increasing the pressure in the tubes. If left unchecked, the tubes would clog. When a tube clogged, the water flow increased in the other tubes causing higher-than-normal pressures that could shear healthy biofilm from the walls of the tubes, thus decreasing the nitrification action of the reactor. To prevent the tubes from clogging, the water team laboratory personnel had to periodically slough the biofilm by manually raising the airflow through each tube to maximum for several minutes until the pressure dropped to a nominal level. As the maturing biofilm continued to grow, the tubes needed sloughing more frequently. Eventually, the staff members were required to come in at night and on weekends to carry out the sloughs. If the staff members forgot or were late in sloughing a tube, as happened on a number of occasions, clogging was the inevitable result. Soon the manual procedures could not keep up with the number of sloughs required.

About midway through the test, the subsystem engineer for the BWP approached the controls group to inquire if the third tier of 3T could possibly help automate the sloughing process. That the engineer understood the usefulness of the top tier in relation to the other tiers indicated that the water team had understood the basic concepts behind the 3T architecture. Of course, this problem was one of simple scheduling. We constructed a scheduler that once an hour checked the pressure in each tube against a maximum allowable pressure for the tube, sorted the tubes according to this pressure difference and the time since the last slough for each tube, then invoked a new RAP that raised the airflow rate for a specified period to force a slough just as the human staff memers did in the previous manual mode. The scheduler interface (table 1) allowed the staff members to adjust each tube's maximum pressure. By having the sequencer broadcast the schedule information, we made the schedule details part of the BWP display GUI as well as the data logs.

Toward the end of the last IWRS test-point run, 3T was sloughing one tube an hour each day and night. Eventually, the air sloughs were insufficient to bring the pressure down in a few of the tubes, so periodic manual assistance was required, with the staff members cleaning the nitrifier trap filter and releasing pressure through manual valves positioned throughout the tube lengths. Nonetheless, without 3T's auto-slough, the water team would not have been able to maintain a viable BWP after the first half of the IWRS test.

Spin-Off Benefits for the AI Community

Since 1995, the authors and their colleagues at the Texas Robotics and Automation Center Laboratories (TRACLabs) and NASA-JSC, as well as a number of research groups around the country, have identified several areas where AI investigations can help with problems we have encountered in support of ALS operations. Such investigations have included planning and scheduling (Schreckenghost et al. 2000), adjustable autonomy (Schreckenghost et al. 2001), human-centered computing (Dorais and Kortenkamp 2001), and machine learning (Kortenkamp, Bonasso, and Subramanian 2001).

The two-year IWRS test has also spawned a number of research efforts. The following is a list of the past and ongoing work inspired by the IWRS efforts:

Evaluating the application of machine learning to the control of advanced life-support systems: The participants were NASA-JSC, NASA-Ames, Rice University, Carnegie Mellon University (CMU), Massachusetts Institute of Technology (MIT), and the Naval Research Laboratories. Researchers at Rice University and JSC identified the periodic nature of optimizing coupled life support systems and evaluated techniques for dealing with such systems (Kortenkamp, Bonasso, and Subramanian 2001). Recent efforts have focused on using logged IWRS sensor and control data to automatically build models of system behavior, which can then be used to monitor for off-nominal operations.

Developing a suite of visualization tools for distributed autonomous systems: The participants were CMU and members of TRACLabs. Using logged data from the IWRS, researchers developed a set of analysis and display tools that allow a user to manipulate the data and look for relationships (Kortenkamp et al. 2001).

Distributed crew interaction: The participants are NASA-JSC and Ohio State University. The objective is to investigate knowledge representations and architectures for distributed collaboration among human and software agents in support of automated life-support control, in particular the 3T IWRS system (Schreckenghost et al. 2002).

Complex event recognition: Participants are TRACLabs and I/NET, Inc. Using IWRS data, the project seeks to develop software tools for detecting and storing information about important control events.

Robust methods for autonomous fault diagnosis and control of complex systems: Participants are Vanderbilt University and NASA-JSC. The objective of this project is to use probabilistic methods to diagnose failures in complex systems (for example, the IWRS) and adapt controllers to recover from these failures.

Finally, data and information concerning the IWRS are available to the AI community at large. Several simulations of the IWRS subsystems (for example, Malin et al. [2002]) and the IWRS control code exist, and data logs for the entire test are available to any interested party.

Summary and Future Directions

After years of grafting parts of 3T to various ALS systems, the integrated WRS test gave us an opportunity to prove the usefulness of the entire intelligent control architecture in a long-running application. The test is over, and we have learned many useful lessons concerning autonomy, unattended operations, and long-duration control. Although much of the hidden power of 3T might have eluded the water engineers, we feel we validated the need for AI in this test when the test team requested an automatic scheduler for the nitrifier sloughs.

Based on our experience with the IWRS, we claim that long-running, unattended autonomous operations will be the norm for ALS systems in the future. There are several implications of this claim, which give rise to a number of interesting research issues that should be pursued. Although one could make the case for many AI technologies being relevant in this area, we list a few that stood out to us during this test:

Providing commanding capability from remote locations: Because of security reasons, we did not investigate executing IWRS procedures from outside the water laboratory, but a number of times, it would have been convenient for the members of the control team or even the water team manager to be able to restart a given subsystem from their home or desk. Remote commanding naturally requires the authorization and authentication of intended users. However, allowing more than one engineer to affect different parts of the system simultaneously from remote locations gives rise to issues of managing conflicting commands, preventing inadvertent ASDs, and informing each user of the actions of other users making changes to the system. The distributed crew interaction project mentioned earlier (Schreckenghost et al. 2002) is currently investigating these issues.

Planning and scheduling complex ALS operations: The IWRS is but one of several systems that make up a complete life-support facility for either space or planetary crews. Each of the systems in figure 13, for example, represents the same or greater level of complexity as the IWRS. Such an ALS system will require generative planning capabilities that must allow for interaction by the test team or by the crew in eventual deployed applications (Schreckenghost et al. 2000).

Machine learning: We discussed earlier the natural drift of instrumentation, the need to adjust the controls to accommodate new wait times or set points, and the need to detect anomalies in the IWRS subsystems. With the large number of systems anticipated for a full ALS, a human crew or test team will be hard pressed to keep track of these changes without automated assists. Machine learning techniques will be necessary to provide these as-

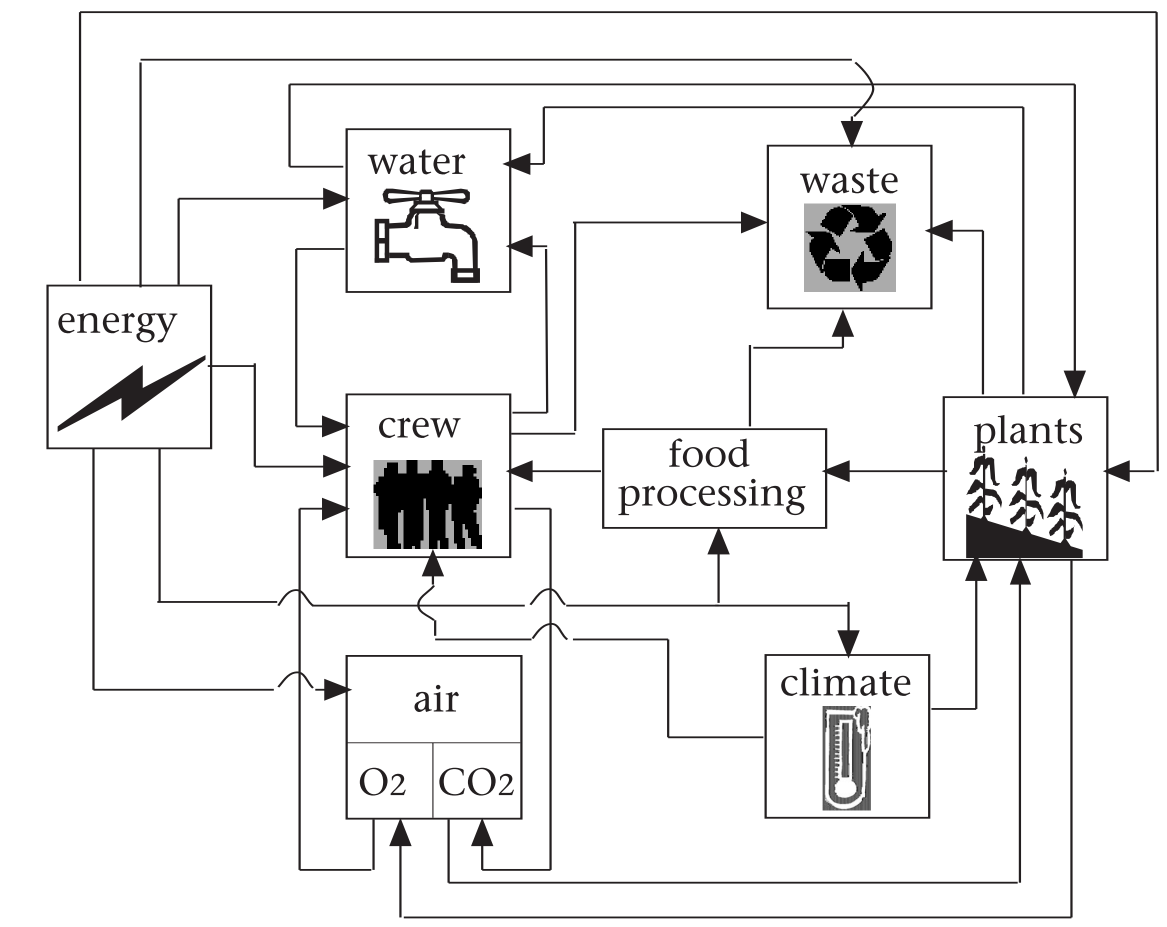


Figure 13. The Interacting Subsystems of an Advanced Life-Support System (Kortenkamp, Bonasso, and Subramanian 2001).

sists, particularly because the human crew–test team will not be monitoring the systems in situ and, thus, will not be as familiar with what constitutes normal data values as it would if it kept close watch on such systems.

Natural interfaces for control: Because users of future ALS systems will spend little time in front of a control console, they will tend to forget how to interact with multiple interfaces such as the one shown in figure 8. This aspect of unattended operations points to the need for more natural interfaces, ones that are multimodal and flexible and allow the user to enter a dialog for discussing the state of the system and any actions that might need to be taken.

Smart data interfaces: As alluded to in our lessons learned, although the analog data displays and their associated data logs (figures 9 and 10) made it possible to detect problems in the iWRS, the process was still time consuming, particularly for system changes that were almost imperceptible without a historic trace of data values. For a full, unattended ALS system, interfaces that quickly show recent trends that will catch the user's eye, and then easily guide her to the particular data source, will be mandatory. As well, the high-level information must be combined with related device-level log data into an integrated situation so that all related information is available to the user for inspection in one place (Thronesbery, Christoffersen, and Malin 1999).

Distributed human interaction: With future ALS systems only needing intermittent monitoring, human-computer interaction will take place from locations remote from where the automation executes. Our simple foray into notifying remote users about the nitrifier slough must eventually be extended not only to all off-nominal events but also to any event individual users might find of interest. It is likely that our simple GUIs and ad hoc event detectors would grow into full-fledged proxy agents for each user, for example, as in Chalupsky et al. (2001). Our IWRS notification work is continuing in Schreckenghost et al. (2002) with software proxies and an analysis of notification schemes.

We believe the ultimate goal for AI in life support is to allow the ALS systems to run "in the background" as it were, just like earth-bound residential climate-control systems. When humans must intervene, the intelligent control system will guide them easily and quickly to the source of the problem. Our experiences with the IWRS show that AI can make significant contributions today in ALS systems, but there is still much work to be done for the future.

Note

1. Our use of the Lisp language was not an issue for the WRS engineers. Because none of them were computer scientists, they depended on the control team to select the appropriate languages for the tasks at hand.

References

Bonasso, R. P. 2001. Intelligent Control of a NASA Advanced Water Recovery System. Paper presented at the Sixth International Symposium on Artificial Intelligence, Robotics and Automation in Space: A New Space Odyssey, June 18–22, Montreal, Quebec, Canada.

Bonasso, R. P.; Firby, J. R.; Gat, E.; Kortenkamp, D.; Miller, D. P.; and Slack, M. G. 1997. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2): 237–256.

Chalupsky, H.; Gil, Y.; Knoblock, C. A.; Lerman, K.; Oh, J.; Pynadath, D. V.; Russ, T. A.; and Tambe, M. 2001. ELECTRIC ELVES: Applying Agent Technology to Support Human Organizations. Paper presented at the Innovative Applications of Artificial Intelligence, August 4–10, Seattle, Washington.

Digitool. 1996. Macintosh Common Lisp Reference Manual. Technical Manual, Digitool, Inc., Cambridge, Massachusetts.

Dorais, G. A., and Kortenkamp, D. 2001. Designing Human-Centered Autonomous Agents. In *PRICAI Workshop Reader,* eds. R. Kowalcyk, S. W. Lake, N. Reed, and G. Williams, 339–352. New York: Springer-Verlag.

Elsaesser, C., and Sanborn, J. 1990. An Architecture for Adversarial Planning. *IEEE Transactions on Systems, Man, and Cybernetics* 20(1):186–194.

Firby, J. R. 1999. The RAPS Language Manual. Chicago: Neodesic.

Gat, E. 1998. Three-Layer Architectures. In *Mobile Robots and Artificial Intelligence,* eds. D. Kortenkamp, R. P. Bonasso, and R. Murphy, 195–210. Menlo Park, Calif.: AAAI Press.

Kortenkamp, D.; Bonasso, R. P.; and Subramanian, D. 2001. Distributed, Autonomous Control of Space Habitats, 2752–2762. In Proceedings of the IEEE Aerospace Conference. Washington, D.C.: IEEE Computer Society.

Kortenkamp, D.; Milam, T.; Simmons, R.; and Fernandez, J. L. 2001. Collecting and Analyzing Data from Distributed Control Programs. *Electronic Notes in Theoretical Computer Science* 55(2).

Lai-fook, K. M., and Ambrose, R. O. 1997. Automation of Bioregenerative Habitats for Space Environments. In Proceedings of the IEEE International Conference on Robotics and Automation, 2471–2476. Washington, D.C.: IEEE Computer Society.

Malin, J.; Flores, L.; Fleming, L.; and Throp, D. 2002. Using CONFIG for Simulation of Operation of Water Recovery Subsystems for Advanced Control Software Evaluation. Paper presented at the Thirty-Second International Conference on Environmental Systems, July 15–18, San Antonio, Texas.

Schreckenghost, D., and Thronesbery, C. 1998. Integrated Display Supervisory Control of Space Operations. Paper presented at the Human Factors and Ergonomics Society Forty-Second Annual Meeting, October 5–9, Chicago, Illinois.

Schreckenghost, D.; Bonasso, R. P.; Hudson, M. B.; and Kortenkamp, D. 2000. Activity Planning for Long-Duration Space Missions. Paper presented at the AAAI Workshop on Representational Issues for Real-World Planning Systems, July 30, Austin, Texas.

Schreckenghost, D.; Bonasso, R. P.; Kortenkamp, D.; and Ryan, D. 1998. Three-Tier Architecture for Controlling Space Life-Support Systems. In Proceedings of the IEEE Symposium on Intelligence in Automation and Robotics: IEEE Computer Society.

Schreckenghost, D.; Edeen, M.; Bonasso, R. P.; and Erickson, J. 1998. Integrated Control of Product Gas Transfer for Air Revitalization. In Proceedings of the Twenty-Eighth International Conference on Environmental Systems, July 15, Danvers, Massachusetts.

Schreckenghost, D.; Malin, J.; Thronesbery, C. E.; Watts, G.; and Fleming, L. 2001. Adjustable Control Autonomy for Anomaly Response in Space-Based Life-Support Systems. Paper presented at the IJCAI-2001 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents, August 6, Seattle, Washington.

Schreckenghost, D.; Thronesbery, C.; Bonasso, R. P.; Kortenkamp, D.; and Martin, C. E. 2002. Intelligent Control of Life Support for Space Missions. *IEEE Intelligent Systems* 17(5): 24–31.

Simmons, R., and Dale, J. 1997. Interprocess Communication: A Reference Manual. IPC Version 6.0. Robotics Institute, Carnegie Mellon University.

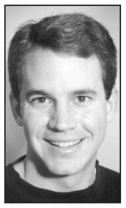
Thronesbery, C., and Schreckenghost, D. 1998. Human Interaction Challenges for Intelligent Environmental Control Software. Paper presented at the Twenty-Eighth International Conference on Environmental System, July 13–16, Danvers, Massachusetts.

Thronesbery, C.; Christoffersen, K.; and Malin, J. 1999. Situation-Oriented Displays of Shuttle Data. Paper presented at the Human Factors and Ergonomics Society Forty-Third Annual Meeting, Houston, Texas, September 27–October 1.

Williams, B. C., and Nayak, P. P. 1996. Immobile Robots—AI in the New Millennium. *Artificial Intelligence* 17(3): 16–35.

Pete Bonasso is a senior researcher with Texas Robotics and Automation Center Laboratories, where he has supported a variety of National Aeronautics and Space Administration projects in intelligent control since 1993. He is the codeveloper of the 3T robot control architecture and has applied this architecture to the control of robotic and life-support machines. Although his primary area of interest is intelligent control of robots (he is coauthor of the MIT Press book *Mobile Robots and Artificial Intelligence,* 1998), he has spent the last four years writing RAPS programs in 3T for ALS water-processing immobots. He is currently involved in building Lisp agents in a distributed agents research project. His e-mail address is r.p.bonasso@jsc.nasa.gov.

Carroll Thronesbery has been developing intelligent systems and designing human-computer interaction with intelligent systems since receiving his Ph.D. in cognitive psychology from the University of Houston in 1981. He developed traditional requirements documents for large Department of Defense projects as well as software prototypes for innovative, small-scale projects. Thronesbery has developed intelligent systems for the National Aeronautics and Space Administration Johnson Space Center, receiving awards for an intelligent system application certified for use in the new Mission Control Center, an innovative World Wide Web technology application to support response to Space Shuttle flight anomalies, recommendations for orbiter upgrades, an intelligent system development methodology, an evaluation of software development tools, and a human-computer interaction design for controlling and monitoring life-support hardware. His e-mail address is c.thronesbery@jsc.nasa.gov.



David Kortenkamp is a senior research scientist at Metrica Inc., supporting the National Aeronautics and Space Administration Johnson Space Center. He received his Ph.D. in computer science and engineering from the University of Michigan in 1993 and his B.S. in computer science from the University of Minnesota in 1988. Kortenkamp directs nearly $800,000 in annual research projects and is on a number of conference and workshop program committees. He is coauthor of the book *Mobile Robots and Artificial Intelligence* (MIT Press, 1998) and is associate editor of the MIT Press series on intelligent robotics and autonomous agents. His e-mail address is korten@traclabs.com.