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 two-year test for a NASA 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, 24/7 for sixteen 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.

"We'll have to go with four 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). "But 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 PDA then spoke up. "Even at 32 mils a minute, the pressure build-up 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 the above exchange was typical of many the AI controls team from the Robotics, Automation and Simulation Division (AR&SD) at Johnson Space Center 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 of 2001. We were building an AI control system for that test that had to handle upwards of 200 sensors and actuators grouped among four water processing subsystems. The control system would run 24/7 and be completely autonomous. It was an applied AI engineer's dream and in the end we were extremely successful. But 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.

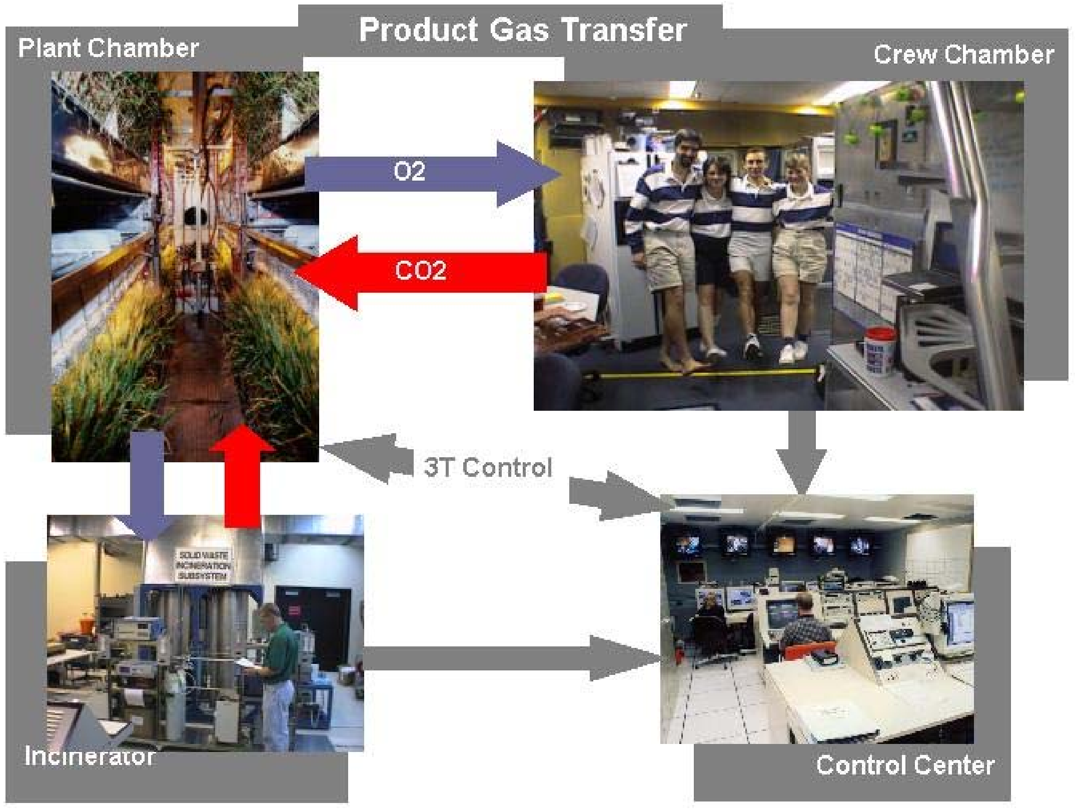


Figure 1 The Product Gas Transfer Environment.

The Early Years

Since 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 ten-foot diameter chamber full of wheat (Lai-fook and Ambrose, 1997). For fifteen days, the man lived, worked and exercised in the chamber while 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 (C&W) for the climate and nutrient environment of the wheat crop.

In 1997 they put two men and two women in a thirty-foot chamber for ninety-one days (Schreckenghost et al., 1998b). A physical-chemical air revitalization system recycled the air for three of the four people, while a wheat crop in the ten-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 (see Figure 1). Operating 24/7, the AI system also employed a generative planner that scheduled waste incinerations and crop planting and harvesting, coordinating those tasks with the day-to-day product gas transfer.

For both of these projects we used a three-layer architecture (Gat, 1998) to design, organize and develop the control software. AR&SD had used a particular implementation of this architecture known as 3T (see sidebar) in a number of robot projects prior to 1995 (Bonasso et al., 1997), and since life support systems are a form of immobots (Williams and Nayak, 1996), its application to ALS projects was straightforward.

The 3T Intelligent Control System (sidebar)

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 (see **Figure 2**):

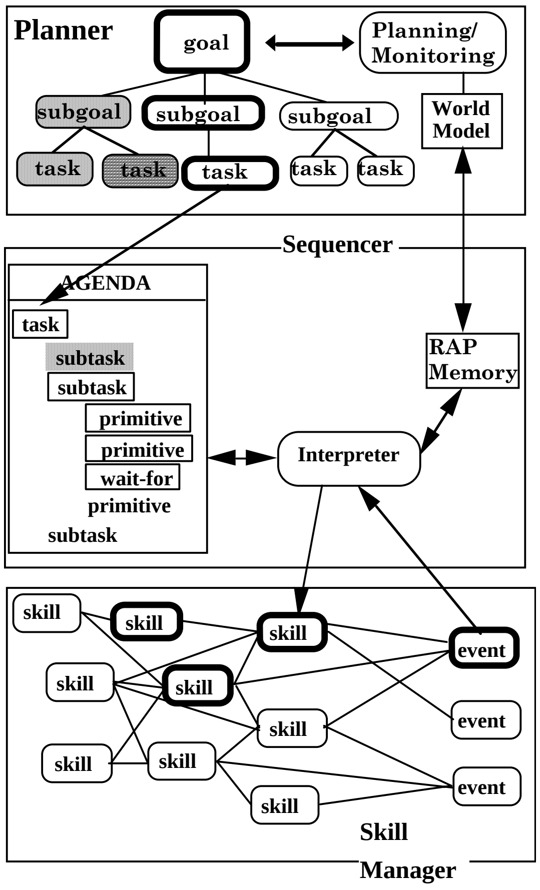


Figure 2 The 3T AI Control Architecture

° A set of robot specific, situated skills (or behaviors) that 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. 3T's implementation includes primitive actions, queries and monitoring events that can be combined to form autonomous behaviors. 3T's 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 (see Figure 7) co-located with the CPUs (we are using a VERSAModuleEurocard (VME) bus, with Vxworks running on Power PCs).

° A sequencing capability that can differentially activate the situated skills in order to direct changes in the state of the world and accomplish specific tasks. 3T 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 that reasons in depth about goals, resources and timing constraints. 3T uses a non-linear hierarchical task net planner known as AP (Elsaesser and Sanborn, 1990). AP 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 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.

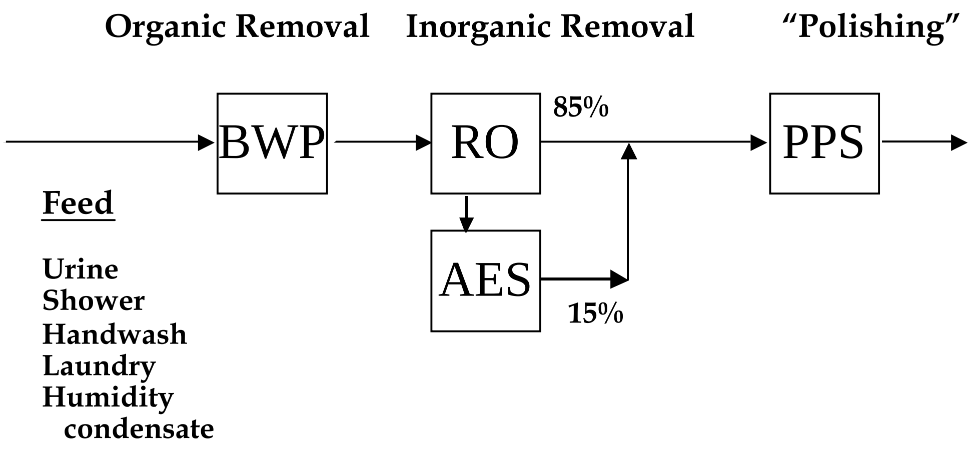
A key aspect of 3T is that it gives developers the ability to integrate the continuous, near-real time control algorithms in the bottom layer with advanced AI algorithms in the top layer -- i.e., automated planners and schedulers -- that are event driven but more computationally expensive. 3T does this through the integrating action of the middle layer. Essentially, the middle layer translates the goal states computed by a planning/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.

3T applications run autonomously due in large part to 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 safing the controlled system; and the planner can synthesize alternative plans in light of the failures of the lower two tiers.

In each of the previous efforts, the 3T team from AR&SD was required to

Advanced Water Recovery System (sidebar)

The advanced water recovery system (WRS) is a set of next generation WRS components, which promise to provide potable water using fewer consumables (filters, resins, etc.) and much less power than the components currently in use on the International Space Station (ISS). Figure 3 and Figure 4 show the four subsystems used in the integrated WRS test (iWRS). The iWRS is comprised of 1) a biological water processor (BWP) to remove organic compounds and ammonia; 2) a reverse osmosis (RO) subsystem to remove inorganic compounds from the effluent of the BWP; 3) an air evaporation system (AES) to recover additional water from the brine produced by the RO; and 3) a post processing system (PPS) to bring the water to within potable limits.



		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 3 . The water flow paths and the target quality values in milligrams and millisemens (an indirect measure of water quality) per liter for the iWRS. The numbers in parentheses for the PPS effluent are those for typical residential tap water.

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, etc.) In contrast the advanced WRS developed and tested at JSC is projected to require only 250 pounds of consumables per year and use 50% less power.

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

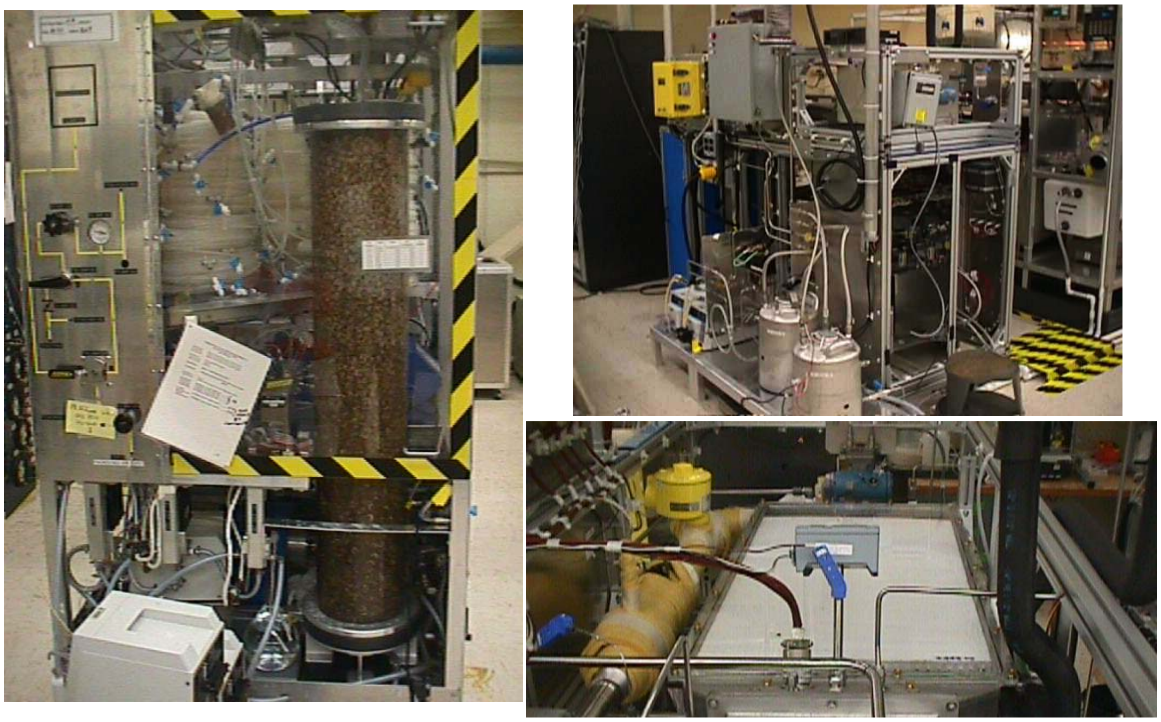


Figure 4 The AWRS subsystems. At the left is biological water processor (BWP). Upper right is the rack containing the Reverse Osmosis (RO) subsystem in the rack bottom, the air evaporation subsystem (AES) at the top of the rack, and the post processing system (PPS) in the rack's left rear. The bottom picture is a close up of the wick in the AES.

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 analog-to-digital (A/D) converter boards used by the sensors and devices to the top tier of the architecture.

In the summer of 1999 we 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 24/7 test. Then in January 2000 the advanced water research group received ALS funding for the year long integrated water recovery system (iWRS) test, involving four advanced water recovery subsystems (Bonasso, 2001) (see Advanced Water Recovery sidebar).

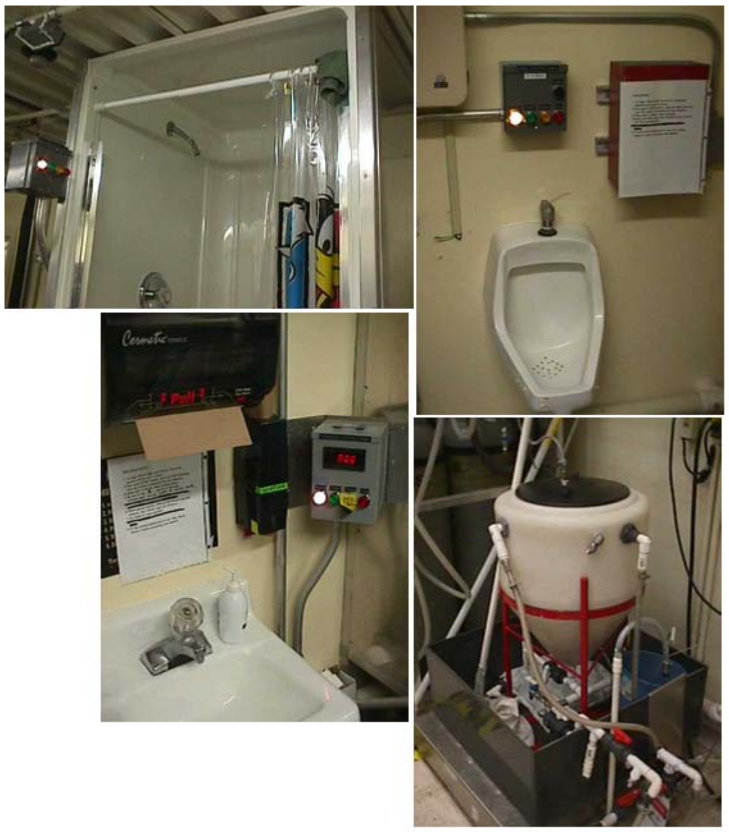


Figure 5 The iWRS waste water collection system. Human volunteers donate urine, showers, and hand washes, using liquid soap with the chemical composition of that to be used on the space station. A computer system responds to the pushbuttons at each donation site to weigh and record each type of donations before sending the donation to the main feed tank for the iWRS. Prepared solutions representing respiration water are added to the feed tank to complete a composition representative of that expected on the space station and/or planetary outposts.

Build-up

Using 3T allowed us to develop the control of for the iWRS in a modular fashion in two ways. First, moving from bottom to top (see Figure 6), each layer has its own data structures, timing constraints and development tools that allow for parallel development of the software. So we were able to develop skills sets based on the evolving hardware specifications while simultaneously developing 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) using "virtual skills", that is, Lisp skills connected to a Lisp simulation of the expected hardware. A virtual simulation of, say, the RO 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 (see Figure 8).

As the hardware specifications became more firm, 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 on line, the skills for

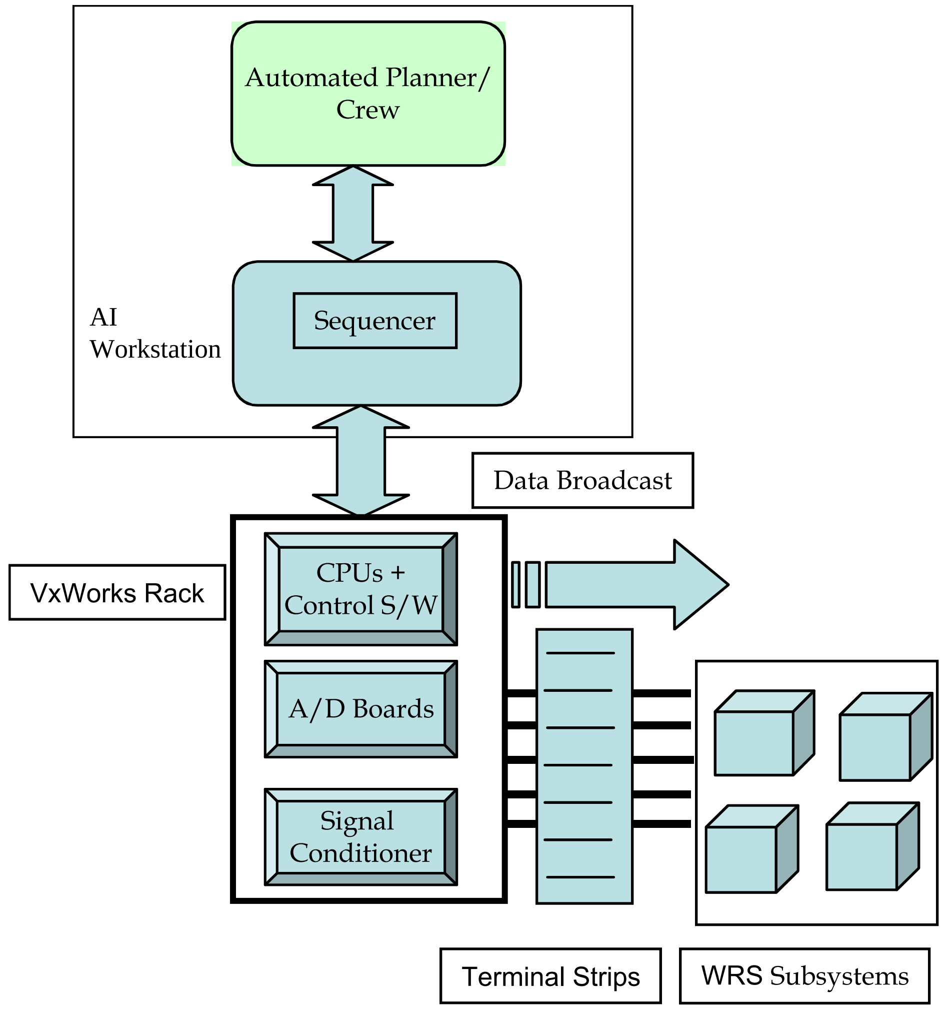


Figure 6 The 3T Implementation for the iWRS test. For each of the WRS subsystems we developed one skill manager, which ran on its own CPU. The skill managers provided the A/D results to the all of the skills modules and broadcasted that data at specified intervals for use by extant clients for analysis 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

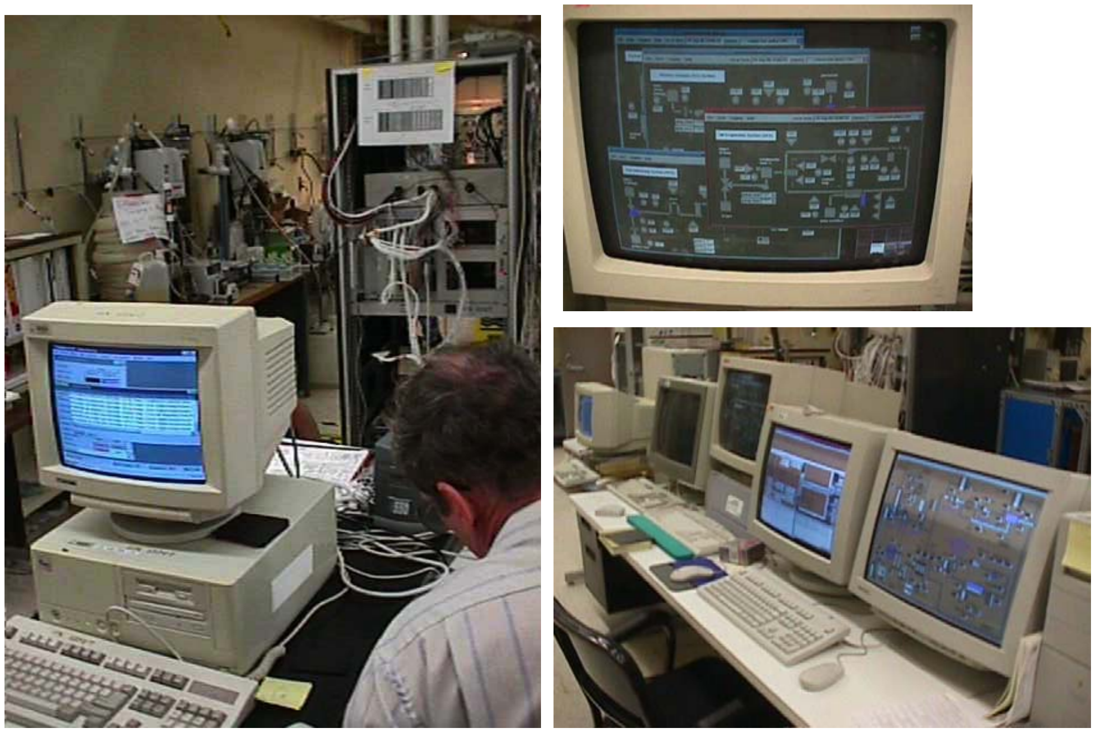


Figure 7 3T Control Computers for the iWRS. On the left is a view of the 3T VME Rack behind the computer that is used as a secondary interface to the RO high-pressure pump. On the right is a view of the (unattended) 3T control table. From foreground to back, the displays are two sequencer/planner displays, the IPC/skill manager display, the display of the broadcast server, and a display associated with the high-pressure pump used in the RO. In the upper right is the display showing the graphical user interfaces (GUIs) for each subsystem generated by the data broadcast from each skill manager.

that subsystem 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 graphical user interface (GUI) to activate and de-activate individual skills. This development approach enabled the 3T team to deliver the low-level control for each subsystem within two weeks after the hardware installation of that subsystem.

Next, the sequencer procedures for the subsystem, known as reactive action packages (RAPs) were installed on the AI workstation and tested with

```
----------------------------------
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 8 Excerpts from the RO Skill Specifications

```
(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 9 A primitive event, a primitive RAP and a high level RAP which use the skills from the skill spec shown previously. 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. There are several methods associated with the processing-start RAP, each indexed by a context clause. The method shown is valid when the required RO stage is purge and the pps-select valve is open to the pps. In this case, the RAP starts the RO main feed pump, stops the recirculation pump and turns the pps-select valve to reject (relieving downstream pressure). Then the RAP turns the RO process valve to the purge position and turns the pps-select valve to the tank position.

the validated skills. An example of the resulting RAPs is shown in Figure 9. The skills level remained relatively stable once the sensors and actuators were in place. 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. This aspect of the control development became important for the WRS team in dealing

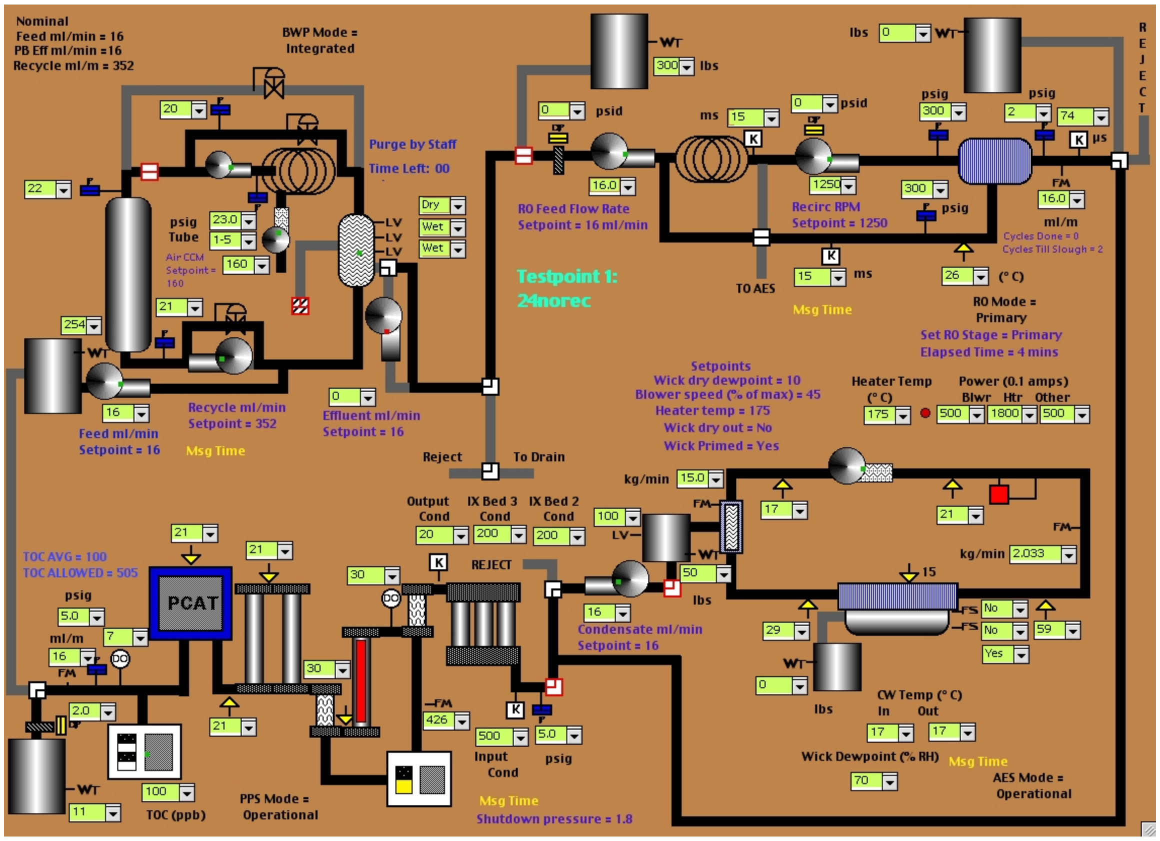


Figure 10 A Schematic Display of the iWRS as seen by 3T. The BWP is in the upper left, the RO in the upper right, the AES in the lower right and the PPS in the lower left. Gray lines indicate flow pipes; black pipes indicate water or air currently flowing. Small boxes with lines at junctures indicate motorized valves.

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 that time period, and even longer if the inoculations were problematic.

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: the BWP and the RO, and then the RO and the other two subsystems. In the iWRS system the pivotal subsystem is the RO. 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 while 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 of 2000, and brought the other two systems online in December of 2000, in time to make a January 2001 full start.

Controlling the iWRS System

In this section we describe for the final iWRS system that went into test in 2001 the control tasks for each subsystem and for the iWRS as a whole (see Figure 10).

The BWP

Feed water from the waste water collection system (see Figure 5) first passes through the biological water processor (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 10) at mid-level. This is accomplished by varying the speed of the feed pump while the draw from the RO main pump remains constant. The other requirement is to monitor the pressures in the recycle loop as well as in the nitrifier tubes and to carry out automatic shutdown procedures (ASDs) 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 that tube are shutdown and a warning is issued.

The RO

The RO (upper right portion of Figure 10) is the lynchpin subsystem since it pulls water from the GLS of the BWP, and delivers its permeate to the PPS and its 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 RO must go through up to four distinct phases in each cycle. The primary phase draws water into a coiled section of pipe that acts like a reservoir, while processing 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. But 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 backpressure on the membranes, permeate conductivity and loss of pressure in the recirculation loops.

The AES

The AES wick absorbs RO brine as it fills the AES reservoir (lower right of Figure 10) during the RO 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 it goes to standby awaiting another load. ASDs concern overheating and loss of coolant fluid in the heat exchanger.

Additionally, the AES pumps condensate to the PPS when the condensate tank reaches a certain level or when the RO 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 PPS

The PPS (lower left of Figure 10) "polishes" the water from the RO and the AES by removing trace inorganic material via ion exchange beds, and trace organics by oxidizing them with ultra-violet (UV) radiation. The PPS controls monitor the input water pressure. When the pressure goes above a threshold that indicates water flow from either the AES or the RO, the O_2 concentrator is started and a number of UV lamps are turned on commensurate with the measured total organic carbons (TOC). 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 inputs and water quality, and only rarely respond to the operation of the other subsystems. Interfaces among the subsystems are:

- The AES pumping condensate to the PPS in the RO's stead (previously mentioned)
- The RO monitors the level of the GLS in the BWP to insure that there is sufficient resource for it to draw upon.
- PPS pressure changes are corroborated by sensing the state of the pumps and the valve configurations of the RO and AES. For example, if the inlet pressure is not high enough to indicate water flow but the AES or RO pump and valve configurations indicate water is flowing, then the PPS will begin operations.
- When there is a complete ion exchange bed breakthrough, the RO and the AES sense the high PPS conductivity and recycle their effluent to the BWP feed tank.

The Test

The iWRS test consisted of 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 non-potable end point occurred when the quality of the water from the last of the three ion exchange beds rose above a pre-defined 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 rejected to reduce the loading on the ion exchange beds; 2 person, 18 hour operation (allowing six hours for maintenance);

- 2 person 18 hour operation with condensate reject.

Each test point called for either different flow rates or full/partial reject of internal flows or both. Besides rejecting AES condensate, four person or 18 hour operations required the RO and BWP to process water at an increased rate, with some of the RO 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 of 2001. Soon they found that the ion exchange beds were performing so well that instead of thirty to forty days, a test point might take three 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 of 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 of 2002 the team maintained the iWRS running in the first test point configuration to support a special antibiotic study by Texas Technical University.

Control Results

We consider the use of the 3T control system a resounding success of applied AI. The resulting software ran unattended for 98.75% of the test period (6684 of 6768 hours), averaging on the order of only 6 hours downtime per month (see the following section on Lessons Learned). In an environment where the experimental hardware is being tested, this achievement is especially notable and can be explained by the combination of the modularity of our control design, and that fact that the upper layers of the architecture is written in Lisp. In this section we discuss these two aspects of the control system results along with findings concerning autonomous, unattended operations.

Advantages of Modular Design

Calibrating instruments is an example situation of how the modularity of the design limited system downtime. For each sensor and variable command output, e.g., pressure and pump speed, the skills had a linear equation to convert the A/D counts to the appropriate device value, e.g., pressure or RPM. Over time the instrument outputs drifted from those 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. Since the instruments were grouped by subsystem, we only had to bring down the given subsystem in order to restart the newly compiled skill, and then only for the few seconds required for the skill to reconnect to the IPC server.

Another situation that exploited the modularity of our design concerned a subsystem shutdown, for example, if the RO experienced a high-pressure event triggering an ASD. With the RO 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 above, whenever the RO 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 put itself in standby mode; without brine to process, the AES also put itself in standby mode. Finally, without the

RO drawing 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 down RO by eventually achieving a standby mode of operation.

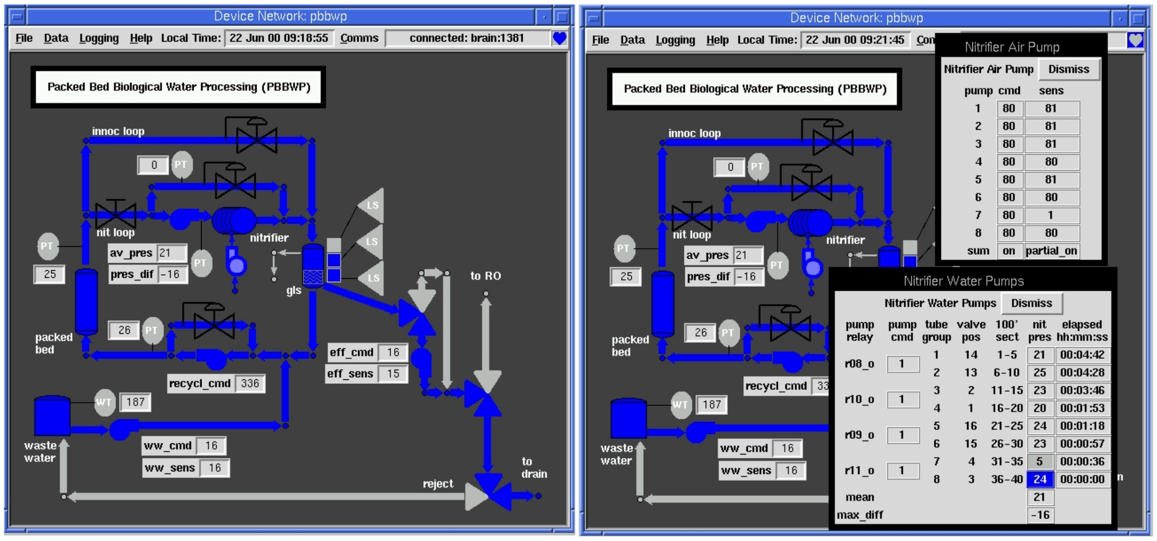
A similar situation took place when a subsystem was taken offline by the staff, such as when the AES wick was being changed out. Each subsystem could be informed through the user interface as to the availability of the upstream and downstream subsystems, and would reconfigure itself accordingly. For instance, if the AES was down, the RO 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 was down, the AES and the RO would redirect their effluent back to the feed tank or to drain, depending on the needs of the test.

Advantages of RAPs/Lisp

That RAPS is a plan interpreter, and that the higher layers 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. So we could store modified RAPs in the library, which would then be picked up the next time the RAPs processing called for them. An example of modifying a RAP concerned the operation of the AES condensate pump. In order to maintain constant operation of the PPS for as long as possible, the AES condensate was pumped to the PPS whenever the RO was not sending its effluent to the PPS, e.g., when the RO 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.

Often, new RAPs were required that were unanticipated at the beginning of the test. New 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 was the one we created to augment the computation of the average and allowed TOC carried out by the PPS skills. 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, 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 RO 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 that threshold. During that time, since no water was being deposited to the product tank, the average and allowed TOC were not updated.

Early in the test, what few high TOC spikes the PPS experienced were of relatively short duration. As the test wore on, however, the AES wicks and RO membranes began to degrade, the high TOC incidents became more frequent and lasted



Nitrifier Air Pump

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

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 11 The GUI client for the 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 be optionally displayed.

longer, and as a result, the TOC calculations were becoming less and less accurate. We needed a way to calculate the volume of water that would have flowed into the product tank in order to update the TOC calculations. Such a volume could be computed from the flow rates of the water coming from the RO and the AES, but since the PPS skills and the skills for the RO operated on two different computer racks, the PPS skills could not access the required flow rates. The obvious solution was to have the TOC calculations performed by the sequencer -- which had access to the flow data in the AES and the RO as well as the instantaneous TOC readings -- 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 re-seed the PPS TOC calculations with the values computed during the low water quality time.

One final aspect of the value of using the incremental compiler of Lisp is worth noting. Frequently, in the early months of the test, the test engineers would desire additional information to be displayed on the main WRS monitor. Examples of additional data displays not called for in the original design include the RO stage elapsed time (upper right of Figure 10) and the allowed and average TOC (lower left of Figure 10). Since 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 above 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.

Staff Acceptance of Autonomous Systems

The 3T control approach is designed for autonomous systems. But in all previous tests, the ALS teams maintained 24-hour vigilance using three eight-hour shifts daily. 3T ran autonomously in these tests, but since human presence was required for the non-automated ALS subsystems, the ALS teams never had to relinquish total control to the software. So early in the iWRS phase, as the hardware groups were setting their test goals, the controls group put forth a control goal of 95% unattended operations as a way of getting the water team to start thinking about autonomous operations. During the BWP-RO functional tests, due to a faulty power supply on one of the controls racks (see Lessons Learned below), the water team manger did not feel confident to leaving the controls unattended over night. As well, for the first two weeks, when all four systems were in test in January 2001, the water team manager allowed the controls team to monitor the system remotely during the workweek, yet still required a member of the test team to be present over night and on weekends. But at the end of the first month, the water team allowed fully autonomous operations, and 3T remained in this configuration from February 2001 under the end of the test, 15 April 2002.

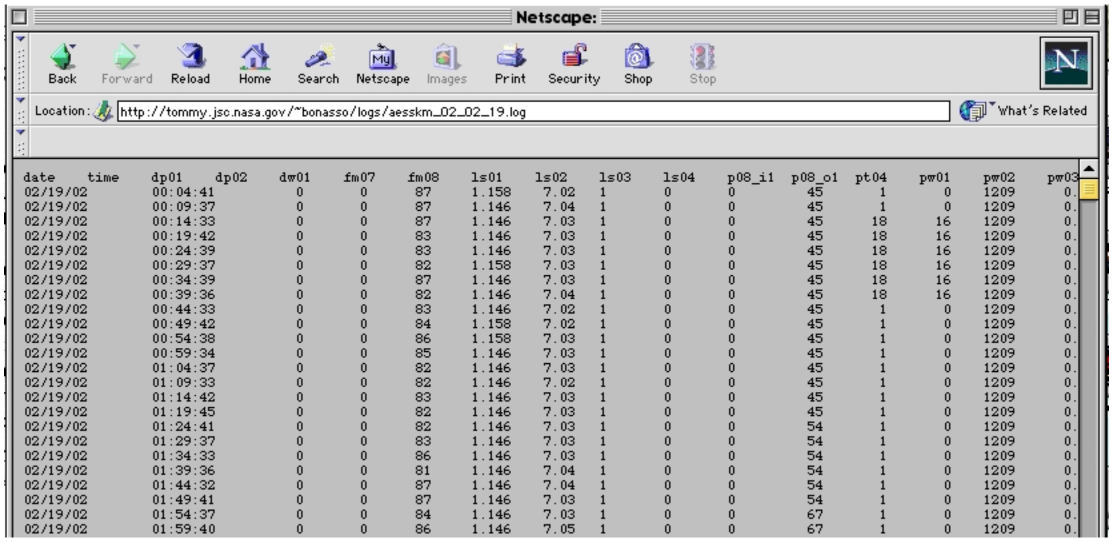
Though autonomy was the order of the day, there was a need to adjust that autonomy at various times during the test. As well, the water team's acceptance of non-vigilant, fully autonomous operations generated several new requirements to support the water staff in their data management and analysis, giving rise to the need for a 3T controls duty roster and a distributed data management system. We cover these issues in the following sections.

Adjustable Autonomy

Although the 3T controls for iWRS were able to run with full autonomy, during hardware build up, functional testing, and for 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. So we provided the test team with interactive interfaces we control engineers used for code testing. These interfaces included commands for individual pumps, valves and relays, using primitive RAPs (see Figure 9 for an example of a primitive RAP for turning a valve), commands to execute mid-level RAPs such as executing an RO 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, mid-way 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. To prevent the ASD during staff purge operations, we modified the ASD RAP to check the state of a RAP memory flag for staff purging. When the flag was present, the ASD would put out the ASD warning but would take no action. Then we added interactive text to the WRS display (see the "Purge By Staff" text in the upper left of Figure 10) that could be triggered to set the staff purge flag in memory and start a twenty-minute timer. At the end of the twenty minutes, the timer code would remove the flag.

Data Management & Distribution System



```
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 12 A browser view of the broadcast data from the AES. Data was normally recorded at 5 minute intervals.

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 both during and after the test. We developed a graphical user interface (GUI) for each subsystem to display in analog form the data broadcast from the skills (see Figure 11), and also to set the logging rate for each subsystem. Menu options on these displays allowed a user to view logged data, to setup strip charts, and to plot any data item being logged.

These GUIs were run on computers in the water laboratory (see Figure 7), but since the controls for the iWRS ran unattended, the engineering staff of the water team desired to view these displays on their PC workstations in their offices. In response to this requirement, we ported the GUIs to the Windows environments used by the staff and installed the code on their workstations. Using these GUIS, the staff could log data from any or all subsystems to their computers as they desired, while the logs of record were maintained in the water lab. Throughout the test, new logging requirements from the water team changed the format of the data and also the variables displayed in the GUIs. The format of the logs allowed viewing from the GUIS, from a browser (see Unattended Operations below) and from Microsoft Excel, a favorite analysis tool of ALS engineers. To give the staff access to the latest GUI code, we made the changes available via a web page. A prompt when a GUI started up would allow the staff to download the new code and to update their GUI display accordingly.

Unattended Operations: Supporting Intermittent Monitoring

Although the 3T control system ran unattended, the control team had to periodically monitor the system for power failure or hardware problems. We set up a duty roster of 3T control engineers to monitor the system. Every six hours, every day including weekends, the control engineer on duty would check in on the system. The engineer could start up the GUI clients on his remote computer 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 (see Figure 12) so that the on duty control engineer could monitor the system from computers 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 that 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 24/7 operations allowed us to expand our understanding of how to support intermittent monitoring.

Maintaining System Awareness

The subsystem GUIs (e.g., see Figure 11) 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 name, component values for a computed value) by clicking on the schematic component in question.

Reviewing Performance History

While monitoring data only intermittently, it was important for the iWRS engineers 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 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 can function autonomously most of the time, but also that it is able to recognize when failures occur and to 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 timestamps 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 (see Figure 8), allowing operators to refresh their understandings of how the controls work and providing access to device nomenclature used in standard iWRS drawings used in the hardware specifications. In addition, the lead controls engineer wrote up some very helpful operations notes which 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.

Lessons Learned

3T was designed for the intelligent control of autonomous robots, robots that never ran for longer than a few hours at a time. Our experience in applying this architecture to long duration ALS control systems and the WRS 3T system in particular has given us insights into the key characteristics of ALS systems and implications of those characteristics for control.

ALS Systems Have 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 three or four 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 five minutes to heat the air circulating in the AES and upwards of ten minutes to cool down. As a result, we developed our sequencer procedures as essentially a mixture of activation steps and monitors as opposed to the normal sequence of primitive enable and wait-for clauses. When one of these long-term events, e.g., waiting for the oxygen concentrator to become operational, failed, it was often due to the fact that over the length of the test, the device was just taking several seconds longer to activate. We became quite adept at recognizing this "activation drift" as the test went on.

Related to the long activation response times is the fact that the WRS system level events occurred on the order of hours or days. So to find out if a system level change was having the desired effect we often waited for days or weeks. An example of this had to do with determining the optimum number of RO cycles before having the controls perform a membrane slough. An RO cycle typically completed every four and half hours. At the beginning of the test, the system was directed to slough the membranes every eight cycles, or every 36 hours of processing. As the test continued, the quality of the RO output water, called the permeate, tended to be worse toward the end of the last two cycles. This suggested requiring a slough 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.

Yet, sometimes, the control system had to "catch" fast moving events. For example, in determining the optimal number of cycles between sloughs the RO engineer needed to correlate the permeate water quality with a count of how many sloughs had taken place. A slough took no more than four minutes to execute, but while the skills broadcast the data every 15 seconds, the data was logged at five minute intervals to minimize the amount of data required for analysis (of the WRS system events, only the RO slough event took less than five minutes to occur). So the slough indicator -- the RO recirculation pump running in reverse at one third the normal RPM -- was often missing from the logs. To assist the RO engineer to quickly determine the number of sloughs that had occurred without having her scan through days of sequencer tracking logs, we built an event detector into the RO GUI which detected the indicator using the fifteen second data, but set it as a yes/no flag for the five minute log. This detector turned out to be important, since with a new membrane or with varying amounts of RO water being recycled in different test points, a new cycles-to-slough value had to be determined as often as every month.

ALS Systems Are 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 insure that the RO 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.

Our loosely coupled agent approach obviated the need for automated generative planning to achieve integrated control of the iWRS, but it did give rise to more complex RAPs code to handle the increased number of contexts, or system states that could arise. For instance, 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 that procedure). But integrating the AES with the rest of the WRS required an additional five methods and a rewrite of the original two.

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 will degrade.

During the twelve months of iWRS operation we witnessed the slow degradation of pressure transducers, flow meters, a dew point sensor, the AES blower and the main RO feed pump. Sometimes the ultimate failure brings the test to a halt, such as in the case of the RO feed pump. With the other equipment, the degradation is gradual and difficult to detect, since the symptoms are often intermittent. The point is that it sometimes takes months for the degradations to occur, 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 us 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 character of each of the subsystems and were able to more easily ascertain the cause of these types of anomalies.

Automation has to last longer than the hardware.

Besides loss of WRS hardware, we had to replace almost every computer used in the control system including 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", i.e., having feed water circulating around the colonies. The power supply to the rack controlling he BWP began to fail during the BWP-RO phase of testing in the spring of 2000. Early on, the only indication there was a problem was that the rack CPUs would reset, zeroing the pump speeds, thus halting feed water to the BWP. When this happened after the last human check around 11 pm nightly, the water laboratory personnel would arrive in the morning to find the colonies destroyed. The first failure required a two-week re-inoculation of the BWP; as a result, the team 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 running open-loop. *The team decided that the chance of a BWP or RO 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.

All software will have memory leaks.

Most software developers delivering an application will write their code carefully enough to make efficient use of resources. But there may be inefficiencies in the resulting code that will not appear with the normal amount of debug testing. Such inefficiencies have a cumulative effect and will not make themselves felt until after weeks of operation. We discovered 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 with releasing those resources. Memory leaks were discovered in the skill managers, the IPC clients and in the sequencer. The lesson here is where possible run the code with memory metering wherever possible for several days before delivery to detect memory leaks.

Safety shutdowns are required at the subsystem level.

No matter the number of precautions taken to prevent system failure, there was always a set of variables outside of our control. Chief among these were network problems and power failures. We discuss the former here and the latter in the next section.

Every six weeks or so, over the course of a twelve-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 around 11 pm), and before the laboratory personnel arrived in the water lab six hours later. With the sequencer down its messages 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 on the pumps and valves on the A/D boards. In two instances, the failure occurred while the condensate pump was on, and 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 RO action and the RO drew air from the BWP GLS at high pressure into its membranes, rendering them useless.

The solution to these network failures was to make the skill managers aware of the loss of communications with the sequencer and execute a safing of their respective subsystem. 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, e.g., the AES skill manager would turn off its heaters and the condensate pump; the BWP would reconfigure itself to stand alone and turn 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. 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 makes for faster, less error prone restarts.

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. Since the staff 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 not arriving until 4 or 5 am), a power failure posed little problem.

However, the loss of power as well as the numerous times the iWRS had to be halted due to hardware failure -- about twenty-five times during the course of the test -- it became important to be able to restart the system quickly, without having to manually determine the state the system was in before the power loss or the hardware failure. So we wrote a RAP to log the internal state of the iWRS every thirty seconds. When the staff restarted the system, the sequencer read the last state of all the subsystems and determined how to resume operations. This was possible because 1) Each primitive checks the condition of the valve or pump before commanding the device, and 2) the RAP executive will skip steps in a procedure that have been obviated by outside or serendipitous events. Thus to restart the RO for example, the sequencer might determine from the logged state file that RO was last stopped twenty 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.

Support for Intermittent Monitoring

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. While 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 that communication. This integrated information was not only available in the GUI and in the log, but it was also sent to subscribing email 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. 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 possible similar activities remotely.

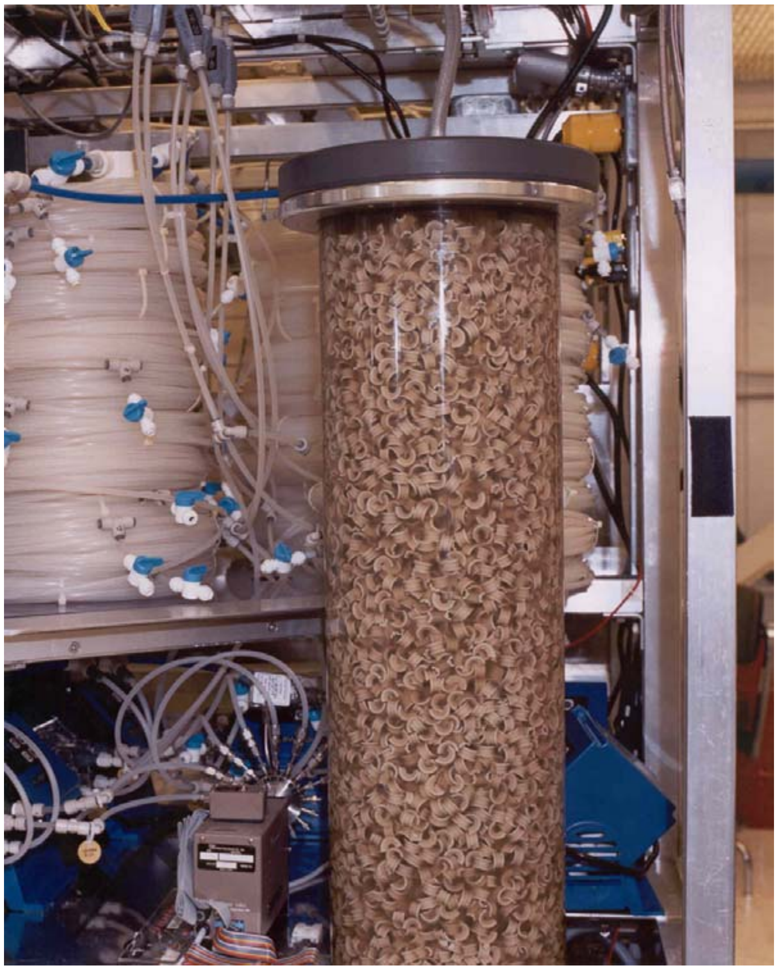


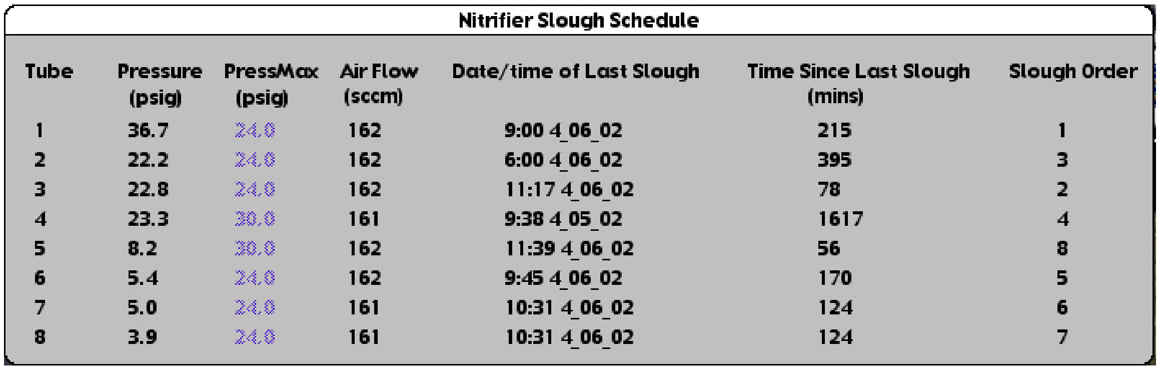
Figure 13 Symbiotic biological reactors. Feed water flows from the packed bed on the right through the nitrifier (eight 300' coils of thin tubes on the left and to the rear), through the GLS (not shown) and back to the packed bed. The nitrifying reactor contains bacteria that use oxygen gas (O_2) to break down the ammonia (NH_4) in the feed water into nitrates (NO_3) and nitrites (NO_2)**.** Bacteria on the ceramic saddles in the packed bed remove organic carbons from the feed water using oxygen from the nitrate and nitrite molecules provided by the nitrifier. The resulting nitrogen and carbon dioxide gases are released in the GLS.

The Best AI Win

The most rewarding experience in the two-year iWRS 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 (see Figure 13) consisted of bacteria that grow 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 through the tubes 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 was required to come in at night and on weekends to carry out the sloughs. If the staff forgot or was 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 was a simple scheduling problem. We constructed a scheduler that once an hour checked the pressure in each tube against a maximum allowable pressure for that 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 air flow rate for a specified period to force a slough just as the human staff did in the previous manual mode. The scheduler interface (see Figure 14) allowed the staff 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 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 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.



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

Figure 14 The nitrifier slough interface. The water staff could set the maximum pressure (PressMax column) for each tube. The scheduler obtained the instantaneous pressures from the BWP data broadcast message.

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 seen a number of 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 et al., 2001a).

The two-year iWRS test has also spawned a number of research efforts. Here 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, CMU, MIT and the Naval Research laboratories. Researchers at Rice and JSC identified the periodic nature of optimizing coupled life support systems and evaluated techniques for dealing with such systems. (Kortenkamp et al., 2001a). 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., 2001b).

• 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

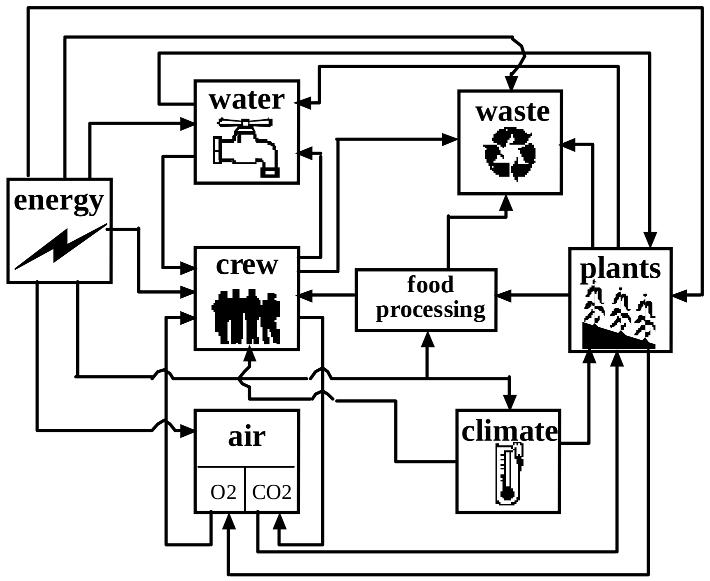


Figure 15 The interacting subsystems of an advanced life support system (Kortenkamp et al., 2001a)

project is to use probabilistic methods to diagnose failures in complex systems (i.e., the iWRS) and adapt controllers to recover from those failures.

Finally, data and information concerning the iWRS is available to AI community at large. Several simulations of the iWRS subsystems exist (e.g., (Malin et al., 2002)), and the iWRS control code, and data logs for the entire test are available to any interested party (contact Pete Bonasso at r.p.Bonasso@jsc.nasa.gov).

Summary and Future Directions

After three 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, but we have learned many useful lessons concerning autonomy, unattended operations and long duration control. While much of the hidden power of 3T may 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. While one can imagine the whole range of AI technologies being relevant in this area, we list a few which 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 of the water laboratory. Yet 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 gives rise to issues concerning authorization and authentication, managing conflicting commands from two or more authorized users, and providing timely feedback to the user as to the results of his command. The distributed crew interaction project mentioned above (Schreckenghost et al., 2002) is currently investigating these issues.

• Planning & scheduling complex ALS operations. The iWRS is but one of several systems which make up a complete life support facility for either space or planetary crews. Each of the systems in Figure 15, 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 above 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/test team will be hard pressed to keep track of these changes without automated assists. Machine learning techniques will be necessary to provide those assists, particularly since 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 they would if they 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 those multiple interfaces such as the one shown in Figure 10. This aspect of unattended operations points to the need for more natural interfaces, ones that are multi-modal, flexible and with which the user can enter a dialog for discussing the state of the system and any actions which may need to be taken

• Smart data interfaces. As alluded to in our lessons learned, while the analog data displays and their associated data logs (see Figure 11 and Figure 12) 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 which 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 et al., 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 should grow into full-fledged proxy agents for each user, e.g., 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 him 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.

References

Bonasso, R. P. 2001. Intelligent Control of a NASA Advanced Water Recovery System. In *Proceedings of the The 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space: A New Space Odyssey*. Montreal.

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: 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. In *Proceedings of the Innovative Applications of Artificial Intelligence*. Seattle, WA.

Digitool, Inc. 1996. *Macintosh Common Lisp reference Manual*, Cambridge.

Dorais, G. A. and Kortenkamp, D. 2001. Designing Human-Centered Autonomous Agents. In *PRICAI Workshop Reader*, Kowalcyk, R., Lake, S. W., Reed, N., and Willaims, G., Eds. 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*, Neodesic, Inc., Chicago.

Gat, E. 1998. Three-Layer Architectures. In *Mobile Robots and Artificial Intelligence*, Kortenkamp, D., Bonasso, R. P., and Murphy, R., Eds.: AAAI Press.

Kortenkamp, D., Bonasso, R. P., and Subramanian, D. 2001a. Distributed, Autonomous Control of Space Habitats. In *Proceedings of the IEEE Aerospace Conference*.

Kortenkamp, D., Milam, T., Simmons, R., and Fernandez, J. L. 2001b. 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. Albuquerque, NM: IEEE.

Malin, J., Flores, L., Fleming, L., and Throp, D. 2002. Using CONFIG for Simulation of Operation of Water Recovery Subsystems for Advanced Conrtol Software Evaluation. In *Proceedings of the 32nd International Conference on Environmental Systems*. San Antonio, TX.

Schreckenghost, D., Bonasso, R. P., Hudson, M. B., and Kortenkamp, D. 2000. Activity Planning for Long Duration Space Missions. In *Proceedings of the AAAI Workshop on Representational Issues for Real-World Planning Systems*.

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

Schreckenghost, D., Edeen, M., Bonasso, R. P., and Erickson, J. 1998b. Integrated Control of Product Gas Transfer for Air Revitalization. In *Proceedings of the 28th International Conference on Environmental Systems*. Danvers, MA.

Schreckenghost, D., Malin, J., Thronesbery, C., Watts, G., and Fleming, L. 2001. Adjustable Control Autonomy for Anomaly Response in Space-based Life Support Systems. In *Proceedings of the IJCAI-2001 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*. Seattle, WA

Schreckenghost, D., Martin, C. E., Bonasso, R. P., Kortenkamp, D., Milam, T., and Thronesbery, C. 2002. Supporting Group Interaction Among Humans and Autonomous Agents. In *Proceedings of the AAAI-02 Workshop on Autonomy, Delegation, and Control: From Inter-agent to Groups*. Edmonton, Canada.

Schreckenghost, D. and Thronesbery, C. 1998. Integrated Display Supervisory Control of Space Operations. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting*. Chicago, IL.

Simmons, R. and Dale, J. 1997. *Inter-Process Communication: A Reference Manual. IPC Version 6.0*: CMU Robotics Institute.

Thronesbery, C., Christoffersen, K., and Malin, J. 1999. Situation-Oriented Displays of Shuttle Data. In *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting*. Houston, TX.

Thronesbery, C. and Schreckenghost, D. 1998. Human Interaction Challenges for Intelligent Environmental Control Software. In *Proceedings of the 28th International Conference on Environmental Systems*. Danvers, MA.

Williams, B. C. and Nayak, P. P. 1996. Immobile Robots - AI in the New Millennium. *Artificial Intelligence* 17 (3).

Pete Bonasso is a senior researcher with Texas Robotics and Automation Center Laboratories (TRACLabs) where he has supported a variety of NASA projects in intelligent control since 1993. He is the co-developer of the 3T Robot Control Architecture, and has applied that architecture to the control of robotic and life support machines. Though his primary area of interest is intelligent control of robots (he is co-author of the book Mobile Robots and Artificial Intelligence), he has spent the last four years writing RAPs programs in 3T for ALS water processing immobots. He is current involved in building Lisp agents in a distributed agents research project.

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 had developed traditional requirements documents for large Department of Defense projects as well as software prototypes for innovative, small-scale projects. Dr. Thronesbery has developed intelligent systems for NASA/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 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.

David Kortenkamp is a senior research scientist at Metrica Inc., supporting NASA 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. Dr. Kortenkamp directs nearly $800,000 in annual research projects and is on a number of conference and workshop program committees. He is co-author of the book Mobile Robots and Artificial Intelligence and is associate editor of the MIT Press series on Intelligent Robotics and Autonomous Agents.