

Intelligent Control of A Water Recovery System: Three years in the Trenches

By Pete Bonasso, Dave Kortenkamp and Carroll Thronesbery

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

The AI controls engineer frowned at the speaker, a young M.E. 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, chemical and fluids 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 iWRS, slated to begin in January of 2001. We were building an AI control system which for that test 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 things that happened for which we were ill prepared and we would come away with a much better appreciation for what we had undertaken.

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

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 the area of advanced life support (ALS). In 1995, they put a man in an airlock linked to a fifteen-foot diameter chamber full of wheat (Lai-fook & 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. Our

control system -- the first for ALS -- monitored and provided caution and warnings (C&W) for the climate and nutrient environment for the wheat crop.

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

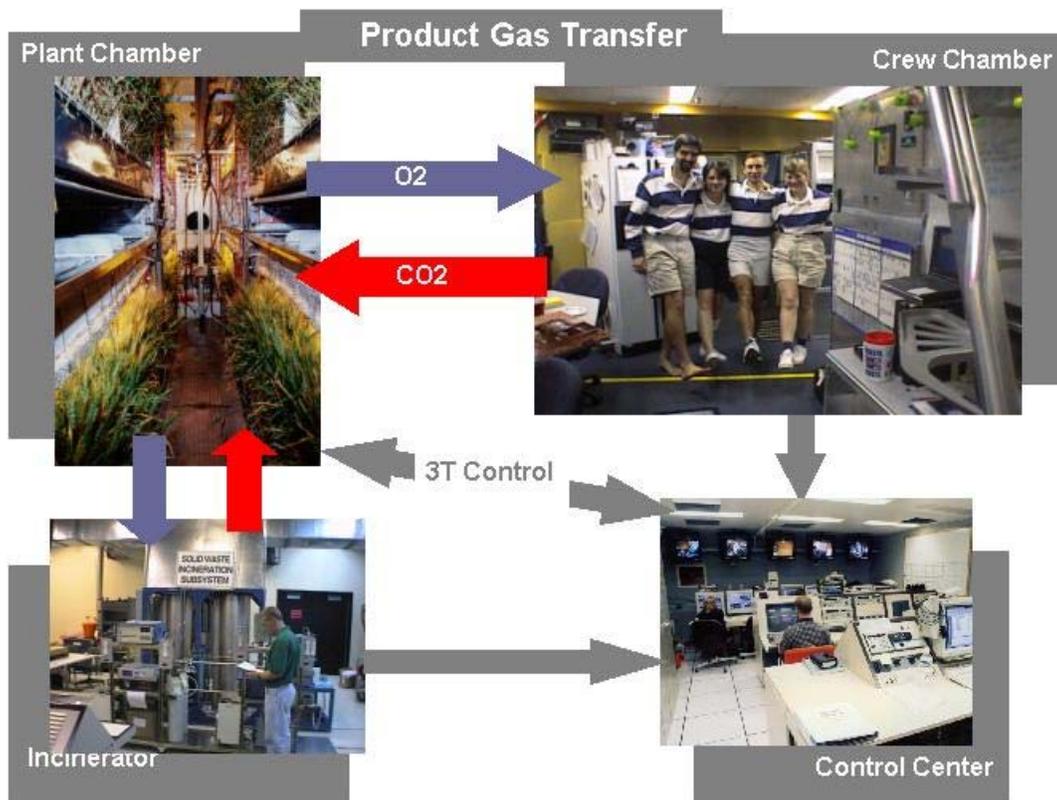


Figure 1. The Product Gas Transfer Environment.

For both of these projects we used a three-layer architecture (Gat 98) 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

The ALS control system uses the intelligent control software for autonomous systems known as 3T (Bonasso et al, 1997) (see figure), which separates the general robot intelligence problem into three interacting tiers (see Figure 2):

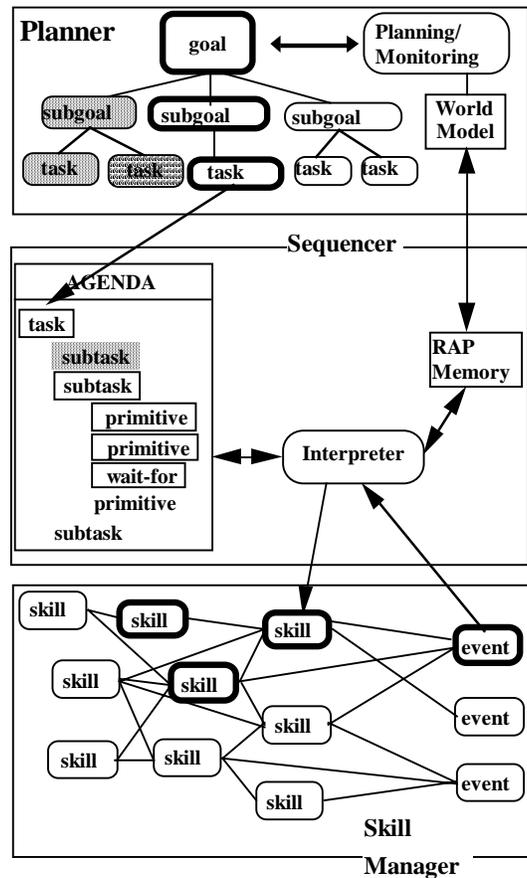


Figure 2. The 3T AI Control Architecture

◦ 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 VME analog to digital (A/D) conversion cards in the controls rack co-located with the CPUs (we are using a 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 99) 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 (HTN) planner known as AP (Elsaesser & Sanborn 90). 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 & James 97). 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 signals when any of the states it must achieve are lost; 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.

END 3T SIDEBAR

.....

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, 1998b). In 1999, however, we began to support advanced water recovery projects that were build water processing subsystems from the ground up. As a "charter member" of the water research group, the AR&SD AI team was able to play an influential role 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 starting from the analog-to-digital (A/D) converter boards used by the sensors and devices of the developing systems.

In the summer of 1999 we used the bottom two layers of 3T to provide autonomous control for 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 yearlong integrated water recovery system test.

.....

Water Recovery System Sidebar

Advanced Water Recovery System

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. Figure 3 shows the four subsystems used in the integrated WRS test. The iWRS is comprised of 1) a biological water processor (BWP) to remove organic compounds and ammonia; 2) an 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 IRS; and 3) a post processing system (PPS) to bring the water to within potable limits.

The WRS planned for use on the International Space Station is a physical-chemical system that requires a 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.

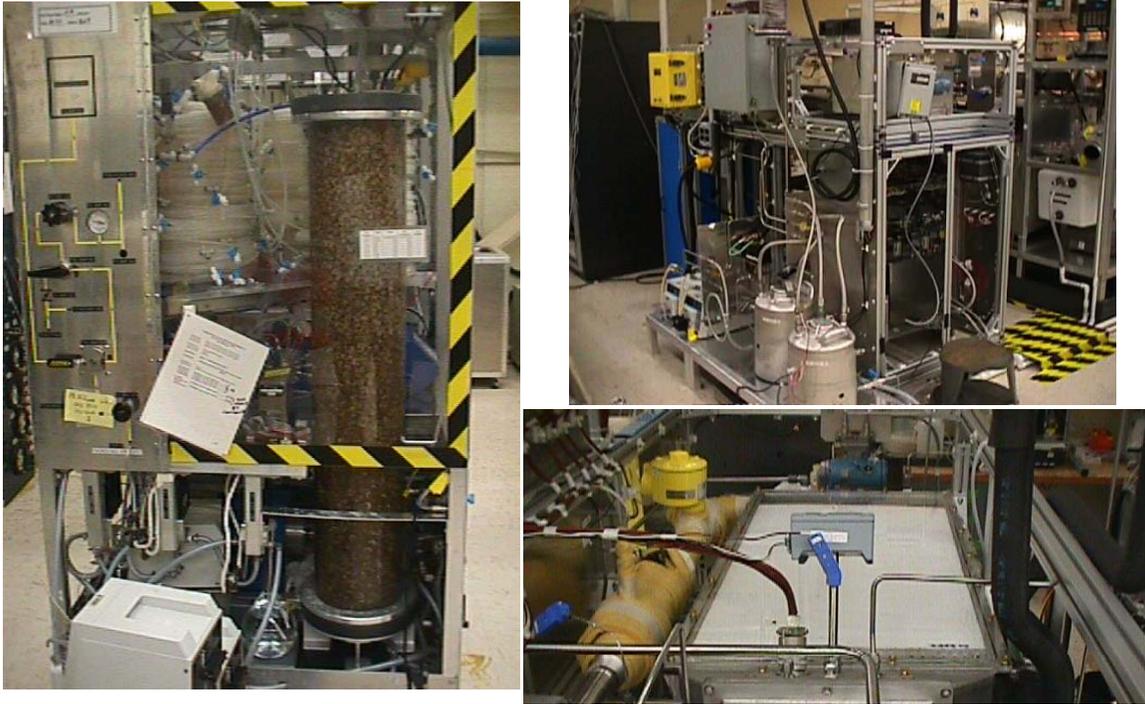
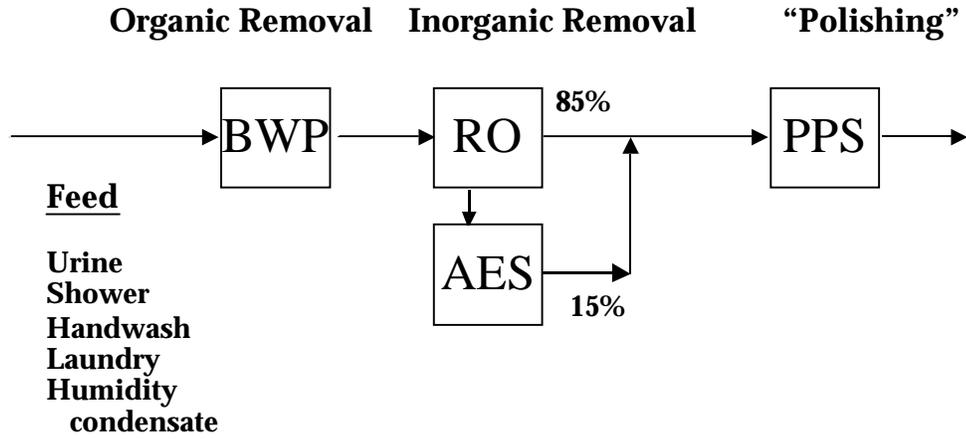


Figure 3 The AWRS subsystems. At the left is biological water processor (BWP). Upper right is the rack containing the Reverse Osmosis (RO) subsystem below, the air evaporation subsystem (AES) on top, and the post processing system (PPS) in the left rear. The bottom picture is a close up of the wick in the AES.



		<u>BWP Effluent</u>	<u>RO/AES Effluent</u>	<u>PPS Effluent</u>
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
HCO ₃	0 mg/L	500 mg/L	50 mg/L	3 mg/L (n/a)

Figure 4. The water flow paths and the target quality values in milligrams or 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.



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.

END of WRS SIDEBAR

.....

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 7).

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 that subsystem were installed in the test rack in the water lab. After testing the individual data channels, the skills developers used a skill-level command 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 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

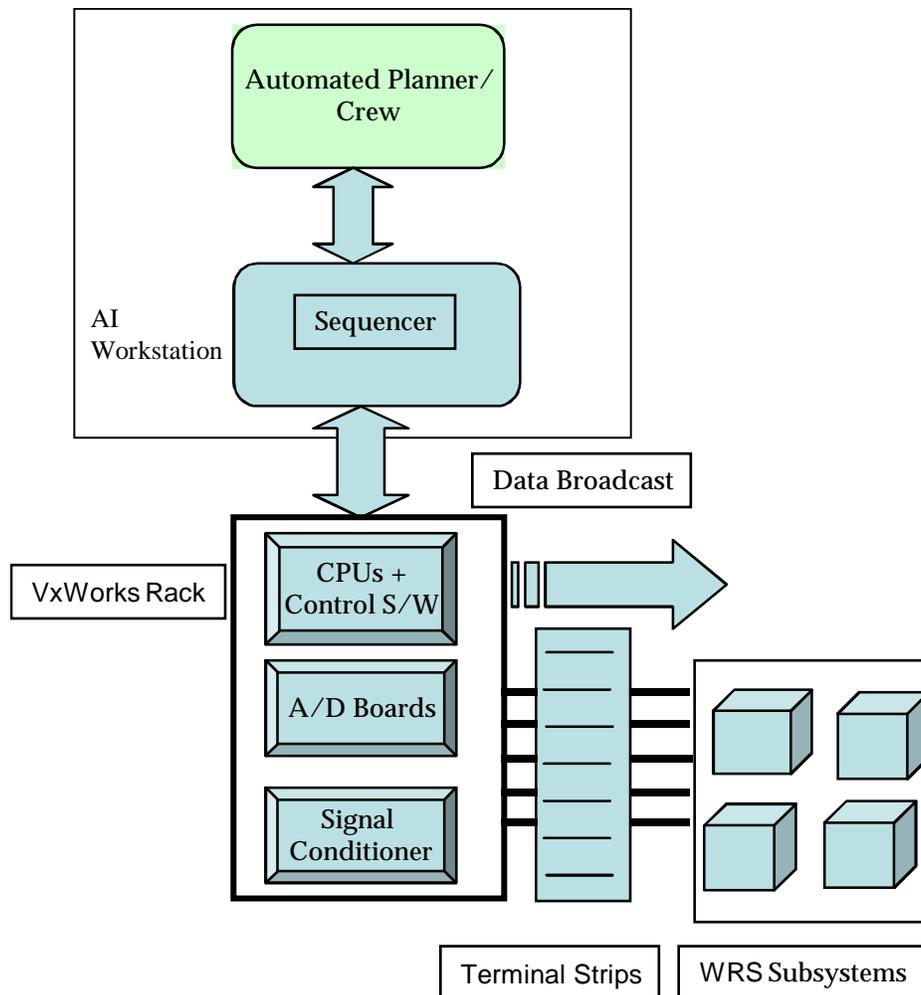


Figure 6a. The 3T Implementation for the iWRS test. For each of the WRS subsystems we developed one skill manager, which ran on its own CPU. The skill managers provided the A/D results to all of the skills modules and broadcasted that data at specified intervals for use by extant clients for analysis and review. The sequencer level managed task control (see Figure 6b), and the top level control was provided for the most part by the engineers running the test.



Figure 6b. 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 left is the display shows the GUIs for each subsystem generated by the data broadcast from each skill manager.

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_o3 to hi the rest to low, except for off when all lines will be set lo.

Figure 7. 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 8. 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 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, turns the RO process valve to the purge position and turns the pps-select valve first to the reject position and then to the tank position.

the validated skills. An example of the resulting RAPs is shown in Figure 8. The skills level remained relatively stable once the sensors and actuators were in place. We repeated the process for each subsystem. Finally, additional sequences that integrated the subsystems were developed and tested. 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 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 for 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 three systems online in December of 2000, in time to make a January 2001 start.

Controlling the iWRS System

The control tasks for the final iWRS system that went into test in 2001 can be described for each subsystem and for the iWRS as a whole (see Figure 9).

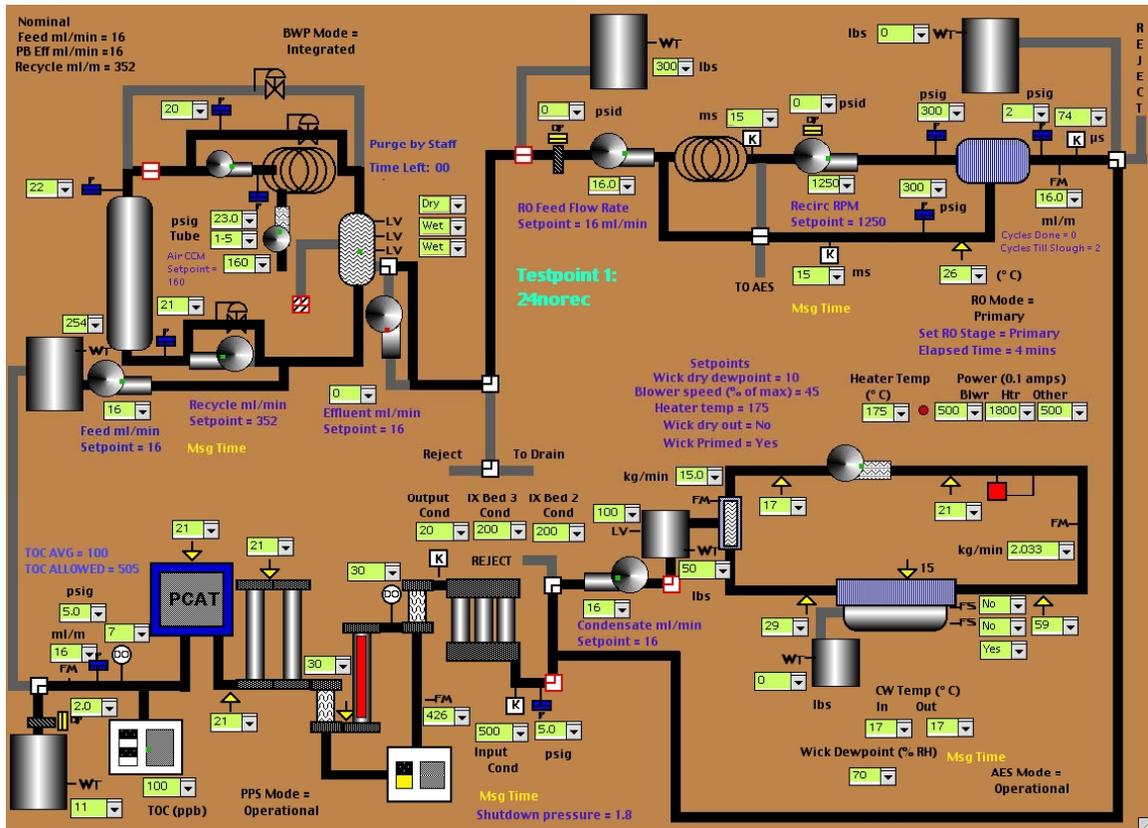


Figure 9: A Schematic Display of the iWRS as seen by 3T. The BWP is in the upper left, the RO in the upper right, the AES in the lower right and the PPS in the lower left. Gray lines indicate flow pipes; black pipes indicate water or air currently flowing. Small boxes with lines at junctures indicate motorized valves.

The BWP.

The main control task for the BWP is to keep the water in the gas-liquid-separator (GLS) 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 is the lynchpin subsystem, drawing water from the GLS and delivering permeate to the PPS and brine to the AES. 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 shown in Figure 9. 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 purged of particulate that collects 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 processes the brine in batches. When the brine fills the reservoir to the first level switch, the AES starts up, processing the brine until the switch reads dry, at which point it goes to standby awaiting another load. The ASDs concern overheating and loss of condensing fluid.

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. When the wick is spent, as indicated by the conductivity of the condensate, the AES engineers initiate a dry out procedure.

The PPS

The PPS controls monitor the input pressure. When the pressure goes above a threshold that indicates water flow from either the AES or the RO, the O₂ concentrator is started and a number of UV lamps are turned on commensurate with the measured 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 accumulation in the product tank to determine whether the PPS output should be rejected to the BWP feed tank. ASPs concern overheating of the lamps and high output conductivity indicating a breakthrough in the ion exchange beds.

Overall 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. The AES pumping condensate in the RO's stead was 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. And PPS pressure changes are corroborated by sensing the state of the pumps and valve configurations of the RO and AES. For example, if the inlet pressure is not high enough to indicate water flow but either the RO or the AES is flowing water to the PPS, then the PPS will begin operations.

Finally, when there is a complete ion exchange bed breakthrough, the PPS will notify the RO and the AES to 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 WRS sidebar). This non-potable end point occurred when the last of the three ion exchange beds started allowing water above a pre-defined level of TOC concentration. These 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; and 2 person 18 hour operation with condensate reject.

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

The first test point was begun in January of 2001. A significant finding was 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 problems encountered). 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.

Advantages of Modular Design

Calibrating instruments is a good example 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 must be recalibrated. The recalibration resulted in a new equation that had 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 and shut down. 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 will empty and the AES will stop sending water to the PPS. Without input water the PPS puts itself in standby mode; without brine to process, the AES also puts itself in standby mode. Finally, without the RO drawing from the BWP, the level in the GLS of the BWP will begin to rise, causing the feed pump to slow down to compensate. This compensation continued until the feed pump was stopped, thus putting the BWP in standby mode. Thus, all the subsystems respond to the down RO by inevitably 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 up- 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. Recall that, in order to maintain constant operation of the PPS for as long as

possible, the AES condensate is pumped to the PPS whenever the RO is 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 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 ER 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 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, and requires 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 was 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 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 solution was to have the TOC calculations picked up 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 note on the value of using the incremental compiler of Lisp. 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 and the allowed and average TOC (see Figure 9). Since the entire interface was written in Lisp (we used Macintosh Common Lisp (ref) running on a Power Mac G4), a control engineer could build, debug and install such changes to the displays 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 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.

Adjustable Autonomy

We designed the iWRS control system to run autonomously. But 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 let the interactive interfaces we control engineers used for code testing for use by the test team. These interfaces included commands for individual pumps, valves and relays, using primitive RAPs (see Figure 8 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 fluent 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 Figure 9) 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

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 interfaces (GUI) for each subsystem to display in analog form the data broadcast from the skills (see Figure 10), 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 the Linux computers in the water laboratory, 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 kept 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. 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.

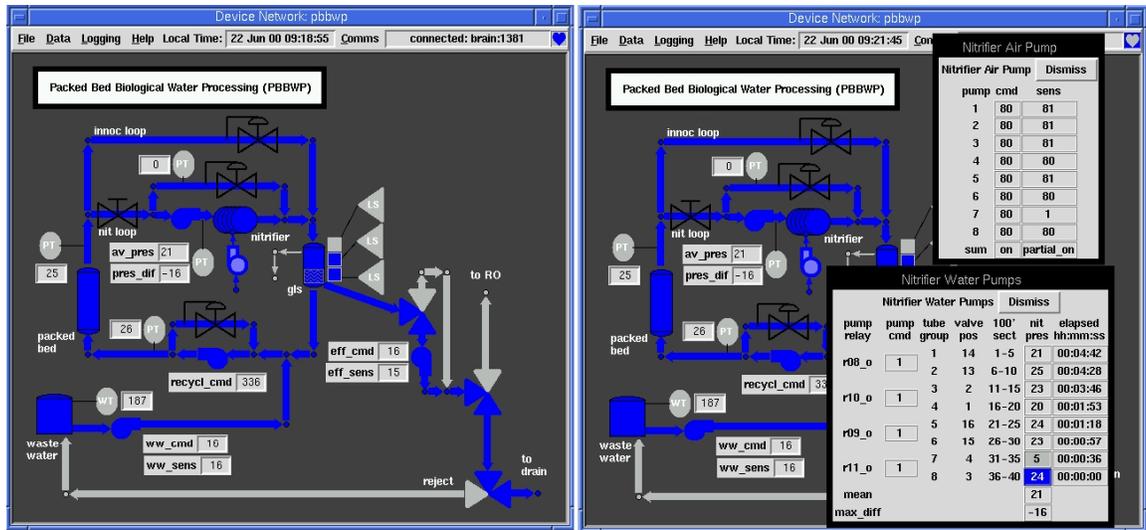


Figure 10: The GUI client for the BWP, showing the data broadcast from the BWP skill manager displayed on an analog of the BWP hardware.

Unattended Operations

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 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 11) so that the on duty control engineer could monitor the system while on travel. If there was a problem, the engineer would contact someone from the water team to handle it.

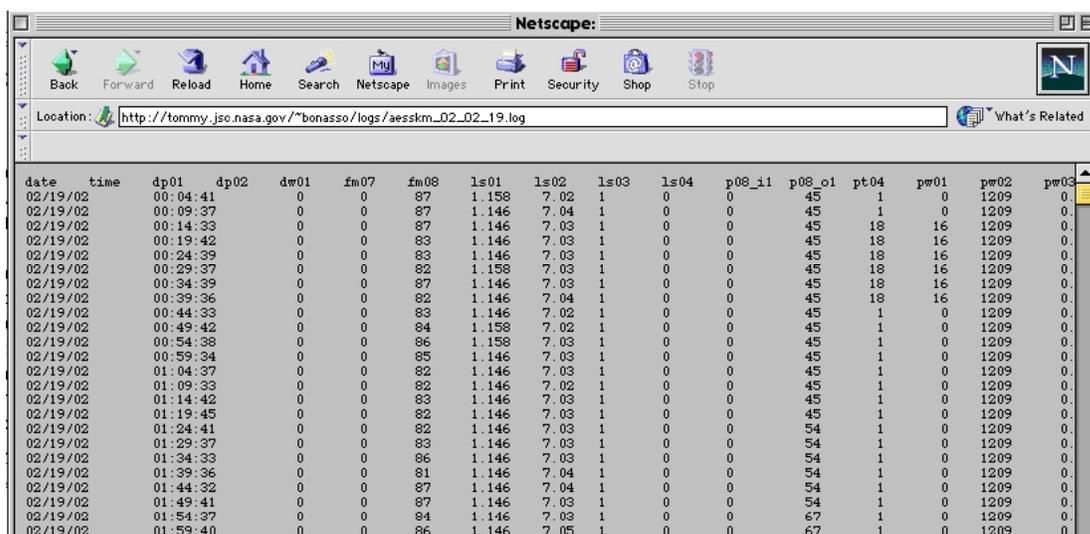


Figure 11: A browser view of the broadcast data from the AES. Data was normally recorded at 5 minute intervals.

Lessons Learned

3T was designed for the intelligent control of autonomous robots. These robots never ran for longer than a few hours. 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 differences between robots and ALS systems as well as their implications 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 warm the air circulating in the AES and upwards of ten minutes to cool down. As a result, we developed our sequence code 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 ended up taking a week and a half to determine that the cycles-to-slough should be set to four to keep the permeate quality consistently high.

In determining the optimal number of cycles between sloughs the RO engineer needed to correlate the permeate water quality with a count 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 noted 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 ASL 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

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.

1. 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 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 problem 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.
2. 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 VMW 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 a few hours without being "fed", i.e., having feed water circulating around the colonies. The power supply to the rack controlling the BWP began to fail when only the BWP and RO were in test. Early on, the only indication there was a problem was that the rack CPUs would reset, zeroing the pump speeds and thus halting feed water to the BWP. When this happened in the early morning hours, 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, and 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 for another two weeks.

3. 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 if possible run the code with memory meters wherever possible for several days before delivery to detect memory leaks.
4. 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. Five or six times 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 arrive in the water lab six hours later. Without the sequencer reading its messages, 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 a typically worst-case scenario had the failure occurred while the condensate pump was on, or 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 would pump the tank dry and start 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 would be pumped dry by the RO action and the RO would draw air at high pressure into its membranes, rendering them useless.

References

Bonasso et al 1997

R. Peter Bonasso, R. J. Firby, E. Gat, David Kortenkamp, D. Miller and M. Slack, "Experiences with an Architecture for Intelligent, Reactive Agents", *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997, pp. 237-256.

Elsaesser & Sanborn 90

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

Firby 99

Firby, James R. 1999. *The RAPS Language Manual*. Neodesic, Inc. Chicago, IL.

Gat 98

Gat, Erann, "Three-Layer Architectures" in *Mobile Robots and Artificial Intelligence*, ed. David Kortenkamp, R. P. Bonasso, and Robin Murphy, AAAI Press, 1998.

Lai-fook & Ambrose 1997

Kristin M. Lai-fook and Robert O. Ambrose. 1997. Automation of Bioregenerative Habitats for Space Environments, in Proceedings of the IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, pp 2471-2476.

Schreckenghost et al, 1998a

Schreckenghost, D; Edeen, M.; Bonasso, P.; and Erickson, J. Intelligent Control of Product Gas Transfer for Air Revitalization. Proceedings of 28th International Conference on Environmental Systems. July 1998. Danvers, MA.

Schreckenghost et al, 1998b

Schreckenghost, D; Bonasso, P.; Kortenkamp, D.; and Ryan, D. Three Tier Architecture for Controlling Space Life Support Systems. IEEE Symposium on Intelligence in Automation and Robotics. May, 1998.

(Simmons & James 97).

Simmons, R. G. and D. James, "Inter-Process Communication: A Reference Manual. IPC Version 6.0," CMU Robotics Institute, 1997.

(Williams and Nayak, 1996)