Architectures and Evaluation for Adjustable Control Autonomy for Space-Based Life Support Systems

Jane T. Malin
Technical Assistant to Branch Chief
Automation, Robotics & Simulation Division
NASA Johnson Space Center/ER
Houston, TX 77058

Debra K. Schreckenghost
Research Scientist
Texas Robotics & Automation Center
NASA Johnson Space Center/ER
Houston, TX 77058

KEYWORDS

Automation, Autonomy, Fault Management, Fault Isolation, Detection and Recovery

ABSTRACT

In the past five years, a number of automation applications for control of crew life support systems have been developed and evaluated in the Adjustable Autonomy Testbed at NASA's Johnson Space Center. This paper surveys progress on an adjustable autonomous control architecture for situations where software and human operators work together to manage anomalies and other system problems. When problems occur, the level of control autonomy can be adjusted, so that operators and software agents can work together on diagnosis and recovery. In 1997, adjustable autonomy software was developed to manage gas transfer and storage in a closed life support test. Four crewmembers lived and worked in a chamber for 91 days, with both air and water recycling. CO_2 was converted to O_2 by gas processing systems and wheat crops. With the automation software, significantly fewer hours were spent monitoring operations. System-level validation testing of the software by interactive hybrid simulation revealed problems both in software requirements and implementation. Since that time, we have been developing multi-agent approaches for automation software and human operators, to cooperatively control systems and manage problems. Each new capability has been tested and demonstrated in realistic dynamic anomaly scenarios, using the hybrid simulation tool.

INTRODUCTION

In the past few years, NASA has been investing in research and development to produce safe, autonomously operating life support systems. During manned exploration missions, intelligent autonomous control of air and water processing systems will reduce crew workload and increase crew independence from Earth-based support. In the Adjustable Autonomy Testbed, an intelligent layered and distributed autonomous control architectures has been developed, demonstrated and used. The autonomous control has been enhanced to be adjustable, so that the system and crew can work together

to resolve problems and perform maintenance tasks. Simulation-based methods have been used for dynamic testing of intelligent control software in complex system-level scenarios.

AIR PROCESSING IN SPACE-BASED LIFE SUPPORT

Advanced Life Support for space systems is designed to maximize the recycling of resources from waste products. The goal is to recover virtually all oxygen (O_2) from the carbon dioxide (CO_2) respired by the crew of a spacecraft or space station node. Three major and interrelated subsystems have been used to convert CO_2 to O_2:

- Variable Configuration Carbon Dioxide Removal (VCCR) System: CO_2 is removed from the crew cabin atmosphere by adsorption into a molecular sieve sorbant bed and then desorbed from the bed under low pressure and stored in an accumulator for recovery of the O_2.
- Carbon Dioxide Reduction System (CRS): reacts CO_2 removed from the cabin atmosphere with hydrogen (H_2) to produce methane (CH_4), which is vented, and water, which is transferred to the Water Recovery System (WRS).
- Oxygen Generation System (OGS): Water from the WRS is reduced to O_2 and H_2 by electrolysis. The O_2 is returned to the crew cabin and the H2 is transferred back to the CRS where it is reused for the CO_2 reduction process.

A schematic of the flows of water and gases among the three ARS subsystems is shown in Figure 1.

Adjustable autonomy for control and fault management has been developed and demonstrated in the life support domain. The software operates in the Adjustable Autonomy Testbed, which provides a dynamic simulation of the subsystem hardware as well as the autonomous agents that control them. Scenarios for demonstration have most often focused on the VCCR molecular sieve system. Figure 2 provides a high level functional view of the subsystems of the VCCR system.

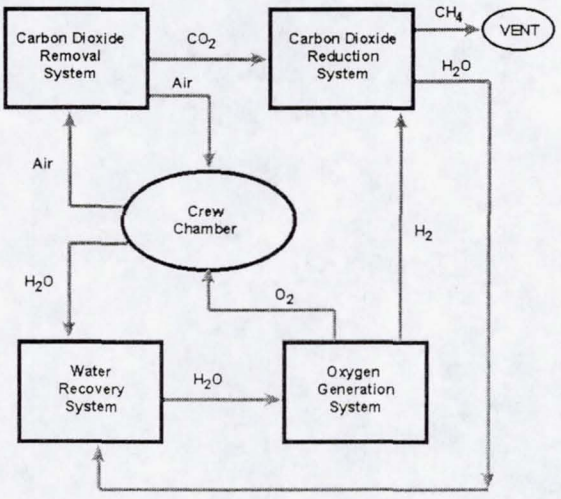

FIG. 1 - SCHEMATIC OF GAS FLOWS IN AIR REVITALIZATION SUBSYSTEM

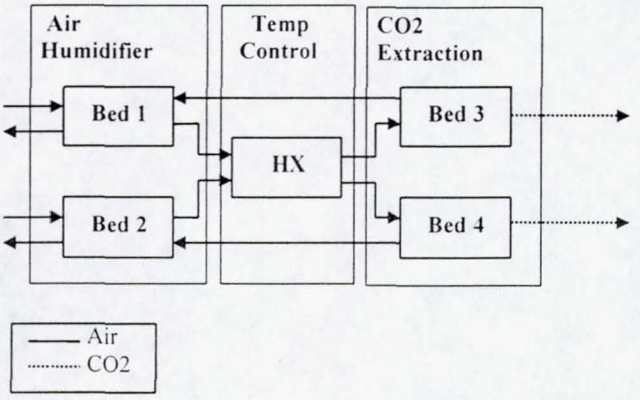

FIG. 2 - SUBSYSTEMS OF VARIABLE CONFIGURATION CO_2 REMOVAL (VCCR) SYSTEM

EXTENDED OPERATIONAL TEST OF AUTONOMY

In 1997, adjustable autonomy software was developed and deployed to manage gas transfer and storage in a closed life support test. Four crewmembers lived and worked in a chamber for 91 days, with both air and water recycling. In addition to the physico-chemical systems for gas processing, wheat crops were used to convert CO_2 to O_2, and an incinerator periodically processed biological waste, producing CO_2. Figure 3 shows the configuration of the chambers and gas storage and transfer during the test.

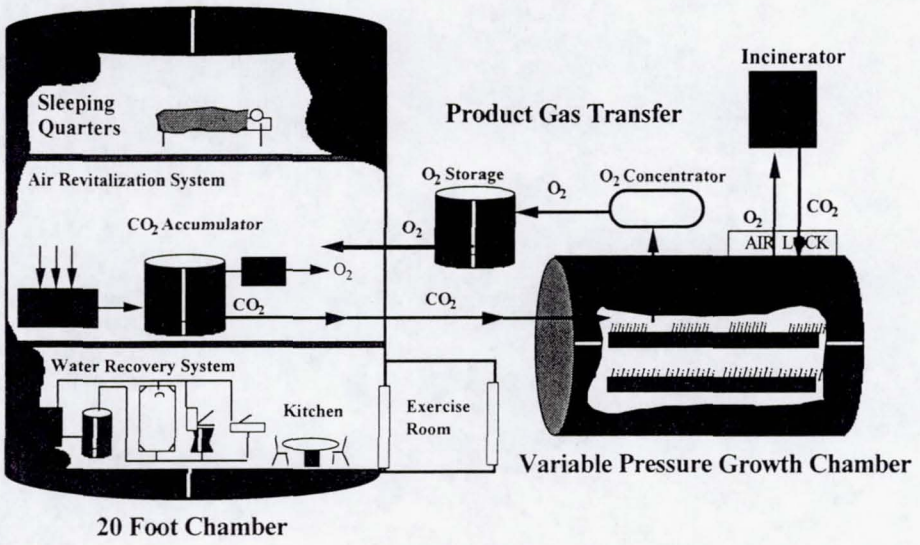

FIG. 3 - PRODUCT GAS TRANSFER IN THE PHASE III TEST IN THE LUNAR MARS LIFE SUPPORT TEST PROGRAM

AUTONOMOUS CONTROL ARCHITECTURES

An autonomous architecture developed for robotic control (Three Tier or 3T) has been modified to support adjustable autonomy (1). 3T consists of three tiers of control processing that operate in parallel. Control flows from the top tier (the Planner) down to the bottom tier (the Skill Manager) that commands the system hardware. Control feedback flows in reverse, from the bottom tier up to the top tier, which enables closed loop control at all tiers of the architecture. Each tier has the following capabilities:

Planner: Predicts activities to achieve control objectives.
Represents and assigns tasks to multiple agents.
Monitors plan execution, detects plan execution failure, and replans at failure.

Sequencer: Selects and orders procedures to implement planned activities.
Chooses procedures reactively, based on current state of environment.

Allocates procedure steps to specific skill managers.

Skill Manager: Implements procedure steps under closed loop control.
Skills are activated to issue commands to control instrumentation.
Events are activated to monitor sensor readings in response to control.

The version of this architecture that was deployed to manage product gas transfer (PGT) and storage for the manned test in 1997 is shown in Figure 4.

Agent architectures generally include a goal-oriented deliberative planner and a task-oriented procedural executive that manages low-level control. In our agent architecture research, we have integrated an additional model-based module (Livingstone) that is a specialist in system level fault detection, identification and recovery (2). Livingstone uses abstract models of system device modes and patterns of symptoms to diagnose failures and differentiate between failed parts and sensors. Livingstone can also identify new recovery configurations after a failure, with the commands required to achieve the configuration.

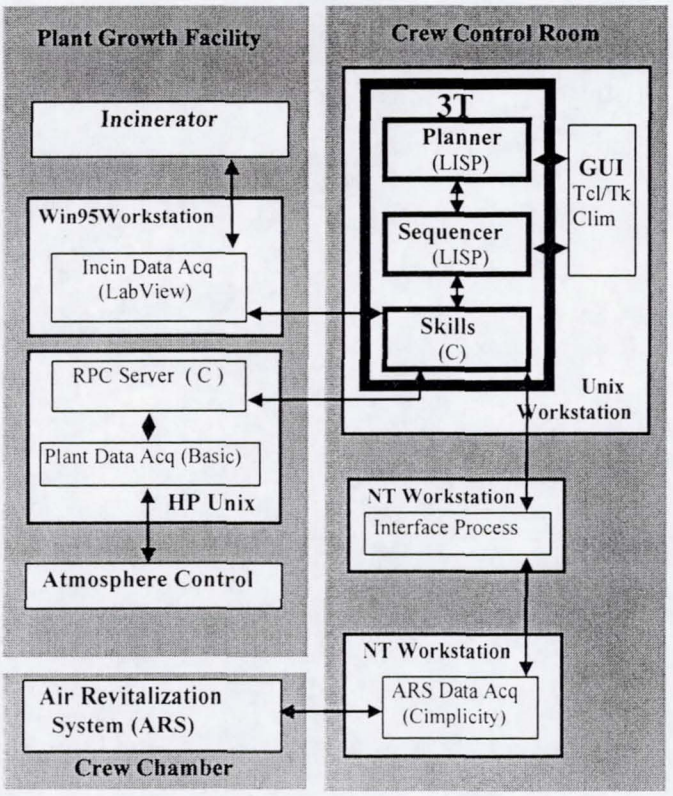

FIG. 4 - PGT CONTROL ARCHITECTURE

A version of this architecture that was used in the Adjustable Autonomy Testbed (AAT), with an integrated model-based fault management module, is shown in Figure 5.

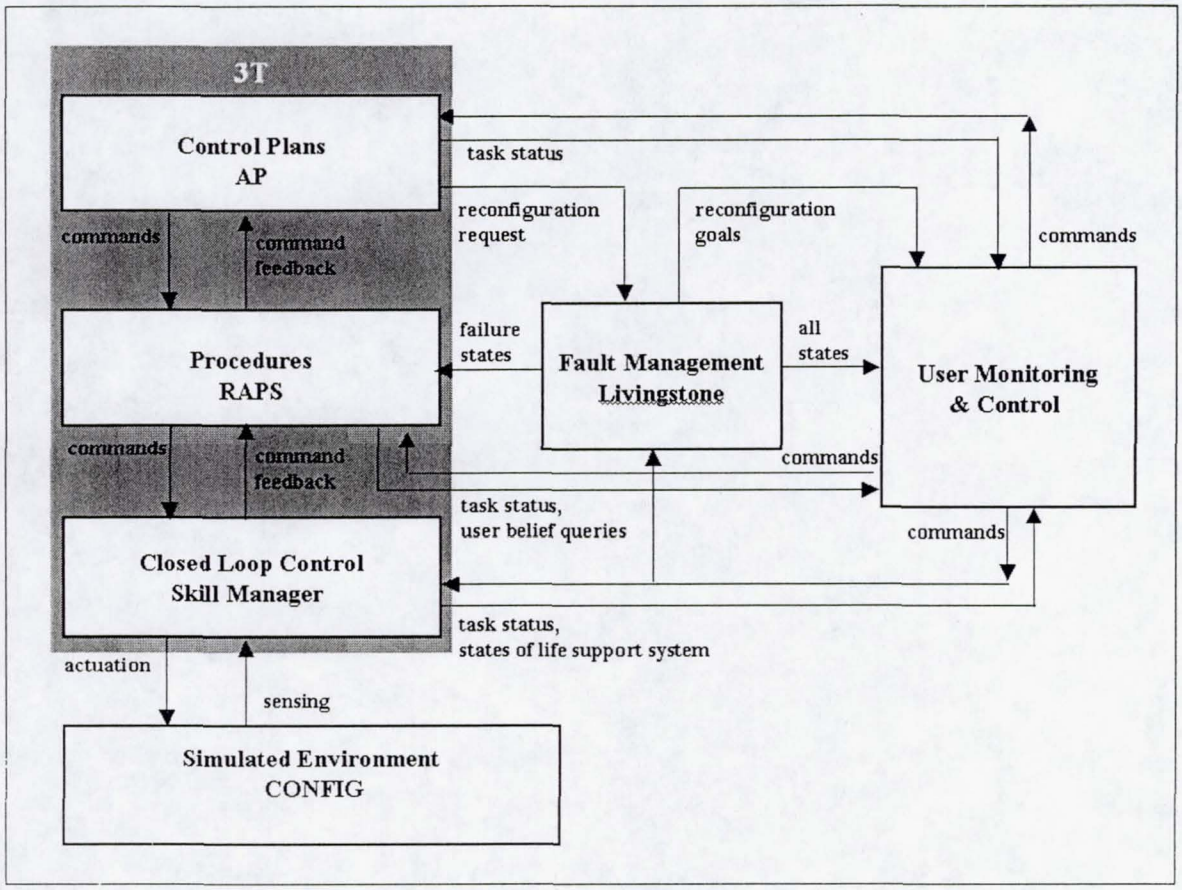

FIG. 5 - AAT CONTROL ARCHITECTURE

BENEFITS OF INTELLIGENT CONTROL OF PRODUCT GAS TRANSFER

Typically, in management of planetary systems such as life support systems, the intelligent software concurrently performs multiple tasks. For example, the Product Gas Transfer System in the Lunar Mars Life Support Project Phase III test in 1997 (3) performed the following tasks:

- Managed transfer of CO_2 and O_2 between crew and plant chambers.
- Managed CO_2 and O_2 levels in the airlock and plant chamber before, during and after periodic incineration (about every four days), by reconfiguring gas flows.
- Maintained CO_2 at appropriate levels for activities in the plant chamber.
- Maintained appropriate O_2 levels in plant chamber and pressures in storage tanks. (Independent conventional software managed the O_2 levels in the crew chamber.)
- Detected alert and alarm situations and responded appropriately.

These tasks were accomplished by reconfiguring gas flows and flow controllers, by inhibiting or starting gas transfers, and by interacting with safety software and human operators.

For the test, the control software was operational on October 6, 1997. It successfully managed the transfer of product gases round-the-clock until the end of test on December 19, 1997. Except when integrating new capability, monitoring hazardous operations, or responding to an anomaly, the system typically operated without human supervision or intervention.

The limited human role in operating the PGT system contrasts significantly with the operation of the more traditional process control software used with the other life support systems developed for the test. Operating the conventional control software was manually intensive, requiring vigilance monitoring and frequent manual adjustment of control parameters. One person was on shift for 16 hours daily to perform these tasks. In contrast to this routine, operating the PGT autonomous control typically required 6-8 hours weekly of shift work, with an additional 3 hours for each harvest and 6 hours for each incineration operation (which required manual reconfiguration of skill interfaces to receive data from the incinerator control software).

BENEFITS OF ADJUSTABLE CONTROL AUTONOMY IN ANOMALY RESPONSE

In the adjustable autonomy concept of operations, routine operations are fully autonomous, including response to expected failures. The human is not required to vigilantly monitor these operations. Activity histories and event summaries are provided for remote, occasional monitoring by the human. When interesting or anomalous events occur or a need for manual action arises, the autonomous system notifies the human. As a result, human intervention is performed *by exception* – during critical operations, anomalies, or unusual situations. The adjustable autonomous software supports a range of such intervention, from informing the autonomous system of changes to control parameters to taking over control manually. Even in the most extreme case where the human takes over control, the autonomous system continues to monitor operations to maintain knowledge of the current control state.

Figures 6a and 6b illustrate the interaction among the user and the different functional components of the autonomous architecture during a failure situation in the VCCR system (4). In Figure 6a, the

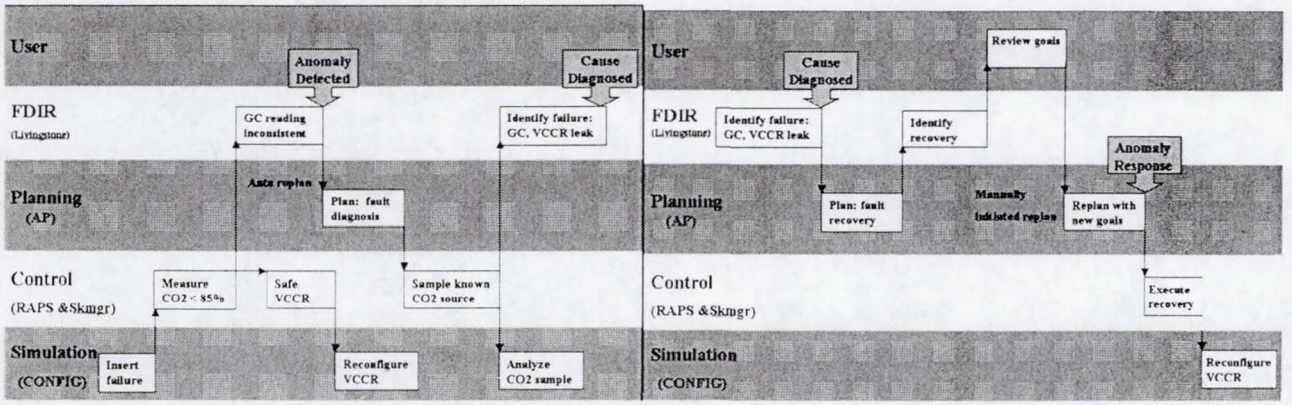

FIG. 6A - EXAMPLE OF FAULT DIAGNOSIS FIG. 6B - EXAMPLE OF FAULT RECOVERY

autonomous system detects and isolates the cause of the failure without human intervention. In Figure 6b, however, the human participates during recovery response to validate the recovery plan constructed by the autonomous system prior to its execution. This approach allocates the more routine fault detection tasks to automation, and takes advantage of human knowledge about the underlying hardware systems to guarantee that fault management procedures constructed on-the-fly have no unintended effects.

SIMULATION-BASED EVALUATION

Intelligent software for systems management is typically designed for flexible and robust operation. Such software uses its resources to make the best of unexpected problems and opportunities. It can autonomously assess a wide range of possible states and contexts and select appropriate procedure variants that are consistent with goals. In anomalous fault management situations, it may use models of expected system behavior to detect and diagnose degradations and failures. Planner and fault management modules can identify appropriate recoveries, consistent with the situation and current goals. The challenge is to evaluate this new behavior, to predict what might happen when the intelligent software executes its reactive plans and tasks in a new situation.

A combination of test and evaluation approaches is needed for validation of such advanced software. The joint behavior of the operating hardware and software is analyzed, to not only verify the software requirements but also discover missing requirements. Many accidents and major losses have been due to flaws in software requirements (5). Experience with evaluation of complex intelligent software has shown that it can be difficult to envision effects of failures and recoveries in complex highly interconnected systems. Consequently, software requirements may not be developed to handle some of these situations (6).

An incremental scenario-based simulation approach is being used in the testbed to evaluate intelligent software. Hybrid models of hardware and operations have been used to evaluate intelligent software for autonomously managing advanced gas processing systems (7). Simulation-based evaluation of the PGT intelligent software application for the Phase III Test was accomplished by running scenarios where the software interacted dynamically with the simulation model. The CONFIG hybrid simulator was used to model the life support hardware, controls and schedules, and plants and crew. CONFIG has been developed in-house in Lisp, to extend discrete event simulation with capabilities for continuous system modeling and qualitative modeling (8). CONFIG uses graph analysis to handle changes in the model structure and associated pressures and flows during a simulation, as system reconfigurations cause changes in the direction and activation of couplings between component models.

For software evaluation, an interface was established between the 3T reactive sequencer and the system model, which included low-level controllers. Nominal 120-day schedules were simulated, and PGT performance was evaluated for each software task. CONFIG supports fast scenario-based simulation of systems in operation. The ratio of simulated time to real simulator execution time was generally 20:1. Figure 7 shows a CONFIG graphical interface for the model of the system that was controlled by PGT software in the Phase III manned test. Simulated hardware failures were inserted to test the sequencer under each off-nominal condition specified in the requirements. Anomalous conditions and failures in simulated hardware and control can lead to emergent software behavior. When the software appeared to

not be operating properly, the problem was investigated and fixed. This approach uncovered problems that had been missed in more conventional software testing.

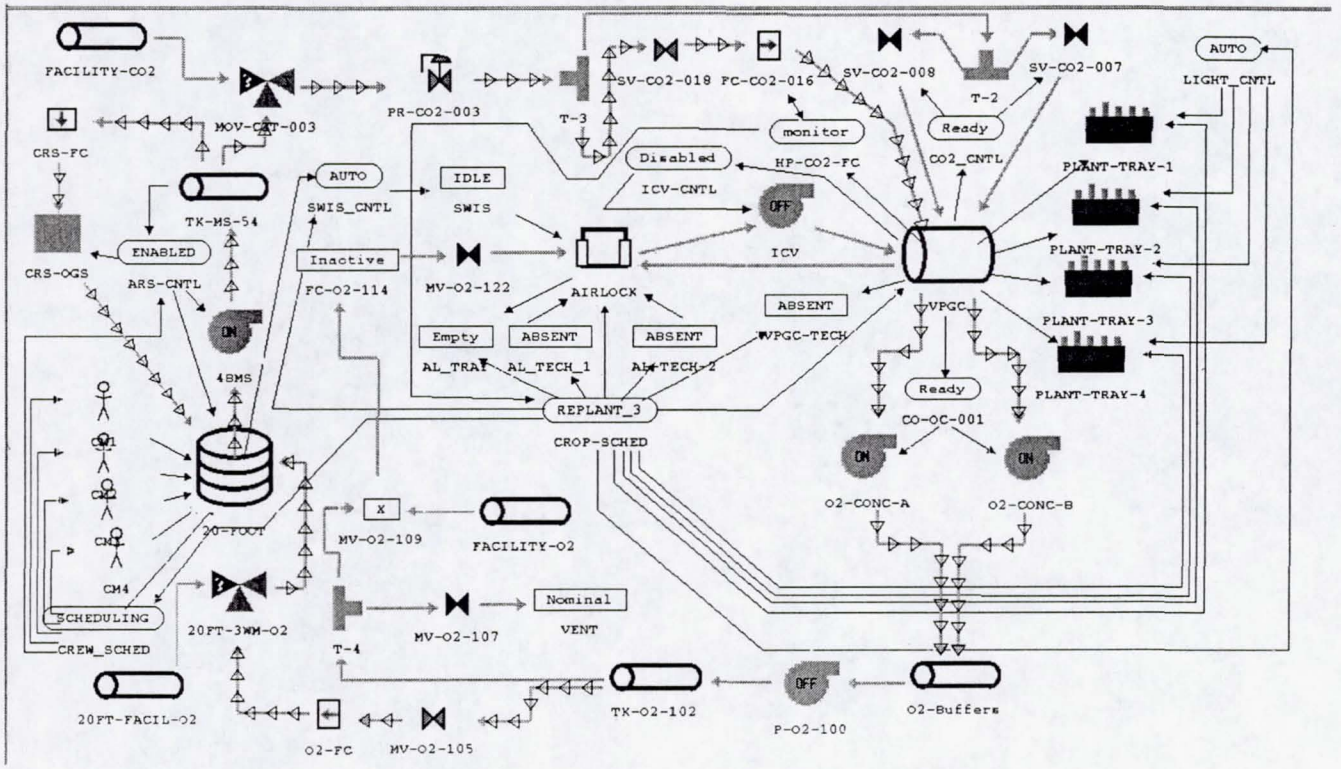

FIG. 7 - CONFIG MODEL OF PHASE III TEST HARDWARE, ENVIRONMENT, CREW AND WHEAT PLANTS

In more recent work, the interface between the models and the intelligent software has been through the low-level control layer. With this low level interface, it is not necessary to model the low-level control software, and it is possible to test all the interacting modules of the intelligent software.

System-level validation testing of the software by interactive hybrid simulation has revealed problems both in software requirements and implementation. This simulation approach has proved useful for developing and demonstrating applications of the new architectures for control of complex systems with embedded failures.

CONCLUSION: USES OF TESTBED SOFTWARE

An adjustable autonomy control architecture has been developed to support dynamic fault management with human-in-the-loop operations. This architecture has been used to control space life support systems, a type of advanced process control system. The Adjustable Autonomy Testbed software has been useful in evaluating concepts of space operations using adjustable autonomy. It also is being used to provide software, data, or an evaluation environment for a number of other NASA projects. It provides simulated space system data for investigating concepts and prototypes for assisting ground

control operators. It provides user interface software for control of life support hardware in the Advanced Water Laboratory at Johnson Space Center. It is being used as a simulated space system for evaluating the use of models of space flight crew activities. It will be used as an environment for evaluating concepts and prototypes for distributed crew interaction with space systems. It provides a simulated autonomous system control for evaluation of machine learning algorithms for control. Domain models of control from this test are being used to develop advanced model-based control software.

REFERENCES

1. Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D., and Slack, M."Experiences with an Architecture for Intelligent, Reactive Agents," Journal of Experimental Theory of Artificial Intelligence, Vol. 9, 1997, 237-256.

2. Williams, B. C. and Nayak, P. P. "A Model-based Approach to Reactive Self-configuring Systems," Proceedings of the 13th National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA, 1996, 971-978.

3. Schreckenghost, D. Ryan, D., Thronesbery, C., Bonasso, P., and Poirot, D. "Intelligent Control of Life Support Systems for Space Habitats". Innovative Applications of AI, AAAI Press, Menlo Park, CA, July 1998.

4. Schreckenghost, D., Malin, J., Thronesbery, C., Watts, G., and Fleming, L. "Adjustable Control Autonomy for Anomaly Response in Space-based Life Support Systems". Workshop on Autonomy, Delegation and Control at Int'l Joint Conference on Artificial Intelligence, August, 2001.

5. Leveson, N. *Safeware: System Safety and Computers*. Addison-Wesley, Reading, Mass., 1995.

6. Malin, J. T., Fleming, L., and Hatfield, T. R. "Interactive Simulation-Based Testing of Product Gas Transfer Integrated Monitoring and Control Software for the Lunar Mars Life Support Phase III Test." Proceedings of SAE 28th International Conference on Environmental Systems, SAE Paper No. 981769, 1998.

7. Malin, J., Nieten, J., Schreckenghost, D., MacMahon, M., Graham, J., Thronesbery, C., Bonasso, P., Kowing, J., and Fleming, L. 2000. "Multi-agent Diagnosis and Control of an Air Revitalization System for Life Support in Space". IEEE Aerospace Conference, CD, March, 2000.

8. Malin, J. T., Fleming, L. D., and Throop, D R. "Hybrid Modeling for Scenario-Based Evaluation of Failure Effects in Advanced Hardware-Software Designs," Model-Based Validation of Intelligence, Technical Report SS-01-04, AAAI Press, Menlo Park, CA, 2001.