Intelligent Control of Life Support Systems for Space Habitats

Debra Schreckenghost, Daniel Ryan, Carroll Thronesbery, Peter Bonasso, Daniel Poirot

Texas Robotics and Automation Center
NASA Johnson Space Center/ER
Houston, TX 77058
d.schreckenghost@jsc.nasa.gov

Abstract

The Interchamber Monitoring and Control (IMC) system is semi-autonomous, intelligent software that controls life support systems designed for recycling air in remote space habitats. The IMC system was developed using the 3T autonomous control architecture. This architecture integrates traditional control with reactive task sequencing and deliberative planning technology. The IMC system was used during the Phase III test of NASA's Lunar/Mars Life Support Test Program (LMLSTP). For this test, four crew members lived in a closed habitat for 91 days. The objective of using an intelligent control system was to reduce the need for crew involvement in nominal control of life support by automating control operations. Prior to this test, manually intensive traditional process control software had been used to control LMLSTP life support systems. It is impractical and inefficient for crew at a remote site to continuously monitor day-to-day operation of life support systems. Our intelligent control software autonomously handles nominal and expected anomalous situations. The crew only intervenes in exceptional or novel situations. During the Phase III test we demonstrated the viability of using intelligent control for such automation.

Description of Life Support Tasks

The Interchamber Monitoring and Control (IMC) system is semi-autonomous, intelligent software that controls life support systems designed for regenerating the air in remote space habitats. This software was used during the Phase III test of NASA's Lunar/Mars Life Support Test Program (LMLSTP). For this test, four crew members lived in a closed habitat for 91 days. Wheat for air recycling and food was grown in a separate closed chamber. The wheat was planted and harvested in stages, separated by 16-24 days. At any given time, there were four different stages in the plant chamber. This crop could provide enough oxygen (O2) for one of the four crew members. Crew solid waste was incinerated every four days; effluent from this incineration contained carbon dioxide (CO2) that the plants recycled into O2. Other sources of CO2 included gas removed from the air in the crew chamber and a pressurized bottle.

The IMC system managed the transfer of O2 and CO2 (called product gases) among gas reservoirs to ensure crew and crop health and to recycle gases produced by waste incineration (figure 1). These reservoirs included a crew habitat, a plant chamber, an airlock, and a number of pressurized tanks. The IMC system performed the following tasks autonomously during the 91 day test:

- Controlled the concentration of O2 in the plant chamber to maintain a setpoint
- Controlled the injection of CO2 into the plant chamber to maintain a setpoint needed for plant growth
- Managed the storage and transfer of O2 from the plant chamber to the crew habitat for use by the crew
- Managed the storage and transfer of O2 from the plant chamber for use during solid waste incineration
- Selected and configured for use the best available source of CO2 for the plants

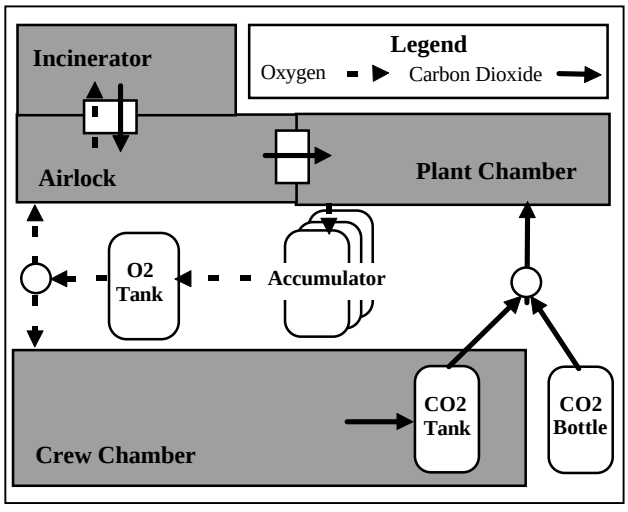

Figure 1. Product Gas Transfer for Phase III Test

The objective of building this intelligent control system was to reduce the need for human involvement in nominal control of life support systems by automating control

operations. Prior to this test, traditional process control software had been used to control life support systems developed for LMLSTP. Operation of this software is manually intensive, requiring vigilance monitoring and frequent manual adjustment of control parameters. It is impractical and inefficient for crew at a remote space site to continuously monitor day-to-day operation of life support systems. Our intelligent control software autonomously handles nominal and expected anomalous situations. The crew only intervenes in exceptional situations. During the Phase III test we demonstrated the viability of using intelligent control for such automation.

Description of Control Application

The basis of the IMC system is the three tier (3T) autonomous agent control architecture (Bonasso, et. al.,1997a). 3T integrates traditional control with reactive task sequencing and deliberative planning technology. The lowest tier is the skill manager, which interfaces with the life support hardware controllers and sensors. The middle tier is the sequencer, which dynamically selects groups of skills to effect standard operating procedures and alarm handling. The top tier is the planner, which predicts product gas needs and availability and generates a plan based on these predictions that is implemented by the sequencer. All tiers of the 3T architecture continuously monitor data from sensors and reactively alter control at each level based on changes in the environment (figure 2).

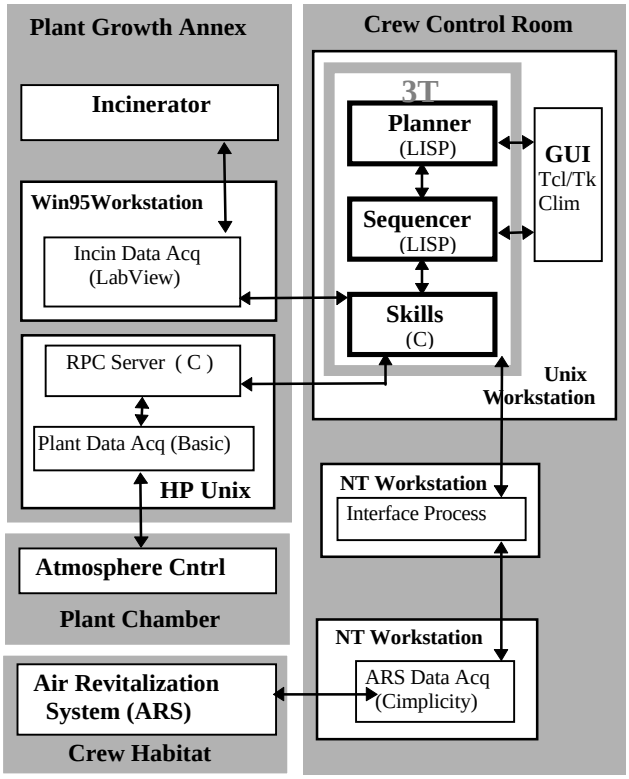

Figure 2. IMC Control Architecture

Each tier of 3T corresponds to a program. In the lowest tier, each skill manager is a C program. At the middle tier, the sequencer is the Reactive Action Package System (RAPS) (Firby, 1989). At the top tier, the planner is the Adversarial Planner (AP) (Elsaesser and MacMillan, 1991). Both the planner and sequencer are developed in LISP and are executed in separate LISP threads. All processes execute on a Sun Unix workstation.

Hierarchical Distributed Control

3T utilizes a different knowledge representation at each tier. The tiers usually are distributed as separate processes on different platforms. Thus, in addition to the usual communication between skill managers and the controlled system's effectors and sensors, 3T also defines protocols and a methodology for communication between the layers of the architecture. The sequencer tier acts as the locus of mediation between multiple skill manager processes, user interface processes, and the planning tier by employing a communication channel abstraction for integrating the various interprocess communication interactions.

The communication channel abstraction is implemented in object-oriented fashion with specialization for different types of channels possible on an abstract *link* class. These link specializations depend on the dimension of description including: (1) structure of the data being communicated (e.g., per-byte encoding, simple strings, nested structures), (2) lower level transmission mechanism (e.g., sockets, RPC, or CMU's IPC package), and (3) communications protocol shared with the connecting process (e.g., monitoring and control process such as a user interface). In addition to supporting the fundamental job of sending and receiving messages, every comm link responds to a standard set of messages for creating, destroying, starting, stopping, pausing, resuming, resetting, and clearing. Thus, the design for distributed communications makes possible dynamic and incremental reconfiguration of the architecture without affecting continuous operation of active processes.

The IMC system was distributed over four platforms:

- Sun Ultra Sparc running sequencer & planner (Lisp) and Tcl/Tk user interface with Solaris operating system
- Sun Sparc 5 running eight skill managers written in C and running under SunOS
- Pentium class PC running the user interface in the crew chamber under Windows 95
- Pentium class PC running one of the data servers described below under Windows NT

Beginning at 3T's lowest level with the skill interfaces to the existing life support hardware and software, there are 3 unique and custom interfaces to life support systems:

- Bi-directional data exchange with the data acquisition system for the crew chamber. IMC did not directly control crew chamber air regeneration hardware but it

influenced this control by providing data about product gas transfer to the air regeneration control software.

- Data and command interface with the data acquisition system for the plant chamber
- Data monitoring interface with the control system for the solid waste incinerator

We used 8 skill managers to interface to these 3 systems.

In turn the sequencer interacts with each skill manager using an instantiation of a skill manager link class. These links utilize sockets for low-level transmission duties. The communications between sequencer and skill manager links are based on a protocol where the sequencer activates and deactivates control skills leading to contexts in which event occurrences are monitored. The sequencer may query the skill managers for local state information in addition to reconfiguring and resetting each skill manager individually. In the IMC application , 3T was operated in a number of skill configurations, depending upon ongoing operations. A skill manager was only executed when activities using its skills were in progress (e.g., incinerator skill manager was executed only during an incineration).

The interaction between planner and sequencer proceeds differently. Instead of the planner controlling the sequencer similar to the way the sequencer controls skill managers, the planner suggests modes of operation for the sequencer. In the IMC application the mechanics of this interaction were supported by having the planner and sequencer run in the same Lisp process but in different threads. The planner could then directly read from and write to sequencer memory. We describe this integration in more detail in the section on the uses of AI technology.

Interaction with the User

The 3T architecture was developed originally for control of autonomous systems. Practical demands of the IMC required user-initiated intervention in autonomous control. The integration of user initiative into the autonomous control architecture supports reactive manual control and reconfiguration in response to novel situations. Situations where such manual interaction was needed include:

- *Uncontrollable Events and Unanticipated Operations*: provide manual capability for responding to events the autonomous software cannot control (e.g., failure of external data source, such as rebooting data acquisition)
- *Unavailable Instrumentation*: accommodate events that could not be instrumented sufficiently to enable autonomous control (e.g., manual sensor calibration)
- *Safety*: permit a user to override the autonomous system for safety reasons (e.g. erroneous sensor measurement results in unsafe action by the autonomous system)
- *User Preferred Tactics*: permit a user to select which redundant sensor should be used for control or to specify control parameters (e.g., control setpoints)

Our application design has the following characteristics that enable such manual intervention:

- *Explicit specification of Level of Autonomy (LOA).* We parameterized the LOA for both O2 and CO2 control. The LOA specifies whether the autonomous system or the user isssues control commands (Bonasso, et al, 1997b). The LOA is specified at startup for each type of gas and can be changed during system execution.
- *Separation of monitoring and control.* The sequencer both changes controller states to handle nominal and expected anomalous situation (control) and detects caution and warning (C&W) situations (monitoring). Our sequencer design separates the modules performing monitoring from those performing control. This approach permits operating the system with only monitoring activated as well as with full monitoring and control. When LOA is manual, no changes to controllers are made by the autonomous system but the user is provided with C&W information useful in maintaining situation awareness while performing tasks manually. We also modularized separable control segments (O2 and CO2 control). This permits manual intervention in the control of one type of gas while remaining autonomous on the other type of gas.
- *Manual activation of each hardware controller.* For the 3T system, this corresponds to providing for user activation of skills to control life support hardware. We implemented this through the sequencing tier, which guarantees that the control system maintains accurate values for the current state of all manually controlled hardware (Kortenkamp, et al., 1997).
- *User-specified control preferences.* The product gas transfer system includes redundant and alternative sensors and controllers. IMC supports user in specifying preferred instrumentation when multiple are available. These preferences are used by the sequencer as context for selecting methods. Only data from a preferred sensor is monitored when determining control changes. Specifying a preferred controller reconfigures the gas flow network through the selected controller if possible.
- *Interlocks permitting manual override of autonomy*. For product gas transfer, interlocks are implemented to inhibit O2 concentration and CO2 injection in the plant growth chamber. These interlocks were needed to accommodate plant lighting profiles and to ensure human safety. Since these interlocks can be activated by either the user or the autonomous system, we provide a priority scheme where manually activated interlocks override autonomously activated interlocks.
- *Manual reconfiguration for hardware maintenance*. It was important to provide manual capability to take sensors out of the control loop (*offline*) temporarily with minimal effect on the control of product gas transfer. Every 2 days the gas analyzers needed to be calibrated. Occasionally a broken sensor needed to be

repaired. The control tactics to take a sensor offline were implemented at the sequencing tier. When multiple sensors were available, we switched to a redundant sensor when one was offline. If all sensors were offline, we stopped all active control relying on that sensor until it became available. We also alerted the user.

- *Parameterized control setpoints and C&W thresholds.* The values of control setpoints and C&W thresholds could be changed during execution by either a user or the planning tier (e.g., planner lowered the O2 setpoint to accelerate CO2 consumption by the plants). These changes were interleaved with the execution of autonomous control to guarantee control was not executed with inconsistent settings.

This manual intervention capability in combination with modularity in control segments provides the flexibility to respond to situations that could not have been anticipated. For example, it was decided well after the Phase III test had started that we should change the control scheme for raising CO2 levels in the plant chamber after a crop harvest. Since there were few harvests remaining when this decision was made, we avoided major changes to the autonomous software by setting LOA manual for CO2 and manually activating the preferred method of CO2 injection until chamber concentrations were nominal. Throughout this operation, O2 control was autonomous.

The user interface supports supervisory monitoring as well as manual intervention. It provides test engineers with a summary of the current control situation. It can be scanned quickly for critical parameters, configurations, and anomalies. It logs control actions taken and alarms detected. Such a summary is useful when the control system is untended during round-the-clock operations. This user interface was developed using Tcl/Tk.

Uses of AI Technology

The IMC system combines a number of AI technologies with traditional control. The basis of the control system is an autonomous agent architecture. This architecture includes a deliberative planner, a reactive sequencer, and a set of tight sense/act processes that interface to control instrumentation. We use planning technology to predict gas needs and to specify activities that change control strategy to meet those needs. We use reactive task sequencing to automate standard operating procedures and alarm handling. The sequencer selects control tactics from a set of alternative approaches. This selection is based on matching a sensor-based assessment of situation to the method best suited for that situation. The sequencer also alters control tactics in response to anomalies in the environment or the control instrumentation.

This test was a 91 day experiment, so equipment and control schemes were changed and new data values obtained after analysis of previous activities. Thus, a 91 day plan would become infeasible after a few weeks. We made use of the fact that there are regular, repeated activities in the plan to partition the planning operators. Solid waste needed to be incinerated every 4 days and some portion of the plants were harvested and replanted every 16, 20 or 24 days. Lower level planning operators were developed to generate various types of four day plans (e.g., with planting and harvesting, with changes in setpoints), and higher level operators could compose these into longer plans based on plant crop rotations. Thus, plans of varying length could be generated, and new plans could be made at any time. This replanning was possible without stopping the sequencer since the planner draws its state conditions from the current sequencer memory.

To specify control strategy, the planner's primitive operators invoke top level sequences which change values in the sequencer memory that represent the desired strategy. The sequencer reacts to effect this control strategy as best as it can, given the current situation in the environment. This is an example of using plans as guides rather than as blueprints for execution (Agre and Chapman, 1990). This approach allows the sequencer to select an alternative to the preferred strategy, should there be a problem with implementing that strategy. For example, the planner may specify that the strategy for oxygen control is to accumulate O2 for an impending incineration. If the pressure in the O2 accumulator gets dangerously high because of an unexpectedly high output from the plants, the sequencer will ignore the preferred strategy to store O2 and will transfer gas out until the pressure is at a safe level. This approach also supports our phased integration approach described later by permitting the planning tier to be replaced by manual intervention.

Integration with User Interface

An important modification made to the 3T architecture was developing a principled approach for user-initiated interaction with the autonomous control software. We modified the sequencer for bi-directional communication with a user interface process and developed user interface capability that aided the user in communicating with the sequencer. The IMC supports a variety of modalities for the user to interact with the control system, including:

- User or the control system volunteers information
- User responds to information requests by the control system (e.g., the planner predicts a time region for incineration, but requests the user to confirm start time)
- User executes a function in the sequencer

There were several sequencer extensions required for human interaction. First, the approach whereby the sequencer monitors each skill manager for context-dependent events was generalized to allow for events received by *monitoring and control links*, such as a user interface. This enhancement included the ability for the sequencer to make arbitrary queries of monitoring and control links without blocking for answers. Second, a

dynamically configurable facility for process and data subscription was created so that each user interface could subscribe to ongoing operational changes reflected in the sequencer memory or data store. Finally, a communication protocol message semantics for the monitoring and control links was defined to convey these new type of messages.

The graphical user interface (GUI) was an independent process which communicated with the control software by:

- *Data Updates* received periodically on a change-only basis. The GUI displayed this information on a continual basis, regardless of the level of autonomy.
- *Messages* from the sequencer containing information about caution and warning, observed states, and control states. The GUI logged this information in a file and optionally displayed messages as they were received.
- *Sequencer Function Execution Requests* from the GUI. User-initiated functions altered sequencer memory to accomplish changes in control strategy (e.g., change LOA) and tactics (e.g., change to a different controller).
- *Queries* from the sequencer to the user to obtain data not available from instrumentation. Queries could arrive in any order and could be answered in any order.
- *Hardware Control Commands* from the GUI. These commands were executed through the sequencer to avoid conflict with automatic controls by the sequencer.

Integration with Traditional Control Software

For the Phase III test, the IMC system was integrated with two legacy control systems (control for the plant chamber and for the crew habitat) and one new control system (for the incinerator). Intelligent control is integrated with these traditional control systems at the skill tier of 3T. Skills interface with the continuous control instrumentation, providing data which the sequencer translates into symbolic state information. The skills developed for the IMC ranged from simple binary switches to sophisticated adaptive controllers (e.g., feed forward & PID controllers).

IMC was not allowed to issue control commands through two of these legacy systems (crew habitat and solid waste incinerator). We integrated by monitoring data from these systems and making sure product gas transfer control actions that affected them were consistent with their control objectives and caused no harm. For example, IMC stopped providing oxygen to the crew habitat when O2 concentrations in that chamber were above normal. The O2 alert levels that we used to stop injection were well below the alert levels used by the legacy system so we would not reverse or impede its control actions to remedy the situation.

For the integration with the plant chamber control, no software could be changed in the legacy system. We were required to make all control inputs as if they were inputs from the user interface to the legacy system. This system only permitted one user input every data cycle (~15 seconds). Thus, we were unable to take advantage of parallelism inherent in the control problem because of the integration approach with the legacy system.

In all of these cases, however, our ability to devise strategies that integrated data transfer among these heterogeneous systems with minimal change to legacy software was pivotal in gaining the customer's cooperation and support. Another confidence-building integration strategy we implemented was the ability to hand over control to the legacy system in case of a serious alarm. During the test, we never exercised this capability because we never saw a serious alarm (i.e., IMC always recovered from anomalies before hand-over conditions occurred).

Application Development and Operation

The IMC system was developed at a 2 staff level in collaboration with a product gas engineer using an iterative development methodology. To demonstrate feasibility, we developed a subset of the control system that maintained gas concentrations for wheat grown in the plant chamber. This 5-month development included testing in the lab with an emulation of life support hardware and recorded data. Then we integrated this prototype with the life support hardware for the plant chamber and tested it during a wheat growth test (3.5 months). During this test, we demonstrated 2 modes of operation: monitoring only, and monitoring and control.

Once we had demonstrated feasibility, we began developing the operational system. The operational design was based on the feasibility prototype, with significant changes to address what we learned in the wheat test and to add full product gas transfer capability. Developing the operational system took 2 months. As with the feasibility prototype, we tested the operational system in the lab before integrating with the life support hardware. Integration took 2 months, and extended past the start of test because of delays in the installation of life support hardware, and network loading problems that required changing network hardware. Once these issues were resolved, the IMC operated round-the-clock for 73 days.

Phased Integration of Control

We integrated the 3T application in phases, starting with the lowest tier and moving to the top tier. As each tier was integrated, we provided a usable (though incomplete) capability. When the skills were integrated, we could manually control the life support system using a computer. When the sequencer was integrated, control tactics were automated but manual intervention was required for changes in control strategy. The full system capability was available only after the planner was integrated.

Product gas transfer was a new life support system with hardware installed just prior to using it in the Phase III test. The installation and checkout of this hardware did not always occur as planned, resulting in subsets of hardware becoming available for integration with the control software in an unpredictable order. The customer desired that we control "as much as possible as soon as possible". To achieve this objective, we modularized the control system into separable control segments, and provided the ability to control each segment either manually or autonomously. O2 control was a separate module from CO2 control. Since the CO2 gas transfer hardware was installed much sooner than the O2 gas transfer hardware, we were able to complete integration with CO2 control software and begin autonomous operation while still performing manual integration testing on O2 gas transfer hardware.

For phased integration of new capability into an operational system, the developer must be able to meet operational objectives and test new software at the same time. Software errors discovered while integrating new capability would bring down the operational system occasionally. It was important to provide a computer capability to monitor life support instrumentation for assessing system state and to take manual control actions for maintaining safety while the autonomous system was down. For the Phase III test, this capability was implemented below the skill tier. For future systems, it would be useful to include this capability at the skill tier.

Application Use

This application was developed to support the Phase III test of the LMLSTP. This test started on September 19, 1997. The control software was operational on October 6. It successfully managed the transfer of product gases round-the-clock until the end of test on December 19. Except when we were integrating new capability or responding to an anomaly, the system typically ran without human supervision or intervention. Three times a week a user would take gas analyzers offline for calibration. When waste was incinerated (every 4 days during portions of the test), we manually reconfigured skill interfaces to receive data from the incinerator control software. We also manually transferred data files logged by the IMC to a centralized computer 3 times a week.

The limited human role in operating the IMC contrasts significantly with the operation of the more traditional process control software used with the other life support systems developed for LMLSTP. Operation of other 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. Operating the IMC typically required 6-8 hours weekly of shift work, with an additional 6 hours for each incineration and 3 hours for each harvest.

An interesting aspect of the IMC application was the use of Lisp in a long duration operation. Lisp's use of garbage collection has been mentioned as a liability for its use in “real world” applications. We did not find this to be the case. We were able for the most part to perform garbage collection during times that did not affect system performance. Only when the Lisp process size increased significantly (above 30Mb) due to an as yet undiagnosed memory leak did garbage collection time become a factor.

Another benefit of using Lisp was the ease with which changes could be incorporated into the sequencer while executing. Using the Emacs to Franz Allegro Lisp interface, several Emacs Lisp buffers were used to interact with the running sequencer while receiving low-level status information from the Lisp process. This status information proved useful in determining possible sources of communications problems between the sequencer and the various interacting processes. One challenge we overcame was limiting the maximum size of the status buffer (i.e., ordinarily Emacs buffers retain a complete output history), and yet maintaining continuous output of status information to the buffer. An automated solution was implemented by which Emacs periodically checks the buffer size and deletes the oldest status information when the buffer history length reaches a specified size.

As described elsewhere, the IMC was integrated with a heterogeneous set of legacy software processes operating on different platforms. This diverse set of data connections resulted in a number of communications-related problems during operations, including loss of communication due to high bandwidth network traffic, failure to terminate another node on our network, and rebooting of legacy systems. Since most of these problems resulted in suspension or termination of communications among our processes, we could have recovered automatically from many of these problems if we had added software to monitor the health of process communication.

Future Work

The successful demonstration of autonomous control using AI technology during the Phase III test resulted in the selection of this architecture for controlling life support systems being developed for the Space Station. We are currently developing 3T control software for use in a 1999 ground-based demonstration of life support systems.

Acknowledgements. This work was funded by the National Aeronautics and Space Administration. We would like to thank M.Edeen/NASA for domain knowledge, R. Burridge/ Texas Robotics and Automation Center for CO2 injection skills, and D. Overland/NASA for an interface to a legacy control system.

References

Agre, P.E., and Chapman, D. 1990. What are Plans for? *Journal of Robotics and Autonomous Systems*. 6: 17-34.

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

Bonasso, P., Kortenkamp, D., and Whitney, T. 1997b. Using a Robot Control Architecture to Automate Shuttle Procedures. In *Proceedings of 9th Conference on Innovative Applications of Artificial Intelligence*, 949-956 Providence, R.I.

Elsaesser, C., & MacMillan, T.R. 1991. Representation and Algorithms for Multiagent Adversarial Planning. Technical Report MTR-91W000207. MITRE, Wash. D.C.

Firby, J. 1989. Adaptive Execution in Complex Dynamic Domains. Ph.D. diss., Yale University.

Kortenkamp, D., Bonasso, P., Ryan, D, Schreckenghost, D. 1997. Traded Control with Autonomous Robots as Mixed Initiative Interaction. *AAAI-97 Spring Symposium Workshop on Mixed Initiative Interaction*.