Augmenting Automated Control Software to Interact with Multiple Humans

C. Martin, D. Schreckenghost, R. P. Bonasso

NASA Johnson Space Center, TRACLabs
1012 Hercules, Houston TX, 77058, USA
cmartin@traclabs.com, ghost@ieee.org, r.p.bonasso@jsc.nasa.gov

Abstract

It is clear that automated control systems for applications such as power generation and advanced life support do not operate in isolation. Unfortunately, support for interaction with humans often lags support for basic automation functionality in the development of such systems. Two goals of the Distributed Collaboration and Interaction (DCI) project are to (1) enhance interaction capabilities between humans and autonomous systems and (2) better understand design requirements for future autonomous systems to support this enhanced interaction. This paper describes current work in DCI with respect to a command and authorization capability. This capability provides a coordination mechanism through which humans and an autonomous control agent can take actions on the same underlying hardware system with a decreased risk of interfering with each other's commands.

Introduction

Autonomous systems deployed to realize control automation are needed to reduce human workload, to increase efficiency, and to perform routine operations, such as vigilant monitoring or continuous iteration through processing cycles. Such automation does not operate in isolation. The automation must be deployed in an environment, and this environment typically contains humans who are affected by or overseeing its operation. Humans must explicitly interact with the automation for tasks such as supervisory monitoring, modifying goals, repairing underlying hardware or software, responding to anomalies, and taking advantage of opportunities.

Unfortunately, practical and cultural considerations (Schreckenghost et al., 2002) drive designers to give the basic functionality of the automation both priority and temporal precedence over supporting interaction in the development of such systems (i.e., the system must be shown to work before interaction with humans becomes a priority). As a result, automated control systems are not typically designed from the outset to fully support interaction with humans, even though designers may recognize this need or plan to address it in the future.

This paper describes one element of our current research to both (1) enhance interaction capabilities between humans and autonomous systems and (2) better understand design requirements for future autonomous systems to support this enhanced interaction. In pursuit of the first goal, we have developed a Distributed Collaboration and Interaction (DCI) environment that allows humans to interact with automated control agents whose function is not primarily human-centric but who must be supervised by or coordinated with humans (Martin et al., 2003). In particular, this paper describes a command and authorization capability recently developed for the DCI environment. This capability provides a coordination mechanism through which humans and a control agent can take actions on the same underlying physical system with a decreased risk of interfering with each other's commands.

In pursuit of the second goal, we hope use our experience with the DCI environment to define requirements, which can be considered up front with other functional design requirements to make the automated control systems interaction-ready. In doing so, we hope to enable future designs for autonomous systems to support enhanced interaction with humans from the ground up. One critical element of our approach is to avoid placing the entire burden of supporting human interaction on the automation itself. Because autonomous control systems must often handle time-critical tasks, we want to avoid introducing extra processing overhead in the autonomous system, when possible. Our approach uses *augmenting software* that is tightly coupled to the automation through shared models or data but has its own processing resources. The separation of processing for interaction into augmenting software has the added benefit of accommodating the natural lag between implementing basic automation and interaction capabilities. With respect to the command and authorization capability, we aim to define what models and data should be made visible by the autonomous system as well as what basic functionality the autonomous system must exhibit to allow interaction.

The treatment of interaction among humans and automated control systems as a coherent design and research issue is a relatively new endeavor. This paper describes our approach through DCI. The following section describes the automated control system to which we have applied our current research. The command and authorization capability is then presented in detail. The paper concludes with a summary of contributions and lessons learned as well as a discussion of future work.

Automation for the Water Recovery System

Our work to enhance human interaction with autonomous control systems builds on our previous experience developing automation for advanced life support at Johnson Space Center (JSC) since 1995 (Bonasso et al., 2003; Schreckenghost et al., 2002). The work described in this paper has been applied to the advanced Water Recovery System (WRS) (Bonasso et al., 2003). The WRS removes organic and inorganic materials from wastewater (hand wash, shower, urine and respiration condensate) to produce potable water. The hardware and control software for the WRS operated unattended in a continuous 24/7 integrated test from January 2001 through April 2002, (Bonasso et al., 2003). The WRS is comprised of four subsystems as pictured in Figure 1:

(1) The *biological water processor (BWP)* removes organic compounds and ammonia by circulating the water through a two-stage bioreactor. The first stage uses microbes to consume the organic material using oxygen from nitrate molecules. The second stage uses microbes to convert the ammonium to nitrate.

(2) The *reverse-osmosis (RO)* subsystem removes inorganic compounds from the output of the BWP, by forcing the water to flow at high pressure through a molecular sieve. The sieve rejects the inorganic compounds, concentrating them into brine. At the output of the RO, 85% of the water is ready for post-processing, and 15% of the water is brine.

(3) The air *evaporation system (AES)* removes the concentrated salts from the brine by depositing it on a wick, blowing heated air through the wick, and then cooling the air. The inorganic wastes are left on the wick and the condensate water is ready for post processing.

(4) The *post-processing system (PPS)* makes the water potable by removing the trace inorganic wastes and ammonium using a series of ion exchange beds and by removing the trace organic carbons using a series of ultra-violet lamps.

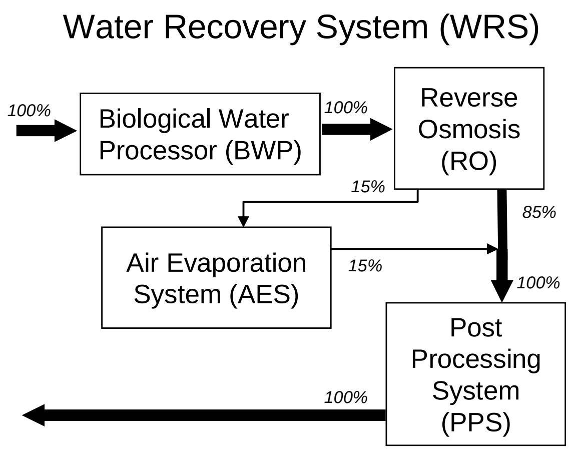
Water Recovery System (WRS)
100%
Biological Water Processor (BWP)
100%
Reverse Osmosis (RO)
15%
85%
Air Evaporation System (AES)
15%
100%
Post Processing System (PPS)
100%

Figure 1. WRS Subsystems

These subsystems are loosely coupled, and their primary interdependencies are related to input and output of the water to be processed. In total, the automated control system for the WRS manages more than 200 sensors (measuring pressure, temperature, air and water flow rates, pH, humidity, dissolved oxygen, and conductivity) and actuators (including pumps, valves, ultra-violet lamps, and heaters). The automated control system uses a layered control architecture, called 3T (Bonasso et al., 1997), which has been applied to control both robots and advanced life support systems. 3T's top tier is a hierarchical task net (HTN) planner, the plans of which are executed through a reactive middle tier that in turn manages the sensors and actuators of the hardware via a low-level control tier.

Commanding the WRS

Although the 3T-based automated control system operates the WRS hardware unattended most of the time, there are several cases in which humans must also take actions on the life support system. Actions that humans take can be either *manual* or *mediated*. *Manual* actions are those that the human carries out directly on the life support system hardware, for example, physically turning a valve. A human conducts *mediated* actions by giving instructions to the automation software, which carries out the actions. Mediated actions can be requested from other external sources as well, including software elements of the DCI system. In contrast, *automated* actions are those taken by the control software during its normal operation without any requests from an external source. Manual and mediated actions are needed for two possible reasons (1) the action must be manual because the automation has no appropriate actuator or (2) the action could be carried out either by a human or via the software but is motivated by circumstances outside the scope of normal operation for the automation. An *RO slough*, as described in the following subsection, exemplifies the latter.

When a human wishes to perform actions on the WRS using the DCI command and authorization capability, he or she requests the appropriate *commanding* permission[1] for a particular activity. To grant commanding for a given activity, DCI must first, if possible, grant authorization for the set of manual or mediated actions required by the activity, and then reconfigure the WRS hardware and control automation to the proper state required for those actions. In general, the reconfiguration process may include setting the states of particular hardware such as valves open/closed or pumps on/off, adjusting the autonomy of the automation to allow for manual actions (Schreckenghost et al., 2002), bringing the state of the system to a particular point such as getting tube pressures

[1] Throughout the paper, the term "authorization" implies a license to take action on the WRS. We use the term "commanding" to convey this authorization *plus* whether the system is ready for the execution of a particular activity associated with a pre-defined procedure, which may contain multiple manual or mediated actions.

and temperatures within a specified range, or commanding a subsystem to a particular processing mode. The following subsection describes the current support implemented for commanding and authorization in the DCI system, which includes a subset of these reconfiguration steps.

Current Support for Commanding the WRS

The command and authorization capability in DCI coordinates multiple humans and the control automation, allowing each to take actions on the same underlying hardware system without interfering with ongoing tasks. Our current work concerning command and authorization interactions addresses the coordination of multiple humans with each other and with the automation before, during, and after the execution of human-initiated actions on the WRS hardware. We currently support these four activities:

- *BWP nitrifier slough* – The biofilm that grows on the insides of the tubes in the nitrifying portion of the BWP will thicken over time, slowly constricting the passage of water and air. To minimize clogs, the control system periodically sloughs the biofilm by sharply increasing the airflow. This automatic slough is only partially effective, and eventually a human is required to manually slough the nitrifier. The configuration for this activity requires ensuring that water is flowing in the BWP, as well as suspending the automatic shutdowns (ASDs) that the control automation will normally enact if tube pressure readings go outside the nominal range.
- *RO slough* – Inorganic deposits may accumulate inside the RO's tubular membranes. If the water flow is reversed, a small ball in each tube will slide along the tube length, sloughing this buildup away. The automated control system carries out this RO slough at a predetermined frequency. If the RO output quality degrades, a human may manually command the control system to slough the membranes again. This is the only *mediated* action we currently support. Reconfiguration for this activity requires the RO to be shutdown.
- *RO membrane change out* – Eventually the RO membranes lose their efficiency and must be physically replaced. The RO is shutdown, and the upstream and downstream subsystems are placed in standby mode.
- *BWP pressure calibration* – Pressure sensors are the primary input used to control the BWP. These sensors require calibration about every three months. In order to conduct the calibration, the BWP must be disconnected from the downstream subsystems and placed in a standby mode.

Actions required to achieve the reconfiguration necessary for these activities may be either manual or mediated. In the current system, reconfiguration affects both hardware (the states of eight valves and ten pumps) and software (the operating characteristics of the automated control system). When possible, commanding is granted for concurrent tasks, which must have compatible configurations.

Commanding and Authorization in DCI

In general, the command and authorization capability is needed in DCI to allow humans and control automation to make efficient progress toward their individual goals without (1) the risk of having their work immediately undone or destroyed (2) the risk of interfering with or preventing the work of others and (3) the risk of putting the underlying hardware system in an unsafe state (for example, a state where pumps may be damaged by attempting to pull water from a blocked source).

In the DCI environment, each user is represented by an Ariel agent (Martin et al., 2003), which acts as a liaison between the user and the rest of the software environment, including the automated control agent. An Ariel agent provides a human-centric interface into the software environment and provides a number of services including notification, task tracking, and location tracking. In particular, the Ariel agent provides a Command and Authorization Service, which assists its user with command and authorization requests. Figure 2 shows the Ariel agents, the WRS system and the following two components:

• **Command and Authorization Manager (CAM)**. The CAM accepts requests for commanding from users through their Ariel agents. Each request is associated with an activity that the user wishes to perform. The CAM first queries the AFC (see next bullet) for information about the effects of the requested activity as well as any configuration conflicts between the current system configuration and the configuration required for the activity (currently, only hardware conflicts are reported). The Managing Authorizations section, below, describes how the CAM uses the results of this query to grant or deny authorization. If authorization is denied, this result is returned to the user along with a description of the configuration conflicts. If authorization is granted, and the user wishes to continue, the CAM asks the AFC to carry out any required reconfiguration on the WRS. Once the reconfiguration, if any, is complete, the CAM informs the

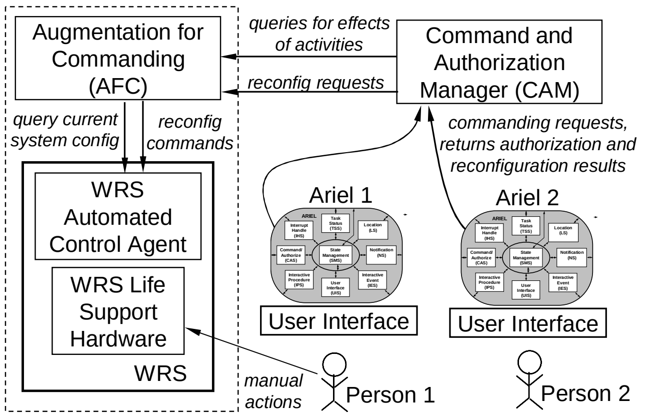
Augmentation for Commanding (AFC)
queries for effects of activities
reconfig requests
Command and Authorization Manager (CAM)
query current system config
reconfig commands
commanding requests, returns authorization and reconfiguration results
WRS Automated Control Agent
WRS Life Support Hardware
WRS
Ariel 1
Ariel 2
User Interface
User Interface
manual actions
Person 1
Person 2

Figure 2. Commanding and Authorization in DCI

user through his or her Ariel that the WRS is ready to command. The user can then proceed with the actions required by the procedure for the requested activity. When the user has completed the activity, he or she requests the CAM to release commanding for the activity. The CAM informs the AFC that the activity's reconfiguration is no longer required and then releases the authorization.

• **Augmentation for Commanding (AFC).** The AFC is a piece of augmenting software in the DCI architecture (shown by the dotted lines indicating tight coupling to the WRS). It is coupled to the automated control system in that they share static models of both the physical WRS system and the procedures that can be performed on the system (including reconfiguration procedures). Using these models, the AFC predicts how various activities will affect the WRS. The AFC can also query the WRS control agent dynamically to get the current system configuration.

We found that models of reconfiguration procedures could be used to (1) determine what parts of the WRS would be affected by (reconfiguring for) an activity and (2) allow the AFC to trigger the WRS control agent to perform the reconfiguration necessary. Except for mediated activities, such as the RO slough, models of reconfiguration procedures were not originally developed for the WRS control agent because they were not necessary for autonomous operation. In support of the DCI commanding capability, we added models of the reconfiguration procedures for the other three activities described above. In the future, we also plan to add procedure models of the actual manual activities. This will further help us determine how the WRS would be affected: by both the body of the activity itself and the reconfiguration (initial and final). Because the human actions during the RO slough activity are mediated by the control system, the WRS already has a model for this activity.

When the CAM queries the AFC about the effects of an activity, the AFC provides two results. First, the AFC decomposes the associated reconfiguration procedure (as well as the activity's procedure model, if available) to determine and return all components of the WRS that may be affected by the activity. In the current implementation, this result is highly abstracted and consists of an indicator for the highest-level system or subsystem that is affected. This system/subsystem approach is made extensible by also returning the specific decomposition of subsystems that are affected by the reconfiguration (in the future, subcomponents of the subsystems may also be used here). Second, the AFC queries the WRS automated control agent for the current system configuration (i.e., the current state of the eight valves and ten pumps) and returns a list of conflicts between the current state and the state required after reconfiguration. The CAM uses the first result to determine whether to grant authorization for the activity, as described in the next section, and passes the second set of results back to the user.

If the CAM asks the AFC to reconfigure the WRS for a requested activity, the AFC triggers the WRS control agent to perform the reconfiguration procedure, if any. During the course of the reconfiguration, some manual actions may also be required. When it is time for a manual reconfiguration action, the WRS control agent, through the AFC, CAM, and the Ariel agent's user interface requests the user to perform the action and waits for a return indication from the user that it is accomplished. This feedback from the user is needed because manually operated physical devices are not normally instrumented for computers, so manual actions are not easily observable by the software for tracking a user's progress in the reconfiguration. Once all reconfiguration actions have been completed, the CAM informs the user that the WRS is ready for commanding.

Managing Authorizations

Authorization to act on the WRS is managed by the CAM. The CAM is centralized to provide synchronized access from multiple entities (various Ariel agents and, in the future, the automated control system itself) to a single model describing which entities hold which authorizations. In general, granting authorization to one entity for a given scope of action blocks other entities from receiving authorization overlapping that scope until the first authorization is released. This blocking authorization paradigm prevents multiple entities from acting on the WRS simultaneously for activities within the same scope, which may therefore conflict or interfere with one another. Because multiple entities can *request* authorization simultaneously, the CAM's synchronized access to the authorization model, along with the requirement to obtain authorization to act on the system before reconfiguring for a particular activity, prevents simultaneous conflicting actions (including reconfiguration actions) from being performed.

When possible, the CAM should authorize concurrent activities. We believe that the maximum concurrency without risking conflicts can be achieved by authorizing activities Act1 and Act2 concurrently as long as (1) their configurations do not conflict and (2) no action taken for Act1 (during reconfiguration or the procedure itself) affects the same component or state value (i.e., tube pressure) as any action taken for Act2, and vice versa. For our initial approach, we used models already within the WRS control agent to support command and authorization and limited our development of new models. Unfortunately, (1) the existing models for the required configurations are not detailed enough to guarantee no conflicts (e.g., they have not been extended to include required state values or operating characteristics of the automation) and (2) we do not have models of the procedures for activities that require only manual action. Although we did develop models of the required configurations for each activity, we found that actions during activities may still conflict, even when the configurations are compatible. Until we extend these models, we have initially adopted a conservative approach to authorization that works well with the existing models but does not allow the maximum possible authorization concurrency. The approach is conservative in that it locks authorization for an entire subsystem (e.g. the RO) if any component of that subsystem is affected by an activity (by the reconfiguration - or the activity itself if a model exists), and it locks authorization for the entire WRS if multiple subsystems or the dependencies between subsystems (e.g. water flow) are affected. For the small set of actions and scenarios we have considered, the conservative nature of this approach has not been a disadvantage.

When a user requests commanding permission for a given activity from the CAM, the CAM obtains information about the highest-level system or subsystem affected by the activity from the AFC. The CAM translates the system/subsystem decomposition into a model of scopes for granted authorization. Let Φ be the set of all system components such that authorization can be assigned for the scope of that component. For the current implementation Φ = {WRS, BWP, RO, AES, PPS}. For the variables x and y, let $x, y \in \Phi$. Let $Sub(x, y)$ define a predicate that indicates whether component x is a subsystem or subcomponent of component y in a hierarchical decomposition of the system. For the current implementation, the following hold: *Sub*(BWP, WRS), *Sub*(RO, WRS), *Sub*(AES, WRS), *Sub*(PPS, WRS).

Let α be the set of all agents (including humans and the automated control agent) that can act on the system. For the variables a and b, let $a, b \in \alpha$. Let $Auth(a, x)$ define a predicate indicating that agent a has authorization to act over the scope of system component x.

The CAM uses the following rule to assign authorizations: When b requests $Auth(b, x)$, then grant $Auth(b, x)$ if and only if no other agent holds the authorization for x, for any of x's subsystems, or for any component that has x as a subsystem. In other words, when *request*($Auth(b, x)$),

if $\forall a, \neg Auth(a, x)$

  $\wedge \forall a, \forall y, Sub(x, y) \Rightarrow \neg Auth(a, y)$

  $\wedge \forall a, \forall y, Sub(y, x) \Rightarrow \neg Auth(a, y)$

then $Auth(b, x)$.

The current CAM implementation assumes that every entity requesting authorizations possesses the necessary credentials (authentication, skills, and/or certificates) for the authorization to be granted. We would like to add credential checking in the future. However, it is not currently critical in our application because (1) we assume all possible users (NASA crew) are highly trained and (2) our authorization process is used primarily for coordination rather than access control. Although users must log in to use DCI (authentication), they can currently act on the WRS by circumventing DCI completely. Users are motivated to request commanding permission through DCI primarily to minimize the risk of conflicts and to obtain assistance from the AFC in reconfiguring the WRS for the desired activity.

If the CAM denies a user authorization to act on the system, the user should (by policy) wait until the authorization can be granted before taking any action. However, enforcing such a lockout could prevent a user from taking needed action in an emergency, which is a particularly troubling prospect with respect to a critical life support system. The development and use of more sophisticated models for the effects of activities on the system will allow us to avoid being overly conservative, maximizing the number of activities we can authorize concurrently. However, these advances will not address situations in which a low-priority ongoing activity may block authorization for an emergent higher-priority activity. We are currently working on building a user override capability for denied authorizations as well as policies for notifying other users who are impacted when such overrides are exercised.

The current WRS implementation offers limited options for enforcement of either denied authorizations or denied system access in general. There is some password protection for mediated actions, but anyone could theoretically walk up to the system and power down a pump at any time. We hope to improve enforcement as the override capability is developed. Suri et al describes relevant previous work on policy enforcement (Suri et al., 2003). In the interim, when an authorization is denied, the CAM reports back to the requesting user the set of pre-existing authorizations that conflict with the request as well as the list of conflicts between the current system configuration and the requested activity's configuration. The highly trained user can consider this information to determine how to proceed. He or she may ask other users holding a conflicting authorization to release it, or he or she may proceed manually with the desired reconfiguration and activity with foreknowledge of possible conflicts that may arise. Although much work remains, making users aware of possible conflicts arising from ongoing activities by other users on the WRS is an important first step toward supporting the coordination of multiple humans and an automated control agent working on the same underlying physical system.

Conclusions and Future Work

Two research goals for the DCI project are addressed by our current work to support commanding and authorization for action by multiple humans and an automated control agent on the same physical system. In support of the first goal, to enhance interaction capabilities between humans and autonomous systems, we have enhanced coordination among users by making them aware of possible conflicts arising from ongoing activities by other users on the WRS system. Further, as an integral part of processing a human's request to perform an activity on the physical system, we provide previously unavailable assistance in reconfiguring the system for that activity. By suspending or modifying automatic responses in the control system for the duration of human-initiated activities, we have also enhanced coordination between humans and the automation. To provide these capabilities, we use models of system connectivity and configuration that previously existed in the automation, and we created new models of the reconfiguration procedures needed to support four human activities. We developed new ways to interpret these models (for example, examining a procedure to determine its effects on the system), and we updated the automation to perform the newly modeled reconfiguration procedures when requested. Finally, we developed and implemented a conservative policy for granting authorization to act on the system, which ensures that no more than one user at a time has authorization at a given scope.

In support of the second goal, to better understand design requirements for future autonomous systems to support enhanced interaction, we have determined that such systems should support sophisticated models of the effects of activities, including those consisting of manual actions, as well as the reconfiguration procedures required to support those activities. Because designers can never know all needed human-initiated procedures prior to system deployment, there should be a mechanism for dynamically adding these models and procedures (ideally without taking the system offline to do so). Finally, the automated system must exhibit some form of adjustable autonomy to suspend or modify its operation for the duration of a human-initiated activity.

We have identified a great deal of future work with respect to both goals. First and foremost, we need deeper and more comprehensive models. In order to move beyond our conservative approach to authorization, we need to extend both the breadth of models for activities (modeling the effects of manual and mediated actions in the body of procedures rather than only the configurations required as initial conditions) and the depth of these models (modeling not only hardware states such as pump on/off but also required operating parameters and performance constraints such as required pressure or temperature ranges). Once these models are available, we can update our authorization policies to permit more informed concurrency by using a more detailed understanding of the possible conflicts revealed by these models. We will also investigate using reconfiguration and procedure models that account for potential procedural changes, which may be required at loss of capability due to system faults. Sometimes these changes require a change in scope of authorization.

We would like to further enhance our authorization capabilities by supporting credential checking as well as authorization enforcement and override capabilities. To support override capabilities, we are currently beginning to explore how reconfiguration should be managed for multiple (possibly overlapping) required configurations. This work focuses on determining the target system state once commanding for a particular activity is complete. We have developed preliminary algorithms to determine which reconfiguration actions can be "undone" and which actions shouldn't be undone because they overlap with required configurations for concurrently authorized activities. As the models of required system configuration for each activity become more sophisticated, determining the target system reconfiguration before and after commanding will also become more demanding.

Finally, we plan to extend this work to better support coordination with the autonomous system. This additional support would include (1) extending supported activities to those containing a mixture of manual, mediated, and automated actions, (2) making more extensive use of the adjustable autonomy and traded control capabilities of the automation, and (3) granting explicit authorizations to the automation in addition to humans such that humans are protected from unknowingly acting on the system when the automation is performing a critical operation. Although much work remains to fully support human commanding and authorization in coordination with autonomous systems, the preliminary work presented in this paper provides both enhanced capabilities and encouragement that we have defined a reasonable path forward.

References

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

Bonasso, R. P., Kortenkamp, D., and Thronesbery, C. 2003. Intelligent Control of A Water Recovery System: Three years in the Trenches. *AI Magazine* 24 (1): 19-44.

Martin, C. E., Schreckenghost, D., Bonasso, R. P., Kortenkamp, D., Milam, T., and Thronesbery, C. 2003. An Environment for Distributed Collaboration Among Humans and Software Agents. In *Proceedings of 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, 1062-1063. Melbourne, Australia.

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

Suri, N., Bradshaw, J. M., Burstein, M., et al. 2003. DAML-based Policy Enforcement for Semantic Data Transformation and Filtering in Multi-agent Systems. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, 1132-1133. Melbourne, Australia.