A Control Architecture for Safe Human-Robotic Interactions During Lunar Surface Operations

David A. Wagner[*], Daniel L. Dvorak[†], Lynn E. Baroff[‡], Matthew B. Bennett[§],
Michel D. Ingham[**], David S. Mittman[††], Andrew H. Mishkin[‡‡]
California Institute of Technology, Jet Propulsion Laboratory, 4800 Oak Grove Dr. Pasadena, CA 91109

Long duration human-robotic missions to the Moon and beyond will require increased use of automation beyond current Space Shuttle and International Space Station practice. This paper explores the application of a model- and state-based goal-oriented control architecture to solving the problem of coordinating activities between humans and robots to improve the reliability and safety of these interactions. A goal-oriented control system continuously enforces constraints on states of the system to achieve not only control goals, but also to enforce passive constraints, such as safety constraints, on those activities.

I. Introduction

Long-duration missions to the Moon and beyond will be neither affordable nor safe without significantly increased use of automation. The Apollo missions, as well as current human space missions, achieve safety and reliability in part through rigorous detailed planning, testing and rehearsal, and through a large operations staff, highly-trained to continuously monitor and control mission activities, and respond in real-time to unexpected events as they occur. While it will always be more cost effective to offload work from astronauts to controllers on the ground where that is possible, it won't always be possible. As the number of people and activities increase, and as the distance from earth increases, ability to communicate becomes a significant impediment to ground-based monitoring. Controllers on the ground simply won't be able to monitor every detail, or react fast enough to ensure safety all the time.

One of the key drivers for this disparity is the level of communications latency typical to each mission class, as shown in Figure 1. Near-Earth human spaceflight missions allow rapid and voluminous communications of data, imagery, and voice, between the spacecraft and operators on Earth who can interactively supervise and monitor operations for safety. Procedures are generally well planned, but performed by people.

In contrast, most deep-space missions are constrained to operate with long communications latencies of hours to days, and much smaller data rates. Such constraints force deep-space missions to rely on more autonomy than is normally afforded to piloted missions. The Mars Exploration Rovers, for example, perform an entire Martian day's worth of roving and observing according to a predefined plan with no interactions from the ground until the results are relayed back at the end of the day. Capabilities such as this rely heavily on pre-planning, and are designed to respond to most unplanned

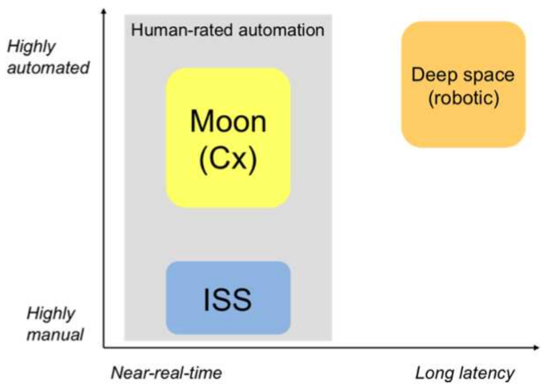

Figure 1. Comparing levels of automation with communications latency for different

[*] Senior Engineer, Flight Software Applications and Data Management Group/MS 301-240
[†] Principal Engineer, Planning and Execution Systems Section/MS 301-270/AIAA Member
[‡] Program Element Lead, Mission Systems Concepts Section/MS 301-270
[§] Senior Engineer, Flight Software Systems Engineering and Architectures Group/MS 156-142
[**] Senior Engineer, Flight Software Systems Engineering and Architectures Group/MS 301-225, AIAA Senior Member
[††] Senior Engineer, Planning Software Systems Group/MS 301-250D
[‡‡] Principal Engineer, Planning and Execution Systems Section/MS 301-250D

surprises by canceling the plan, reverting to a safe state, and calling for help, which can cause the loss of a day's worth of observation. But losing a day's worth of observation is better than losing the entire mission. Even greater levels of onboard autonomy would enable the system to diagnose and recover from (or work around) a greater number of off-nominal situations, or to return more science by performing onboard data analysis and making appropriate decisions about follow-on measurements without having to wait for ground intervention (Ref. 12).

A second key difficulty of applying the traditional paradigm of intensive ground control to long duration exploration missions is the greater number of environmental components that must now be monitored and are subject to ongoing control. Ongoing operations aboard the International Space Station (ISS) have already shown that crew members can become overwhelmed with "housekeeping chores" that seriously subtract from time available to conduct science experiments. Automated systems that contain both monitoring and controlling features will enable considerably more exploration time for the explorers, and allow the ground controllers to save their energies for serious, uncertain, or off-nominal situations. As in the operations approach used on many robotic missions, the actions performed by the system, and the controls and adjustments made to the system during an operations period, can contribute to the planning for the next period.

Future human spaceflight to remote locations like the Moon and Mars will eventually force a convergence of these operational styles, but in the nearer term mission planners would like to be able to leverage automation to reduce operating costs where possible, both on the spacecraft and on the ground. Key to this goal is the need to apply automation in a way that preserves or enhances the safety of the astronauts, while providing them more flexibility to perform their exploration duties.

The research described in this paper aims to address some of these issues by applying a goal-oriented control system architecture to the problem of human interactions with automated and robotic systems. We believe that many of the problems relating to human interactions with automation arise from the ambiguities created by the traditional command oriented interfaces used to control them. By describing control intent in terms of a constraint on a state of the system over an interval of time (our definition of a goal), a goal-oriented architecture makes operator intent unambiguous. However, in space exploration activity people are often an intimate part of the system, not only as operators, but also providing the eyes, ears, and hands that can dramatically extend the system's capabilities.

II. Motivating Scenarios

The Constellation program (Ref. 2) envisions a return to the Moon as a next step in human space exploration. Plans for creating the exploration infrastructure include building semi-permanent habitats on the Moon allowing astronauts to stay on the lunar surface for long periods, performing in-situ scientific exploration and additional construction. Living and working on the Moon will require living quarters, new vehicles, and new tools to allow astronauts to move about on the surface exploring their environment. When this vision is fully realized there could be many astronauts working outside on many different tasks all at the same time. Semi-autonomous robotic vehicles for transporting people and cargo, such as the ATHLETE platform, are already being developed (Ref. 3)

Contrast this vision with the way NASA currently conducts manned missions on the International Space Station (ISS). Activities on the ISS are highly choreographed, every move planned in advance, tested in ground simulations, documented, and coordinated between astronauts and mission controllers on the ground. Safety is a key consideration here. But more importantly, mission planners are trying to make optimal use of the astronaut's expensive and limited flight time by working out procedures and problems on the ground to the extent possible. Extrapolating the workforce size and costs of ISS operations to the much larger scope of Constellation lunar operations reveals a key challenge in the mission: how to enable astronauts to work more independently and flexibly on the Moon without compromising safety or efficiency.

Consider a scenario where one team of astronauts is about to take control of a new habitat module recently landed on the lunar surface. New modules are landed at a safe distance from the habitat, and arrive with a robotic transportation system such as the ATHLETE (Ref. 3), which can carry the module to the habitat and dock it to the existing structure. Under guided control from the ground, the ATHLETE can separate itself from the lander, and navigate its way to the vicinity of the habitat. If the program achieves its vision, the scene around the lunar habitat should be busy with activity: astronauts performing various scientific and engineering tasks, riding in or operating various vehicles, deploying and maintaining autonomous mobile and fixed experiment systems. Each of those activities will be formally planned, and some plans will be more detailed than others. For example, an ad-hoc science experiment conducted by a single astronaut over a three hour period might require much less coordination than a construction task involving several astronauts, complex construction equipment, and several weeks of work.

But even the simplest and most routine tasks on the lunar surface involve sufficient hazard potential to justify careful oversight of some sort.

Traditional space flight operations plan every anticipated task, including many “just-in-case” contingencies, in minute detail, review them for safety, and document them as procedures. The procedures are then executed with care paid to staying within the prescription because any excursions would violate the *assumed* safety of the plan – assumed because safety constraints may only be reviewed and checked against assumptions at plan time, and only weakly asserted during execution. The need to manually review and verify safety constraints produces a brittle system where significant effort and time are needed to develop a plan in the first place, and where minor unanticipated events can force plan changes that take significant time and effort to review before execution, making it cumbersome and slow to take advantage of new information and new opportunities in real time.

III. Driving Requirements

NASA's policy of protecting the health and safety of people who are involved in space activities is implemented through the application of NASA directives and standards, especially the NASA Procedural Requirement (NPR-8705.2B, Ref. 13) that describes human-rated systems. Human-rating is an integral part of all program activities throughout the life cycle of all human space flight systems. All crewed space systems are required to be human-rated: spacecraft and their launch vehicles, planetary bases, planetary surface mobility systems, and Extra-Vehicular Activity (EVA) suits. The requirement for human-rating also applies to any system elements that are attached to the crewed elements during the mission.

Actual human-rating requirements have been developed by defining what is fundamental, and essential, about systems that take humans into space. NASA’s experience over the last 45 years in many human space flight programs, from Mercury to the International Space Station, have led to this key definition:

> A human-rated system accommodates human needs, effectively utilizes human capabilities, controls hazards with sufficient certainty to be considered safe for human operations, and provides, to the maximum extent practical, the capability to safely recover the crew from hazardous situations.

As described in the NPR, human-rating consists of three fundamental tenets:

1. *Human-rating is the process of designing, evaluating, and assuring that the total system can safely conduct the required human missions.*

2. *Human-rating includes the incorporation of design features and capabilities that accommodate human interaction with the system to enhance overall safety and mission success.*

3. *Human-rating includes the incorporation of design features and capabilities to enable safe recovery of the crew from hazardous situations.*

The Constellation program continues to develop specific detailed requirements derived from this basic policy, but because the policy is so general, a large gulf remains to explain how the various detailed mechanisms will collectively enforce the policy. So, part of our task is to consider more detailed aspects of the general policy that might be applied systematically to a system design to provide measurable benefits. These can then serve as the basis for testable requirements.

Fundamental to the first two guidelines for engineering systems for human use is the need for control systems to behave not only as designed, but also *as expected by the people operating the system*, and in all situations, including those for which the system may not have been specifically designed. A decision maker, whether it is a control system or a human operator, can only make decisions based on the available evidence. Bad decisions (as opposed to slips, or inadvertent actions, which can generally be prevented through confirmations, or other system actions that force the user to focus attention on the action (Ref. 15)) result from having insufficient or incomplete evidence, or incomplete world models that fail to accommodate the given evidence. Engineers often attribute the need for *situational awareness*, to human operators, but the requirement applies as well to automated systems if the system will be required to make decisions and perform actions on behalf of the people who depend on it. Ideally, both operators and control systems would share a common model of the world. Until computers become significantly more capable, though, we must accept the reality that designers are often forced to optimize implementations to fit within available resources, and to limit the scope and precision of awareness that a control system can have. Even so, operators need to know how the system is modeling the world so that they can model, and thus understand, its behavior.

Simply reporting a deluge of information to the operator is not sufficient to focus attention on what really matters at any given point in time, and therefore does not ensure *awareness*. Human perceptual capabilities come equipped with filters that can be learned and adapted to different real-world situations. These filters will allow people to see or hear or even smell what is important to their needs at the time, and filter out distracting additional sensory input. But even relatively simple automated systems can produce data at much higher rates and volumes than people can ingest in raw form. Filters in the presentation system and graphical visualizations can help people distill meaning out of this torrent, but in many cases the result still requires the user to interpret the information against a model of the world, and control intent, to assess whether or not control is being effective. Part of our approach is to demonstrate how a goal-oriented control architecture can make control intent explicit, and continuous over time, allowing the system and human operators to clearly identify any divergence, and allowing operators to have much more confidence in the system's performance *as expected*.

Another key aspect to human expectations is that the system will respond in a particular way to control inputs. Systems can be designed, and operators trained to respond to faults and failures in the system, and to some unexpected conditions in the environment, but they cannot easily respond to a situation where more than one supervisor is trying to exercise control over the same system at the same time. Traditional systems address this question of control authority through a combination of mechanisms: either by completely eliminating the possibility of any remote control at design time (as in airliners), or by establishing procedural protocols to manage control handovers. Piloted vehicles (e.g., airliners) often eschew the potential benefits of remote control where those benefits are outweighed by the potential risks of someone usurping control for nefarious purposes, partly because the available protocols for managing control authority – even when supplemented by secure authentication mechanisms – still require some level of cooperation to work.

Future systems will have to consider the fact that robotic helpers like the ATHLETE are inherently multi-function systems, likely to participate in many different tasks, which may be supervised by several different operators, possibly for different, but concurrent tasks. Some larger construction activities may also require the coordination of several semi-autonomous systems. Solving these problems safely will require consistent protocols for coordinating *control authority*.

IV. Control Architecture

Prior publications (e.g., Ref. 7) describe a growing problem in designing and building software-intensive systems for space exploration: that traditional engineering methods are not keeping up with the complexity of the kinds of systems being conceived. Although rigorous architectural design is generally applied to the engineering of hardware systems and subsystems, the same is not generally done for software, even though software is responsible for a growing share of engineering effort and cost, and software is responsible for performing increasingly critical functions in increasingly subtle ways. Our solution to this situation is a rigorous system engineering methodology, State Analysis, and a control-oriented software architecture based on JPL's Mission Data System (MDS Ref. 1,4).

State Analysis is a model-based systems engineering methodology that enables the specification of a control system according to rigorous architectural principles (Ref. 8). This methodology focuses on the state variables of the physical system to be controlled, and the state effects relationships between them, describing these relationships in the form of explicit models. A key benefit of State Analysis is that it provides a formal definition for a goal: *a constraint on a state variable of the system over an interval of time*. A goal is a requirement on the system that is meaningful to a human reader, but also a runtime artifact of the control system that can be verified during execution.

MDS is an implementation of a model-based, state-based, and goal-oriented control architecture. MDS specifies a consistent set of methods for proposing and verifying coordinated activity plans, executing them, detecting failures that occur during execution, and responding to failures by interactively re-planning in the context of deep-space robotic systems, where autonomy is more important than real-time interactivity. Key features of this control architecture include:

- *State as a first-order consideration.* State variables represent the system's best estimate of the state of the system under control. State variables accurately represent the continuity and uncertainty of this knowledge over time.
- *Separation of estimation from control.* State variables are first class entities in the control architecture intended to formally separate the process of state estimation from that of state control. Mixing estimation with control allows controllers to have their own private estimate of system state, which can be inconsistent with other controllers' views of the same state, and thus lead to control conflicts.

- *Distinction between estimates and evidence.* State estimates are distinct from the various forms of evidence that can be used to determine these estimates, including measurements and commands.
- *Goal-directed operations.* Operator intent is expressed as goals rather than commands. Control system commands typically specify only the intent to change the state of the system at a point in time. They say nothing about what the expected prior state was, or what the state should be in the future. Although a command may *imply* intent for a state change to persist into the future, this is insufficient information to enable the control system to detect the conflict that would arise if a subsequent command changed the state again, or if the state changed spontaneously. A goal explicitly says what the state should be (or what the state should not be, or more generally, what the bounds on the state are) over an interval of time. This allows the control system to continually verify satisfaction of the goal, and to immediately detect conflicts if two incompatible goals overlap in time (example: a switch cannot be both ON and OFF at the same time).

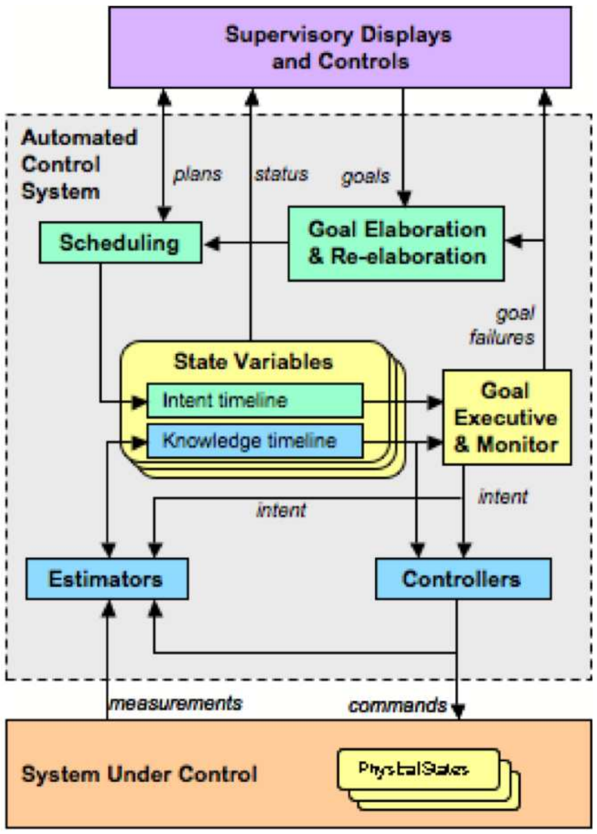

Figure 2 MDS Control Architecture

The MDS reference architecture defines two layers of control management as shown in Figure 2. Individual state variables are estimated and controlled by goal achievers (estimators and controllers) that are responsible for determining system state, and performing the immediate control actions needed to keep their estimated state value within the constraint described by their given goal. A second management layer is responsible for elaborating, planning, and verifying external requests for action, also in the form of goals, and then coordinating the distribution of goals to individual achievers at the appropriate time according to the resulting plan.

A key benefit of a goal-oriented control system is that it exerts continuous closed-loop control over all system state variables, not just the dynamic ones. Goals express continuous intent over time. Contrast this with a traditional command-oriented system (Ref. 10): when a command is issued to close a valve, the system and its users generally assume that the valve will remain closed after the command takes effect. A realistic model of the given system would agree that the valve will usually remain in the commanded position ***unless*** something else happens, such as another command, a manual actuation, or a fault. In a goal-oriented system, a goal for the valve to be closed over a given time interval would ensure that the valve is closed, and *stays closed* over that entire interval.

Furthermore, goals, as constraints on state values over time, can express constraints even on state variables that cannot be directly controlled. Any state variable that can be estimated can be constrained. This provides a mechanism by which a plan can express prerequisite values on state variables such as resource margins, or other safety-related state variables in order to explicitly and formally verify safety requirements at plan time and at run time.

Goals can express dependencies in the form of elaborations: supporting sub-goals needed to ensure the achievability of the parent goal. And, a goal can suggest, through alternate elaborations, different tactics, or alternate ways it might be achieved. Elaboration refines a plan recursively, adding supporting goals that might be needed to allocate resources, stipulate prerequisites, or extend intent expressed on an aggregate state to the states of its elementary parts. Unlike command macro expansion, goal elaboration is a cumulative process: the parent goals remain in the executable network rather than being compiled away, so as to ensure that intent remains explicit. Retaining the elaboration hierarchy enables a robust fault response that can include goal-specific fault responses, or plan repair in the form of goal re-elaboration. This form of fault response benefits from using the same mechanism as is used to expand and verify plans in the first place, thus helping to ensure consistent behavior, and avoiding conflicts between execution and a separately engineered fault response mechanism.

V. Application of the Control Architecture to Human Interactions

When the MDS architecture was conceived it was always assumed that the system under control was hardware. Thus, earlier documentation of the architecture describes the interface between system under control and control system in terms of "hardware adapters". We can conceive of systems, though, where humans are not only present as the operators of a system, but also as integrated elements of the larger system: providing input to the system other than purely directive information, and responding to system actions. If the system is responsible for achieving some goals, and that requires some human participation, then it is possible to model those interactions as part of the system under control using patterns analogous to those defined for hardware.

The MDS architecture defines specific roles for elements of the control system and the system under control. For example, *sensors* are elements of the system under control that can provide information describing the physical state of the system (measurements). *Actuators* are elements of the system under control that the control system can use to change the physical state of the system. The architecture assigns particular responsibilities and constraints on the behaviors of each role, along with guidelines defining how they must be analyzed and modeled. Part of our research was to analyze what might happen if we assigned each role to a human being. The following describes this analysis.

Human as supervisor – When robots support and assist human work, the ideal situation would have the humans supervising, but not having to specify the actions of the robots at a detailed level. Just as a foreman prefers to hire workers who are already trained to perform a given job, human supervisors will prefer robots already suited to the type of task they will perform, so that they actually help the human without requiring inordinate amounts of his time and effort. Ideally, a supervisor only has to issue occasional directions in the form of *goals*, and the robots will be able to determine how to accomplish these goals on their own, using their innate capabilities and sensory capacities. The supervisors may need to think ahead to plan and coordinate the actions of multiple robots, and eventually verify their performance, but they should be able to rely on the abilities of the robots to accomplish the given goals on their own. Supervised robots may occasionally need clarification of their goals or redirection if they run into obstacles.

Human as actuator – The role of an actuator is to perform a predictable physical action when so commanded. At first it seems problematic for the control system to be issuing commands to a human – particularly if that person is also the system operator or supervisor. This would seem to be creating a cyclic dependency, or a role reversal. However, people are already used to interacting with computer programs by responding to prompts to take certain physical actions (put a CD in the drive, provide input data, etc.). People might take issue with the notion of this interaction being in the form of a command, and reasonably so. Thus the fact that the interaction is being modeled in the control system as an "actuator interaction" might only be visible in the control system – the user might prefer to view it differently. What's important is that the control system uses a model of the physics of the system under control, including the ways it can fail. When a computer program must interact with an operator to execute a goal, system design and system operations design must both take into consideration the medium through which the system communicates with the operator, and other activities the operator may be performing in addition to the task at hand (the operator may be multitasking). How should the prompt be displayed to the operator? What if the operator fails to respond? What if the prompt message is lost? What if the operator simply doesn't want to respond at the moment? What if the operator does respond, but the response is lost, or garbled?

Human as sensor -- Humans can be effective sensors, at least when it comes to physical states that they actually can perceive. Computer users are familiar with the paradigm of providing various forms of input when so prompted, and even with the idea of serving as the eyes and ears for a computer program.

From the point of view of the control system, getting information from humans requires analysis and modeling similar to that described above for the actuator role. That is, human inputs, with a large contribution from typical input devices, are known to be noisy in the sense that information transfer is imperfect. Human input is potentially discontinuous, and subject to some latency. Inputs can be mis-typed, mis-clicked, or even mis-spoken. Miscommunication errors can include not only mis-interpretations, but inadvertent false responses (I didn't mean to say that). Human language contains special mechanisms that make corrections for these automatic, so that the participants of conversations usually don't notice them (Ref. 16). Automated systems do not yet tolerate the same level of ambiguity! State Analysis would suggest that, just like any other noisy sensor, human input must be modeled and treated with an appropriate grain of salt. Human-system interface design principles usually address many of the same issues: that questionable or critical inputs should be confirmed with the user and against other evidence, and that inputs should be reflected in a display for user feedback before being accepted.

Human as estimator -- For certain kinds of state variables the human can perform well as an estimator. The ability of people to observe and integrate multiple sources of information, and quickly make sense of the situation is a key justification for putting people in space. However, sharing that understanding with a control system remains problematic. For instance, a person can only pay attention to so many things at once. When a control system depends on human input, what happens if the human walks away, or is distracted by other tasks? What if multiple elements of the control system all demand input at the same time? All of the problems associated with human inputs as sensors apply here as well. However, the role of the estimator defined by State Analysis includes the responsibility to update the state variable as needed by users of that state information. This leaves two options available to the system designer: either all the possible users of that state information must be able to accept the state information on an irregular, and possibly unpredictable delivery schedule, or a human cannot be the sole source of that state information – especially if the information must be collected and compiled routinely, since human error rates (slips) will increase the more routine an operation becomes. Certain state variables lend themselves to irregular updates: parameter states, or states reflecting user preferences. These are states of the system that change infrequently.

Estimators are also responsible for providing truthful representations of uncertainty in the state estimates. Such expressions of uncertainty (whether represented probabilistically or less quantitatively with qualifiers like "possibly" or "probably) may not be straightforward for a human to produce. At a minimum, the human as estimator would need to be careful to remain objective about their estimates, to not bias them or overstate their level of certainty.

State estimates must also be clear about their applicability in time, and it is usually the job of the estimator to label state estimates to describe when they were produced, and when they are relevant. Functions such as these may be routine enough to be implemented as part of a user interface adapter.

Human as controller – A controller's role is to try to achieve a control goal, generally by changing the state of some aspect of the system under control (via commands to an actuator) and continuing to keep it that way as long as the goal is in effect. The first part of this, changing states, sounds like a good match to a humans capabilities. However, vigilance over a period of time, when the chance of an off-nominal event is small, is not one of the more reliable aspects of human behavior. An important consideration in this case is how the system should respond when the state does not match the goal. The MDS architecture separates responsibility for trying to achieve goals from that of responding to goal failures. So, a separate goal monitor is responsible for detecting when state variable values (estimates) are not satisfying their currently applied goals. When this happens the goal monitor initiates a fault response.

Contrast this with the case where the human is merely acting as an actuator. The request to the human to perform some action in either case may appear similar to the human, however, a goal implies a requirement to be continuously achieved over its period of applicability, where a command only expresses the need for some immediate action. Continuous closed-loop control is within the ability of a human as long as the number of concurrent tasks is small (e.g., driving while texting is dangerous because a human cannot devote attention to the task of texting while remaining safely attentive to the task of driving). Furthermore, a human is likely to base control actions on internal mental estimates of system states, and mental models not visible to the rest of the system. On the one hand, this is one reason why humans are often given such roles: because they can be trained more easily than a robust system can be engineered; on the other hand, these hidden models make it difficult to integrate this behavior with other activities safely, because it is difficult to model human behavior in the rest of the system.

If the human is acting as a controller or estimator (i.e., part of the overall control system), the software elements of the control system will still need to incorporate components that can act as interfaces to allow communication between the human and the state variables of the system (which are an integral part of the control system). Once you add any filtering, interpretation, or reminder logic in these components, they inherit some of the responsibilities of estimators and controllers, and thereby weaken the architectural assignments of responsibilities. For this reason it seems simpler and safer to avoid trying to model humans as achievers. Even though they may be providing highly refined information as input, modeling this input as a measurement will allow the control system to manage when inputs are requested, and to filter and interpret the inputs through a filter that considers the qualities of the communications medium, and other uncertainties.

Another thing that should not factor into the modeling considerations is the perception of subordination that might be implied by the use of the terms actuator and sensor as compared with estimator and controller. Clearly, an astronaut might have problems with the idea of accepting *commands* issued by a computer. These terms and roles are used here merely for the purpose of accurately modeling behaviors to achieve a robust control system design, and need not show up explicitly in the actual user interface.

Human as goal elaborator – MDS assigns two distinct roles to a goal elaborator. At plan time, the elaborator produces (i.e., adds to the plan) one or more supporting sub-goals for a given parent goal. Supporting goals may express pre-requisite conditions, or parallel constraints on other state variables that must be enforced to ensure success of the parent goal. For example, to ensure that pressure in a chamber remain above a given threshold over a given interval, it may be necessary to constrain the upstream valve to remain open over that intervl, and to transition open prior to the start of that interval. An elaborator may also define more than one set of sub-goals, each expressing a different tactic for accomplishing the given parent goal. An elaborator usually chooses a tactic according to predefined rules, but there may be cases where an optimal choice is more easily inferred by a human, using evidence that may not be available to the control system, or using those human processes that are considered "creative" or "insightful" – the processes that allow us to see relationships between apparently unrelated things, also let us make strangely correct decisions using little, or even faulty, evidence (Ref. 17).

A second key role of the elaborator is to act as mediator of a fault response at runtime. If any of a goal's elaborated sub-goals fails at runtime, the parent goal's elaborator is asked to determine an appropriate response, which might consist of ignoring the fault, re-planning with an alternate tactic, invoking the fault response of an ancestor goal (failing up), or safing the system. Here again, the responses are usually engineered, but there may be cases where a human would have a broader scope of information to inform an appropriate response decision.

Resolving role conflicts – It may not be immediately obvious that modeling humans as part of the system under control conflicts with the strictly hierarchical organization of the MDS control architecture if the same humans are also supervising the system, which would typically be the case. The control architecture's organization is based on the principal that designers and operators of a system have built and operate the system according to models that define their understanding of the underlying physical system, and that they do so to attain some higher goal that may not be explicit in the implementation. The system under control is something to be modeled so that the control system can react to its behavior to achieve its goals. If the system under control includes a human, who can make supervisory "meta-control" decisions, isn't it impossible to model the behavior of this human in the control system? Not necessarily. The model of the system under control can never be perfectly precise, and must necessarily limit its view of the underlying physics to just the details that are relevant to its particular purpose. In the case of modeling a human as a sensor or actuator, the model need only assume that the human may fail to respond to a request for action, and need not ponder why. Essentially, this assumes that we can effectively distinguish, and distinctly model, the behavior of a human in each role he/she plays.

Is it possible that a human operator might be confused by having multiple roles, and might this confusion lead to inappropriate interactions? The nature of the interactions demanded of each role is distinct: supervisors provide goals, and view status, while actuators and sensors are given specific requests for action. This distinction should be preserved and enhanced by the user interface implementation to clarify the nature of each interaction. (People can always be reminded that, within their own social order, they are also likely to play more than one role, especially if they have children: mentor, judge, nurturer, chauffer, and caterer…)

Human as a self-contained autonomous "subsystem" – On the surface, it seems like the most comprehensive and accurate way to model the behavior of a human element of a system is to model them as another self-contained, semi-autonomous system. This at least describes the relationship in a way that doesn't attempt to simplify or ignore the complexities of human behavior. However, modeling the full depth of this complexity is a significant challenge beyond the scope of our current research.

Associating these roles with human decision-makers is only natural, given that many of the roles were traditional human occupations before the control architecture adopted them, providing a means to automate their functions. However, we have to ask whether incorporating these functions into a larger architecture has imposed new operational constraints that might conflict with the ability of a person to perform them effectively and safely. Answering those questions requires the development of a test system with which we can experiment.

VI. Demonstration System

To focus our investigation we implemented a control system for an air lock (Figure 3), chosen in part because of its relevance to space operations and safety, but in particular because it offers some unique relationships between the astronaut and the control system. An astronaut cannot simply open the front door of a spacecraft or lunar habitat and walk outside without first balancing the air pressure across the door: the inside is pressurized for human habitation, while the outside is at a vacuum. Simply evacuating the entire habitat is expensive, and perhaps an unwelcome discomfort to any other inhabitants. An air lock is a chamber with an inner and outer door or hatch that allows an astronaut to proceed outside (egress) without having to decompress the entire habitat. After entering the air lock the astronaut can close both hatches, then decompress only the air lock, while the habitat stays at normal pressure. Once the air lock is at a vacuum, the outer hatch can be opened, and the astronaut can proceed outside. Similarly, an astronaut can ingress by performing this procedure in reverse order: enter the air lock from outside, close the outer hatch, re-pressurize the air lock, then open the inner hatch.

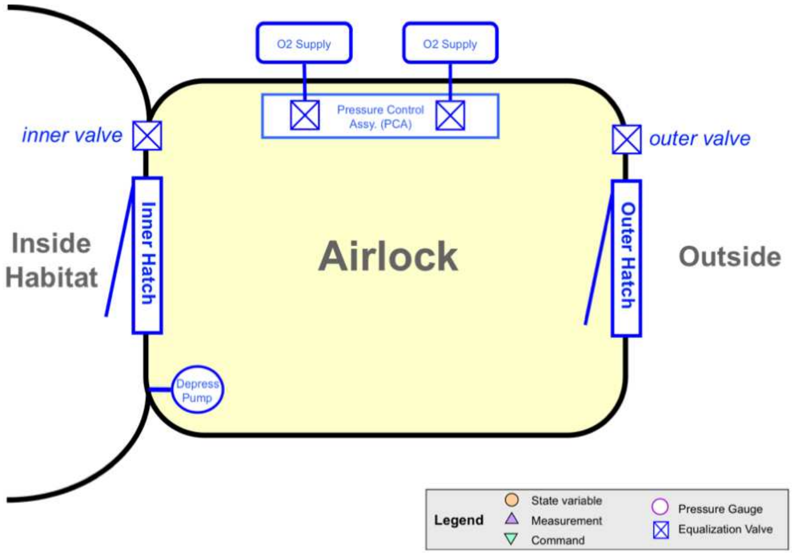

Figure 3. Airlock Physical Configuration

We modeled our air lock system after one in current use on the International Space Station (ISS, Ref. 14). This air lock system has some semi-automated elements, and some manually operated elements. The hatches, for example, can only be operated manually: the astronaut must turn a handle to open the latches, and move the hatch. Valves that control the introduction of supply air are controlled by a Pressure Control Assembly (PCA) subsystem. The PCA is designed to try to maintain a given set pressure. Operation of the ISS air lock is performed as a procedure where astronauts and mission controllers coordinate actions according to the steps of a procedure document they all share.

The fact that the hatches, and equalization valves in the hatches, are manually operated is an intentional design feature of the air lock. Simple, manually operated mechanisms minimize the risk of accidental actuation, as well as the number of failure modes. Astronauts can easily assess the state of a hatch visually while in the air lock. This also means that the astronaut has to keep track of the state of the hatch, along with the states of any other manually actuated parts of the system, of which there can be many.

Procedures can and do help remind a person what they should be doing in a particular situation, but procedures, in particular procedural logic, whether executed by a person or a computer, focus on verifications and branching at a point in time, and they rely on implicit models of the system to ensure that a valve once closed remains closed. As the number of these simple mechanical states grows, so does the number of things the astronaut has to stay aware of to remain in control of the situation. People can only focus attention on so many things at once, so it quickly becomes necessary to rationalize that some things aren't likely to change, and so can be pushed farther from attention. That is, until things don't perform according to expectations, such as when a closed hatch seal begins to leak. Then, quite a bit of effort is needed to review all of the assumptions about the state of the system.

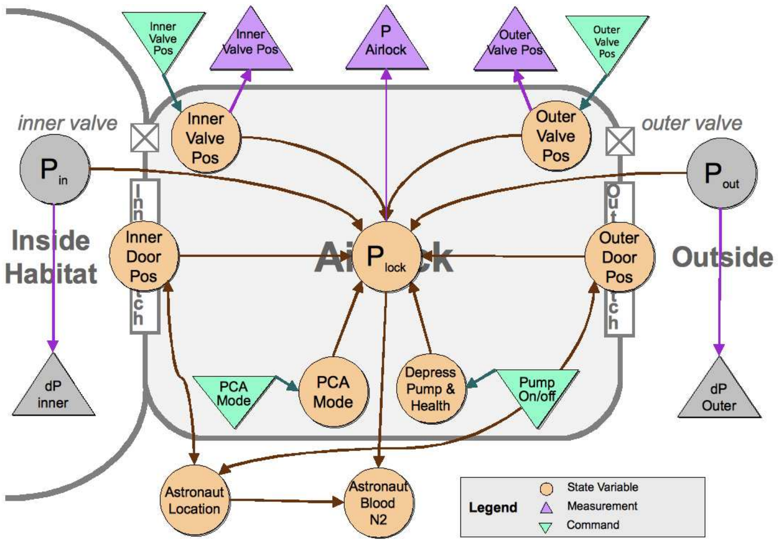

Figure 4. Airlock State Effects

In our example air lock we employ a goal-oriented control system to encode this procedure as a goal network. Our airlock control system cannot directly control all of the elements of the air lock (hatches, e.g.) because they can only be manually operated. The control system can still indirectly coordinate control of these elements through the astronaut by engaging the astronaut to actuate valves and hatches, and determine their current states. From the point of view of the astronaut, this control system departs from the traditional paradigm for a control system. Here, the astronaut is delegating partial responsibility to the control system to coordinate the egress procedure. In effect the control system is reading the procedure (as a goal network), controlling states that it can (pressure supply) directly, and double-checking safety constraints continuously through the process. Figure 4 depicts the relevant physical states, with arrows indicating state effects. This model informs and guides the design of the control system, including its goals: detailed models for each state effect inform estimator and controller algorithms, and elaborations can extend sub-goals onto affecting states to effect control.

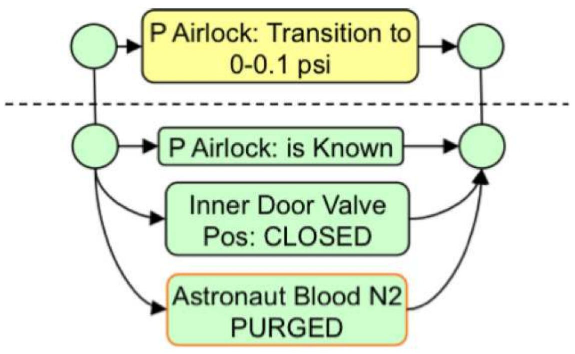
P Airlock: Transition to 0-0.1 psi
P Airlock: is Known
Inner Door Valve Pos: CLOSED
Astronaut Blood N2 PURGED

Figure 5. Partial Goal Elaboration Showing Safety Sub-Goals

To illustrate the ability of the system to continually respond to safety constraints, our demo system includes astronaut blood nitrogen level as a key system state variable. Space suits supply oxygen to the astronaut at a pressure significantly lower than the normal atmospheric pressure maintained within the habitat. When preparing to exit the airlock care must be taken to eliminate nitrogen from the astronaut's bloodstream prior to lowering pressure to the point where the nitrogen could vaporize, causing a condition known as "the bends." This is accomplished by breathing pure oxygen over an empirically-determined period of time. Current procedures merely stipulate a prebreathe period, which was determined as a worst-case best practice. However, the model that informed this value is no longer present in the operational system. In our demo system, a model relating nitrogen density to time is directly used to estimate the astronaut blood nitrogen level state in the system (and this estimation could be made even more accurate through the use of direct measurements of blood nitrogen). As depicted in a fragment of the goal elaboration for a pressure transition goal in Figure 5, the goal network for egress elaborates parallel safety constraints on the blood nitrogen level, requiring a safe nitrogen level any time an astronaut will be exposed to pressure below normal. In this diagram, the parent goal above the dashed line elaborates the supporting goals shown below the line. Thus, this fragment expresses the requirement that whenever pressure is reduced to unsafe levels, the astronaut blood nitrogen level must concurrently be at a PURGED (safe for that condition) state. This example also demonstrates how an elaboration can express a precondition (which is also a safety constraint): that the inner equalization valve must be closed in order for pressure to be able to achieve that level. If a constraint is violated during execution of the egress plan, an automatic response can be invoked: safe the system, or at least suspend further pressure reductions.

It should be noted that similar work has been undertaken by Kortenkamp and his collaborators in the development of a system to automate the execution of some ISS procedures (Ref. 11). Their work starts with the shared premise that some of what is performed in traditional procedures could be automated, but that many more actions still require manual performance. Their system for executing procedures also incorporates human interactions as part of the system, and seeks to make the intent of a procedure (the goal), along with its preconditions, explicit. One difference is that the nature of a goal in a procedure-oriented system is transient: a target to be achieved at some point in time, and then dismissed. Contrast this with the MDS definition of a goal as a constraint on a state of the system over a specified interval of time. Thus, while a procedure-oriented system may achieve a given goal, it does not necessarily ensure the goal remains that way over time. Furthermore, the system describes no underlying models of system behavior (at least not explicitly) that would guide and support the automatic verification and validation of a procedure at design time, or at run time. While this is clearly a step in the right direction, its execution engine retains the procedural semantics of the paper documents it intends to automate. Thus, it can branch conditionally based on inputs from the user or from telemetry, but it does not (as far as we know) continually verify constraints over time. Specific fault branches can be encoded into the procedure at specific points, but there is no general mechanism for addressing faults other than deferring to the wisdom of the user. In effect, in this system the human retains the role of goal achiever, delegating only a supporting role to the procedure executive.

VII. Conclusion and Future Work

An architectural approach to a given problem is intended to provide some guiding principles and unifying concepts that help to make solutions elegant (in the mathematical sense of the word), or at least ensure that solutions are consistent within their context. This work brings together two fields where some architectural principles have been applied separately in the past, and attempts to unify them into a single consistent architecture in which human beings and machines can work together as part of a single system. Viewing the user interface from a control perspective helps to ensure that control requirements such as uncertainty and latency are addressed in the analysis, design, and implementation of a control system where human interactions are involved.

At the supervisory interface, a goal-oriented control system architecture, such as the one provided by MDS, significantly narrows the semantic gap in the communication of intent between the operator and system. Goals express continuous intent over time, eliminating the ambiguity that can result in systems where commands are used to simply change state at a point in time.

Modeling humans as sensors and actuators is simpler and easier than modeling them as controllers and estimators, even the roles might sound more applicable. Even though the information that humans provide as input may have a denser information content, or apply to a higher level system state than a typical sensor, the interface is still noisy and latent. Furthermore, since humans may not be willing or able to continuously provide accurate or usable inputs, they can only be direct estimators of states whose values are only needed intermittently, or where estimates can remain valid for a relatively long time.

Projecting future behaviors of the system is particularly affected when humans are involved in any role. When actuation and sensing is accomplished through directly connected hardware, a control system may need to model the health of that hardware, or its interconnecting or supporting parts, in order to predict how it might behave in the future, and so, whether planned goals will be achievable. Even when humans are an integral part of a given system, designers cannot assume that they are welded to the console. The system may exist in a transient state. For example, the airlock system only exists as modeled when an astronaut is physically present.

Our current work continues to refine the ability of the MDS control architecture and software frameworks to provide the most relevant information to users explaining constraint violations that occur at runtime or at plan time, along with relevant choices that can be applied to the response. Constraint violations that occur at runtime are usually the result of some unexpected change in system state (e.g., a hardware fault), or use some inaccurate models during planning: for example, when starting a car trip you expect to be able to reach your destination with the fuel you have, but later realize that you are consuming fuel faster than expected. Planning is usually a process seeking an achievable arrangement of the given activities in time, where achievability is determined by comparing projected states with the goals (state constraints over time). Explaining constraint violations here helps to explain why a proposed plan was rejected.

Current work also includes the definition of mechanisms to manage supervisory interactions – so called *control authority*. More than one person or system can potentially be in a position to have a supervisory relationship with a controllable asset, but conflicts can arise if more than one of them are trying to control the same system at the same time, and such conflicts can have very negative safety consequences.

We are demonstrating basic capabilities of a goal-oriented control architecture using relatively simple goals, and where the supervisory interaction is limited to issuing individual goals. What if both the supervisory system, and the subordinate system were goal-based systems? For example, an individual ATHLETE robot might have a goal-oriented control system that accepts goals to move and articulate according to some specific activity plan. That plan might show up as just one of many integrated activities in a goal-oriented planning system, where assorted surface activities are being coordinated. Ideally, the planning system would only need to know about subordinate system goals at an abstract level, allowing the subordinate system to elaborate those goals and execute them according to tactics chosen locally to fit current conditions. How much would the supervisory system need to know about the details of execution in order to coordinate the activities of one system with another? How would these interactions work in larger hierarchical organizations? These questions are among those we hope to address in the next phase of our work.

Acknowledgement

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Electronic Publications

[1]Wagner, D., "Mission Data System," [online reference website], URL: http://mds.jpl.nasa.gov [cited 22 September 2008].

[2]The White House, Office of the President, "A Renewed Spirit of Discovery: The President's Vision for U.S. Space Exploration," URL: http://www.nasa.gov/pdf/55583main_vision_space_exploration2.pdf, Released to accompany the President's NASA FY 2005 Budget, January [cited 4 March 2009]

Periodicals

[3]Wilcox, B., et-a, "ATHLETE: A Cargo Handling and Manipulation Robot for the Moon," Journal of Field Robotics [online journal], Vol. 24, No. 5, URL: http://www3.interscience.wiley.com/journal/114211098/issue, Wiley Periodicals, May 2007, pp. 421–434.

Proceedings

[4]Dvorak, D., et-al, "A Unifying Framework for Systems Modeling, Control Systems Design, and System Operation," *The International Conference on System, Man and Cybernetics*, ISBN: 0-7803-9298-1, Vol. 4, IEEE, Kona, HI, 2005, pp. 3648-53

[5]Dvorak, D., et-al, "Goal-Based Operations: An Overview," *Proceedings of the AIAA Infotech@Aerospace Conference*, AIAA-2007-2724 , May, 2007.

[6]Wagner, D., et-al, "An Architectural Pattern for Goal-Based Control," *Proceedings of the Aerospace Conference 2008*, ISBN: 978-1-4244-1487-1, IEEE, Big Sky, MT, 2008, pp. 1-17

[7]Rasmussen, R., et-al, "Achieving Control and Interoperability through Unified Model-based Systems and Software Engineering," *Proceedings of the AIAA Infotech@Aerospace*, AIAA-2005-6918, September, 2005.

[8]Ingham, M., et-al, "Engineering Complex Embedded Systems with State Analysis and the Mission Data System," *AIAA Journal of Aerospace Computing, Information and Communication*, Vol. 2, No. 12, AIAA, Reston, VA, 2005, pp. 507-536

[9]Dvorak, D., et-al, "Comparison of Goal-Based Operations and Command Sequencing*," SpaceOps 2008 Conference*, AIAA-2008-3335, May 2008

[10]Dvorak., D., "NASA Study on Flight Software Complexity ," *Proceedings of AIAA Infotech@Aerospace Conference 2009*, AIAA-2009-1882, April 2009.

[11]Kortenkamp, D., et-al, "Developing and Executing Goal-Based Adjustably Autonomous Procedures," *Collection of Technical Papers - 2007 AIAA InfoTech at Aerospace*, Vol. 1, AIAA, Reston, VA, 2007, pp. 88-97

[12]Estlin, T., et-al, "Enabling Autonomous Science for a Mars Rover," *SpaceOps 2008 Conference*, AIAA-2008-3241, May 2008.

Reports, Theses, and Individual Papers

[13]NASA, "NASA Procedural Requirements (NPR) Human-Rating Requirements for Space Systems," NASA NPR-8705.2B, 2008.

[14]NASA, "International Space Station: Environmental Control and Life Support System (ECLSS) Astronaut Training Manual," NASA TD-9706, 1998.

Books

[15] Norman, Donald A. "The Design Of Everyday Things". *pp 105-114.* Basic Book, New York, NY, 1998.

[16]Ibid, *p 105.*

[17]Ibid, *p 125-129.*