Managing and Instructing Information Assistants

Jane T. Malin[1], Arthur Molin[2] and Carroll Thronesbery[2]

[1]NASA Johnson Space Center
2101 NASA Parkway, ER2
Houston, TX 77058
281-483-2046
jane.t.malin@nasa.gov

[2]S&K Technologies, Inc
201 Flint Ridge Plaza, Ste 102
Webster, TX 77598
281-480-1453 X36
amolin@sktcorp.com

Abstract

We describe agents that are customizable information assistants for space control center personnel who monitor mission operations and equipment and who assist space crew. These monitoring agents can provide flexible services that are not provided by autonomous system management agents on a space vehicle. These agents watch for the occurrence of pre-specified patterns in telemetry data and perform selected actions (notes made in the console log, written reports, notification of people performing specified roles) when those events are recognized. We have prototyped tools to help flight controllers and engineers create, review and supervise these agents. We have developed a higher order specification interface for viewing original specifications and making new, similar specifications. We have studied an existing tool that performs similar functions in a control center, and have applied some of the lessons learned to our own work. This study has led to design of a new user interface for constructing and customizing agent instructions.

Introduction

There are numerous types of persistent assistants that could perform various types and combinations of services for human customers and collaborators. The type of assistant addressed in this paper is an information assistant for space control center personnel who plan, assist and monitor mission operations, who evaluate the health and status of systems and equipment and who respond to anomalies. In the control center context, these assistants provide services to monitor telemetry and software tool use. These services include logging and reporting on events and system performance. They also include opening workspaces, creating actions and sending notices. This limited role contrasts with agents that act autonomously a majority of the time (Martin et al., 2003); nevertheless, the development of such assistants presents challenges that are common with other types of persistent assistants (Malin and Johnson, 2003).

These agents work for specific groups and roles in a control center. Owner groups are responsible for constructing and customizing instructions for these agents. There are users that are very familiar with some agent services and much less familiar with others. There are numerous issues concerning ease of specification, modification and review.

Owner group members (authors and users) need to review and assess these instructions. Simple instructions can be easy to understand while more complex and powerful ones are less understandable. Other groups or roles may also need to review, because of interest in the services and products of assistants they do not control. Finally, owner groups operate and supervise these assistants. They need easy access to assistant status and health and easy means of intervention. Our work on information agents is addressing all these issues of construction, review and supervised operation, in the context of multiple communicating owner groups. Past efforts to implement useful agents have often fallen short in these areas: people find the agents too difficult to create, too difficult to understand, and too difficult to supervise.

Mission Control Centers

The Mission Control Center (MCC) at NASA's Johnson Space Center is responsible for ensuring the safety of a crew of astronauts in a space vehicle, the safety of the vehicle itself, and the success of the mission.

The Mission Control Center is divided into three groups—the front room, where the flight controllers have direct responsibility for the mission; the back room, where their assistants support the front room controllers; and the Mission Evaluation Room (MER), where engineers monitor system health during operations and investigate anomalies. The flight controllers and engineers of MCC track the plans of the astronauts while on orbit, monitor the telemetry that shows the status of the vehicle, and investigate anomalies that develop during a mission. They

record events, conversations, analyses and things to do in the console logs.

Flight controllers and engineers within MCC and MER are divided into groups called disciplines, each with a specific responsibility—Life Support, Power, Medical, and so on. Currently, one or a few individuals who are on shift at any given time represent each discipline within MCC. The flight controllers are a large and changing group of people; new flight controllers are constantly going through a rigorous training process, in order to be ready to replace those who leave.

Currently, flight controllers spend a significant amount of time monitoring telemetry data. Some software is available for monitoring data and logging anomalies, but it has not been widely adopted. One such tool, the Automated Logging Function, or ALF, is currently used by the MER engineers who monitor the Space Shuttle Remote Manipulator System (RMS). ALF users can specify rules involving telemetry values, to trigger creation of log messages when the rule is matched. Rules can match when a telemetry value exceeds a threshold, or takes a discrete value. This tool has been used by one engineering discipline, but it has not been broadly accepted.

The Space Shuttle MCC at Johnson Space Center is fully staffed at all times during a mission, but that is impractical for the International Space Station MCC (JSC-29229, 2003). Because of the continuous mission, MCC is not fully staffed during the night shift. The current night and weekend staffing is handled by Gemini flight controllers. Each Gemini flight controller has expertise in a few disciplines. In the first version of this program, Station Duty Officers from the Communications discipline manned the night shift, and followed a set of instructions from each discipline. These Anomaly Response Instructions specified out-of-bounds telemetry values, and the actions that should be taken if the values were observed. The instructions called for things like logging telemetry values, waiting until a value had exceeded the threshold for a certain period of time, and notifying or calling in flight controllers in the correct discipline. These Anomaly Response Instructions were the artifacts and inspiration for Briefing and Response Instructions (BRIs) for the information assistants.

Our Vision for the Future

Our team has proposed a future view of the Mission Control Center, where tools are available via the World Wide Web, in the MCC itself, in the other offices at Johnson Space Center, and potentially in contractor offices and partners' locations around the world. Software agents will take over the tasks of monitoring data and logging events. When an anomaly occurs, the software agents will notify members of anomaly response teams and support them (Malin, 2004). When the team is called in, they will find the data that they need already available. The software agents and the people will use the same collaboration software to update and track events and issues.

We envision that each discipline will develop its own set of instructions for software agents to help within its area of expertise. These agents will be interdependent, in the same way that the disciplines are interdependent. For example, if the Life Support discipline foresees a problem with oxygen generation, they will inform the Medical discipline. The software agents can communicate with each other and with the flight controllers and engineers. New software agents will be created to monitor specific, short-term issues; when the issue is resolved, that agent could be eliminated, without touching any of the other agents in the environment.

These agents will communicate with the flight controllers using the same software that the flight controllers use to communicate with each other. Flight controllers will be able to easily review the actions of the software agents, and determine what they are doing and why.

This vision takes concrete form with the Briefing and Response Instruction (BRI) System, which consists of the BRI Editor, an easy-to-use, general-purpose editor for developing BRIs, and the BRI Engine, an engine that can process data sources and apply BRIs to them. The term BRI was developed as an analogy to the Anomaly Response Instruction.

Using the BRI Editor, a person can develop the BRIs, which are instructions to be performed when a particular pattern is observed in input data. When the BRI Engine is loaded with a set of BRIs, and connected to a set of data inputs, we call the result an instance of an Intelligent Briefing and Response Assistant, or IBRA. These IBRAs are envisioned as supporting ground operations. They will perform many of the simpler tasks currently performed by the flight controllers in the front and back rooms. They will monitor the telemetry stream from the vehicle, detecting anomalous conditions. When the conditions are detected, the IBRA can log the event, send a message to the discipline responsible for the system, or create an anomaly workspace with status information and references about the condition, to support the engineers and anomaly response team.

Our team has been designing, developing and evaluating a suite of web-based tools as a prototype of the collaborative environment we have envisioned. Extensive human centered analysis has guided the design and development of the tool suite (Malin et al., 2002; Rinkus et al., 2005; Rukab et al., 2004; Zhang et al., 2004). This tool suite, called Team Work Center, has the following teamwork support features:

Team members and IBRAs use the same tools, to facilitate incremental agent development and learning approaches that reuse techniques of users.

Web-based and database-based tools facilitate global and tool-independent access and search.

Explicit content links between tools enable finding things and keeping track of their locations, minimizing extra data entry effort or errors.

The Team Work Center consists of

- Databased Logger - to create a database of log entries that supports review of large log files, automated logging, and generation of reports and specialized logs
- WorkIT Workspace Manager - to collect and share items related to an issue or anomaly or work topic in one accessible workspace, with capability to handle files, links, actions, logs, and paperwork
- Report Maker - to create report formats and collect information from multiple sources (log entries, data, data analyses, notices, actions, procedures, links to workspaces and references) into formatted reports such as shift handover reports and reports about anomalies.
- Notifier - to manage notification of team members on or off console
- Team Work Center Viewport - to provide a summary view and access to recent products of the other tools.

Intelligent Briefing and Response Assistants (IBRAs) use these tools to manage information, and to increase team understanding of events, assessments, priorities, and plans. IBRA actions can be triggered by data from the tools and from from intelligent system management agents (ISMAs) on a space vehicle or base. ISMAs manage spacecraft systems and handle anomalies. IBRA actions include making console Logger entries and assembling reports and briefings in issue workspaces. IBRAs can pass on information from ISMAs concerning goals, procedures, control regimes, configurations, states and status.

IBRAs display their current status in the Team Work Center Viewport so that flight controllers can track the agents associated with their own and other disciplines. It is a way for the people to be aware of what software agents are running, what they are doing, and why.

Design for Use

In designing IBRAs and tools for developing Briefing and Response Instructions (BRIs) we have identified several human-centered considerations. These include ease of construction, ease of review, and ease of supervision.

One of the central goals of our project is to allow domain experts to develop intelligent assistants to help them do their jobs. This is the task of constructing the agent. It involves specifying the conditions under which the agent must act, and what it must do under those conditions. The domain expert must be able to enter this knowledge and annotate it. Often, this will involve looking at existing agents that perform related jobs, extracting the knowledge from them, modifying it and inserting it into the new agent.

Once the agent is created it must be reviewed. Other people than the original creator must be able to understand the agent; the agent might well outlive the tenure of the original creator. The agent must be debugged and verified, possibly by someone other than the creator. As situations change, the agent must be modified. Flight controllers and engineers will also review agents from other disciplines, to ensure that their concerns are being met, and to subscribe to receive products of the agents.

Finally, the agents must be supervised while they are operating. In order for software agents to work as team-members with people, the people must be able to oversee what the agents are doing. People will always have a need to determine

- What the agent is doing
- Why it is doing that
- What it is going to do next

ALF: Problems of Construction and Review

To aid our design work, we have studied the Automated Logging Function (ALF) tool and reviewed the rules that are routinely used. ALF has been used by the Mission Evaluation Room engineers who monitor the Space Shuttle RMS, but to date not by any other groups. We were interested in the kind of rules that engineers would want when using a tool of this nature. In addition, we were interested in why the ALF tool, which has been in use for at least 10 years, has not been more widely accepted.

We determined that one of the limits to wide usage of the tool is the lack of tools to specify and review rules. The rules are specified in a C-like programming language. The target users, while computer savvy, are not computer programmers. A simple rule in the ALF language is shown in Figure 1. Even our programmer evaluators of ALF had to ask for expert help to understand ALF rules.

```
if (changed(#V72X2931J))
  if (#V72X2931J  <
ABE_TEMP_ACC_HI_LIMIT + 1.0)
    logmesg(STATUS_CAUTION,
"ABE_TEMP_QUAL_HI_LIMIT is below
ABE_TEMP_ACC_HI_LIMIT + 1.0");
```

Figure 1 Sample ALF Code

We reviewed the ALF rules that were in use, and determined that almost all of them fell into one of a few general patterns. We identified three basic types that covered more than 90% of the existing ALF rules:

Event Definition: This type defines an event that occurs when a telemetry value or internal variable is set to a value above, below, or equal to a threshold level. In response to the event, the system will either set an internal variable, or generate a log message.

Limit Specification: The range of legal values for a telemetry or variable are divided into high, nominal, low. The high and low sections can be further divided into caution and warning ranges. A "deadband" can be defined around any of the transitions, which prevents bogus messages when a telemetry is hovering right around a threshold value. When the telemetry or variable crosses a threshold, the appropriate log message is generated.

Timing Definition: This defines an event that is made up of a start event and an end event, with a duration associated with the compound event. Both start and end events will individually match Event Definitions, and the same interface would be reused to define them. When both events have occurred, a log message will be generated with the time taken between the events.

We developed designs for template interfaces for domain experts, to help them easily specify these types. We developed prototypes of the interfaces, and are preparing to have ALF users evaluate the prototypes. One of these prototypes is shown in Figure 2.

A user who opens this interface will see three sub-windows. The top window is used to define the conditions under which the rule will become active. The first decision made is to choose the telemetry field or internal variable the rule is referring to. An interface into a mission-specific database allows the user to search for the defined telemetry values, and a scrolling list gives all of the variables that have been defined by other rules. Next, the user will choose the comparison that will be applied to the value. Finally, the condition can be specified. This is an expression involving telemetry field, variables, and constants. Some telemetry fields take discrete values; these values are available in the telemetry database and can be displayed in a list. Other telemetry fields are numeric, in which case arithmetic operations can be applied to the field.

The next sub-window of the interface allows the user to see the log message that would be printed for the given rule. We observed that the log messages printed out by the existing ALF rules usually follow a strict format, describing the event that triggered the message, and we implemented that format as the default message. The user can override the default with a specific message if desired.

In the final sub-window, the user can create or set a variable. These local variables can be used in the other rules.

We are developing capabilities to allow new rule templates to be developed by more sophisticated users; these new rule templates would then be available to be used by non-programmers, to create rules that instantiate them.

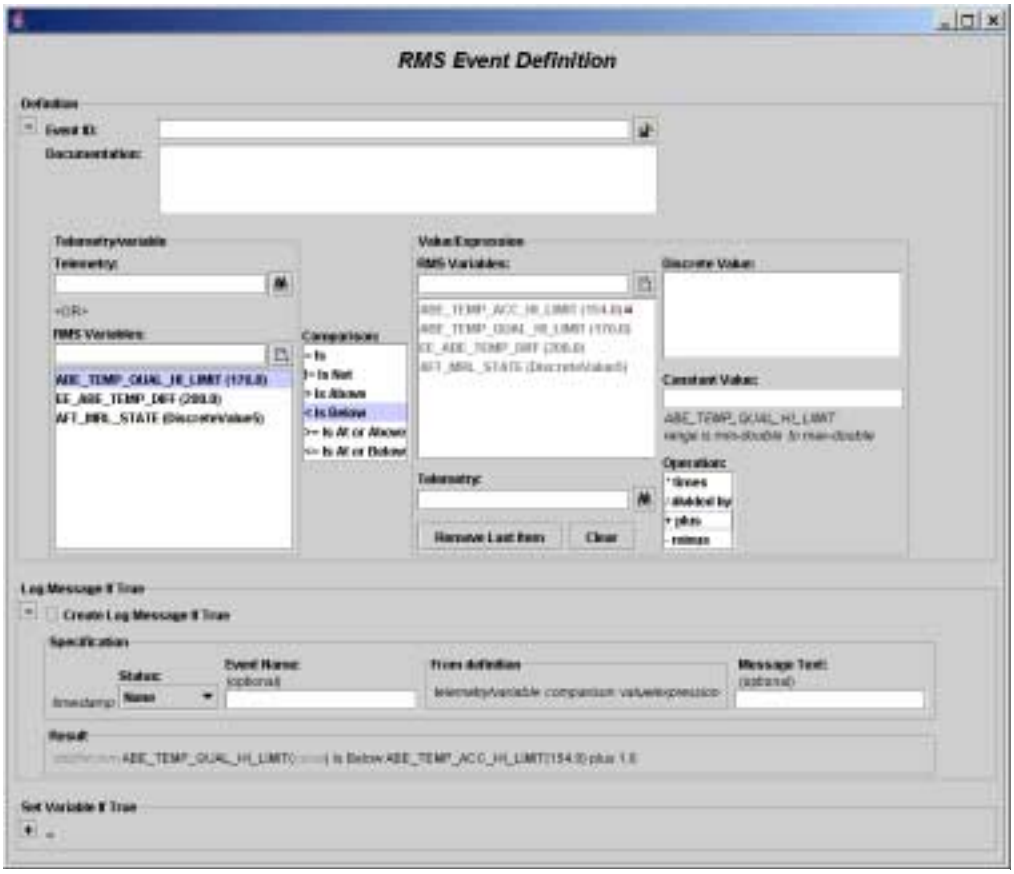

Figure 2 Prototype of a Rule Template

IBRA Construction, Review and Supervision

Construction

Prior to the ALF study, we had developed a general purpose, graphical editor for creating BRIs. The editor has a drag-and-drop style interface, allowing users to choose from a palette of objects, drop them into a graphical window, and connect them together. The editor can be seen in Figure 3, showing a window for defining the conditions under which the rule will become active. The BRI Editor shows the rule triggers and the actions in separate windows.

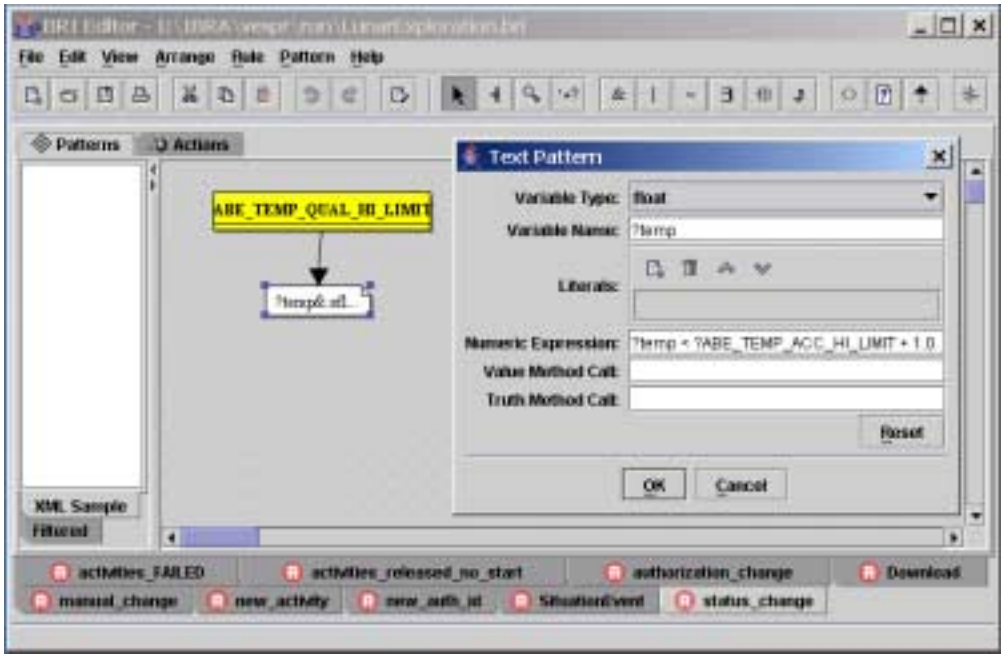

Figure 3 The BRI Editor

This editor has been evaluated twice, first by members of the development team. This evaluation led to a number of changes that were implemented in 2004. The outside evaluator who studied the ALF rules also re-implemented these rules using the BRI Editor. One of the resulting recommendations led to the design of a template interface. Others have not yet led to design changes.

We intend to develop the template editors in such a way that they can generate either ALF or BRI rules. This will give us an alternate editing platform to evaluate. In use, if a user needed to create a "standard" BRI, one that matched an existing template, the template editor could be used. If the BRI did not match any existing template, then the BRI Editor could be used directly.

The BRI System is built around XML data. We have found that the flight controllers have a good understanding of the data produced by the systems they monitor. One of the goals of the design is to have the representation of the pattern that describes data to be as close as possible to the representation of the data itself. As XML is inherently tree-structured, the patterns match that structure.

When building a BRI, it is necessary to be able to explore the data that the BRI is intending to match. The BRI Editor has a number of specific features to support that. Users can load a file of sample data, or load a DTD file that specifies the possible contents of the data file.

If a sample data file is loaded, it is simple to select a portion of the XML tree in the sample data window, and drop it into the pattern window. This creates a pattern that will match exactly that sample subtree, but it is easy to then modify the pattern until it meets the needs. If a DTD is loaded, then the patterns can be build in a "top-down" fashion. At each level, a drop-down list shows the data items that might be found as children of the selected data items. Alternatively, the user can type in the data that is expected, even if that does not match the DTD.

We find that most users prefer to find an existing example that is "close" to the problem that they have in mind, and then modify it for their exact situation. The BRI Editor makes it easy to review the existing BRIs and quickly determine their purpose and function. The BRI Editor also has a number of features to make it easy to cut, paste, and modify existing BRIs once found.

Review

There are two general purposes for reviewing a rule. The first is to understand the details about why a rule has fired. The regular output from the rule firing (new information written to the console log, notices sent out to specified individuals, written reports generated) is intended to contain sufficient information to take initial action. However, additional information is sometimes required for detailed analysis of an anomaly in the monitored system underlying an event recognized by the IBRA agent. In this case, the user can request a view of the rule to see a detailed description of the characteristics required for the rule to fire. Any details the user may have forgotten since

the initial specification of the rule can be reviewed so that a full analysis of the underlying event can be performed.

The second purpose of reviewing a rule is to enable the user to refine the rule specification to reflect new knowledge the user has gained from continued experience with the monitored system. We feel this is a critical aspect of persistent assistant agents: to allow their behavior to be refined as human users gain experience (and knowledge). The refinement can be in terms of either the recognition characteristics or the actions to be taken when the event is recognized. The user interface is essentially the same as for the initial rule specification.

The BRI Editor supports review by allowing the user to load sample data sets and see which of the BRIs are activated by which elements of the data set. This provides a simple testing and debugging environment, as well as a way to review the actions of the BRIs with real data.

The ALF-based templates provide explicit feedback and review of both trigger and action by showing the log message that would be printed. That message typically includes the trigger and implicitly includes the action (logging) for an event rule.

Supervision

To develop visibility into IBRA status we have developed a Status Monitor for human and artificial agents that is available through the Team Work Center Viewport. The Team Work Center Viewport will allow flight controllers to monitor the IBRAs associated with their discipline, to ensure that everything is operating nominally. This interface will allow them to monitor the health of the IBRAs, load and unload specific BRIs for specific situations, and query the IBRAs for expected events.

Since IBRAs are expected to come and go as they are needed, the Team Work Center Viewport is a dynamic view, ensuring that it is up-to-date. Some IBRAs can be marked as "expected" in the configuration files, and if they are absent then their absence can be flagged; others will be transitory, and when they shut themselves down no error message would be displayed.

From the list of all agents present in the environment, the user can choose one, and "push down" to see a more detailed view of its status. This would show the BRIs that are loaded, enabled and disabled, what data sources are connected and which have not produced data lately.

The interface can be seen in Figure 4. This interface is still under development. We need to determine how to present this information to our users in a fashion they can easily understand, and, ultimately, how we can answer the questions that they have about the IBRA agents. We also plan to allow users to control the IBRAs via this interface—to load, unload, activate and deactivate BRIs, to call for reports on recent activities, and to start and stop the IBRAs.

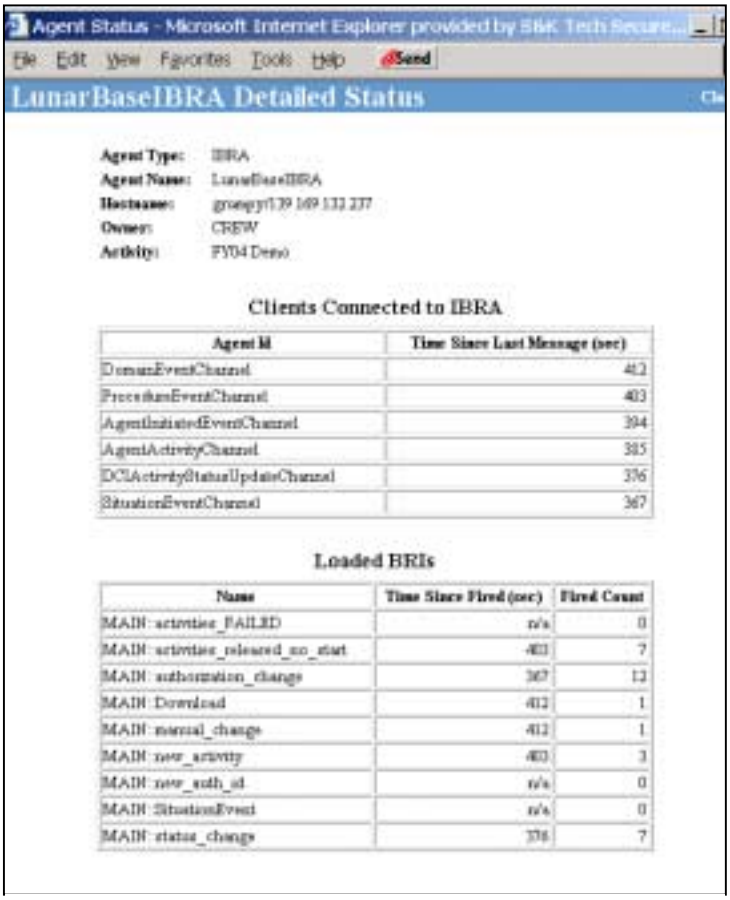

Figure 4 The BRI Monitor

Technical Description of IBRAs

The IBRA system is based on CLIPS, a rule-based programming language developed as an expert system shell at Johnson Space Center. This useful and powerful language is under-utilized, partly because the syntax of the language is unfamiliar and the programming style foreign to most programmers. We wanted to develop a tool that would make the development CLIPS programs easier.

At the same time, we observed that XML was becoming commonly used as a data interchange program. We decided to develop a way to easily process XML data in a rule-based language such as CLIPS. It was important to be able to process arbitrary XML schemas, in order to work with tools that we did not control.

The Team Work Center, Database Logger, and ReportMaker prototypes use ColdFusion and a mySQL database. The WorkIT workspace tool prototype uses ColdFusion and an Oracle database.

Conclusions and Future Work

In the course of this project, we have confirmed that if intelligent software is too difficult to understand, it will not be widely used. The problem of developing intelligent software does not really end with construction and execution. People will have to live with the software for extended periods of time. If they don't understand it, if it does not fit into their world view, if they don't understand what it is doing and why—they will not rely on it or use it.

We have also found that flexibility and configurability are critical. Different user communities have different needs. They must be able to adapt the software to their own processes.

The entire environment has been successfully demonstrated in the past two years. The Team Work Center, WorkIT and Logger tools have been used in simulations in the Mission Control Center, as well as in mission control for NASA Extreme Environment Mission Operations (NEEMO) undersea missions. We are preparing for an evaluation of the BRI System in the Medical discipline of the International Space Station.

We have gone through two design iterations with the BRI Editor, and have incorporated good ideas from several evaluators. Our current focus of attention is the problem of supervision; how the software can be monitored; how it can report its state in a way that can be understood; and what information a user needs about the current status of an intelligent agent.

The largest challenge currently facing NASA is the new charter to return to the Moon and to Mars. This challenge will require a re-evaluation of many of the existing ways of "doing business," and we hope to have our ideas represented. The next-generation manned space vehicles will likely be more autonomous from the ground than are the Space Shuttle and International Space Station, and a manned mission to Mars will have to be more autonomous. This might well be achieved by having many of the functions of the Mission Control Center performed on the vehicle itself, by intelligent software agents much like the IBRAs.

Acknowledgements

Thanks to Nina Patel, who studied the ALF rules and evaluated the BRI Editor. Thanks to Harold (Sonny) White, who helped us understand ALF and the RMS monitoring rules. Thanks also to Patrick Oliver, who implemented the Team Work Center Agent Visibility window, and to Kevin Kusy, who implemented much of the BRI Editor and BRI Engine.

References

Malin, J. T.; Johnson, K.; Molin, A.; Thronesbery, C.; & Schreckenghost, D. Mar, 2002. *Integrated Tools for Mission Operations Teams and Software Agents*, 2002 IEEE Aerospace Conference. Washington, D.C.: IEEE.

Malin, Jane T. 2004 *Interaction between Autonomous Systems and Anomaly Response Teams*. In *Interaction between Humans and Autonomous Systems over Extended Operations: Papers from the AAAI Spring Symposium*. AAAI Press, Technical Report, SS-04-03, Stanford, CA, March, 2004 (pp. 133-135).

Malin, J. T. & K. Johnson, 2003. *Information for Successful Interaction with Autonomous Systems*. In *Human Interaction with Autonomous Systems in Complex Environments: Papers from the 2003 AAAI Spring Symposium*, AAAI Press, Technical Report SS-03-04 (pp. 129-133).

Martin, C.; D. Schreckenghost; P. Bonasso; D. Kortenkamp; T. Milam; & C. Thronesbery. March, 2003. *Aiding Collaboration among Humans and Complex Software Agents.* AAAI Spring Symposium. Workshop on Human Interaction with Autonomous Systems in Complex Environments. AAAI Spring Symposium

Rinkus, S.; Walji, M.; Johnson, K. A.; Malin, J.; Smith, J. W.; Turley J. P.; Zhang, J. 2005, in press. Distributed cognition in knowledge management design. *Journal of Biomedical Informatics.*

Rukab, J. A., Johnson-Throop, K. A., Malin, J., & Zhang, J. 2004. A Framework of Interruptions in Distributed Team Environments. In M. Fieschi, E. Coiera, & Y. C. J. Li (Eds.), *Proceedings of 11th World Congress on Medical Informatics* (pp. 1282-1286). Amsterdam: IOS Press.

Thronesbery, Carroll; Jane T. Malin; Ken Jenks; David Overland; Patrick Oliver; Jiajie Zhang; Yang Gong; & Tao Zhang. 2005, in press. *A Data-Based Console Logger for Future Mission Operations Team Coordination*. *Proceedings of IEEE Aerospace Conference 2005*. Washington, D.C.: IEEE.

International Space Station Flight Controller Operations Handbook (FCOH) (JSC-29229, rev DCN0006). October, 2003. Houston: Johnson Space Center.

Zhang, T., Aranzamendez, D., Rinkus, S. M., Gong, Y., Rukab, J., Johnson-Throop, K. A., Malin, J., Zhang, J.

(2004). An Information Flow Analysis of a Distributed Information System for Space Medical Support. In M. Fieschi, E. Coiera, & Y. C. J. Li (Eds.), *Proceedings of 11th World Congress on Medical Informatics* (pp. 992-996). Amsterdam: IOS Press.