Lessons Learned from the Introduction of Autonomous Monitoring to the *EUVE* Science Operations Center

M. Lewis, F. Girouard, F. Kronberg, P. Ringrose, A. Abedini, D. Biroscak, T. Morgan, and R. F. Malina

Center for EUV Astrophysics, 2150 Kittredge St., University of California, Berkeley, CA 94720–5030

Abstract

The University of California at Berkeley's (UCB) Center for Extreme Ultraviolet Astrophysics (CEA), in conjunction with NASA's Ames Research Center (ARC), has implemented an autonomous monitoring system in the *Extreme Ultraviolet Explorer (EUVE)* science operations center (ESOC). The implementation was driven by a need to reduce operations costs and has allowed the ESOC to move from continuous, three-shift, human-tended monitoring of the science payload to a one-shift operation in which the off shifts are monitored by an autonomous anomaly detection system. This system includes Eworks, an artificial intelligence (AI) payload telemetry monitoring package based on RTworks, and Epage, an automatic paging system to notify ESOC personnel of detected anomalies.

In this age of shrinking NASA budgets, the lessons learned on the *EUVE* project are useful to other NASA missions looking for ways to reduce their operations budgets. The process of knowledge capture, from the payload controllers for implementation in an expert system is directly applicable to any mission considering a transition to autonomous monitoring in their control center. The collaboration with ARC demonstrates how a project with limited programming resources can expand the breadth of its goals without incurring the high cost of hiring additional, dedicated programmers. This dispersal of expertise across NASA centers allows future missions to easily access experts for collaborative efforts of their own. Even the criterion used to choose an expert system has widespread impacts on the implementation, including the completion time and the final cost. In this paper we discuss, from inception to completion, the areas where our experiences in moving from three shifts to one shift may offer insights for other NASA missions.

Introduction

The *Extreme Ultraviolet Explorer (EUVE)* launched on a Delta II rocket in June of 1992. The Explorer class spacecraft carries a set of science instruments designed and built at the University of California, Berkeley (UCB). The *EUVE* mission was designed to conduct the first multi-band survey of the entire extreme ultraviolet (EUV) sky followed by spectroscopic observations of EUV sources. Mission operations are run from Goddard Space Flight Center (GSFC) while health and safety monitoring of the science payload is carried out in the *EUVE* science operations center (ESOC) at the Center for EUV Astrophysics (CEA), UCB.

Shortly after launch, it became clear that NASA's mission operations and data analysis budget faced drastic cuts. With *EUVE*'s early scientific success, CEA sought ways to dramatically lower the mission operations budget in the hope that cost reductions would allow *EUVE* to continue operating past the end of

the nominal mission. We looked at many areas of the project including the possibility of reducing staffing by introducing autonomous monitoring. Because of our lack of experience in this area, we began the process by looking for a partnership with someone possessing relevant experience. We found the NASA Code X and NASA Ames Research Center (ARC) had the knowledge and the desire to help us.

The Explorer Platform is an inherently robust spacecraft designed to last 10 years on orbit. It has both software and hardware safing conditions that can be entered with ground commanding or autonomously by the spacecraft. To date, we have never entered the hardware safehold mode but have autonomously entered the software safepointing mode twice by human error. Like the spacecraft, the science payload is very robust. The payload protects the science instruments with on-board hardware and software safety measures, such as heaters for the mirrors and control of the detector voltage level in the event a detector is being overexposed (detector doors close in the event of a serious threat). The ability of the payload and spacecraft to protect themselves from immediate threats inspired confidence for the development of a system that would detect threats of a less immediate nature, without requiring full-time human monitoring.

Selecting a Package

Lacking the time and resources to create an artificial intelligence (AI) system from scratch, we decided to evaluate off-the-shelf packages. To select an off-the-shelf package, we needed to examine what would be required of it. As a first step, we limited the scope of the problem to ensuring the safety of the science payload. Thus, a heater might be on for two days without sending anything out of limits. This situation would indicate a problem but not an immediate threat. This kind of anomaly was not considered a priority as a human controller could notice it during the dayshift, and until a limit is exceeded it does not put the instruments at risk.

From the point of view of the *EUVE* science operations center (ESOC), we concluded that three areas require monitoring:

1. The *EUVE* science payload. This monitoring is done from telemetry that includes electrical, thermal and physical systems. Appropriate responses to changing conditions are essential to ensure the continued performance of the science instruments.

2. The communications links (Explorer Platform -> NASA -> ESOC). AI software cannot monitor the telemetry unless data are being received. If one of the communications links goes down, the software must recognize the situation and be able to summon someone to restore communications.

3. The hardware and software ground systems in the ESOC. In the event of a failure, the system must be able to summon someone to resolve the problem.

We conducted an extensive search for off-the-shelf products that would meet our needs. CEA tested products in-house for applicability, speed, and ease of use. The cost of the competing products was a factor, as was documentation and technical support. As the search progressed package stability clearly became the critical criterion. With limited manpower and a short schedule, we could not risk software deficiencies. A stable package also helps ensure the accuracy and utility of documentation. This consideration was very important, as we intended to customize the software ourselves. Several good products lacked adequate documentation. These prod-

ucts would have required us to hire the software company for implementation of the system. This would have been prohibitively expensive for our program.

Ultimately, we selected RTworks by Talarian Corporation of Mountain View, CA. RTworks displayed solid performance coupled with excellent documentation and technical support. Importantly, the generalized nature of the RTworks tools allows customizing to our needs. Moreover, the open architecture allows us to easily plug in previously existing code.

System Overview

The ESOC was formerly staffed 24 hours per day, 7 days per week by a payload controller and an engineering aide student. The ESOC is now staffed for only one 8 hour shift per day, 7 days per week. During the off shifts, the customized version of RTworks called Eworks monitors the payload. The ESOC receives telemetry, via GSFC, on a secure X.25 line. We receive real-time data, during contacts with the satellite and postpass production tape dumps. Immediately upon arrival the data are autonomously archived and decommutated using CEA software. During the dayshift, the telemetry is monitored by a controller using SOCtools (an interactive, workstation-based monitoring system developed at CEA). The data are also fed into the RTworks data acquisition module (RTdaq) and the inference engine (RTie). If Eworks detects an anomaly, requests are made to the Epage system to page an on-call payload controller. For visual monitoring of the Eworks software, the human computer-interface (RThci) module is activated.

Lessons from Implementation

We broke the implementation into two teams: one to handle the ground systems and communication issues and the other to deal with monitoring of the science payload. Although the tasks are equally important, this paper deals primarily with the payload monitoring. The communications/ground systems group did come to a very important realization. Monitoring of communications links and local ground systems boils down to the same basic question; is the software receiving data? Or, more accurately, is the science payload being monitored? We determined that the software does not need to know the state of every link in the communications path, it only needs to know if data are being regularly supplied. So, if Eworks does not receive any telemetry for more than 6 hours the on-call controller will be paged. For more detailed information on the ground systems, see Abedini & Malina (1994).

The payload monitoring team consisted of a small group of controllers, hardware scientists, and programmers (from CEA and ARC). The team chose a small set of critical engineering monitors needed to ensure the health and safety of the science payload. The creation of an expert system was not approached by working from the hardware blueprints but rather proceeded based on expert and comprehensive knowledge of the functionality and performance of the payload. After identifying the critical areas to monitor, the team developed and tuned the method of knowledge capture.

We decided to develop an intermediate knowledge representation that would serve as a deliverable product from the domain experts to the knowledge base developers. We used informal flowcharts in a series of documents for each of the major subsystems for which we were automating the monitoring. This approach proved very useful as it cleanly separates the issues of implementation and knowledge representation from the actual knowledge itself. We had some difficulty in representing the domain knowledge in flow-

charts until we freed ourselves from the perceived need to represent the knowledge in a sequential way. On several occasions we found that we were attempting to make the knowledge representation fit into a preconceived, causal flow when it is more naturally and correctly represented by an event-driven model ("event-driven" in that nothing occurs until new data are received). The data are often received in what appears to be an asynchronous fashion because of issues of data quality, dropout, or other effects of receiving our data after the level-zero processing performed at GSFC, as well as the basic complexity of our telemetry stream.

The data-driven nature of the system in itself presents a problem since one of the very things we want to detect is a lack of current data. Ultimately, we cannot predict when we should be receiving real-time or production data since the spacecraft contact schedule often changes at the last minute. Instead we determined that it is sufficient to check whether or not data has been received within a certain number of hours. If we do not receive data from the payload for 6 hours then an alarm is raised. However, since the RTie and in fact the whole RTworks system is driven by the reception of data, we had to create external clients that trip time-out alarms. Fortunately, the RTworks architecture provides a convenient application programming interface (API) for interfacing with custom external clients and the external clients proved easy to implement. Not only is it important that the chosen product proved flexible, but it is also important to note the recasting of the problem. While it is often essential to have an existing working system, to ensure the success of automation one must often recast what needs to be automated. Before our implementation of RTworks, we had operations personnel verify that data were received for every real-time contact, but the essential activity was the verification of payload health and safety on a regular basis. No pressing need exists to verify that data are successfully received on every contact, alleviating the need to predict the contact schedule.

The nature of our telemetry format gives us problems in several areas. While existing missions rarely have the capability to change the nature of their telemetry stream, future missions should carefully examine the form in which their telemetry reaches them. The form can have profound effects upon the ease of automation. Our telemetry is level zero processed by the Packet Processing facility (PACOR) at GSFC before being sent to the ESOC. Thus, we are left in the unfortunate position of having a real-time data stream that has been stripped of all quality information. Since the data delivery format (PACOR messages) does not allow for in-band quality information but, rather, provides it at the end of each data message, we must handle reasoning on uncertain data. This format greatly complicates the implementation and can easily be avoided by providing the full data stream. In today's world of relatively cheap processing and storage resources it does not make sense to marginally compress the data delivered (stripping our quality information compresses the data by less than 5%). Other significant advantages result from keeping the original data stream, including quality, intact. For example, almost all data storage has some life expectancy beyond which the data becomes corrupted. If the full data stream is stored with the original, quality information, it can be reverified every few years.

The basic nature of our telemetry challenges us in that each frame of data does not contain a complete snapshot of all engineering channels. In our contacts with various people and various monitoring and control systems we encountered a widespread assumption that each frame of data contains a sample from all

available engineering channels. It actually takes 128 frames (over 2 minutes) of *EUVE* data to sample every engineering channel, although many are updated every one or two frames. We found that this issue, and the simple fact that the data may contain dropouts (from transmission problems), was not handled gracefully by the RTworks product.

A basic, underlying assumption is that one can reason between the last sample from every engineering channel. This assumption is inadequate because values from the current frame can only be reasoned in conjunction with the most recent expected value, which is not necessarily the most recent value received. Our interactive SOCtools package uses a shared memory segment to decouple the decommutation from display of the engineering channel values. The shared memory segment uses individual timestamps on every engineering channel to deal with this issue (and the timestamps also conveniently serve as a semaphore for multiple, asynchronous, client accesses at the individual engineering channel level).

The RTworks product does not maintain individual timestamps on the most recent values. However, because of the product's flexibility and the quality of the documentation, we were able to modify our customized RTdaq and RTie to handle this issue by supplementing the basic message types between the RTworks modules with a new message type. For gaps in the input stream, the engineering channels expected but missed are sent to the other RTworks clients in one of these new messages. In the case of the RTie receiving such a message, it sets the internal values to *unknown* for the slots corresponding to the given engineering channels. Rules do not fire when the slots they reference have *unknown* values (*unknown* is the default start-up value for all slots). In this way, all slots will either contain the most recent expected value or *unknown*, and thus the integrity of the most recent value model is maintained.

Our reuse of existing code played an important role in how quickly we were able to implement our system of autonomous monitoring. Appropriate data abstractions and code modularization really paid off. The fact that much of our operations software is available through APIs has proven extremely beneficial. It enabled us to rapidly develop the previously mentioned customized RTdaq. Also, since we already had extensive limit-checking code, we did not attempt to create rules in the inference engine (RTie) to do limit checking, but rather we pass in the results of the external limit checks. This procedure has the added benefit of allowing us to easily take advantage of existing code to handle limit checking our real-time data that lacks quality information. This feat is accomplished through the use of what we call tentative limit checking. The first time a value exceeds a particular limit it is treated as only tentatively out of limits, and it is not until a second consecutive update, which also exceeds the limits, that a value is considered out of limits.

Paging

Initially we planned a combination paging and telephony system, but we were forced to scale back our efforts because of resource and schedule constraints. The current, very simple system relies on standard Unix utilities, like cron. We have postponed all efforts in the area of telephony. A key feature of our paging system is its persistence. It continues to page at regular intervals, escalating the number of people being paged after certain timeouts until an acknowledgment has been received. The system requires that the on-call personnel login to the CEA computer system and acknowledge the page(s). Ideally we would support a telephony system that

allowed the page requests to be reviewed from any phone and acknowledged by one or more button pushes. There are a number of services provided by several local and long-distance phone carriers, but to our knowledge none currently allow a customizable feedback feature (non-email based). We have found that the delivery of pages is unreliable, a fact not of common knowledge to most users. Aside from the possible human problems such as turning the pager off or forgetting to wear it, and the occurrence of dead batteries, many structures can cause shielding or interference that prevents the reception of pages. Our operations center is one such location. There is also paging service provider downtime. The low-cost solution is simply persistent pages that continue until some form of acknowledgment is made.

Another ability we did not plan for, but clearly need, is the sophisticated grouping of page requests. Our automated systems focus on the detection of problems and then bring a person into the loop. We have no automated diagnostics that can take multiple alarms and group them together into a single problem (page request). The paging system can handle an unlimited number of page requests, but the user interface is too primitive to allow convenient handling of (acknowledging and closing) multiple, simultaneous page requests.

Living with the System

As we are settling in to our new one-shift scheme, we are discovering the significance of removing humans from the control room. This move has had a profound effect on the flow of information. In the past, records were kept, but a great deal of information was exchanged face to face. During shift changes, noteworthy events could be discussed by the controllers before the ending shift departed. In our current mode of operations, controllers are separated by time and distance. As a result, record keeping and documentation have become critical issues. A controller paged at 2 A.M. will be asleep at 8 A.M. the next morning when the dayshift arrives. In order for the members of the team to act as a cohesive unit, the records left by the paged controller must be clear, complete, and unambiguous.

We also find we are not using the system as we suspected we would. Many expert systems are designed to assist operations personnel, rather than replace them. As such, the graphical human interface is very important. In our case, the display system is secondary, since the major focus is on automating the monitoring of payload systems during unstaffed shifts. As it turns out, our rule base, so far, is not very large (< 500 rules), and over half the rules exist simply to support the human computer interface. This fact is particularly significant, as RTworks compiles the rules for the inference engine at runtime. The unnecessary rules introduce a performance penalty when developing an automated batch processing system. We are considering removing the display rules from the rule set used to process our tape dump data.

The Future

The development of our system is an ongoing process. The broadening of the rule base to include more of the engineering monitors is an obvious route for improvement. But we are working on other important areas as well. For instance, our system only reacts when something goes out of limits. It cannot predict a monitor will go out of limits based on past history and current trends. This kind of predicting is a normal part of human monitoring. We are currently working with software from NASA's Jet Propulsion Laboratory that does raise alarms based on predictions that a monitor will go out of limits. This kind of addition to our inference engine will significantly

reduce the remaining dayshift human monitoring functions, ultimately allowing a move to zero shifts. In some cases, anomalous situations may be detected early and avoided altogether.

Our current system monitors for anomalies; when an anomaly is detected, a person must be called in to deal with the problem. With the expanding capability for on-board fault detection and reaction, the next logical step is to move the autonomous monitoring software from the control center to the satellites themselves. In an ideal situation, autonomous monitoring software would have the ability to take corrective action. If the software can be taught to recognize certain types of anomalies, then it could be taught what action is necessary to deal with the situation. This applies primarily to known anomalies, but it would be an important step toward greater autonomy.

Conclusion

In the current fiscal climate, we are all going to have to find ways to reduce mission operations costs. With the development and availability of low-cost AI packages and proven mission operations software, elimination of labor intensive activities is an attainable goal. At CEA, we are proving that you can remove humans from the control room and obtain a more reliable, safer, and lower-cost operation. As our experience increases and our system matures, we become a model for other missions to follow. Likewise, our collaborative efforts across NASA centers can only help to increase the expertise available for other missions to call upon.

Acknowledgments

We would like to thank M. Montemerlo, P. Friedland, D. Korsmeyer, and D. Atkinson for their support of the development of innovative technologies for NASA missions. We thank Dr. Guenter Riegler of NASA Headquarters and Dr. Ron Polidan and Peter Hughes of GSFC for their championing of innovation on the *EUVE* project. We would also like to thank all the members of the CEA staff who helped to make one-shift operations a reality in the *EUVE* science operations center.

This work has been supported by NASA contract NAS5-29298 and NASA Ames grant NCC2-838.

References

Abedini, A. & Malina, R. F. 1995, Designing an Autonomous Environment for Mission Critical Operations, *Proc. SPACEOPS 1994*, "Third International Symposium on Space Mission Operations and Ground Data Systems," in press.

Malina, R. F. 1994, Low-Cost Operations Approaches and Innovative Technology Testbedding at the *EUVE* Science Operations Center, presented at the *45th International Astronautical Congress*, IAA Symp. on Small Satellite Missions, Sess. on "Low Cost Approaches for Small Satellite Mission Operations and Data Analysis," Jerusalem, Israel, 1994 October 9–14.

Morgan, T. & Malina R. F. 1995, Advances in Autonomous Operations for the *EUVE* Spacecraft, "Robotic Telescopes," *Proc. Astron. Soc. Pac.*, in press.

RTworks, Talarian Corporation, 444 Castro Street, Suite 140, Mt. View, CA 94041, (415) 965–8050.