

Handling Emergencies in Autonomous Systems with An Episode-Incident-Alert Workflow

Paul Baker, Kai-dee Chu, and Cindy Starr

Global Science and Technology, Inc.

Julie Breed

NASA-Goddard Space Flight Center

Jeffrey Fox

Pacific Northwest National Laboratory

Mick Baitinger

NEXTGEN Solutions, Inc.

1.0 Introduction

A low-cost ground system operation must succeed with little attention from operators or engineers. For this reason, there is a strong incentive to design future spacecraft and their support systems for automatic operation. Thus, we anticipate that all future manual interventions will occur in unplanned, emergency situations. In the past, designers needed to worry about maintaining operator proficiency in the face of tedious daily repetition. In the future, our concern must be for the performance of people thrown into an unfamiliar emergency situation.

The *Software and Automation Systems Branch* (Code 520) of NASA Goddard Space Flight Center is developing an *Emergency Response System* (ERS) to coordinate the response of staff who are assigned on a contingency basis and who are not necessarily dedicated to one project. Staff will remain on-call at distributed locations from which they are expected to handle spacecraft anomalies with tools for distributed, group work.

The ERS project has adopted Lotus Notes, E-Mail and WWW protocols as its primary implementation means. These elements are familiar to many. Although we review their application in Section 2.0, “The Emergency Response System”, the primary topic for this discussion is a workflow model that organizes the elements of the ERS to mitigate to several serious drawbacks:

1. When people receive notification of an anomaly from an autonomous system, they are not actively engaged in the operation and lack any awareness of the current situation.
2. Engineers might recognize a problem earlier than an autonomous system and therefore they might correct it with less loss of data. However, no engineers will be stationed at a low-cost ground system.
3. The autonomous system may generate a flood of alert messages that builds an excessive queue before any action can be taken in response to the actual problem.

Our workflow model is called *Episode-Incident-Alert* or E*I*A. An *Episode* is any time sequence that deserves examination. Episodes are recognized automatically; then, telemetry data are assembled to describe what happened during the episode. Automatic analysis programs examine these telemetry data sets in various ways to characterize the engineering state of the spacecraft. Based on that analysis, an episode may be set aside or elevated to the status of an *Incident*, which is an episode that demands attention. Every incident carries with it the telemetry data from the episode as well as any numeric output from the

analysis. An incident is recorded in a data base where it is accessible remotely to anyone with proper authorization.

Lastly, a workflow system sends *Alerts* to obtain help with the incident. An *Alert* is a notification of an Incident that is sent to an engineer or a specialized autonomous agent. If the engineer or agent does not respond in a timely manner, the workflow system issues alerts to other parties until the Incident has been handled.

In summary, many episodes are recognized during normal operation. All are examined automatically. A few may show indications of trouble, and those few are raised to the status of an incident. This step filters the episodes to avoid raising unnecessary concerns. For every incident, one or more alerts are sent to obtain help. When help arrives, there is a package of contextual data available to inform the diagnosis of the incident and the correction of any problem. This contextual data mitigates the difficulties the staff will experience when they are suddenly confronted with an emergency. That is the primary benefit that we claim for the E*I*A workflow model.

A ground system that employs this E*I*A model can provide valuable functions:

- The system coordinates an effective response to a problem without requiring dedicated engineers or operators. See Section 2.0, “The Emergency Response System”.
- The system captures engineering expertise and applies it automatically and routinely to recognize incidents before the problem escalates to a severe level. See Section 3.0, “An Engineering Case for E*I*A”.
- The system consolidates messages and status information so that it does not overwhelm the staff when they must respond to an emergency.

The next section will provide background information on a closely related development, the *Emergency Response System*. Subsequent sections will present a motivation for the E*I*A model. Finally, we will describe how model has been implemented in software.

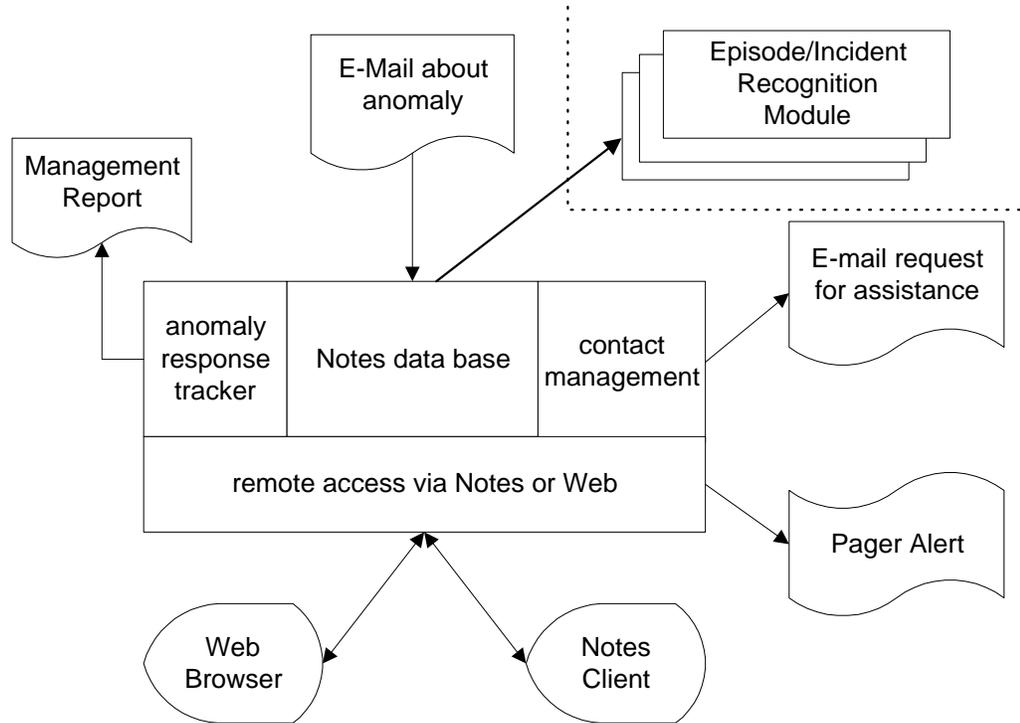
2.0 The Emergency Response System

The E*I*A Model is not a system, rather it is a concept for using a system to provide better operations. To demonstrate the concept in a real operation, we need to imbed the model in an actual system. Currently, we are using the E*I*A Model to enhance a system called the *Emergency Response System* or ERS. In this section, we will introduce the ERS and point out its interfaces to the additional components needed to implement the E*I*A model. The ERS has been and will remain a project in its own right; consequently, this section will digress somewhat from the main topic to explain why the ERS was developed and how it works.

The essential core components of the ERS handle anomaly messages, engage assistance from standby resources, and track the response to an anomaly. Currently, the ERS uses Lotus Notes to implement workflow and store the information where it is accessible

remotely. The configuration of the essential system is illustrated in Figure 1. This figure also illustrates additional components that are used to implement the E-I-A workflow.

FIGURE 1. Core Components of the Emergency Response System



We are currently cooperating with new missions to assist the mission and obtain field experience that will guide the refinement of the ERS. The primary interface from the ERS to the mission ground systems is currently via E-Mail messages as illustrated in Figure 1. The messages are filtered to recognize anomalies, which are then handled individually or in small groups. The ERS implements the following scenario. The steps that use built-in features of Lotus Notes are marked with bold type:

1. An anomaly arrives via **E-Mail** from the event message filter. An **agent** activated by E-mail arrival converts it into an anomaly **document**.
2. Every anomaly **document** is the start of a **hierarchy of response documents** that record the history of the response to the anomaly.
3. An **agent** activated by new documents reads the anomaly, consults a **data base** of recommended resources and **sends an E-Mail message** to the resource. A resource may be a trained person or a specialized software agent.
4. The person (or agent) that receives the **E-Mail message** describing the anomaly will respond to the problem and then file a **response document** with the Notes system.
5. **Agents** activated by a watchdog timer look for anomaly documents that lack a proper **response attachment**. If a preset time is exceeded, the agent triggers a second alert.

6. Outside engineers and agents can use **remote access** to examine the anomaly **documents** using **database forms and views**.

The ERS must be enhanced to support of the E*I*A model because the original E-Mail notifications do not provide enough information. The enhancements are installed as separate modules that are activated by Notes itself. There are many such modules, because there are many possible types of Episodes. In the next section, we describe one type of module. Section 4.0, "Recognizing the Episodes" will summarize the general types of episodes.

3.0 An Engineering Case for E*I*A

It is our opinion that engineers who design spacecraft and test them before launch possess an understanding of the hardware systems that surpasses that of the operator teams working in today's control centers. Operators may overlook a problem that an engineer might detect easily using numerical analysis. Automating the operator's job is not enough. In addition, we need to apply the engineer's expertise, as the following example will illustrate.

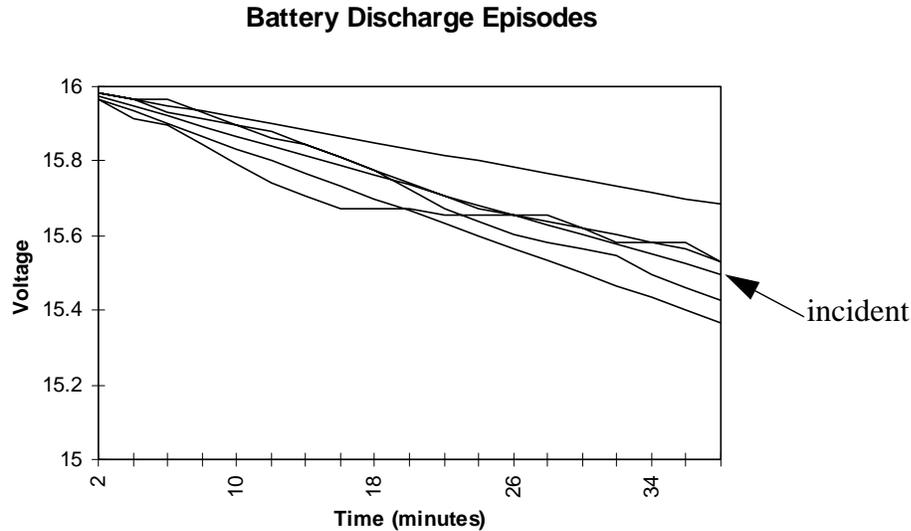
Suppose we have asked the automatic system to record episodes of time when the spacecraft is in the Earth's shadow and the batteries are losing charge without solar power. For every episode, the system assembles measurements of the currents and voltages as they vary over time. A set of such measurements might resemble the simulated set shown in Figure 2. The battery starts at full charge when the Sun goes into eclipse and the episode starts. Thereafter, the voltage drops as charge drains away. The current load varies depending upon what experiments are active, so the episodes produce curves that spread over a band. The simulated data in the figure contains one curve generated for a battery with a severe capacity loss. Although the problem is annotated in the figure, an unaided eye cannot find the problem curve among the normal curves.

In a typical automatic monitor, a Red Alert is triggered when the voltage falls below a certain value. Obviously, that value must be set lower than the minimum voltage that is reached during a normal episode. A fixed voltage limit is only a very inaccurate measure of battery health, unfortunately. The problem curve in Figure 2 stays well above the normal minimum and hence above the trigger level for a Red Alert. Neither would a trend analysis program find the problem because the curve lies within normal slopes and levels.

The battery problem is not easily seen in this one curve because the problem coincided with a period of time when the current load was low. At some later time, the current drain will be higher, the voltage will drop more, and the Red Alert will trigger. Because the problem is real it will surface - when it impacts the rest of the spacecraft! It would be far

better to recognize the problem early and reschedule the loads so that the battery voltage is never lower than expected.

FIGURE 2. Example of Trends during Several Episodes



Once an engineer is alerted to such a problem he or she can easily detect earlier incidents by looking at the records and quantitatively analyzing the response of the battery to its electrical-current load. Normally however, engineers have too many responsibilities to analyze the performance quantitatively on a routine basis. Many technical problems are not be detected until very late.

What we suggest for the future is a new agreement with the engineers establishing a cooperation with the project. The engineers are the key. Only the design and test engineers are really qualified to develop the necessary quantitative tools for problem detection and analysis. Once developed however, the tools can be incorporated in a routine, automatic analysis system. Operations automation can then free the engineers to use their talents creating new systems while still providing an early detection system for technical problems. The E*I*A model was designed to support this operations concept.

Engineers may be interested in how the concept will impact their work. Are they required to build analysis tools in a particular form? Must they learn to program? In the ERS, we offer engineers the option to provide spreadsheet models for various engineering subsystems. Most engineers understand how to use spreadsheets, and no special computer equipment is needed to develop and test the spreadsheets. Ground system designers can then integrate the spreadsheets using new tools for software integration such as OLE automation. A specific implementation of the concept will be described in Section 5.0, "Processing Engineering Episodes".

4.0 Recognizing the Episodes

An episode is an interval of time that is worthy of study to verify the condition of the spacecraft. There appear to be three important categories of episodes and each must be handled slightly differently in the software:

Engineering Episodes. Are intervals bounded by a start time and a stop time that were derived by examining the continuous variation of engineering parameters. In general, we don't know the length of such intervals in advance. Consequently, the selection process must recognize the interval in the real-time data stream and then retrieve the data for the interval from a playback data stream. Usually, we are interested in continuous changes during an engineering episode so the definition of the episode will specify which parameters to sample and a sampling time. The motivation for using engineering episodes was discussed above in Section 3.0, "An Engineering Case for E*I*A". The implementation approach we are using will be described in Section 5.0, "Processing Engineering Episodes".

Scheduled Episodes . Are intervals defined around a scheduled event such as a command. The time interval is known in advance so that the selection process can wait for the arrival of the data and sample data as it arrives. Our current implementation does not work with scheduled episodes for two reasons. First, prototype is not connected with the command management subsystem. Second, we have not had time to develop rule-based software to evaluate the content of a scheduled episode.

Regular Episodes . Are intervals defined with a fixed duration. One such episode follows another indefinitely. Regular episodes have two important applications. First, Regular episodes are necessary when the test for an incident is a statistical test. Such a test must be applied to an interval rather than at a point. Second, a regular episode can be used to consolidate events to prevent overloading the emergency response system with redundant alarms. Our current implementation handles regular episodes, but this will not be discussed because of space restrictions. An unabridged version of the paper is available over the Internet (See <http://abita.gsti.com/July97.htm>).

A key feature of the episode concept is that episodes focus on specific issues - hardware components, performance, etc. It is really the focus that makes them valuable because the specific nature of each episode will allow the ERS to recruit well-qualified contingency staff to handle a problem. Moreover, the automatic system can assemble contextual data on a specific subject that will help the staff resolve any problems. If each episode is focused, however, there must be many different types of episodes to cover all important issues. Therefore, the episode recognition software must run many algorithms simultaneously.

The telemetry data stream must be broadcast to a series of data processing pipelines operating in parallel. Each pipeline takes responsibility for one kind of episode and operates independently of the others. Within each pipeline there are a uniform series of steps beginning with the recognition of the time intervals that define the episode. The next step selects data from the time interval. The selected data are then packaged and sent for classification. If an incident is recognized in the data, its description and associated data are for-

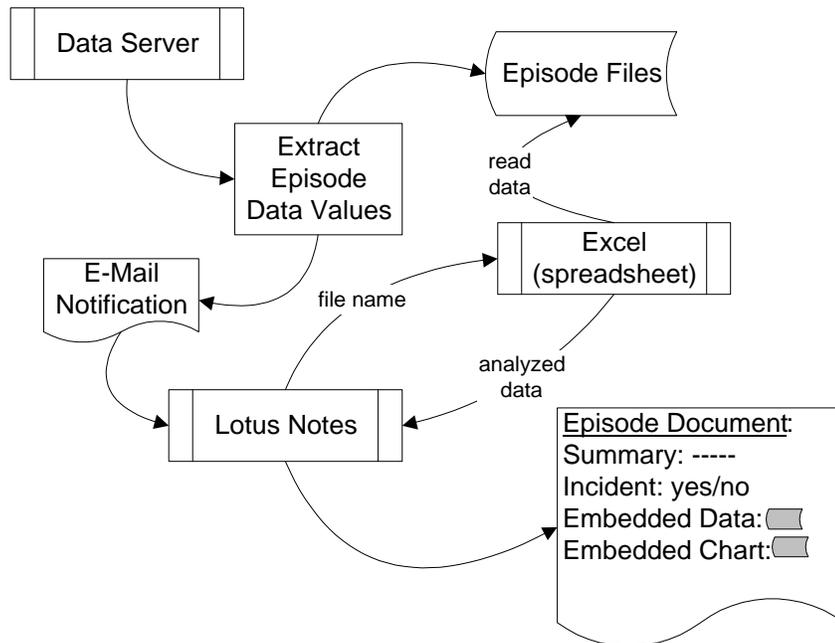
warded to the core components of the ERS which handle the alerts to engineers and operators.

It is important that there are multiple, independent episode handlers because each handler should be devoted to a single, obvious engineering issue. When that is the case, the ERS can take appropriate action following an incident based on the clear definition of the circumstances leading up to the recognition of the episode.

5.0 Processing Engineering Episodes

In this section, we will outline our current approach to processing engineering episodes.

FIGURE 3. Engineering Episode Processing Flow



The data flow for processing one type of engineering episode is shown in Figure 3. Each episode type has a separate processing flow operating concurrently, as discussed above.

The flow begins with the delivery of data values to an extraction program. This program uses numerical criteria to find the start and stop of an episode. In the example cited above, the program might use the output of the solar panels to distinguish periods when the battery is undergoing pure discharge. In any case, it will select a predefined set of data parameters and record their values over the time interval. The results are placed in an episode file, one file for each time period. When the file is complete, the program sends an E-Mail to Lotus Notes to inform it that the file is available. In our current version, Notes then sends the file name to a spreadsheet program: Excel from Microsoft Inc. Excel copies the file data into a template spreadsheet that was prepared earlier.

The expertise of the engineers enters the system via this spreadsheet. The spreadsheet contains a numerical model of a physical process, e.g. battery discharge. When the measured values are inserted in the spreadsheet, the engineer's formulae are used to generate predicted values that are then compared to the actual measurements. In the example we have

been using, the spreadsheet uses the initial voltage of the battery and the measured currents to predict the discharge of the battery and the change in its voltage over the time interval. The prediction is then compared to the measured voltage during discharge. The spreadsheet reviews the differences and compares them to a numerical tolerance, which is also set by an engineer. If the differences exceed tolerance, a spreadsheet cell is set to the classification "incident". When Excel has finished, it has produced a quantitative evaluation of battery performance similar to one an engineer might produce. As a last step before returning to Lotus Notes, Excel uses the data to build a graph that enhances the display of the data, if it must be examined later.

Lotus Notes builds a new document for the episode and then does two things with the spreadsheet. First, it reads selected values and fills in fields in the document. The most important field is the classification of the episode; that is, whether it is an incident or not. If it is an incident, then the ERS handling system will be activated as soon as the document is complete. Second, Lotus Notes will embed the spreadsheet itself in the document. This step ensures that the engineering evaluation will accompany the episode document wherever it goes.

The processing system uses a mixture of software. We are using RTserver and RTplayback from Talarian's RTworks to provide the data server function. We use commercial software, Excel and Notes, plus scripts for that software in Visual Basic and Lotus Script, respectively. The initial extraction of the episode is accomplished with a simple C program.

6.0 Summary and Forecast

We have developed a system, the ERS, that aids in the resolution of unexpected situations on an emergency basis. We anticipate that, very soon, the only situations involving people will be emergencies, because future control centers will run without staff. Our system is based on groupware so that a team can be recruited quickly from remote sites and the team can begin work on an emergency using networking tools.

The E*I*A Model guides the implementation of many operational modes in this system. The model describes a way to organize and present data for an effective response from the emergency staff. The principles of organization are: 1) isolation of a time interval - the episode - 2) focus on one subject - the issue programmed in an episode handler - 3) attention to all subjects - via concurrent operation of episode handlers - and 4) documentation of the incident through embedded data and analysis results.

For additional information please contact:

Paul L. Baker
Global Science and Technology, Inc.
6411 Ivy Lane, Suite 610
Greenbelt, MD 20770
(301) 474-9696
pbaker@gsti.com

and see <http://abita.gsti.com/July97.htm>