

Handling Emergencies in Autonomous Systems with An Episode-Incident-Alert Workflow (Unabridged Version)

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.

The following report is an expanded version of a paper presented at the 2nd International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, held at Keble College, Oxford in July 1997.

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. See Section 4.0, “A Practical Case for E*I*A”.

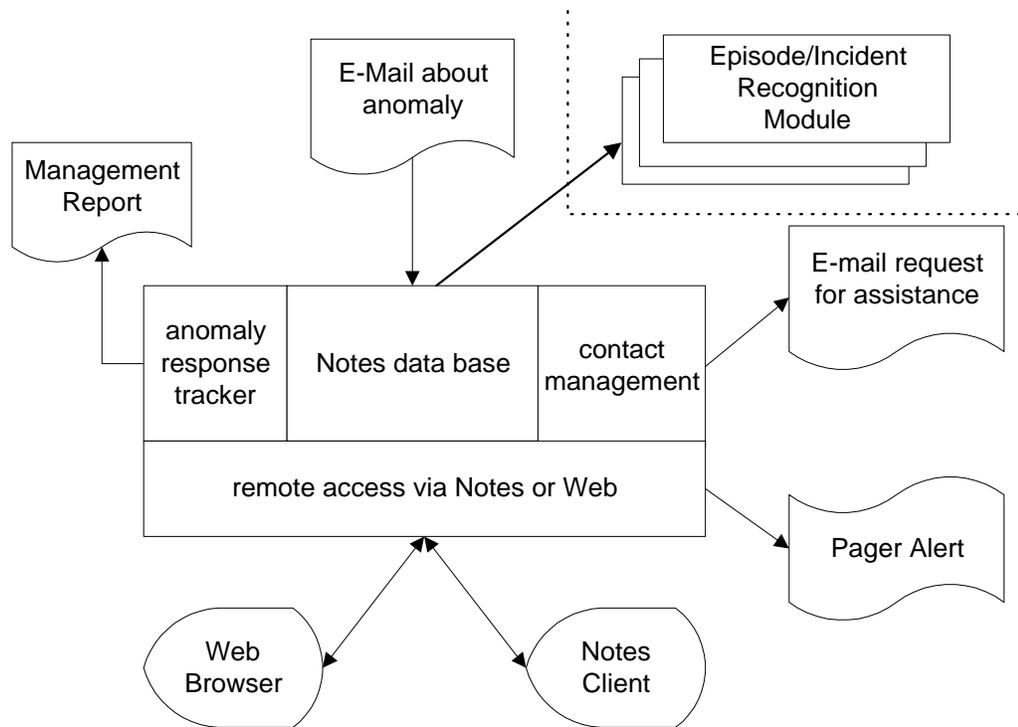
The next section will provide background information on a closely related development, the *Emergency Response System*. Subsequent sections will present a motivation for the E*I*A model. Finally, we will describe how model has been implemented in software.

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



A few words on Lotus Notes may be in order. Notes is a commercial product built upon an atypical database that has some relational and some object oriented features. The database allows distributed access to data and provides optional replication of information to redundant servers. The original goal for replication was the support of mobile computing; that is, the continuation of service during an interruption of the connection to the primary server. For ground-systems, replication promises redundancy and the possibility to limit direct Internet access to the primary server for security reasons. The product incorporates E-Mail and news group features as well as a full set of tools for form-based interaction with the data base.

Lotus Notes workflow is handled by scripted agents, which operate together according to an overall design. Unfortunately, the Notes system implements design with many isolated code fragments, which are hard to maintain. In our application, these scripts recognize problems, identify what person or agent can help, and notify the person or agent.

The most recent version of Notes incorporates an HTTP server to handle requests from Web browsers; although, the normal Notes browser is inexpensive and has better hyper-text features than any Web browser. All Notes documents may contain embedded copies

of document fragments from other applications. These embedded components can be viewed in any client browser, Unix, Mac, Windows. A client with the proper access privilege may also edit embedded documents if the original application is installed locally and OLE is running. We use the embedded documents to distribute context data concerning the ground system incidents.

Notes software is robust and free of internal problems, but it is exceptionally hard to install; moreover, documentation for Notes installation and configuration is very poor. Consequently, it requires considerable time and effort to start an operation based on Lotus Notes. Lotus Notes is the leader in its field, a status it has achieved through a unique combination of features. Any particular feature in Notes is often inferior to the one found in a more specialized competitor. For our projects, it was important to select a mature product with a full set of features. Lotus Notes fit that description best.

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. The next two sections will describe two important types of episodes. Section 5.0, "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

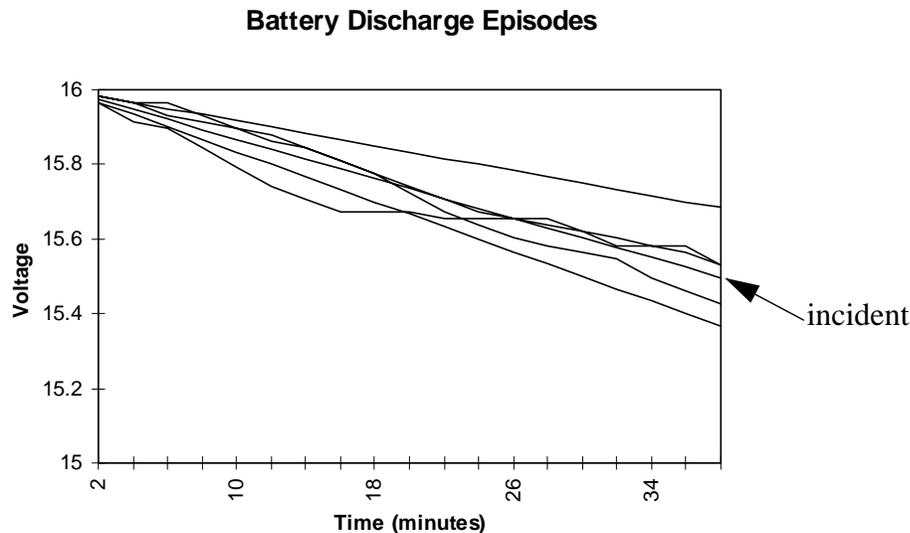
addition, we need to apply the engineer's expertise, as the following example will illustrate.

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 6.0, "Processing Engineering Episodes".

4.0 A Practical Case for E*I*A

In Section 2.0, "The Emergency Response System", we described a system to handle unexpected problems during automated operation. In early tests, the system recognized and responded to event messages that signify a problem. This approach is very common. Many control centers issue event messages and use them to draw attention to problems. The event-based approach has some practical difficulties when it is applied to a highly automated operation. Here are a few of the practical problems:

1. When the satellite system goes from nominal to abnormal operation, the event stream changes from a stream containing no alarm events to a stream that typically contains many, redundant alarm events. Because it takes some time for the ERS to recruit help, alarm events will continue to arrive and great numbers of redundant alarms will be issued. This situation is unacceptable because, when the staff finally contact the operations center, they must respond to each of the redundant alarms.
2. The event stream merges many types of events that pertain to different technical subjects and conditions. The contingency staff has difficulty using this event stream to understand the situation because relevant and irrelevant events are interleaved. Event streams should be filtered into subset streams that are specific to a subject.
3. The staff may decide that a condition cannot be corrected immediately and they may wish to override the system to temporarily disable the alarms. If the alarms are set to fairly general specifications and if the alarm is then turned off, then it is possible that a new condition may occur but no alert will be generated. For example, the alarm may be triggered by a message beginning with "RED_X". If the engineers decide that "RED_X1" is acceptable temporarily and, therefore, they disable the alarm, then a new message such as "RED_X2" will not cause an alarm.

4. The event stream is a poor way to show causal connections and patterns, without considerable additional processing. Consequently, events alone are not the ideal material for the context data that we would like to supply to the contingency staff who must respond to an alert.

We are currently applying the E*I*A model to regular episodes in a way that solves the first three problems and mitigates problem four.

5.0 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 6.0, "Processing Engineering Episodes".

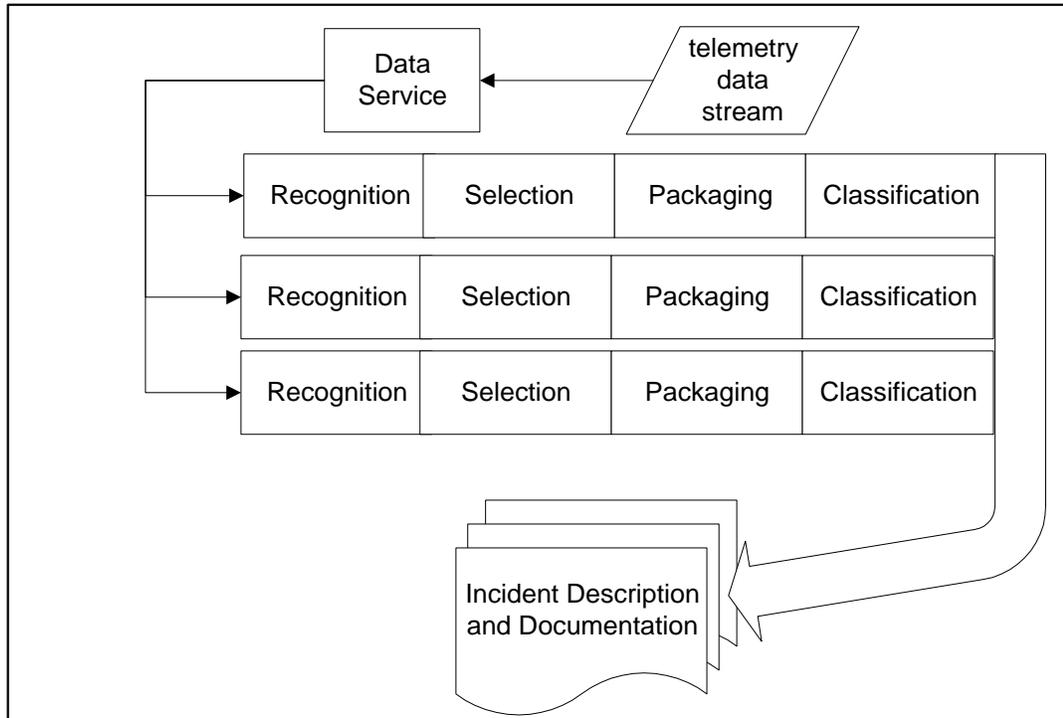
Scheduled Episodes . Are intervals defined around a scheduled event such as a command. The time interval is known in advance so that the selection process can wait for the arrival of the data and sample data as it arrives. Our current implementation does not work with scheduled episodes for two reasons. 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. This issue was discussed above in Section 4.0, "A Practical Case for E*I*A" and our current implementation approach is described in Section 7.0, "Processing Regular Episodes".

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.

Our current implementation uses the software architecture illustrated in Figure 3.

FIGURE 3. A Typical Software Architecture Implementing the E*I*A Model



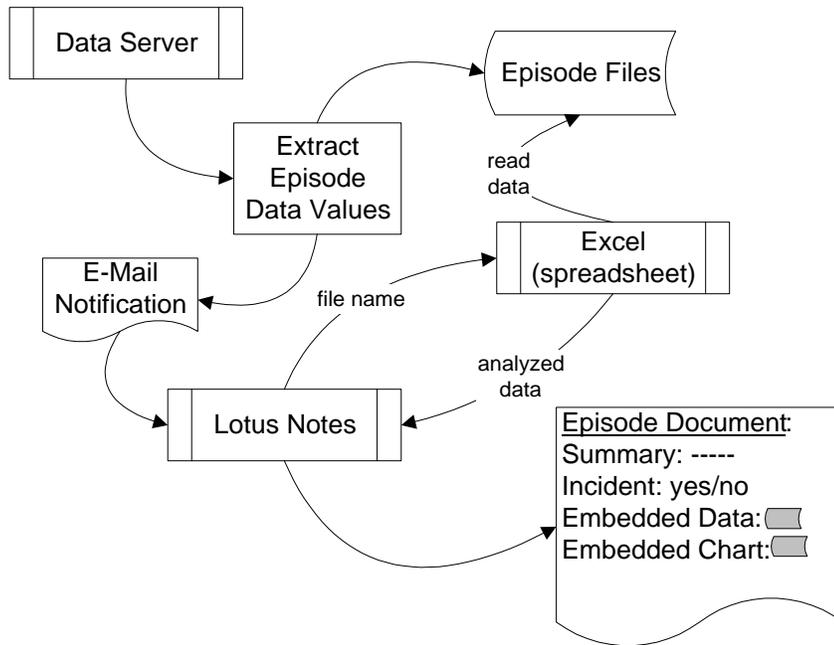
The telemetry data stream must be broadcast to a series of data processing pipelines operating in parallel. Each pipeline takes responsibility for one kind of episode and operates independently of the others. Within each pipeline there are a uniform series of steps beginning with the recognition of the 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 forwarded 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.

6.0 Processing Engineering Episodes

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

FIGURE 4. Engineering Episode Processing Flow



The data flow for processing one type of engineering episode is shown in Figure 4. Each episode type has a separate processing flow operating concurrently, as illustrated earlier in Figure 3.

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 evalua-

tion 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.

7.0 Processing Regular Episodes

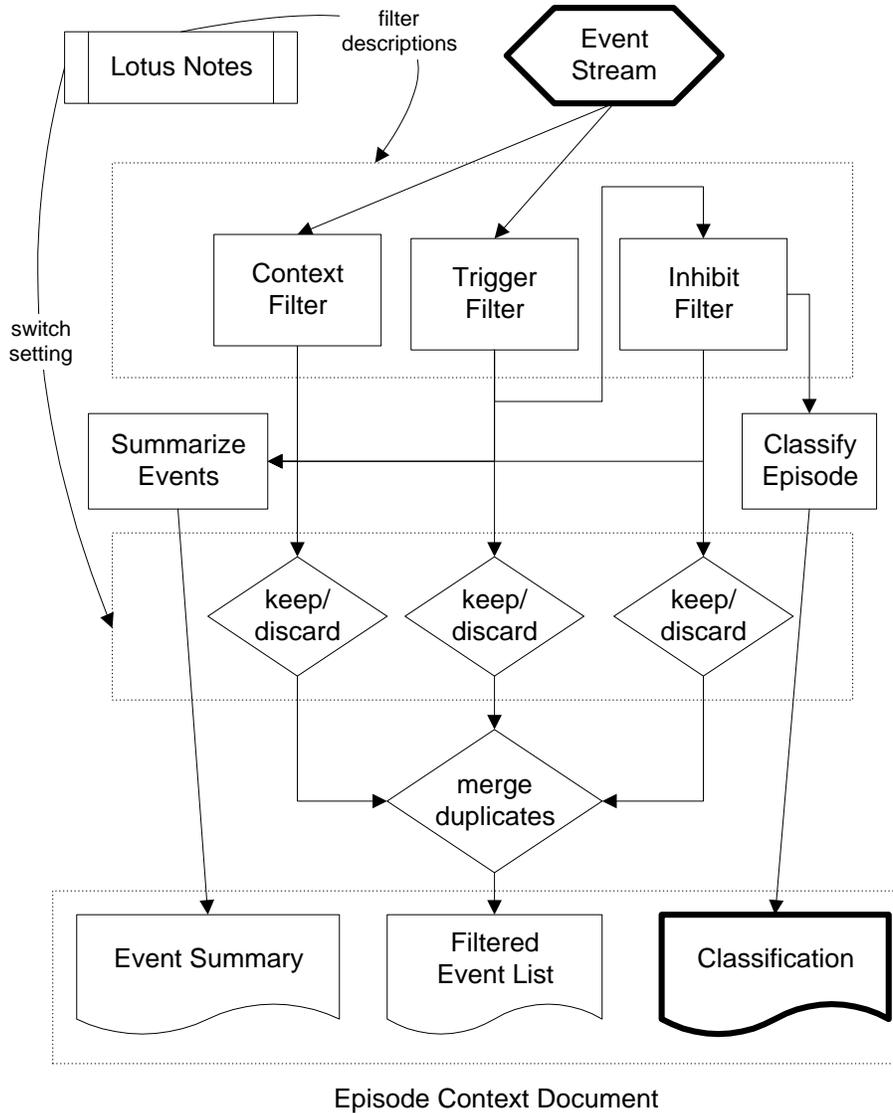
The time interval for a regular episode is determined by a fixed, periodic series of intervals, without regard to what is occurring during the episode. The time interval for the regular episode is chosen strictly for convenience. When the interval is long enough, we can be certain that the ERS will issue only a small number of alerts to the contingency staff. On the other hand, no alert will be issued until the episode interval is complete; therefore, we cannot make the interval too long or an alert will be unreasonably delayed.

Currently, we envision the regular episodes as a means to handle event messages from the ground system. For this reason, the processing of regular episodes must resolve the issues discussed in Section 4.0, "A Practical Case for E*I*A". In any case, the event messages are basically textual; therefore, our implementation of processing for messages is fundamentally a text pattern matching algorithm.

There can be multiple definitions for regular episodes, just as there are multiple definitions for all episodes. Each definition establishes an independent search that will be applied to the regular time intervals. Each search looks for evidence on a particular subject or condition and then assembles contextual data on that subject. Currently, the search technique uses one or more regular expression text patterns, which the software can match against

the text of the event messages. The text patterns are used in each of a set of three filters that are combined as illustrated in Figure 5.

FIGURE 5. Regular Episode Processing for Event Messages



The three filters are:

- Context filter - selects events that are needed to understand the subject of the episode.
- Trigger filter - selects events that indicate the episode must be treated as an incident. If the interval is classified as an incident, it will be handled by the ERS.
- Inhibit filter - may be set to reject any event that passes the trigger filter. In effect, the inhibit filter will selectively override the trigger filter.

These three filters are set by descriptions stored in Lotus Notes, which is used here only as a configuration data base. The description for each filter includes 1) one or more regular

expressions, 2) an instruction to match the text expression against a particular message field or the whole message, and 3) a flag to specify whether the system should save the records that pass through the filter.

The primary features of the processing are indicated by the elements drawn with heavy lines in Figure 5. An Event Stream is the source of information, and the result is a classification. The classification is achieved by sending the events through the trigger filter. There is one enhancement to this basic flow however.

We realize that an engineer will sometimes wish to disable the trigger on certain events if he or she has already studied a problem but cannot correct it immediately. There is a risk if the engineer disables the trigger filters directly. Instead, the engineers and operators are encouraged to disable the trigger only for a narrow range of message types. They can selectively disable the trigger action by entering a vary narrow filter pattern in the inhibit filter. Events that activate a trigger filter but match the narrow inhibit filter will be blocked and have no effect. At the same time, the trigger filter can still respond to other events.

While the filters are operating, the system accumulates a sequence of filtered events that become part of the context data that accompanies an incident. Of course, trigger events should be saved, but there may also be a reason to save a wider range of events. For example, it may be useful to trigger on “red alerts” but store both “red” and “yellow” conditions to better characterize the situation. The filter system allows an easy way to expand the scope of what is stored. Any message that passes the context filter will be stored with the filtered event list.

While the filters operate, the system also produces a summary of events. This summary includes counts of how many events were processed and how many passed each filter. For each filter, the summary also includes a set of event types. Every regular expression has an annotation that specifies an event type that describes events that match the expression. When a filter sees an event that matches an expression, it places the corresponding event type in the summary set.

These summary sets of event types show an overall a pattern that characterizes the situation. In the future, we expect that expert system tools may be able to perform some analysis on the elements of the event type sets. For example, a case-based reasoning tool might use the sets to identify similarities between situations.

The episode filters are implemented as a set of software objects that follow the *View* or *Observer* pattern and observe a common event source object, which follows the *Model* or *Observable* pattern. These software patterns are implemented by reusable code from the RogueWave class library, Tools++. The same library supplies the code for the regular expressions. The time intervals for the episode are set for us by the current ground system. This system delivers events to us as a series of log files occupying about one hour of time each. Because one hour seems like a reasonable interval for our purposes, we have not changed any aspect of this ground system feature.

This concludes the description of the current event message processing. In the future, we would like to add a system that reviews patterns of events for patterns of cause and effect. Such an enhancement would require a rule based system, but none has been integrated yet with the ERS or the E*I*A software.

8.0 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.

In the future, we expect to add additional levels of automation. For example, an incident could be sent initially to an expert system for resolution. If the expert system can suggest and implement an effective response, then no alter needs to be sent to human operators. As a second example, we envision adding case-based reasoning that tracks the incidents and the manual responses, learns how the two are related, and eventually forms its own decisions without human intervention. In other words, we are improving the information flow now for the contingency staff, but the same improvements should also support future developments in automatic operation.

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>