

Situation Views: Getting Started Handling Anomalies*

Carroll Thronesbery
S&K Technologies
Houston, TX, U.S.A.
c.thronesbery@jsc.nasa.gov

Debra Schreckenghost
Metrica
Houston, TX, U.S.A.
ghost@ieee.org

Abstract - *Event-oriented recognition and display software has been developed to assist human supervisors of automated control systems in maintaining situation awareness and getting started handling anomalies in those systems. The recognition software encapsulates data sets describing related events (i.e., a situation), for review by the human supervisor. The display software supports quick overviews plus details on demand, as well as reminding the operator of definitions for the events in the situation and the expected values of associated parameters. This work is part of a larger effort at Johnson Space Center to develop intelligent aids for use by crew and flight controllers during mission operations.*

Keywords: Information visualization, data visualization, situation awareness, intermittent monitoring.

1 Introduction

At Johnson Space Center (JSC), we are developing agent-based software to aid crew and flight controllers in interacting with automated control software during mission operations [9]. When automation is used to control complex systems, like advanced life support systems [1,8], it is important for the automation to support intermittent monitoring and situation awareness [12] by humans, who are ultimately responsible for the safe, reliable operation of those systems [14]. When the automation performs safely and reliably, human awareness of ongoing operations is reduced and human interaction with the underlying system becomes infrequent. As a result, even highly trained human supervisors will become less familiar with the details of the complex, controlled system (e.g., typical performance levels, timing of expected mode transitions). Yet, those very details become critical when human supervisors must understand and respond to an anomaly.

We have developed an approach for automatically detecting and reporting system performance in a way that supports routine system evaluation for situation awareness and anomaly handling. Central to this approach is the *situation*, an encapsulated collection of

data structured according to human understanding of operational events. A situation is organized around a hierarchy of these events involving both nominal and off-nominal conditions or procedures, and it can be contrasted with individual, low-level data changes that are commonly found in data streams. By organizing the data in this way, the top-level view of the situation consists of highly abstracted events derived from a human's mental model of such situations. This top-level view summarizes the transitions among events within the situation and indicates whether these transitions took place as expected. The user will often need to look no further. However, if needed, the details of the situation can be readily explored because they are hierarchically organized according to both the domain system-subsystem relations and the event hierarchy inherent in the situation. For instance, when tracing the event hierarchy, the user can see all the events that are expected within the situation and when they were actually observed. Alternatively, the user can request summary performance parameters for each major sub-system (e.g., a bioreactor within a water recovery system). If a sub-system summary is not sufficient, the user can further request a full set of parameters for the sub-system of interest. At every level, data plots are available showing the observed and expected values as well as the times of observed and expected events.

In this paper we discuss our approach for event-oriented data capture and display. We describe our software architecture for situation detection, notification, and review. We illustrate the use of our software with an example from an actual anomaly involving a crew water recovery system at JSC. We relate our work on situation assessment to that of other researchers. We close by summarizing our results and future work in this area.

2 Architecture

Our support for situation assessment consists of three main components:

- *Situation Detection*: software that recognizes when the events comprising a situation occur,

- *Notification of New Situations*: software that notifies users when a situation is recognized, and
- *Situation Inspection*: software that aids the user in reviewing and annotating a situation summary.

In this section we describe these components and discuss how they work together to provide the user with an integrated situation assessment capability.

The three components of situation assessment are implemented using the Distributed Collaboration and Interaction (DCI) environment[11] (Figure 1). DCI is an agent-based system developed for use in manned space operations. It's purpose is to aid crew and flight controllers in performing their operational tasks. It accomplishes this by providing each person with a liaison agent, called an Attentive Remote Interaction and Execution Liaison (ARIEL) agent. Each ARIEL agent provides services for its user that are customized to the roles its user holds within the space operations organization. These services are listed below:

- **Notification Service**: filters incoming notices based on a specification of what its user is interested in, and informs the user of the notices of interest based on the user's state (e.g., roles, location) and the importance and urgency of the notice.
- **Task Status Service**: tracks completion of tasks on the user's schedule and requests acknowledgement of tasks deemed critical to operations.
- **Location Service**: tracks the current location of its user; location is modeled as a physical location (e.g., in building 4) and a cyber location (online or offline).
- **Commanding and Authorization Service**: authorizes users to interact with vehicle and crew systems, and assists in reconfiguring these systems for interaction.
- **State Management Service**: manages updates to the model of user state maintained by the agent.
- **User Interface Service**: determines how to inform the user of new information (e.g., notices, schedule changes) and manages the resulting interface changes

The DCI environment also includes *augmenting software* designed to interface legacy domain systems with the ARIEL agents. The Event Detection Assistant (EDA) is an example of augmenting software within DCI that detects events in vehicle and crew systems of interest to the users within DCI. The EDA consists of recognition software to detect when events occur and communication software that imports data from vehicle and crew systems into the recognition software and exports events detected by the recognition software to the ARIEL agents.

Situation Detection is accomplished using the Event Detection Assistant within DCI. EDA is implemented using the Complex Event Recognition Architecture (CERA) [5]. CERA was developed in Lisp by I/Net, Inc., under a NASA grant. A CERA application defines event recognizers that specify the environmental conditions that

must hold for an event to be true (e.g., a loss of control communications event occurs when a safety message is observed). CERA extends the concept of *understanding as recognition* [6,7] that comes from natural language processing by recognizing complex temporal relationships among events and by instantiating new events based on the recognition of patterns of constituent events. Composite, high-level events can be constructed from patterns of simple low-level events, resulting in a hierarchical event structure. For our application, situations are high-level CERA events. When executed, CERA monitors a telemetry data stream from the crew systems. Signals within this data stream activate the CERA recognizers, resulting in the creation of event data structures that are exported to the ARIEL agents.

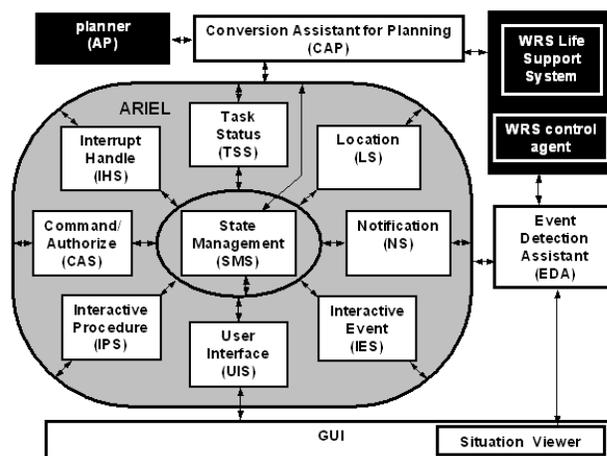


Figure 1. DCI Architecture

Notification of new situations is accomplished using the ARIEL agents within DCI. Situations detected by CERA are processed by the Notification Service of each ARIEL agent to determine if they are of interest to the agent's user and, if so, how the user should be notified [10]. The Notification Service includes an XML pattern-matcher implemented in Java. This pattern matcher compares the contents of a recognized situation to a set of predefined XML patterns that specify which events are of interest to the user. There is a unique set of predefined patterns (called *notice specifications*) for each role that a user can fulfill (e.g., flight surgeon). Each notice specification defines the conditions to be matched as well as the directives to perform when the match holds. The notice directive specifies how urgently and emphatically to inform the user of an event, as well as what interaction mechanism to use (e.g., computer display, pager). If an incoming situation matches the conditions of a notice specification, it is annotated with the information below:

- **Latency**: the urgency of the notice (immediate, deferred, or archive)
- **Focus of attention**: the emphasis to be placed on the notice (primary, secondary, or no-shift)

- Modality: the interaction mechanisms to be used when notifying the user

These three annotations are used by the User Interface Service to notify a user of the event in the specified manner. For example, a situation of immediate latency and primary focus of attention might result in a page with a 911 in the subject line.

Situation Inspection is supported within DCI by providing the capability to launch a Situation Viewer from the ARIEL user interface for all situations of interest identified by the ARIEL agent. The Situation Viewer is a Java process that displays the event information captured by EDA in a situation and overlays it with comments added by users reviewing the situation and knowledge about the expected timing of events and the expected value changes of parameters within these events. The details of a situation can be inspected either from the perspective of the domain system-subsystem relations of systems affected by the situation or the event hierarchy inherent in the situation. The remainder of this paper describes the Situation Viewer and illustrates its use for an example anomaly in the crew water recovery system.

3 Example In Use

Since the purpose of the situation viewer is to allow operators to gain insight into situations that have occurred, it is important to follow the use of the situation viewer through an operator's investigation of a specific occurrence of a situation within a specific situation. This enables the reader to see how the situation viewer exposes information under the operator's control. Later sections will present the software constructs that make this viewer reusable for new types of situations and in new application domains.

Monitored System. Johnson Space Center is conducting a series of test for regenerative life support systems that can operate continuously for long periods of time in a largely autonomous fashion to support science crews in remote habitats. The data shown in the scenario below is actual data from one of these tests.

In the mission operations concept that these life support systems are intended to support, the crew is responsible for most of the day-to-day operations of science experiments and maintenance of the habitat. Most of these functions are highly automated to lighten crew workload. Unfortunately, even the best automation requires occasional human attention to ensure that it is functioning properly. The crew are expected to be familiar with the principles behind the life support hardware systems and the software systems which control them. However, when humans are responsible for a large number of such systems that do not require much

attention, they will inevitably forget some of the details of operation (variable names, detailed control policies).

The example below is based on actual logged data from an experiment with an advanced Water Recovery System (WRS) at Johnson Space Center. The WRS consists of four sub-systems: a Packed Bed Biological Water Processing System (PBBWP) which removes nitrogen and carbon compounds from the water, a Reverse Osmosis (RO) system which filters out small particles, an Air Evaporation System (AES) which evaporates the brine waste product from the RO to remove salts, and a Post Polishing System (PPS) which performs a final polishing of the water to remove trace carbon compounds. In addition, there was a software control system which controlled the four sub-systems of the WRS. The software controls system consists of a skills layer, which performs low-level closed loop control operations and reports the low level data which is logged in an archive file. A second layer in the controls system manages higher level operations (e.g., switching operating modes for a sub-system in response to changing conditions of the WRS). This layer is called the Reactive Action Procedures (RAPs) layer.

Situation Review Scenario. In this scenario, the RAPs layer loses communications with the skills layer (called a Loss of Raps Communication, or LORC). When the skills layer detects the loss of communications, it responds by safing all four sub-systems and sending a safety message indicating that RAPs communications have been lost. A human is called in to handle the anomaly. This requires manually resetting the control software, which results in a restoration of RAPs communication and the restarting of all four sub-systems (called a Restoration of RAPs Communication or RORC).

Figure 2 is the initial view of this situation. From this view, the operator is able to see that a LORC-RORC Situation occurred, lasting from last in the night of May 23, 2001 until the next morning. If the operator has temporarily forgotten the acronym, the tool-tip text provides quick assistance, showing that it is the loss and subsequent recovery of RAPs communications. This display also informs the operator that the LORC-RORC Situation consists of two sub-situations, occurring in sequence: a LORC Situation followed by a RORC Situation. All four sub-systems (RO, AES, PBBWP, and PPS) completed safing operations shortly after receiving the safety message. From general knowledge of the WRS controls software, the operator will know that regaining RAPs communications is a manual procedure that could not occur until people arrived at the WRS laboratory the next morning, which explains the delay in restarting the subsystems. Finally, all four sub-systems restarted shortly after RAPs communications were restored.

During this short review, the operator has been reminded of what the LORC-RORC Situation is and has seen generally how this instance of that situation played out.

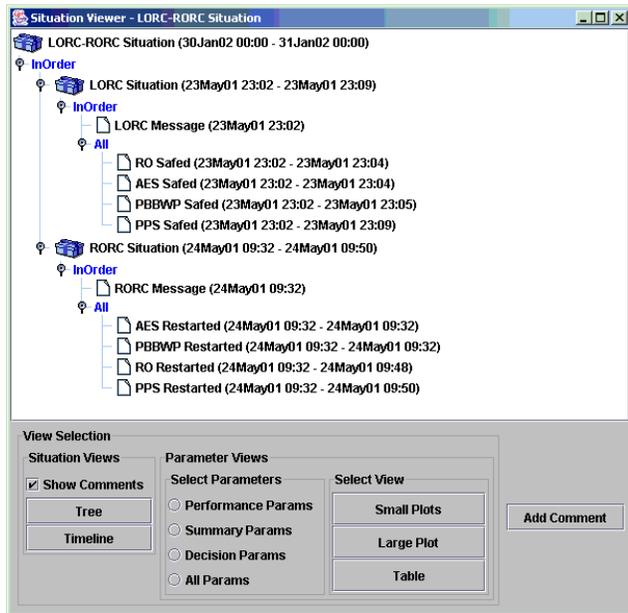


Figure 2. Situation Hierarchy

If the operator wishes to have a closer look at the details behind a portion of the event hierarchy, a click on a leaf-level node (e.g., RO Safed) will result in a view like that shown in Figure 3.

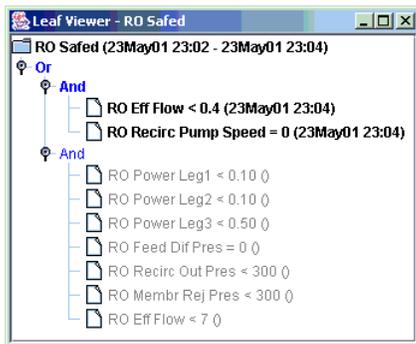


Figure 3. Event Logic and Related Observations

From this view, the operator can see the logic required for concluding that the RO sub-system has been safed. This view tells the operator that if either of two complex conditions exist, the RO will be designated as safed. It also indicates that on this occasion, the first of the two complex conditions was observed (indicated by the bold text and the timestamps next to the conditions).

A summary of system functioning during the situation is available by selecting the “Summary Params”

view. By selection the “Small Plots” option, the view in Figure 4 will then be displayed.

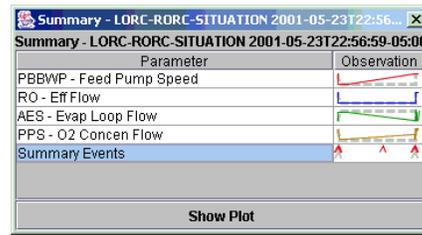


Figure 4. Small Plots of Summary Parameters

For each parameter, the dashed gray line represents expected values over the course of the situation and the solid colored line represents observed values. Generally, the colored lines follow the course outlined by their respective expected lines, although some seem to start drifting a little in the middle of the situation. The operator notes that this may warrant further investigation. The bottom row of the small plots view marks where events within the situation occur. As with the parameters, gray marks indicate where events are expected and colored marks indicates where events were observed. There is an event at mid-situation which has no corresponding expected mark. To see if this “unexpected” event is related to the drift of some parameters, the operator requests a table view of the summary events (Figure 5) by clicking on the lower line of the small plots table.

Summary events			
Observed Time	Expected Time	Name	Description
23May01 23:02	23May01 23:02	LORC	Safety Message: Loss of RAPs Communications
23May01 23:09	23May01 23:07	Safe	Safety configuration, all four systems
24May01 05:19	-----	Loss Skills	Loss of Skills Comms
24May01 09:32	24May01 09:32	RORC	Recovery of RAPs Communications (RORC)
24May01 09:50	24May01 09:50	Restart	Restart completed, all four systems

Figure 5. Summary Events Table

The event in this table with no expected time is the loss of skills communications. From their general knowledge of the WRS controls system, operators will know that without the skills processes, no data are recorded. As a consequence, the apparent drift in the parameters beginning at mid-situation may be the result of a plotting artifact (using a straight line to estimate the data between the two observed points). To investigate this hypothesis, the operator requests a large plot, which shows observed points (see Figure 6 below).

The large plot view in Figure 6 marks observed points by plotting a shape (+, X, triangle, square). Where the parameters are drifting in mid-situation, there are no observed points, so the operator’s hypothesis is confirmed. If the operator needs to see exact data values

for any of the parameters, a tabular view is available. At this point, the operator has become satisfied that everything is according to expectation. The operator records as comments these conclusions about the mid-situation drift of parameter values so that future reviewers of this situation will not be confused by them.

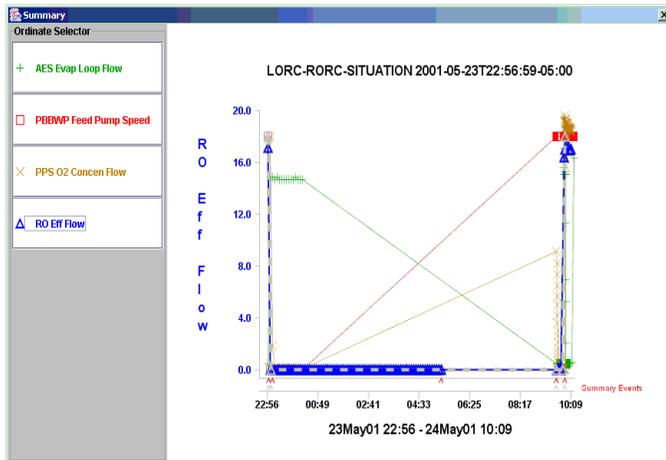


Figure 6. Large Plot of Summary Parameters

4 Information Sources

The use of standard data structures in the information sources for the situation viewer make it adaptable to new classes of situations and to new domains beyond life support systems. Some of the information sources are defined a priori, by a situation designer before data are collected from the monitored system. Other information is collected at runtime, identifying parameter values and observed situation instances.

Data Log. The first source of data used by the situation viewer is a log of system parameter values collected at runtime. While this log is quite large, the information kept in the log is rather simple, identifying the parameter, the time of the observation, and the parameter value. In the example above, the parameter values of each sub-system were logged every 5 minutes.

Situation Instance. This is the report from the situation recognition software indicating what situations were recognized and the structure of those situations. A situation instance includes which events were recognized, which conditions within the structure were observed, and the detection times for each event in the hierarchy.

Situation Specification. The situation specification defines how to recognize situations. It is class level information defined a priori. It includes information about the hierarchy of events to be recognized within a situation, as well as key times and expected values for parameters that characterize these events.

Parameter Specification. The Parameter Specification is also made a priori, identifying the links between parameter identifiers used to retrieve data and more descriptive names for the parameters, units of measure, display format, and ideal plot ranges. While not strictly required, this additional information makes it possible to provide more informative displays to operators who may have forgotten a few of the parameter details.

5 Related Work

The task supported by our situation recognition and display software is the review of meaningful events in a monitored system. These can be events that occur during the course of normal operation or events that characterize an anomaly. In both cases, the user must first understand the situation indicated by these events. Endsley [4] has investigated the use of task-oriented software design to support such situation awareness. According to Endsley, situation awareness can be thought of as an internalized mental model of the current state of the operator's environment. The event recognizers used in our approach capture the user's mental model of operational events and reflect this model in the event hierarchy represented in the situation view. Endsley's view of situation awareness includes the perception of the elements in the environment, the comprehension of their meaning and the projection of their status in the near future. While the meaning of situation for Endsley is more encompassing than our integrated event-oriented data, the encapsulated event-oriented data structure presented by our situation viewer provides an important part of the information necessary for constructing such situation awareness.

Christoffersen, Blike, and Woods [3] have investigated the human process of formulating event-oriented perceptions from low-level data. They indicate that the perception of events (or situations) as unified, organized collections of data occurring over time is analogous to the perception of objects in space. This work has strong implications for the types of software that will support this human cognitive process. They analyzed anesthesiologists' interpretations of emergency room data. They found that events are knowledge-driven (e.g., even the absence of change was informative when change was expected). Our approach to automated situation capture and presentation is to assist the operator in applying knowledge-driven identification of events and the encapsulation of related data. They also found that events are data-driven (i.e., most events were indicated by sudden changes in properties of system data). While we use automated event recognition based on pre-defined patterns of data changes, we also provide manual options to display any variable of interest. This allows operators to discover relationships not defined a priori.

Christoffersen also found that human-identified events are multi-level and nested, which is accommodated by the hierarchical definition of events in our software.

A third area of related work is information visualization [2,13]. According to Card, Mackinlay, and Shneiderman, information visualization is the use of computer-supported, interactive visual representations of abstract data to amplify cognition. They present a reference process model of knowledge crystallization that supports cognitive amplification through visualization. The reference model includes such processes as information foraging, searching for schema, instantiating schema with data, and packaging patterns found in output products. Our approach to situation recognition and presentation assists these processes by supporting data exploration and by organizing data into meaningful collections for detailed examination and display to others.

6 Conclusions

We have developed situation display software that works in coordination with situation recognition software to support mission operations personnel in maintaining situational awareness. We have tested this software using data recorded during a series of experiments with advanced life support systems at NASA/JSC. This software identifies important events as they occur, encapsulates information describing those events, and presents it to assist human supervisors in understanding those events. The software design uses data structures that allow it to be applied to other situation types and to new domains. This effort is part of a larger effort to develop intelligent aids for crew and flight controllers.

We intend to evaluate the recognition and display software with engineers responsible for supervising intelligent control systems. We also will expand to new situation types and to new life support systems. Based on what we learn, we will develop software for users to specify situation definitions.

References

- [1] Bonasso, Pete; David Kortenkamp; & Carroll Thronesbery. "Intelligent Control of a Water-Recovery System: Three Years in the Trenches." *AI Magazine*, 24(1), pp. 19-44, 2003.
- [2] Card, Stuart K.; Jock D. Mackinlay; & Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, Inc.: San Francisco, CA, 1999.
- [3] Christoffersen, Klaus; George Blike; & David Woods. *Making Sense of Change: How Practitioners*

Extract Events from Data Telemetry Streams. Institute for Ergonomics/Cognitive Systems Engineering Laboratory Report, ERGO/CSEL 02-TR-04, 2002.

- [4] Endsley, Mica R. "Designing for Situation Awareness in Complex Systems." *Proceedings of the Second International Workshop on Symbiosis of Humans, Artifacts, and Environment*. Kyoto, Japan, 2001.
- [5] Fitzgerald, Will; R. James Firby; & Michael Hanneman. Multimodal Event Parsing for Intelligent User Interfaces. *Intelligent User Interfaces*. Orlando: ACM, 2003.
- [6] Fitzgerald, W. *Building Embedded Conceptual Parsers*. Ph.D. Thesis, Northwestern University, 1994.
- [7] Martin, C.E., *Case-based parsing and Micro-DMAP*, in *Inside Case-Based Reasoning*. C.K. Riesbeck and R.C. Schank, Editors. 1989. Lawrence Erlbaum Associates: Hillsdale, NJ.
- [8] Schreckenghost, D.; D. Ryan; C. Thronesbery; P. Bonasso; and D. Poirot. Intelligent Control of Life Support Systems for Space Habitats. *AAAI Innovative Applications of AI*. Madison, WI, July 1998.
- [9] ^aSchreckenghost, D.; C. Thronesbery; P. Bonasso; D. Kortenkamp; & C.I Martin. "Intelligent Control of Life Support for Space Missions." *IEEE Intelligent Systems*, 17(5), pp. 24-31, September 2002.
- [10] ^bSchreckenghost, D., C. Martin, and C. Thronesbery. Specifying Organizational Policies and Individual Preferences for Human Software Interaction. *Proceedings of AAAI 2002 Fall Symposium on Etiquette for Human-Computer Work*. November 2002.
- [11] Schreckenghost, D., Martin, C., Bonasso, P., Kortenkamp, D., Milam, T., & Thronesbery, C. Supporting group interaction among humans and autonomous agents. *Connection Science*. 14(4) pp. 361-9.
- [12] Thronesbery, C.G., & D.L. Schreckenghost. "Human Interaction Challenges for Intelligent Environmental Control Software." *Proceedings of the 28th International Conference on Environmental Systems*. Danvers, MA, 1998.
- [13] Tufte, E.R. *Envisioning Information*, Cheshire: Graphics Press, 1990.
- [14] Woods, D.D. "Steering the Reverberations of Technology Change on Fields of Practice: Laws that Govern Cognitive Work." Plenary Address, Annual Meeting of the Cognitive Science Society, August 2002.