CLEAR: Automating Control Centers with Expert System Technology

Peter M. Hughes
Automation Technology Section / Code 522.3
NASA / Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

The Communications Link Expert Assistance Resource (CLEAR) is a fault -isolation expert system to be utilized in the operational environment of the Cosmic Background Explorer (COBE) Mission Operations Room (MOR). CLEAR will assist the COBE Flight Operations Team (FOT) during periods of real-time data acquisition by isolating faults in the spacecraft communication link with the Tracking and Data Relay Satellite (TDRS), providing advice on how to correct them, and logging the events for post-pass evaluation.

After a brief introduction to the problem domain, this paper describes the system requirements, tool selection, development approach, system operation and lessons learned during the transformation of the system from the prototype to the delivered, operational system.

Introduction

The Cosmic Background Explorer (COBE) is a scientific satellite that will carry three instruments to allow scientists to investigate the possible origins of the universe. This satellite will utilize the Tracking and Data Relay Satellite (TDRS) for four or five 20-minute real-time communication events daily. These events will primarily be used for uplinking stored commands, ranging, and monitoring of the satellite's health and safety.

The Flight Operations Analysts (FOAs) in the Payload Operations Control Center (POCC) are responsible for the health, safety, command and control of the COBE spacecraft. This includes the monitoring of the communications link between the COBE and TDRS which demands the real-time evaluation of more than 100 TDRS and spacecraft performance parameters. In order to isolate problems and to select appropriate courses of action for resolving them, this evaluation of real-time data must be correlated with a comprehensive understanding of the TDRS and COBE systems and their communications services. The task is complex, and, if not handled quickly and properly, can result in poor utilization of TDRS services, inefficient spacecraft operations and a potential hazard to the spacecraft's health and safety.

At present, extensive training and communication of actual experience are used to develop the capabilities of the COBE FOAs. Regardless, the size and complexity of the task places a large burden on the analyst.

The Communications Link Expert Assistance Resource (CLEAR) is a fault-isolation expert system to be utilized in the operational environment of the COBE Mission Operations Room (MOR) (the COBE unique portion of the POCC). This expert system was developed to provide quick problem detection and isolation in the communications link thus producing a more efficient and reliable system of operations.

CLEAR will execute on one of the seven Engineering Analysis Workstations (EAWs) used for console operations in the COBE MOR. The Applications Processor (AP), which is the computer that processes COBE telemetry for display in the MOR, will provide the CLEAR system with COBE telemetry, TDRS performance parameters and network information. Using this data, CLEAR will monitor the COBE-TDRS communications link in the search of problems and advise the analyst how to correct them.

CLEAR will be the first real-time diagnostic expert system utilized in a control center to support operations at NASA Goddard Space Flight Center. Currently, spacecraft communication links with the Tracking and Data Relay Satellite are used routinely and are planned to be utilized even more frequently and extensively by upcoming missions. This will provide a high degree of utility of the technology introduced by the CLEAR system.

System Requirements

The following are some of the functional and performance requirements specifications for CLEAR.

CLEAR is to have no effect upon the Application Processor (AP) and is to be transparent to other systems in the control center. The CLEAR system will be a strictly passive component of the system supporting COBE real-time operations. As an advisory, diagnostic expert system, CLEAR will monitor real-time data in an attempt to isolate problems, providing advice on how to correct them when they occur.

CLEAR is to be transportable within the COBE POCC. The system will run on any Engineering Analysis Workstation in

the COBE POCC without hardware modification and with the same operating system level software, e.g., communications package, graphics routines and device drivers, used by other application programs on the EAW. The workstations are AT-class personal computers running DOS and using non-standard graphics cards that support the Intelligent Systems Corporation (ISC) video format for compatibility with the POCC display systems.

CLEAR is to use the standard communication package developed for POCC workstation applications. The data furnished by the AP will be ordered ASCII text. The system will extract the TDRSS performance data, Operations Data Messages (ODM), and spacecraft status parameters from the communication buffer and convert them to the internal format required by the expert system.

CLEAR is to allow the operator to input COBE and TDRSS configuration parameter values for the subsequent TDRS support. CLEAR will utilize these parameters to identify misconfigurations of the communication systems in which case the system will notify the analyst of such discrepancies.

CLEAR is to be driven by ODM and status data sent by the AP. The expert system will monitor real-time Network Control Center (NCC) Operations Data Messages (ODMs) and TDRS and COBE performance parameters. When CLEAR isolates a problem, it will advise the operator how to correct it. The system will also monitor the input data frequency and will warn the analyst if data is not received within the expected interval (3 to 5 seconds).

CLEAR is to diagnose the faults identified during an event. The system will determine possible sources or causes of a fault, rank multiple possibilities in order of probability and present the results to the analyst. The system will also recommend the proper actions necessary to correct the problem. If requested, the system will explain why it believes that the fault exists.

CLEAR is to log all expert system activity for post event analysis. The system will time tag all identified faults and will record the inferences, the diagnoses, the recommendations offered to the operator. The system will provide non-realtime utilities to print a formatted copy of the log, to trace and analyze the activity of the expert system during the event and to extract statistics for evaluation of system performance.

CLEAR is to operate in real-time with a performance requirement derived from the expected 3 to 5 second communication buffer (input data) arrival frequency. The expert system will convert input data, check parameter values and perform inferences within this time interval. Event logging, operator dialog and explanations are not real-time events subject to the performance requirement.

Tool Selection

The real-time response required of the CLEAR system translated into a performance requirement for the expert system. The data driven and diagnostic nature of the expert system placed interface and inference logic requirements on the tool selected to build the application. Further selection criteria came from the hardware and software compatibility requirements.

A number of secondary (desirable rather than mandatory) requirements also used in the selection included cost, number of tool users, length of tool usage, stability of supplier, development environment and availability of source code. The secondary selection criteria were used to rank the expert system building tools that satisfy the mandatory requirements.

At the time of selection, several commercially available expert system building tools met the mandatory requirements based upon available information including independent benchmark tests, first-hand experience and product reviews. However, none was ranked higher than the 'C' Language Integrated Production System (CLIPS).

CLIPS, a tool for the development of expert systems, was created by the Artificial Intelligence Section of the Mission Planning and Analysis Division at NASA/Johnson Space Center. CLIPS is an inference engine and language syntax which provide the framework for the construction of rule-based production systems.

CLIPS was entirely developed in the programming language 'C' for performance and portability. The key features that attracted us to CLIPS were:

- Forward Chaining Rules
- Portability
- Satisfactory Performance
- Provision of Source Code
- Completely Integrated With 'C'
- Extensibility
- Fully Documented

The CLEAR development team is quite pleased with CLIPS. It suits the requirements of the system nicely, has been easy to integrate with the data and user interface subsystems, and has demonstrated respectable performance. The provision of the source code with the tool turned out to be quite useful. It provided the capability to modify the tool to accommodate unanticipated needs of the system thus avoiding the wasted time and effort required to switch to another expert system shell.

System Operation

To facilitate ease of use and to reduce demands on the analyst, user input is minimal. The only user input required is the pre-event initialization of parameters in the Configuration Table display (provided by the CLEAR system). Default settings are provided for each parameter and, to further expedite the process, a menu of common event configurations, called "prototype-event" codes, are provided in a menu-type format. Hence, the user only needs to select a prototype-event code and enter the event start-time in order to configure the expert system for operations.

The CLEAR user interface (figure 1) utilizes textual and graphical output in a windowed format to provide the analyst with information about the status of the communications links. The graphics window in the top left of the screen displays the current status of the communications links and elements of the communications network between the COBE spacecraft and the POCC. If the parameters indicate that a link or processing system is degrading or down, the associated icon will turn to yellow or red, respectively, thus providing the analysts with a reference of the status of the communications event in a quick glance.

When the expert system isolates a problem, a description of the problem will be displayed in the "Problems" window with the associated advice displayed in the "Advice" window. If

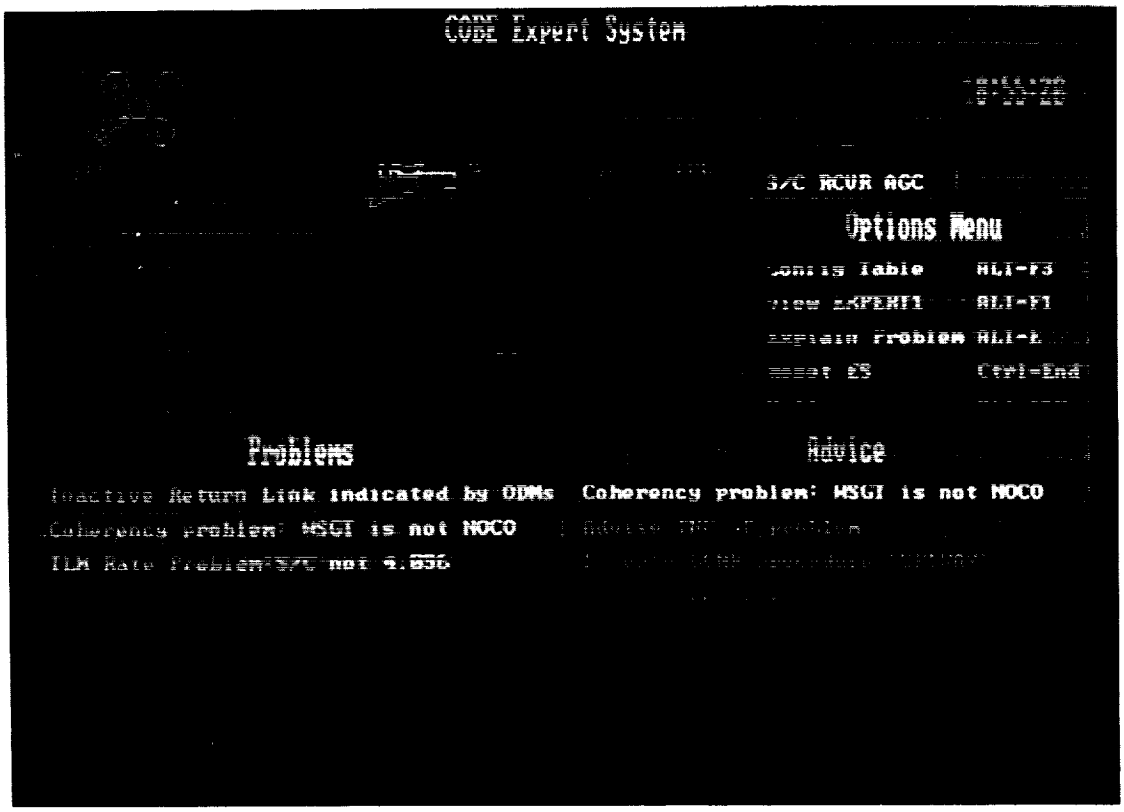

Figure 1. - Photograph of the CLEAR User-Interface

there are multiple problems, they are displayed from top to bottom in order of criticality. By default, the advice of the most critical problem is displayed in the Advice window; however, the system can display the advice to any problem that the analyst selects.

To assist the analyst and to promote system credibility, the CLEAR system provides an explanation facility. When commanded to do such, the expert system will replace the Advice window with a static explanation of why it believes that a specified problem exists. No backtracking or backward chaining is conducted since the expert system must continue to monitor the data in real-time to search for problems. Also, the analyst can view the display page of raw data from which CLEAR obtains its data. This can be used by the analyst to verify the expert system's results in the manner that he/she would have previously used.

Throughout the communications event, the CLEAR system logs data and problems isolated in an "event log". At the conclusion of the real-time support, the event reporting utility automatically converts the log into a more detailed and readable "event report". Also, a history file which contains a listing of the problems isolated in each event and a summary of the total number of occurrences of each problem for the mission to date is updated. The event report is useful for verification and debugging purposes and the history file provides information that allows the identification and analysis of problem trends which can be used to enhance both the knowledge base and operational support.

Development Approach

The CLEAR Expert System was developed in three phases: a rapid prototype phase, an operational prototype phase and an operational development phase. The rapid prototype phase was executed on a Lisp Machine using a high-powered, hybrid expert system development shell. In this phase, knowledge acquisition, requirements specification, and experimentation with knowledge representation schemes were begun. The system was demonstrated using simulated data provided by a data simulator software program that ran on a mainframe and was developed by the CLEAR team.

The operational prototype phase was conducted to further investigate the technical feasibility of the system within the functional requirements of the operational platform. The system was fully redesigned and developed using the programming language 'C' and CLIPS. The performance of the system in the run-time environment was investigated using data transmitted by the simulator at rates anticipated in operations.

The third and final phase involved the integration, testing and evaluation of the system in the COBE POCC. The system has a physical interface with the computer from which the expert system receives its data. CLEAR was tested using simulated data from the data simulator used to test all operational software developed for the POCC. Finally, the expert system was demonstrated and further tested using "live" data during spacecraft communication simulation events (used to prepare the Flight Operations Team in commanding and controlling the spacecraft). Full operational use will be realized after launch of the COBE spacecraft which, at the time of this writing, is scheduled for no earlier than mid-November, 1989.

Lessons Learned

A discussion of lessons learned, based on a retrospective analysis of the CLEAR development experience, is presented in this section. These points, and others, are more fully discussed in the paper "Integrating Expert Systems into an Operational Environment"2.

Involve the user. Involvement of the end-user in the evaluation process of computer systems is widely recognized as an important factor for attaining the functional goals of the system. This involvement seems to be more important for expert systems because, for an expert system to be successful, the users must have confidence in the problem solving capabilities of the system in order to fully accept it as a tool that will enhance their productivity, consistency, etc. This is highly unlikely if a system is brought in from "outside" and they are required to use it. When users are allowed to contribute in the development of the system, not only will the system more likely meet their needs, but they will also have greater faith in the system thus ensuring wider acceptance.

For many expert system development teams user involvement only consists of consulting users for feedback late in the development process. However, if users are included throughout the development process, many benefits will result. For instance, early involvement will emphasize their importance in the success of the project and, hence, they will actually feel like integral members of the development team. And, as integral members of the development team, not only will they strive harder for the success of the system, they will also become system advocates who will promote the system to others. This is valuable in situations where the user group is widely distributed or so large that it is unreasonable to involve all of them in the development process. Perhaps more importantly, an involved user will become a credible advocate when selling the project to management.

Finally, involvement of the user will also help control the cost of development since the user will be able to identify undesirable aspects of the system early in the development phase. As with any software project, the sooner problems are recognized and corrected, the less impact they will have on the project schedule and the less costly it will be to correct them.

Provide a robust software development environment. All too often the prototyping phase is conducted with a rich development environment while the development environment of the operational system is not nearly as supportive. The flexible development environment of the prototype is chosen to facilitate rapid coding and easy experimentation thus allowing speedy development of a system prototype. However, upon completion of the prototype, the development team mistakenly may assume that the problem domain is fully understood and the system architecture is completed. This is seldom the case. Changes to the knowledge base and system design will be necessary throughout the development of the operational phase, even after delivery. Many changes will result as the knowledge base and the user's needs evolve. Some people will only give the system full consideration during the development of the operational version when they realize that the system really is going to be used in operations.

Not only will the choice of a fully supportive software development environment enhance the development process, it will also greatly assist in the maintenance phase. Software maintenance, now considered an integral element of system development by software engineers, is often over-looked by expert system development teams. It is not uncommon for the maintenance of conventional software systems to cost up to 50% of the original development effort4. This figure is greatly increased by a lack of an accurate design specification and robust development environment (that supports easy debugging). Failure to provide a flexible software development environment for the development and maintenance of the operational version of an expert system is extremely short-sighted and often crippling to the success of the project.

Conduct a thorough prototyping phase. Success of an operational expert system is contingent upon a thorough prototyping effort. The purposes of the prototyping phase should be to:

- understand the problem domain,
- acquire the knowledge,
- identify appropriate knowledge representation schemes,
- establish rule precedence and exception handling,
- devise a suitable system architecture,
- define the requirements specification,
- analyze the business justification of the project,
- sell the project to management and the users.

Typically throughout the iterative process of prototyping, many of these critical purposes are not addressed thus severely jeopardizing the success of the project. Although this phase is often viewed as an experimental process, it should be conducted as an investigatory/ preparatory phase necessary for the proper development of an expert system intended to support operations.

The prototype must be demonstrated to management for approval and to the users for acceptance and feedback. Problems can result. It is a mistake to demonstrate a prototype with an underdeveloped user-interface (maybe consisting of textual output, and windows for the editor, debugger); it is equally wrong to demo a system that attempts to incorporate any and every function related to the problem solving process. An underdeveloped user-interface will likely appear complex or unfriendly, thus failing to secure approval for continued development. On the other hand, an overdeveloped user-interface can cause two problems. First, during development, the team will find the actual implementation of such an over zealous prototype to be more difficult than anticipated, delaying the project and wasting funds. Second, if the functionality of the prototype is cut back during the development phase, the expectations of the users will not be met thus reducing their enthusiasm and support for the system.

A problem resulting from demonstrating a handsome, well-behaved prototype is that some managers may want to deliver it "as is". Although the system seems to be complete from the outside, the developers realize that in the rush to produce a working prototype, overall software quality and long-term maintainability issues were not considered. Delivering a system in this state would not only be a software maintenance engineer's nightmare, but the resulting lack of robustness would rightfully scare away users and, even worse, convince operations managers that expert systems are too unreliable to support anything outside of the research labs.

A minor problem while prototyping is the failure to record all problems encountered, solutions used and explanations why chosen. These records help tremendously during the design of the operational system and maintenance. More importantly, these records will help ensure a smooth development of the operational system by avoiding the duplication of costly errors.

Thoroughly assess the effort required to interface the expert system to the user and the data source. A major issue usually focused upon in the prototyping phase is the acquisition, structure, and representation of the domain knowledge. Although this issue is important, it should not overshadow the necessity of conducting a complete system analysis to determine the amount of effort required to interface the expert system to the user and the data source. As with other software development projects, this figure is often grossly underestimated.

An accurate estimation of the effort required to develop the interfacing subsystems is best accomplished by a system analyst who has an understanding of the expert system, the data source, and the dynamic nature of an expert system development project. The knowledge engineer usually assumes this role; however, if he has limited experience in system analysis, the project's success can be endangered.

Test with "real" data as early as possible. Many groups develop an expert system that must interface with other systems for data without testing with "real" data until the final phases of development hence relying on the data to perfectly match its specifications as documented. This is a mistake because real world data is often incomplete and inaccurate.5 Slight variations from the specs can create potentially serious problems during integration and testing. Again, the earlier problems are found, the better off the project.

If access to real data is not feasible, develop a data simulator to facilitate testing. Although this sounds like a waste of time and money, the effort expended to develop one will not only be recouped through development, but it will also increase the quality of the system. Testing can be further augmented by the development of test suites with drivers to administer the tests automatically, logging the results into a file for debugging, maintenance, and record keeping. For large systems involving many programmers, automated test drivers will ease the laborious testing procedure that must be conducted after each modification.

Allow an independent group to test the system. If resources allow, arrange for an independent group to test the system. They will usually uncover a high proportion of the bugs due to their unbiased position and the fact that they have been challenged to discover as many bugs and weaknesses as possible.

Software testing is a critical element of quality assurance that is often executed in an unorganized and haphazard manner. The earlier bugs are isolated, the easier and less costly the removal process will be. Bugs found late in development may force major design changes and hence, increase development costs and delay system delivery. Even worse, if a user isolates a bug, his faith in the system will be diminished thus jeopardizing acceptance of the system.

Conclusion

Over the past few years, expert systems have emerged from the research labs to enhance a diverse array of operations. One of the prime areas of application of this maturing technology is in the area of real-time fault-isolation and diagnosis where the synergistic combination of the speed and tireless attention of the computer and the problem-solving knowledge of an expert (embedded in an expert system) have created powerful systems.

The CLEAR system readily demonstrates how expert systems can relieve the analysts of the tedious and demanding chore of monitoring screenfuls of data so that their time can be dedicated to solving higher level problems. While CLEAR is confined to fault isolation within the COBE-TDRS communications link, the techniques used by this system can be applied to other functions within the POCC. However, the ultimate goal would be to develop a larger expert system or network of cooperating expert systems that could perform all of the functions in the control center.

Acknowledgements

I would like to thank Walt Truszkowski, Dolly Perkins and Bob Dutilly for their assistance in preparing this paper.

References

1. Cholawsky, E.M. "Beating the Prototype Blues." in AI EXPERT San Francisco, Vol. 3, No. 12, December, 1988, pp. 42-49.

2. Hughes, P.M., "Integrating Expert Systems into an Operational Environment." in Proc. for Computers in Aerospace VII Conference, Monterey, CA. 1989.

3. Hughes, P.M., and Hull, L.G. "CLEAR: Communications Link Expert Assistance Resource" in Proc. 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics, Greenbelt, MD: 1987.

4. Pressman, R.S. Software Engineering- A Practitioner's Approach. McGraw-Hill, Inc., New York, NY: 1987.

5. Smith, D.L. "Implementing Real World Expert Systems." in AI EXPERT San Francisco, Vol. 3, No. 12, December, 1988, pp. 36-41.