The Load Shedding Advisor: An Example of a Crisis-Response Expert System

Terry B. Bollinger
Software Productivity Consortium
1880 Campus Commons Drive, North
Reston, Virginia 22091

Eric Lightner / John Laverty
Bendix Field Engineering Corporation
10210 Greenbelt Road, Suite 200
Greenbelt, Maryland 20706

Edward Ambrose
Goddard Space Flight Center, Code 534.1
Greenbelt, Maryland 20770

Keywords: *Expert systems, Prolog, load shedding, entity-relationship model, relational databases, human factors, hybrid methods.*

Abstract: *The paper describes a Prolog-based prototype expert system that was implemented by the Network Operations Branch of the NASA Goddard Space Flight Center. The purpose of the prototype was to test whether a small, inexpensive computer system could be used to host a load shedding "advisor," a system which would monitor major physical environment parameters in a computer facility, then recommend appropriate operator responses whenever a serious condition was detected. The resulting prototype performed significantly better than was originally anticipated, due primarily to efficiency gains achieved by replacing a purely rule-based design methodology with a hybrid approach that combined procedural, entity-relationship, and rule-based methods.*

1. Introduction

Successfully operating a large computer facility is a task that encompasses far more than simply knowing how to run computers. A computer facility is complex, integrated combination of physical, environmental, and computational systems that must work in unison to achieve the overall purpose of the facility; for example, the failure of a small valve that supplies chilled water to an air conditioning unit can cause a computer system to grind to a halt just as surely as the loss of a critical system file. When the interlinked support systems of a facility work smoothly, it is quite easy to forget about the safety net that they provide; however, when one of those support systems suddenly fails or is seriously damaged, a good understanding of its relationship to data processing and communications equipment can suddenly become critical.

After a support system failure has been recognized, the decisions made during the (often short) span of time available for responding to the problem can make the difference in whether critical processing must be abandoned, and in certain cases may determine whether facility equipment is physically damaged. The problem of how to respond to a support system failures is aggravated when a facility is part of a larger real-time communications network, since a loss of key functions in such a facility can have a direct impact on sites throughout the network.

At the NASA Goddard Space Flight Center, the problem of how to minimize the impact of support system outages is a very real operational issue. Goddard is the home of the Network Control Center, or NCC, which is the central control facility for the Tracking and Data Relay Satellite System (TDRSS) Network. The TDRSS Network combines a ground based communications network with a geosyncronous relay satellite, and it is used to provide communications support to a variety of satellites and spacecraft, including the Space Shuttle. Due to the central role of the NCC in the TDRSS Network, the failure of one of its support systems can have an impact that goes far beyond the NCC, affecting communication nodes and TDRSS customers at various remote sites. Such outages can in certain instances result in the loss of irreplaceable scientific data; in the case of the Space Shuttle, such an outage could make it necessary to fall back to the Ground Network, an older ground-based communications network with less coverage.

2. The Load Shedding Study

In August 1985, the Network Operations Branch (Code 534.1) of the Goddard Space Flight Center began a study to determine whether expert system methods could be used to assist NCC operators in responding to failures in NCC support systems. The specific area selected for investigation was load shedding, which is defined for the NCC facility as the selective reconfiguration and shutdown of equipment during power, temperature, or humidity crises. Specific goals of the investigation were:

a) *To determine the applicability of expert system methods to the load shedding problem.* Due primarily to the need for real-time responses, expert system methods were not automatically assumed to be the best approach to the load shedding problem. Although a few examples of real-time expert systems (such as Navex[1]) were known when the study began, most examples of expert systems were for stand-alone use only.

b) *To provide a mechanism for formally capturing load shedding expertise.* Even if the load shedding application did not prove to be a good candidate for expert system methods, it was felt that the formal capture of operator expertise would be a valuable result in itself, since it could be used to formulate better manual procedures for load shedding.

c) *To determine processing and storage needs of an operational expert system.* By developing a prototype of the expert system, it was hoped that firmer estimates could be made of the processing and storage resources needed for an operational load shedding expert system.

A key component of the Code 534.1 strategy was to build an actual small-scale prototype of a Load Shedding Advisor, using an off-the-shelf expert system shell on a PC-class computer system. Although it was recognized that a PC-class system was unlikely to be powerful enough to host a fully functional real-time load shedding system, the PC approach had the important advantage of providing a convenient and readily available host system for developing the major features of the knowledge base.

3. Results of Prototype Development

The load shedding study was completed in October 1986, and the results were encouraging. In particular, the final version of the prototype performed significantly better than anticipated, leaving open the possiblity that a PC-class computer could be used to host an operational load shedding expert system. Gains in both efficiency and maintainability were achieved through the use of a "hybrid" design approach that was developed as the prototyping effort progressed. This hybrid methodology, which is described in more detail later in this paper, replaced a purely rule-based design with a combination of procedural, entity-relationship, and rule-based methods.

Three distinct prototypes were constructed for the load shedding study. Each of these prototypes used a different shell or shell version, and each one concentrated on a different aspect of the load shedding problem. All of the prototypes were implemented on an IBM PC XT with 640K bytes of memory, a 30 megabyte hard disk, two half-height floppy disk drives, an EGA graphics board, a mouse, and an MS-DOS operating system. The number of people working on the project varied from one to three, with one person always assigned full time. The prototyping activity lasted fourteen months.

The first two prototypes were aimed at determining the actual rule sets for an NCC load shedding system, and were directly based on expertise gained by interviews with NCC facilities and operations personnel. The third prototype was used to investigate structural and human factors issues, and was designed for use as a demonstration system. To avoid having sensitive data in the demonstration prototype, its knowledge base was constructed around a hypothetical facility that demonstrated features found in most large computer facilities, rather than being based on the NCC. Descriptions of the three Load Shedding Advisor (LSA) prototypes are given below.

4. LSA-1: Interactive Diagnosis of Load Shedding Problems

Design of the LSA-1 Prototype. The first prototype, LSA-1, was a classic Mycin-style expert system that used Newell/Simon production rules[2] to represent load shedding expertise. The rules were implemented with Version 1.3 of Teknowledge's "M.1" expert system shell, a rule based, backwards-chaining shell that is similar in syntax to the AI language Prolog[3]. Like most

Mycin-style expert systems, the operator interface for LSA-1 was "conversational;" the expert system acquired facts by engaging in a selective question-and-answer session with an operator.

The LSA-1 Knowledge Base. A substantial body of rules was collected and formalized for the LSA-1 effort, but for reasons described below, only a small subset (about 30) of these rules were actually coded into the knowledge base. The implemented subset covered diagnosis and response to power distribution problems for a hypothetical facility that included several load centers (power conversion transformers), two commercial power feeders, and two backup power feeders.

Evaluation of Results. Within the limits of the data available to it, the LSA-1 system performed reasonably well. Response times were acceptable, ranging from a few tenths of a second to one or two seconds, depending on how much text was displayed and how much inferencing was required. Certain aspects of the M.1 Version 1.3 user interface, such as the requirement that all entries be terminated with a period, were cumbersome, but the system converged rapidly to conclusions and needed relatively few entries from the operator.

Although the production rule model of LSA-1 provided a good mechanism for collecting and formalizing load shedding expertise, it did not adequately satisfy the primary goal of determining whether a real-time, on-line load shedding advisor was possible. Factors which made LSA-1 inadequate for accurately assessing the load shedding expert system problem were:

a) *Incomplete coverage of the problem domain.* Since the LSA-1 system was built on the assumption that all decision data would be obtained from an human operator, its coverage was necessarily limited to faults whose effects were visible to the operators. Such an approach suffers from the dual problems of poor fault coverage (since only a small subset of the potential range of fault indicators would be used) and poor resolution (since there was not always enough data to distinguish between distinct faults).

b) *Reliance on operators for time critical status data.* Another problem with LSA-1 dependence on the operator interface for data was the time critical nature of many load shedding problems. In a load shedding emergency, the operator may need to respond in less than a minute; in such a case, it is very unlikely that he will want to spend that time getting the expert system "up to date" on what has happened.

The Domain Status Acquisition Problem. The most significant problem with the LSA-1 prototype was its reliance on a question-and-answer dialog to obtain the data it needed to make load shedding recommendations. While this type of dialogue is adequate for situations where the problem is stable over the time of the dialogue (most patients don't die while Mycin is asking questions), it can be highly inappropriate in a crisis situation where a fraction of a minute may make the difference between success and failure. The problem is aggravated by the fact that good load shedding recommedations require more than a simple identification of the problem; unless the advice can be "customized" to the current status of a facility, the system can only give generic advice on how to deal with a problem.

In a crisis situation, generic advice is a poor second to a list of specific instructions. The difference between the two may be seen in the example of telling an operator to "shut down non-critical disk drives," as opposed to giving him a list of exactly which drives should be shut down. Such customization is dependent primarily on the availability of good status data, rather than on the inferencing power of an expert system. Without such customization, a load shedding expert system would in effect become an on-line documentation system, where operators would "look up" standard procedures by entering a short list of index conditions. While such a system could be valuable as an automated replacement to paper-based operating manuals, it would fall substantially short of the full potential of a load shedding expert system.

In the sections below, the set of data that describes the current status of an expert systems's problem domain (in this case a computer facility) is referred as *domain status data*. The problem of how to acquire such data effectively is referred to as the *domain status acquisition problem.*

5. LSA-2: Simulated Domain Status Monitoring

Purpose of the LSA-2 Prototype. Since LSA-1 did not adequately address some of the broader issues of how to implement a load shedding expert system, the LSA-1 system was abandoned in favor of a new approach in which the domain status acquisition problem would be explicitly addressed. To test the effectiveness of integrating domain status data into the expert system, a new prototype, the LSA-2 system, was implemented.

Design of the LSA-2 Prototype. LSA-2 was implemented in M.1 Version 2.0, which provided a number of significant enhancements over Version 1.3. Version 2.0 was written in C rather than Prolog, and as a result it was about five times faster. Screen display commands, which were one of the weakest points of Version 1.3, were also significantly improved, although they still fell short of the capabilities provided by many conventional PC-based programming languages.

The major thrust of the LSA-2 effort was to simulate entry of domain status data through the the use of simple menu-style operator query screens. The operator could selectively modify the recorded status, which would then be recorded as facts in the knowledge base. Upon completion of such entries, the system would evaluate the status values for potential problems, and would automatically prompt the operator if any were found; thus, unlike LSA-1, the new system was able to initiate its own diagnostic sequences without waiting for an explicit operator query.

Like LSA-1, the LSA-2 system was implemented using only production rules. Features such as the menu displays could also have been accomplished by writing a C program that interfaced with the expert system, but it was decided to instead try using the enhanced I/O features of M.1 Version 2.0 to create a more-or-less conventional menu interface.

The LSA-2 Knowledge Base. The decision to implement conventional menu-style interfaces with production rules instead of external code turned out to be rather a disaster, particularly from the perspectives of clarity and maintainability. Since M.1 Version 2.0 permitted a maximum of 16 variables in any one rule, displays that showed more than 16 variables at once (of which there were several) had to be "coded" by using remarkably opaque trees of display rules. The ability to iteratively modify a parameter, a must for those of us who don't always get it right the first time, could only be implemented by using M.1 metacommands to selectively reset (clear) facts from the knowledge base, an approach which again led to opaque "code."

By adding the requirement that the knowledge base handle more complex data about the status of the facility, it also became necessary for LSA-2 to perform calculations on moderately large sets of data; for example, the total electical load of a facility could only be obtained by adding the individual loads of all active equipment items. Although M.1 was capable of performing arithmetic calculations fairly quickly, the design used in LSA-2 used inferencing as its data access mechanism, an approach which led to very slow evaluations. For example, one simple summation of a few dozen real values took over 60 seconds to perform, a figure that clearly leaves room for optimization.

Evaluation of Results. Unlike the LSA-1 system, which performed reasonably well withing its defined limits, the LSA-2 experiment was obviously no where near its optimal level of performance or structure. Its main value was conceptual; by providing a first-draft attempt to organize and use explicit domain status data in a knowledge base, LSA-2 suggested new ways for organizing such data in a more coherent fashion. After completion of a slow (but functional) LSA-2 system, the prototyping effort shifted its focus to a new tool and a new representation of domain status data.

6. LSA-3: Entity-Relationship Problem Modeling

Switching Over to Turbo Prolog. As described above, the LSA-2 prototype had run into difficulities in its use of rule-based methods for menu displays and numeric calculations. The best solution to this problem appeared to be convert I/O and numeric functions to C, and to use the M.1 shell only for inferential problem solving. Another possibility was translation of the M.1 knowledge base into Prolog, a language which shares many features with M.1, but which contains a fuller range of low-level I/O and numeric functions. Unforunately, most of the PC-based Prolog systems that were available at that time were slow and rather limited, and they often lacked the standard features found in large-machine versions of Prolog.

The situation changed when Borland released Turbo Prolog Version 1.0. Turbo Prolog is actually a subset of full Prolog, since it omits an important feature known as metaprogramming, but the product has a powerful set of high-level and low-level I/O routines, and it is very fast. After a short period of testing, a decision was made to translate the major features of the LSA-2 prototype into Turbo Prolog.

Penalties involved in switching from M.1 to Turbo Prolog included the loss of a rich set of M.1 commands and metacommands, a switch to a less English-like syntax, and loss of the built-in conversational interpreter. These losses were offset by that fact that the LSA-3 design would be built almost entirely around menu-based interfaces, and would rarely need a conversational interface.

Design of the LSA-3 Prototype. As a result of evaluations of the LSA-2 prototype, it was decided that the LSA-3 prototype should explicitly partition the load shedding problem into three components, each of which would use a different conceptual model. The components of this hybrid approach were:

a) *Procedural Programming.* The experiences gained in the LSA-2 model indicated that for many of the current expert system shells, conventional coding may be the best way to implement support functions such as I/O and mathematical calculations. The general rule for deciding whether to use procedural code is that if a function attempts to use inferencing to perform simple, non-heuristic accesses to data, I/O devices, or other rules, then it probably should be implemented with some form of procedural programming. Inferencing is a powerful look-up mechanism, but it is also very expensive in its use of computer time; careless use of inferencing in an expert system can very quickly lead to serious performance problems.

   In Turbo Prolog, "conventional" programming was simulated by selectively using the cut operator to constrain predicates into non-backtracking behavior. Predicates in this form could be used as close analogs of conventional subroutines and functions, and various Turbo Prolog compiler optimizations provided excellent speed and memory performance for constrained predicates.

b) *Entity-Relationship Modeling.* Entity-relationship (E-R) modeling[4,5] is an idea that has become popular in recent years in the database management community. The E-R model of a problem is actually a form of the well-known relational database model[6], differing only in that "relationships" between tables of information ("entities") are explicitly named and defined, rather than being implicit as they are in the relational model.

   The importance of E-R models to expert systems is that they can be used to separate problem modeling from problem expertise. In the case of the load shedding problem, E-R models can be used to represent sets of equipment and their properties, while rules for handling support system faults can be be generalized to address entire classes of equipment, rather than individual items. Since most expert system shells are built around relational databases, simple forms of the E-R model can be directly implemented in products such as M.1. Prolog, with its inherently relational structure, is a particularly good language for implementing E-R problem models.

c) *Rule-Based Knowledge.* In the hybrid approach, production rules are reserved for their classical application of modeling human expertise about specific, well-defined problem domains. However, unlike most rule-based expert systems, the rules of a hybrid expert system should make their assertions primarily in terms of an underlying entity-relationship model of the problem domain, rather than in terms of the external world of the expert system user. Changes to status data in an E-R database can be hidden from the expertise rule set, so that these rules can view the E-R database as if it were a direct mapping from the real-world problem domain. The major advantage of having rules address the E-R model instead of the external world is generality; rules stated in terms of formally defined E-R facts will tend to be more powerful than rules that refer directly to less formalized structure of the real world.

The LSA-3 Knowledge Base. Using the hybrid design paradigm described above, the LSA-3 knowledge base was divided into two parts: an E-R component that describes the hypothetical facility with relational tables, and a set of production rules for describing expertise in solving load shedding problems. Conventional procedures, which were implemented by using constrained Prolog predicates, were used to transparently update the E-R database to the current (simulated) status of the facility.

The LSA-3 User Interface. The LSA-3 user interface strongly emphasized human factors related to the load shedding problem. In particular, the number of operator keystrokes needed to respond to alarms was kept to a minimum, data entry mechanisms were arranged to make invalid entries difficult or impossible, color coding was used to help operators identify out-of-tolerance values at a glance, selective keyboard lockouts were used to prevent invalid data entries, and all screen displays included information on "what to do next." For normal conditions the system the system is passive, requiring no interactions from operators. For serious emergencies, the system goes into an "imperative" mode in which visual and audible alarms are activated. The alarms will remain on until an operator either follows the recommendations of the system, or explicitly overrides the alarms with an explanation of their cause.

Using "instantiation" of general conclusions against the E-R model of active equipment, the prototype is able to convert a general conclusion into a specific list of specific, item-by-item recommendations that are then presented to the operator as a series of imperative menus. These menus are presented in priority order, with the most important sets of equipment given first. Equipment lists are grouped by system to help the menus correspond more closely to the physical components of the facility. If multiple problems are detected, the system responses to those problems are queued in a prioritized order. In such a case, the system will continue to prompt the operator with new alarms until all known problems have been resolved.

Although the primary mode of interaction between the LSA-3 system and an operator is through menus, a special rule interpreter was written in Turbo Prolog for use in cases where reliable domain status data is unavailable. In this situation, it is planned that the load shedding system will fall back to a mode that is similar to to the LSA-1 prototype, in which observable symptoms of faults are used by an operator to access general advice as to what actions to take. The interpreter provides a conversational interface, and is designed to be embedded in a Prolog program, rather than to act as a stand-alone shell. It is backward-chaining, and implements a slightly modified form of Bayesian certainty factors[7] to handle reasoning with uncertain facts.

Finally, the LSA-3 prototype includes a set of routines for creating colored bar gauges, which are used to display temperature and humidity data. The routines are designed for general-purpose use, and can be embedded in any Turbo Prolog program.

LSA-3 Performance. The performance of the LSA-3 prototype was exceptionally good, especially when compared to earlier LSA-2 design that attempted to provide similar features. Evaluations of the E-R status database were completed at a rate of four times a second, and an 80386-based version of the system is expected to run roughly 20 evaluations each second. Both of these speeds are far in excess of anticipated needs. User interfaces were all very prompt in responding, and "smart" display features used in the menus allowed users to see the implications of data entries almost instantly. The only PC resource that was appreciably stessed by the application was memory; in some instances, approximately half of the available 640K bytes of memory was used by the application.

Overall, the performance of the LSA-3 prototype was sufficiently good that the idea of using an 80386-based PC computer to host an on-line Load Shedding Advisor became a real possibility, one which will be investigated in future load shedding work.

7. Advantages of Using Entity-Relationship Models in Expert Systems

The explicit use of an E-R model in the LSA-3 prototype led improvements in both the clarity and performance of the knowledge base, and is a concept that would appear to have considerable generality. Advantages of using E-R models in the construction of knowledge bases include:

a) *Increased Structural Clarity of the Knowledge Base.* In many expert systems, modeling of the problem domain (such as a computer facility or an automobile engine) is intimately mixed with heuristic expertise about the domain (such as which symptoms indicate which classes of faults). The E-R model allows a clear separation of such knowledge, and a corresponding increase in clarity.

b) *Increased Maintainability of the Knowledge Base.* The E-R model of a problem domain will record information that is much more readily accessible than problem solving expertise, and it will record it in the form of easy-to-maintain tables. For example, in a well-structured E-R model, the removal of an equipment item from a computer facility would be shown by simply removing the item from its entity table; the load shedding expertise of the rule base would not need to be changed at all.

c) *Avoidance of Redundant Rule Instantiations.* A common problem in rule-based expert systems is the inadvertant specification of rules which simply repeat a general principle for two or more similar objects. An E-R model helps substantially in eliminating such redundancies by providing a single, coherent terminology for describing classes of similar problem domain objects.

d) *Easier Identification of Generic Rules.* By providing a formal terminology for describing the problem domain, the E-R model can also help an expert notice broader relationships when defining rules.

e) *Increased Clarity and Conciseness of Rule Definitions.* The availability of a formal E-R description of the problem domain also helps make individual rules more precise and less prone to inadvertant use of synonyms.

f) *Faster KB Evaluations Through Specification of E-R Access Mechanisms.* By defining special forms of relationships which order or otherwise modify the appearance of entities at the rule level, explicit search mechanisms can be imposed on rules, without affecting the way in which rules are stated.

g) *Simplification of Interfaces to Conventional Software.* As described for the LSA-3 model, the E-R database can be updated independently by conventional software, allowing the rule base have rapid access to external data without having to acquire it via inferencing.

h) *Support of Constrained Learning Via E-R Modification.* An valuable property of a well-structured E-R database is that it is very easy to update, provided that the changes to it do not violate the particular E-R model used. This feature could readily be used to create knowledge bases that actively learn about changes in the problem domain, a feature that could be very useful for an application such as the Load Shedding Advisor, where the problem domain (a computer facility) may have frequent changes to items such as equipment lists. A limited learning feature would greatly extend the usability of the expert system in such an environment.

j) *Enhanced Support of "Deep" Reasoning.* Finally, the formal problem domain descriptions of E-R models could be used as "axiom sets" for some forms of self-referential reasoning. The analogy is that the expert system can reason more deeply because it can "see" a formal model of the problem domain, rather than having to relay on implicit problem domain knowledge that is scattered all through the knowledge base.

8. Future Directions

Work on the Load Shedding Advisor has continued into a new phase in which mechanisms for linking a load shedding advisor into NCC status systems are being investigated. With the successful of the LSA-3 prototype in demonstrating that a load shedding system is feasible on even a small computer system, the emphasis has now shifted to the significant problem how to acquire timely, up-to-date data for refreshing the E-R status database.

An equally interesting result of the load shedding effort has been the development of the hyprid approach to developing new expert systems. In particular, the idea of explicitly partitioning knowledge bases into an E-R problem domain model and independent expertise about that model seems to hold considerable promise as a way to extend expert systems into a wider range of real-time applications.

[1] Maletz, M.C. 1984. *NAVEX System Architecture and User's Manual.* Los Angeles: Inference Corp.

[2] Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of Artificial Intelligence, Volume I.* Los Altos, CA: William Kaufmann, Inc.

[3] Clocksin, W.F., and Mellish, C.S. 1984. *Programming in Prolog.* New York: Springer-Verlag.

[4] Foard, R.M. 1985. A Data Manager Using Entity-Relationships. *IBM Tech Journal 3, 10* (Oct 1985).

[5] Chen, P.P. 1976. The Entity-Relationship Model -- Toward a Unified View of Data. *TODS 1, 1* (March 1976), 9-36.

[6] Codd, E.F. 1970. A Relational Model of Data For Large Shared Data Banks. *CACM 13, 6* (June 1970), 377-387.

[7] Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of Artificial Intelligence, Volume II.* Los Altos, CA: William Kaufmann, Inc.