Title: SNAP - System Networked Advisor Program - A Diagnostics Advisor for System-Level Troubleshooting

Course:
Year: 1990
Author(s): A. Doumani, R. Martin, R. Ritter and K. Spurgin

Report No: P90007

Abstract:	System Networked Advisor Program, SNAP, is a system level troubleshooting Expert System, currently under development. We propose to develop a prototype portion of SNAP to be used in field services on troubleshooting the disk subsystem. This subsystem provides a reasonable "stepping stone" for introducing expert system technology into production. The knowledge base will contain structural and functional representations of the computer configuration. SNAP provides the troubleshooter to replace the appropriate faculty past, or suggests alternate configurations to further narrow the number of fault candidates.

SNAP - SYSTEM NETWORKED ADVISOR PROGRAM - A DIAGNOSIS ADVISOR FOR SYSTEM - LEVEL TROUBLESHOOTING

A. Doumani, B. Martin, R. Ritter,
and K. Spurgin

EMP - P9007

EMGT 510K
Freiling and Niwa
Spring 1990

Project Team:
Alex Doumani
Bob Martin
Russ Ritter
Kerry Spurgin

SNAP

System Networked Advisor Program -

A diagnostics advisor for system-level troubleshooting

ABSTRACT

System Networked Advisor Program, SNAP, is a system level troubleshooting Expert System, currently under development at Sequent Computer Systems, Inc. This paper proposes to develop a prototype portion of SNAP to advise field service on troubleshooting the disk subsystem.

Disk subsystem troubleshooting is selected due to its moderate complexity, and varied
levels of troubleshooting expertise. This subsystem provides a reasonable 'stepping stone' for introducing expert system technology into production at Sequent. Estimated break even on investment is within a year of introduction. The expected cost savings exceed maintenance costs. Intangible benefits are also considered.

Similar cases are reviewed to help reduce project risk and to provide specification suggestions. A two stage inference engine is proposed for the reasoning process. The Knowledge Base, KB, will contain structural and functional representations of the computer configuration. Matrices and rules guide the reasoning process to determine the fault candidates. SNAP advises the troubleshooter to replace the appropriate faulty part, or suggests alternate configurations to further narrow the number of fault candidates.

Disk troubleshooting: Why prototype the KB system using this problem?

Disks are among the highest failing subsystems, making the problem relevant to troubleshooters and having potential for fast pay back on development costs. Experts accomplish disk troubleshooting today at various efficiencies. Users welcome and support infrastructure and tool developments. The scope of the problem domain is narrow without being a trivial or "toy" issue. More importantly, the tools needed for a disk failure expert system would create a strong foundation for extension into other failure categories.

The task is tolerant of imperfection. Advising actions to take to verify failure root cause lends itself to "good enough" solution. Success is measurable, such as decrease in costs per service incident, or percentage of returned "no trouble found" parts.

Business growth rate drives productivity and training needs. Field populations have doubled in less than two years, with several groups such as hotline experts and systems technicians having to nearly match growth rate due to lack of productivity tools. A successful SNAP will allow existing experts to handle higher product populations, and enhance training of those new troubleshooters hired.

The KB ontological approach to defining knowledge structure has potential to pay back through analysis alone. The structured approach of reviewing cryptic test and diagnostics messages is expected to enable improvements in the tests and messages themselves, even if the KB system were to remain unimplemented.

COST/BENEFIT JUSTIFICATION

Costs to Conduct Feasibility, Implement, and Maintain SNAP

Available resources (sunk costs):
Sequent has numerous data bases and resources which are being or could be applied to this problem. Troubleshooting technicians have access on the Sequent host network to quality and configuration data stored in relational (Oracle) data bases. Many have X-Window graphics terminals, DOS or SUN work stations, and those in the field have modem access with terminals or portable PCs. The variety of expert system shells available for UNIX and ported to Sequent is growing, including the KES shell which has an Oracle interface.

Feasibility:
This stage consists of tool specification, including cross-training KB experts and domain experts, narrowing focusing on problem domain, and prototype a problem solution with enough hardware and software to demonstrate feasibility. Consultants should be leveraged as necessary to achieve early prototype win and to support development of internal specialists attempting to build their first expert system.

10k	KB and data spec review
20k	Problem analysis
30k	Prototype
$60k	Total, Feasibility stage

SNAP initial production system:
This stage expands SNAP into field service and production use with added hardware and tool purchases, KB integration with production systems, and application and robustness upgrade such as enlarging domain scope to other key problems and to develop user training and on-line help documentation.

20k	Hardware/tools purchases
50k	KB production integration
60k	Application development
$130k	Total, SNAP introduction

Expansion and maintenance:
Software tools in the expert system are typically sixty to ninety percent conventional code [13]. Lack of ongoing maintenance and development to capture and code user requests are sited as primary reasons for degradation and eventual abandonment of KB systems. [17] Conventional test software is estimated to take about 50-60% of initial cost to enhance and maintain with growth and changes in product mix and user requirements. These were our basis for estimating annual maintenance costs. Beyond a "critical mass" full-time specialist [17], other funds to expand system could be allocated with discrete justification.

$100k	Annual expansion and maintenance

Benefits

Assumptions used to evaluate potential SNAP benefits are summarized below. Pay back analysis is based on production release of only the KB (disk advisor) prototype.

Field disk population	6000 disk drives	
Disk MTBF	60,000 hours	
Expected disk warrantee/service incidents	750/year	
Current cost per incident	$1000	
Annualized field troubleshooting costs		$750k
In-house system test failures/week	3	
Troubleshooting labor cost/incident	$200	
System cycle time loss/incident	1 day	
Annualized inventory costs (i=20%)	$32k	
Annualized in-house troubleshooting costs		$62k
Total annual disk troubleshooting costs		$812k

Break Even/ROI Analysis

Troubleshooting savings when reported in the KB literature were typically breakthrough rather than minor in nature, with efficiency improvements of 2, 5 and 10 times [19] [12] [11]. More often the strategic wins of the expert system are described, how it enabled functions or quality levels previously unobtainable. Table 1 below provides break even estimates given the disk troubleshooting costs above and a set of "reasonable" improvement factors. An example of a troubleshooting efficiency improvement of 10% (factor = 1.1x) would pay back with a break even of 19 months. The table deals with pay back on prototype and introduction costs of about $180k, and does not take into account either the benefits or costs of expanding the KB system into other troubleshooting areas.

Annualized value, $812k failure cost basis (1990 volume)

Improvement factor	1.1x	1.2x	2x
Annualized savings	81.2k	162k	406k
Estimated break even (for the disk expert)	19 mo	12 mo	5 mo

Table 1: Break Even Analysis

Other Potential Benefits and Opportunities:

Other benefits have been suggested in both the literature and in the problem investigation. Some involve tangible costs which have not been calculated, while others are intangibles such as customer goodwill and employee development and satisfaction. A few of these potential benefits are listed below:

- Reduce customer down time.
- Improve troubleshooting and training tools for OEMs and novice users.
- Improve product design by improving both internal and remote diagnostics.
- Reduce shotgunning and "no trouble found" returned components.
- Reduce system test cycle time (customer lead time) and variability.
- Improve process for building/maintaining system configuration files.
- Enhance technician skills at all levels

SIMILAR CASES

Reviewing reference cases was done to reduce risk of SNAP project failure. The strategy was to categorize or "type" the SNAP project, and then do a select literature search for similar KB implementations. General project management guidelines for introducing KB systems were found in Niwa[17], Liu[13], and Freiling [9].

In the project management category, Liu[13] summarizes key ingredients for KB success as training, end-user involvement, and management commitment. His editing of the CASA/SME round-table discussion covers selection criteria, key project members, and justification, all areas covered in greater depth by Niwa[17] and Freiling[9]. These issues are abbreviated into checklists in Appendix C. Liu[13] also describes "the makings of a production-quality system". Distinctions between prototype and production quality systems are well summarized, and offer guidance to the SNAP team about managing KB reliability and interfacing to existing systems.

Knowledge Based qand expert systems have leveraged the fields of knowledge engineering and cognitive science. KB applications are an attempt to apply artificial intelligence and cognitive psychology to a great number of tasks. These expert systems perform tasks which can be described to two major groups: Analytic and Synthetic. Each of these groups encompass system characteristics[9]. These are:

Analytic:	interpret, diagnose, monitor, predict
Synthetic:	repair, control, plan, design

SNAP has characteristics of both the analytic and synthetic expert systems. The key objectives of SNAP are to **interpret** error messages, and **diagnose** the faulty disk or disk control unit given error messages and configuration input. These are its Analytic functions. The second function of SNAP is to suggest **repair** actions to either correct or further troubleshoot the problem. These specific functions categorize SNAP as a maintenance expert system.

Maintenance applications have limited publication in relation to theoretical AI and Expert System research. Between 5% and 10% of the hundreds of KB articles listed in the Applied Science and Technology Index over the past three years are maintenance applications, with a small portion of that involving troubleshooting (remainder are primarily logistics related). Table 2 summarizes the quantity and categories of articles in the Applied Science and Technology Index.

Subject	Year		
	'87	'88	'89
Artificial Intelligence	76	47	31
Expert Systems	128	72	91
Maintenance Expert Systems		22	12
Knowledge Representation	22	23	28
Total articles:	226	164	162

Table 2: Articles published in Applied Science and Technology Index

One source, a presentation from IEEE [11], summarizes pros and cons of today's expert systems. Table three reflects a few of these generalizations.

PRO	CON
Wide variety of applications	Broad knowledge systems are few
Increase problem solving capability by ~10x	Many systems fall into disuse
Common pay back ~1,000,000	Difficult and tedious to program
Excellent ROI on small systems	Long question and answer sessions
	High maintenance factor

Table 3. The PRO and CON of Expert Systems

The case most similar to the SNAP proposal was found to be a system implemented by Lockheed and described by Laffey, et Al [12]. The Lockheed Expert System, LES, is a framework for building expert systems, similar to Emycin. LES was applied to a large signaling network (3000 printed circuit boards, 1000 cables), comparable to and larger than the multi-component configurations of the Sequent computer systems.

Lockheed applied many of the published general techniques [9][17] for developing successful expert systems which suggest winning strategies for SNAP. These were:

- The use of existing tools to manage costs and allow quick development
- Graphics including system layout and schematics in the user interface,
- The use of second level (hidden until called for) explanations for its reasoning and answering of questions.
- Limiting the number of tedious question-and-answer sessions

One aspect of LES appearing especially applicable to SNAP is the use of "... a structural description of the device in its trouble shooting, as well as allowing the knowledge engineer to explicitly control the reasoning process through WHEN rules." [12][5][6]. These are similar to the reasoning processes proposed below.

The results of LES are very encouraging. LES was reported to have:
- reduced time spent on troubleshooting by a factor of 5,
- correctly diagnosed 41 of 43 real failures seen within the first year.

LES is an example of an expert system successfully implemented. Further information on LES could be useful in developing the fault diagnosis tool for Sequent computers.

KB MODELING

This section suggests a model for implementing SNAP's inference engine. The knowledge for this SNAP prototype was derived directly from the operating principles of the disk subsystem rather than from a domain expert. The objective is to meet the needs of the specific problem, while creating an expandable inference capability for other subsystem domains.

A complete description of the disk subsystem is beyond the scope of this paper, and assumptions to simplify the model described below may need adjustment upon implementation. Appendix A shows system and disk subsystem reference illustrations. The following brief subsystem description is intended to provide a basis for understanding the matrix inference approach proposed.

Disk Subsystem Description

The disk subsystem for Sequent computers is organized around a disk controller card called the Dual Channel Disk Controller. A system can have up to seven DCC's. Each DCC supports up to 8 disk drives. The dual channel design allows independent data transfer on two channels. The data is routed through up to two multiplexers, each serving up to four drives. The first multiplexor (referred to as "MUX 0" by the device programmers) is connected to the DCC by a six meter cable. The second multiplexor (MUX 1) is connected to the first by a 3-inch cable.

The dual channel architecture allows the system to optimize throughput. The disks on each channel (up to four) share common data paths to the DCC. Figure N is an example of a fully configured DCC.

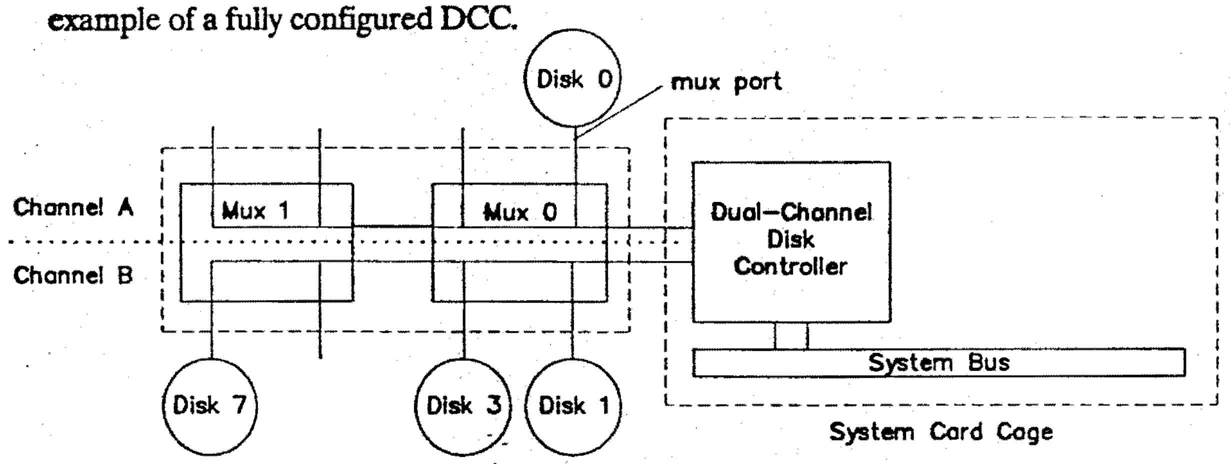

Figure 1: Fully-Configured Disk Subsystem

Program access to a disk is done through a device driver. The driver co-ordinates requests from various processors, drives the DCC and handles completion codes and error indications. Error handling usually returns an error code to the calling routine and may send an error message to the operator console. Error messages may have the following format:

zd%d%c: Error (specific error message); cmd 0x%x at (%d, %d, %d)

where "zd%d%c" defines the controller (zd), the driver (%d), and the partition (%c). cmd 0x%x is the DCC command active when the error occurred, and (%d, %d, %d) is the drive cylinder, track and sector where the error occurred.

Assumptions

- Disk Access: The user can and will access each and all disks in a troubleshooting session. This could mean simply reading a directory from each disk.

- Exclude cables in the basic model: An analysis of the types of debug strategies used showed that few error indications point directly to a cable. Our approach is to consider the cables connecting the drive to the multiplexor to be part of the drive. The 3-inch cable connecting the second multiplexor to the first multiplexor is considered to be part of the second multiplexor. The 6 meter cable between the first multiplexor and the DCC is considered to be part of the first multiplexor. After a component is isolated as faulty, the connecting cable can be verified or replaced and the component reexamined. This reduces the complexity of the debug process and the time required to locate the fault.

- Eliminate trivial: Some error messages relate directly to the DCC indicating memory or microcode problems. Messages which directly identify a DCC problem represent a trivial case and are ignored by this model of the KB system.

Basic Knowledge Components

The system console error message is the basic indication of trouble in the system. We have included a complete set of disk error messages in Appendix B. In general the messages either diagnose problems in the DCC or assume the disks are malfunctioning. In practice, the multiplexor boards and interconnecting cables may also be faulty. System trouble shooters rely on interpreting not only an error message, but a sequence of error messages to verify problems in common data path elements. The basic trouble-shooting strategies are not complicated. Chosing the correct strategy given different user configurations is what distinguishes the expert from the novice troubleshooter.

Initial interviews with the service engineers revealed that they used the sequence of error messages as a key to developing ideas about faulty components. For example, using a system like that in Figure 1, error messages from Drives 0 and 1 might indicate a mux0, while a session of error messages from Drives 0,1,2,and 3 would lead to suspicion of a faulty DCC.

The second major knowledge component is a description of the component configuration. We identify 8 positions corresponding to the possible locations of a disk in a fully loaded system to easier describe system configurations. Each position may contain a drive or be empty.

The final knowledge component is the system model. As described in the assumptions, the model eliminates cables. This is accomplished by modelling the MUX path connecting the channel to the disk drive as a separate component from the data channel, and call this path the mux "port". A port includes all circuitry and cables dedicated to a particular drive on the mux. Mux0 is the mux physically closest to the DCC, and Mux1 is the second or farthest mux. Data passes through the channel path in Mux0 before reaching Mux1. Treating the parts of the mux separately simplifies the rule structure. The Mux would be replaced (after cable is verified) if either its data channel or one of its ports port is bad. Historical failure rate data could be part of the system model improving terminal candidate selection.

The model of the subsystem consists of from 1 to 8 drives, one or two muxes, either 4 or 8 mux ports, one DCC, and 2 data channels running through each mux.

ISOLATING FAULTY COMPONENTS

Overview

Our goal is to develop a system which will analyze a sequence of error messages within the context of a given model and identify a faulty component. If insufficient data is available, the system will use the model to develop hypotheses about the potential faulty components. It will then either indicate the faulty FRU (in trivial cases) or suggest efficient ways to eliminate suspect parts, such as reconfigure the cables or sub-system.

Isolating a faulty component when not trivial is an iteration process. Given a specific configuration and set of error messages, our inference engine should identify the strategy to measure values, replace, or rearrange components to eliminate suspects.

The KB system employs a two stage process for refining the list of potential failure suspects until a single likely candidate is indicated for replacement. During each iteration of the system, the first stage utilizes the error messages and a framework of subsystem configuration to deduce the set of possible faulty components. A list of suspect components is the output of stage one.

Where the output lists more than one suspect component, another error message is typically required to further troubleshoot. The KB second stage provides a set of rules to recommend component reconfiguration prior to executing test instructions to create and log the next error message. The configuration that exists on receipt of the next error message is the basis for configuration knowledge of the next KB iteration, whether or not the site specialist configures as recommended.

Deducing Faulty Components

The KB system takes data in the form of sequences or error messages and deduces a set of possible components that might have failed. Interview with service engineers yielded rules of the form:

"IF the system has configuration X AND drive Y fails
THEN EITHER drive Y is bad
OR mux Z is bad
OR channel W of the DCC is bad".

While it is possible to develop a traditional knowledge base in the IF...THEN form, the rules would grow unmanageably large when expanded to deal with all possible system configurations. We propose a logically equivalent representation using a logic matrix which accommodates multiple configurations without the combinatorial penalties. Similar work has been done by Jau et al. [10].

For each position where a disk may be placed, there exists a data path back to the DCC. We represent this path in a matrix like table 4:

	MP1	MP2	MP3	MP4	MP5	MP6	MP7	MP8	M1A	M1B	M2A	M2B	DCCA	DCCB
P0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
P1	0	1	0	0	0	0	0	0	0	1	0	0	0	1
P2	0	0	1	0	0	0	0	0	1	0	0	0	1	0
P3	0	0	0	1	0	0	0	0	0	1	0	0	0	1
P4	0	0	0	0	1	0	0	0	1	0	1	0	1	0
P5	0	0	0	0	0	1	0	0	0	1	0	1	0	1
P6	0	0	0	0	0	0	1	0	1	0	1	0	1	0
P7	0	0	0	0	0	0	0	1	0	1	0	1	0	1

Table 4

A configuration is determined by associating a set of drives with a set of positions. This mapping is kept in a system configuration table which can be displayed by the operator. When an error message is received, the operator is notified and asked to run the system. When the system is executed, all drives are accessed. The result is a set of error messages from the various partitions. For each error message we identify the drive associated with the partition. From the system configuration table we identify the position of each drive. Using the matrix from table 4, we can identify a set of vectors which represent possible faulty components.

In a similar manner, any drives which did not produce error messages can be mapped to positions and thus to vectors which describe components which must have worked.

If pi is the set of components associated with position i and di is drive i, then we define:

F=P .union. D Where:
pi is an element of P if a drive position Pi failed and
di is an element of D if an error message was received from drive di.

Likewise, we define:

W= P{prime} .union. D{prime} Where:
pi is an element of P{prime} if no error message from a drive at position pi and
di is an element of D{prime} if no error message was received from drive di.

Then define:

S= F and not W

That is S is the set of components which did not appear in a set of working components and did appear in a set of failed components.

Since the system was involved in response to an error message S can not equal the Null set. If S contains one element, it is identified as the failed component. If S contains multiple elements, the operator is asked to identify the known good components. If multiple elements still exist, after this step, the next step is to suggest new configurations.

For example, suppose we had a configuration like Figure 2 (below).

Assume error messages are received from d1 and d7.

Using the configuration table and the matrix from table 4:

p1 = {0 1 0 0 0 0 0 0 0 1 0 0 0 1}

p7 = {0 0 0 0 0 0 0 1 0 1 0 1 0 1}

F = {0 1 0 0 0 0 0 1 0 1 0 1 0 1}

Similarly:

p0 = {1 0 0 0 0 0 0 0 1 0 0 0 1 0}

p3 = {0 0 0 1 0 0 0 0 0 1 0 0 0 1}

W= {1 0 0 1 0 0 0 0 1 1 0 0 1 1}

So:

S = {0 1 0 0 0 0 0 1 0 0 0 1 0 0}

This vector implies that the problem is either mux port 1, port 7, drive 1, drive 7, or channel B of multiplexer 1. A normal tendency in this case might have been to suspect DCC channel B or Mux0 since they are common.

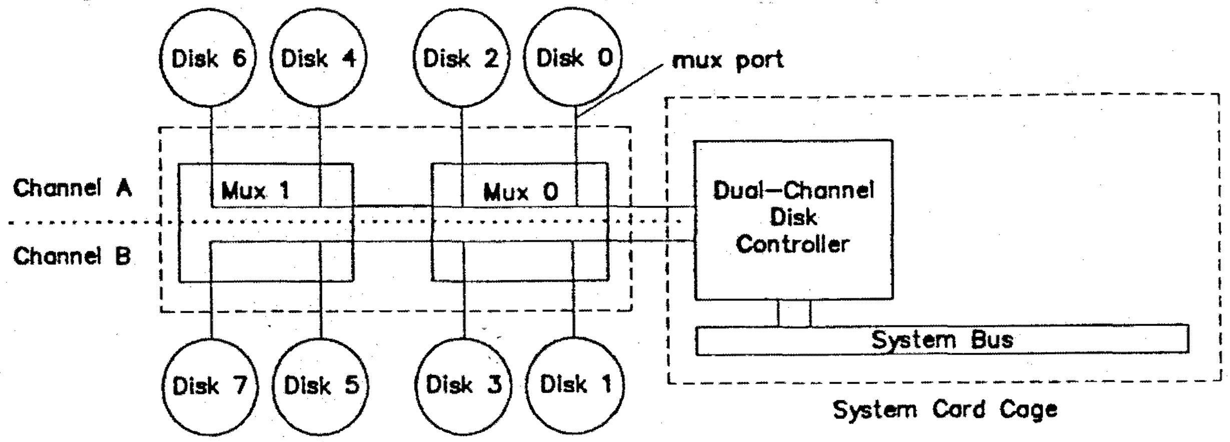

Figure 2: Partially configured subsystem

If the method produces multiple candidates, SNAP will suggest a new configuration be used to isolate failure candidates, or most probable candidate if reconfiguration will not produce more information. This would for example require the service technician or systems administrator on site to switch cables, connecting drives to another Mux port or DCC channel. When the above method is repeated with the new configuration, the set of components which must have worked is combined with the set of components that must have worked in previous passes before subtracting from the new set of possible faulty components. Very few passes would be expected to isolate the faulty component. Using our example, the S vector from the previous configuration will be combined with the F and W vectors from the next configuration to produce the next S vector. If S1 is the previous S vector and S2 is the next S vector, and if F2 and W2 are the F and W vectors from the next configuration, then

$$S2 = (S1 \text{ or } F2) \text{ and not } (W1 \text{ or } W2)$$

Next Configuration Analysis

Example of a series of rules for creating subsequent configuration:

```
IF      there are candidate drives
        AND
        the number of good drives is at least equal to the number of candidate drives
THEN    replace each candidate drive with a good drive.

IF      there are good drives
        AND
        there are more candidate drives than good drives
THEN
{
        IF      the number of candidate channels with at least one candidate drive is
                more than or equal to the number of good drives
```

```
          THEN replace all good drives with candidate drives such that each
                   candidate drive is on a different candidate channel.

          IF       there are less candidate channels with at least one candidate drive than
                   there are good drives
          THEN     select one candidate drive from each candidate channel and replace it
                   with a good drive
          AND
                   replace each remaining good drive with a yet unmoved candidate
                   drive, arbitrarily

          IF       the remaining candidate drives (not replaced with good drives) are less
                   than or equal to the number of non-candidate ports
          THEN     move all remaining candidate drives into non-candidate ports

          IF       the remaining candidate drives (not replaced with good drives) are
                   more than the non-candidate ports
          THEN     move a candidate drive into each non-candidate port arbitrarily,
                   leaving any remaining candidate drives unmoved.
}

IF        there are candidate drives
          AND
          no good drives
THEN
{
          IF       there are at least as many non-candidate ports as there are candidate
                   drives
          THEN     move all candidate drives into non-candidate ports

          IF       there are candidate ports
                   AND
                   there are more non-candidate drives than candidate ports
          THEN     move a candidate drive into each non-candidate port, arbitrarily
}

IF        there are no candidate drives
THEN
{
          IF       there are candidate channels with one candidate port and one non-
                   candidate port
                   AND
                   there are at least as many good drives as there are candidate channels
                   with one candidate port and one non-candidate port
          THEN     move a good drive into the non-candidate port of each such channel

          IF       there are less good drives than there candidate channels with one
                   candidate port and one non-candidate port
          THEN     move each of the good drives into the non-candidate port on a
                   candidate channel
}
```

Following are the rules for terminal cases at which the iteration through the system will stop. Also included is the recommended user action when each of these terminal cases is arrived at.

```
IF      no candidate components are left
THEN    the tests terminated successfully

IF      a candidate drive is attached to a good port
THEN    the drive is bad. Replace it.

IF      all components in the system are candidates
THEN    no further isolation is possible, so:
        IF      there is only one MUX in the system
        THEN    replace the MUX

        IF      there are 2 MUXes in the system
        THEN    replace the DCC

IF      a good drive is attached to a candidate port on a good channel
THEN    the port is bad. Replace the MUX

IF      a channel is the only candidate along the path between a good drive and the
        DCC (i.e. channel is on the second MUX and all other components along
        path are known good)
THEN    the channel is bad. Replace the MUX.
```

Sample Session

This section consists of an example interactive troubleshooting session. "System" represents messages from the AI system while "User" represents responses supplied by the session troubleshooter in the following dialogue. Parenthesized statements are included for clarification of this example only, and are not part of the dialog. Example of session initialization is a customer call to field service hotline. Typically a modem line is set up for remote acquisition of error messages, system configuration, and to run diagnostics.

```
System>         Welcome to the Disk Drive Troubleshooting Aide.
Please type in your answers following the "User" prompt. You may type "Configuration" or
"Explain" at any time. "Configuration will display to you the current Disk Drive Subsystem
configuration, and "Explain" will describe to you the reasoning which went into the latest
decision made by the Troubleshooting Aide.

This is your current Disk Drive Subsystem configuration:
        (on X-terminals an illustration would appear similar to figure 1, or third page in Appendix A)
        You have a single MUX with drives in positions 0, 1 and 2.
        Would you like to proceed with the troubleshooting?

User>   Yes.
```

(System runs tests on all disks and executes the first stage of inference)

```
System>         Candidate components are:
        1) Disk in position 0
        2) Disk in position 2
        3) The Mux
        4) The DCC
        Enter the number(s) corresponding to any of the components above which you know to be
```

good, or 0, if none.

User> 0

(System runs second stage of inference)

System> To continue reconfigure your Disk Drive Subsystem by doing one of the following:
1) Replace Disk in position 0 with Disk in position 1. (graphics showing disk cable reroute)
2) Replace Disk in position 2 with Disk in position 1.
When done, run the remote diagnosis "configuration" test (if connection is established), or enter the number corresponding to the configuration you have selected?

User> 1

(System updates its configuration table, reruns tests on all disks and executes the first stage of inference)

System> Candidate components are:
1) Disk in position 2
2) The Mux
3) The DCC
Enter the number(s) corresponding to any of the components above which you know to be good, or 0, if none.

User> Explain

System> After replacing the Disk in position 0 with the Disk in position 1 and rerunning the tests, error messages from the Disk in position 0 disappeared. Conclusion: Disk 0 is good and should therefore be removed from the list of candidate components. If you also replaced any of the other subsystem parts with known good parts from spares or another working subsystem, you may eliminate those failure candidates.

Candidate components are:
1) Disk in position 2
2) The Mux
3) The DCC
Enter the number(s) corresponding to any of the components above which you know to be good, or 0, if none.

User> 0

(System runs second stage of inference)

System> To continue reconfigure your Disk Drive Subsystem by doing one of the following:
1) Replace Disk in position 2 with Disk in position 0.
2) Replace Disk in position 2 with Disk in position 1.
When done, enter the number corresponding to the configuration you have selected?

User> 1

(System updates its configuration table, reruns tests on all disks and executes the first stage of inference)

System> Candidate components are:
1) The Mux
2) The DCC
No further isolation is possible.
Enter the number(s) corresponding to any of the components above which you know to be good, or 0, if none.

User> 0

(System identifies a terminal condition)

```
System> No further isolation is possible.  We suggest that you replace the Mux with a known
good one from spares depot << name and phone for nearest spare Mux, from Service DB>>.

Any further questions?

User>   Explain

System> After replacing the Disk in position 2 with the Disk in position 0 and rerunning the
tests, error messages from the Disk in position 2 disappeared.  Conclusion: Disk 2 is good and
should be removed from the list of candidate components.  It is not possible to further
isolate among the remaining components.  Applying the terminal condition rules resulted in
suggesting that you replace the Mux.  Given your configuration and known failure rates the Mux is a
more probable candidate for failure than the DCC.

        Any further questions?

User>   No

System> Thank you.
```

CONCLUSIONS AND RECOMMENDATIONS

We have identified disk subsystem troubleshooting as a problem suitable for prototyping an expert system at Sequent. The prototype would show reasonably good pay back if introduced in a small production system, with break even expected within a year.

Integration into production should leverage existing resources such as remote diagnostics, data bases, and technical illustrations from service manuals. Due to the percentage of code required for these standard interface software components, using AI tools that are UNIX-based and include X-window graphics and Oracle interfaces are highly recommended even for the prototype.

A portion of SNAP appears to lend itself well to a combination of rule-based and model-based reasoning. Because of the model simplicity and interactive means of building the configuration matrix around a subsystem rather than the entire computer system, some of the traditional deficiencies of shallow reasoning KB systems may be avoided. SNAP team focus should be on meeting user requirements which include interfacing with their existing tools like data bases. Careful attention to interfaces and modularity will enable SNAP to take advantage of other non-model-based features. Other problems may be better handled by frames (like LES [12]), or even independent or loosely coupled inference engines.

Bibliography

[1] Angeline, P.J., and D'Onofrio, T.W., "A model-based Expert system for component-level fault diagnosis," in Proc. SPIE-Applications of Artificial Intelligence V, 1987, pages 20-27.

[2] Barr, A. and Feigenbaum, E.: "The Handbook of Artificial Intelligence," William Kaufmann Press, CA, 1981.

[3] Barr, A. and Feigenbaum, E.: "The Handbook of Artificial Intelligence: Volume 2," HeurisTech Press, Stanford Calif., 1982.

[4] Bauchannan, B.G., and Duda, R.O., "Principles of Rule-Based Expert Systems," Advances in Computers, Vol 22, 1983, pages 163-184.

[5] Davis, R., et Al., "Diagnosis Based on Description of structure and Function," Proc. Conf. Artificial Intelligence, 1982.

[6] Davis R., "Diagnostic reasoning based on structure and behavior," Artificial Intelligence., Vol 24, 1984, pages 347-410.

[7] Douglas, J., "Delivering Online Expertise," EPRI Journal, Vol. 14, April/May 1989, pages 24-33.

[8] Folley, John D. Jr. and Hritz, Rohn J., "Embedded AI Expert System Troubleshoots Automated Assembly," IE, April 1987, pages 32-36.

[9] Freiling, M., Knowledge-Based Engineering Management, course notes, 1990.

[10] Jau, J. Y., Kiamilev, F., Fainman, Y., Esener, S., and Lee, S. H.: "Optical expert system based on matrix-algebraic formulation," Applied Optics, Vol. 27, No. 24, December 15, 1988, pages 5170-5175.

[11] Laffey, T.J., Mylopoulos, J., and Tennant, H., EXPERT SYSTEMS: Integration with Databases and Real-time Systems, The 34th Videoconference Seminars, Via Satellite, IEEE, March 28, 1990.

[12] Laffey, T.J., Perkins, W.A., and Nguyen, T.A., "Reasoning About Fault Diagnosis with LES," IEEE Expert, Spring 1986, pages 13-20.

[13] Liu, David, Ed., Expert Systems, a round table discussion, published by CASA/SME, Society of Manufacturing Engineers, Dearborn, Mich., 1990

[14] Miline, R., "Artificial Intelligence for Online Diagnosis," IEE Proceedings, Vol. 134, #4, July'87

[15] Miline, R., "Fault Diagnosis & expert systems," in Miline, R.W., and Chandrasekaran, B. (Eds.), The 6th International Workshop on Expert Systems and their Applications, Avignon, France, Apr 1986.

[16] Miline, R.,: "Reasoning about Structure, Behavior and Function," SIGART Newsl., July 1985, (93), pages 4-59.

[17] Niwa, K., Knowledge-Based Risk Management in Engineering: A Case study in Human-Computer Cooperative Systems, John Wiley & Sons, 1989.

[18] Novak, T., Meigs, J.R., and Sanford, R., "Development of an Expert System for Diagnosing Component-Level Failures in a Shuttle Car," IEEE Transactions on Industry Applications, Vol 25, No. 4, July/Aug 1989, pages 691-698.

[19] Schorr, H. and Rappaport, A., Innovative Applications of Artificial Intelligence, AAAI Press, Menlo Park, CA 1989.

[20] Sequent Computer Systems, Inc, 1989 Annual Report, March 1990

[21] Sequent Computer Systems,Inc. Profile, Black and Company, May 1990

[22] Van Melle, W.J., System Aids in Constructing Consultation Programs, UMI Research Press, Ann Arbor MIch., 1981.

[23] White, B. Y., and Fredricksen, J. R.: "QUEST: Qualitative Understanding of Electrical Systems Troubleshooting," SIGART Newsl., July 1985, (93).