An Expert Network Analyzer: Knowledge Acquisition, Fault Diagnosis and Prediction

Nikitas J. Dimopoulos, Andrew Watkins, Stephen Neville and Kin F. Li
Department of Electrical and Computer Engineering
University of Victoria
Victoria, B.C.

Abstract

In this work, we present the framework of an expert system that will be capable of diagnosing and predicting faults in a cable television plant.

The structure of the diagnostic expert system under development comprises equipment specific and network specific parts.

For the equipment specific part, there exist well established diagnosis and calibration procedures. These are captured and integrated into an equipment specific knowledge based diagnosis system with the aid of specifically designed knowledge acquisition system.

The network itself is monitored in real time, and its status data are used to determine failure points and the likely cause of the failure.

Introduction

Engineering systems are becoming numerous and complex. This has resulted in the proliferation of the number and complexity of procedures necessary for maintenance, diagnosis and operations of these systems. The end result is that very few people (if any) within an organization may be cognizant of all the appropriate procedures applicable to any specific system.

Diagnostic expert systems have been developed for several areas of expertise. These include medical as well as engineering diagnosis.

Recently, a number of computer-aided tools have been developed to centralize and make this procedural knowledge accessible. The tools range from simple Hypertext-based renditions of manuals to expert systems which are capable of suggesting and applying the most appropriate procedure.

In a Hypermedia document [4], information is organized nonlinearly and delivered using the most appropriate medium. This may include audio, video, text, graphics etc. The nonlinearity in the organization of the information results from the existence of multiple interrelations linking portions together. These interrelations result in easy access of related topics and avoid lengthy searches frequently encountered in a linearly organized document.

In general expert systems incorporate declarative knowledge in terms of collections of "rules" of the form

if <set of antecedents> **then** <set of consequents>

An inference engine is then used to apply rules from the rule base and advance the inferencing of the system. The order in which the rules have been entered in the knowledge base does not influence the way the inference engine proceeds, rather the rules themselves as they are applied (fired) drive the inferencing to its conclusion.

Many expert systems have been developed based on this premise, as well as several toolsets which help with the prototyping and fielding of such knowledge-based systems.

One of the most difficult problems in developing a knowledge-based system is the acquisition and coding of the knowledge of the domain experts. Knowledge acquisition involves the close collaboration of the domain expert(s) with the knowledge engineer(s). Usually, knowledge is elicited during several sessions with the participation of both the domain expert(s) and knowledge engineer(s). The thus elicited knowledge is incorporated, by the knowledge engineer in the knowledge-base, a minimal working system is rapidly prototyped and delivered. The prototype system is tested in situ by the domain expert, discrepancies and limitations are noted and the cycle repeats until satisfactory performance is obtained.

The limitation of this process lies in the difficulty of knowledge elicitation. Domain experts find it difficult to structure their knowledge in terms of general rules of the form stated earlier and to use categorical statements on exact quantities. Additionally, the knowledge engineer must at least minimally understand the domain so as to accurately code the knowledge presented by the domain expert. The existence of a widely acceptable and easily understood model for the domain knowledge, helps in its elicitation and organization.

Certain domains involve knowledge that is itself pro-

cedural in nature. Examples can be drawn from diagnosis which involves measurement or observation of specific parameters and these must follow a strict protocol either because of the expense involved, or the measurement itself is obtained as a part of a strict sequence of actions (e.g. certain blood analyses cannot be done unless the patient has fasted for a certain period of time). Physicians use protocols for both treatment and diagnosis [8]. Engineers use similarly structured procedures in operating, maintaining and diagnosing systems [2]

Most of this procedural knowledge is aimed towards human experts who are required to perform certain prescribed actions and depending on their outcome either reach a conclusion or perform another set of prescribed actions until enough information has been obtained to allow the establishment of a conclusion.

It is most often the case (especially with novices or if a procedure has not been performed recently) that the procedure, the required actions or the system itself are not familiar to the human expert. In such a case, manuals need to be consulted or the help of another expert is sought. This is time consuming, and it becomes critical in time limited procedures. Systems are currently developing which reposit such knowledge and deliver it in a user-friendly form [6].

Specifically, in the field of telecommunications and communication networks, there have been several examples of expert diagnostic systems which help interpret the status of the system and eventually produce a diagnosis.

Thus, Sugarawa [10] presented a distributed expert system capable of diagnosing local area network problems and has TCP/IP related diagnostic and troubleshooting knowledge. Miyazaki [7] describes techniques for updating switching networks operations and maintenance through expert systems. Nuckolls [9] describes an expert system that performs real time diagnosis of a large digital radio network. Yudkin [11] has introduced a methodology for building up a diagnostic system based on the structure behavior and functionality of the system and its components. In [5], we presented a preliminary system for diagnosis of a large Cable Television Network.

In this work, we shall present our efforts in developing a knowledge acquisition methodology as well as a real-time diagnostic system targeted for a large cable television distribution network. The work is divided into three parts. The first part discusses the development of FLOWTOOL™, our knowledge acquisition system, while the second part discusses the design and implementation of a real time diagnostic system capable of diagnosing a large Cable Television Network.

The current status, conclusions and future work are presented in the third part.

FLOWTOOL

FLOWTOOL comprises a graphical user interface through which the domain expert can easily represent procedural knowledge in terms of a flowchart incorporating decision points and prescribed actions. Additionally, knowledge which cannot easily be represented in the form of rules but it is also important or useful is incorporated in a hypermedia document. The user of FLOWTOOL establishes links to appropriate sections of this document. Once the acquisition phase is completed, the acquired knowledge is automatically translated to a set of rules targeted for a particular inference engine and environment complete with the established hypermedia links. The thus created prototype knowledge-based system can be utilized immediately by the domain expert who created it in the first place without the intervention of a knowledge engineer.

Operational Description

FLOWTOOL is a graphics based, flowchart editing tool which allows the user to quickly create and edit flowcharts by using a computer mouse to point and click. Flowcharts are created by dropping different flowchart symbols on a drawing area (or canvas) and connecting them together. Once a symbol has been placed on the canvas, text can be added and hypermedia links attached. At any time during the creation process, the flowchart can be edited or saved to or restored from a file. Once complete, the flowchart is checked for consistency and compiled into expert system rules.

Figure 1. illustrates the graphical user's interface involved during the operation of FLOWTOOL. The window consists of three sections, a row of control buttons along the top, four different mode buttons down the left side and a drawing area (canvas) in the middle.

The top control buttons allow the user to perform specific actions on the flowchart. They provide menus for loading, saving and editing flowcharts as well as changing the visual properties of the flowchart being displayed. The rightmost 'Flowchart' button contains menu options which let the user attach hypermedia pages and compile the flowchart into expert system rules.

The mode buttons along the side change the current state of the program. The current state determines what action will be taken when the user presses the mouse button on the canvas. While on the canvas, the cursor will change shape to reflect the chosen mode.

Flowchart symbols are drawn by first clicking on a symbol mode button (i.e. a statement or decision button) and then clicking on the canvas. When the mouse pointer is moved back over the canvas, its shape will change to either a rectangle (statement) or a diamond (decision), depending on the symbol to be placed. Whenever a new symbol is dropped, it is highlighted, designating it as the current symbol and a dialog box will appear to display information about it. The sym-

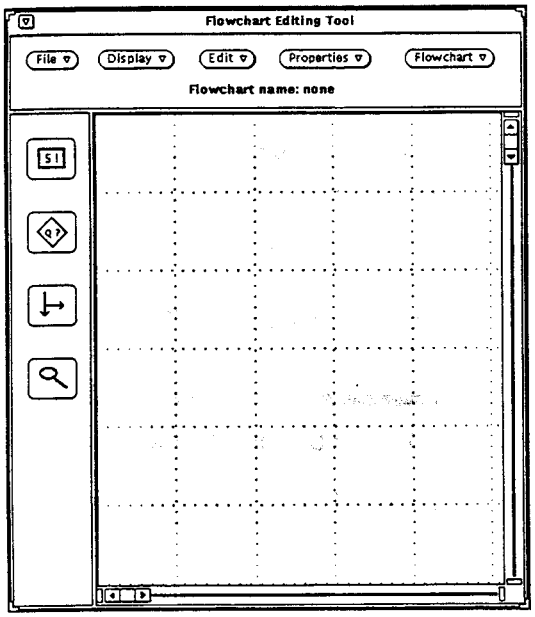

Figure 1: FLOWTOOL's graphical user's interface

bol's dialog box displays the symbol type, its location on the grid and its textual contents. In addition, if a hypermedia page has been attached, then the name of that page and its priority are also displayed.

Once two or more symbols have been placed, they can be connected together to form a directed arc in the flowchart. To connect two symbols, the user must first click on the connect mode button. The user is then prompted to identify source and destination symbols and, having done this, a link is then created between them.

The contents of any symbol can be displayed by clicking on the examine mode button. This causes the cursor to change shape to a magnifying glass, denoting the current mode. When this magnifying glass is then moved over any symbol, a dialog box will appear, displaying the contents of that symbol.

Figure 2. shows a completed flowchart including the contents of a flowchart symbol. The 'Symbol Contents' dialog box displays information about the highlighted decision symbol at the right of the canvas. Besides the full contents of the symbol, a hypermedia page entitled 'AC Distribution Board Page' is also shown with a 'high' priority. This means that when this node in the flowchart is encountered during flowchart traversal, the designated help page will appear (if the user has chosen to accept high priority help pages).

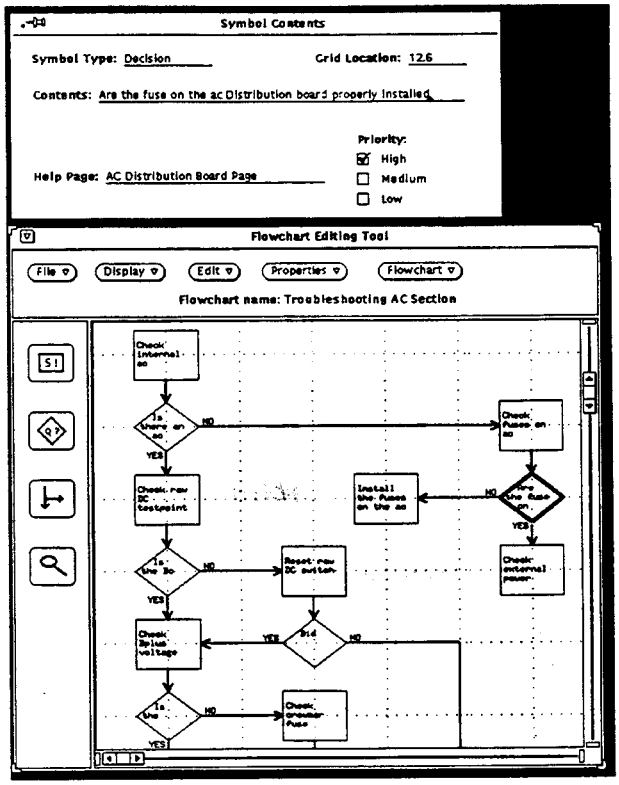

Figure 2: Example of Procedural Knowledge Acquisition.

At any point, the flowchart depicted, can be compiled to a cluster of rules encapsulating the procedural knowledge depicted. The compilation process is straight-forward. Each decision box results into two rules, one for the affirmative and the other for the negative outcome. The contents of a decision box together with references to its preceding decisions constitute the antecedents of the rule, while the statement(s) following a decision path constitute the consequents of the rule.

Additionally, any links to hypermedia pages present, are attached to the rule(s) created. A hypermedia server intercepts requests to display the attached hypermedia pages when the rule fires.

A flowchart incorporates procedural knowledge pertaining to diagnosis, calibration, operation etc.

Once a flowchart has been compiled, the resulting cluster of rules joins the collection of other clusters to form a knowledge base of procedures normally associated with a particular system. At any instant, one or more of these procedures may be applicable. For example, in diagnosis, each cluster would normally be associated with a specific manifestation of a malfunction. When the user wishes to use the expert system to solve a problem, he first describes the problem in terms of the observed "behavior", in the case of diagnosis by enumerating the observed symptoms from a list of known symptoms.

This is done using the 'Possible Problems' dialog displayed in Figure 3.

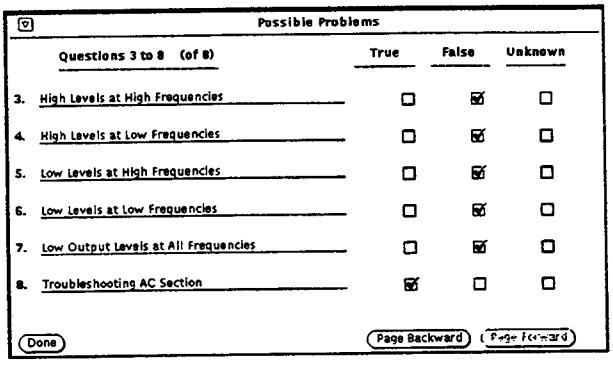

Figure 3: A possible-Problems Screen

Any listed problem checked true will cause the associated cluster to be processed by the expert system. Problems that are checked unknown will also be processed, but only after those checked true have been examined.

When the user hits 'Done', all of the requested flowcharts will be processed. Any hypermedia pages attached the flowchart symbols will also appear. Figure 4 shows how the user is prompted to answer a question from the 'Troubleshooting AC Section' flowchart. A hypertext page appears with the question to help the user make the correct response.

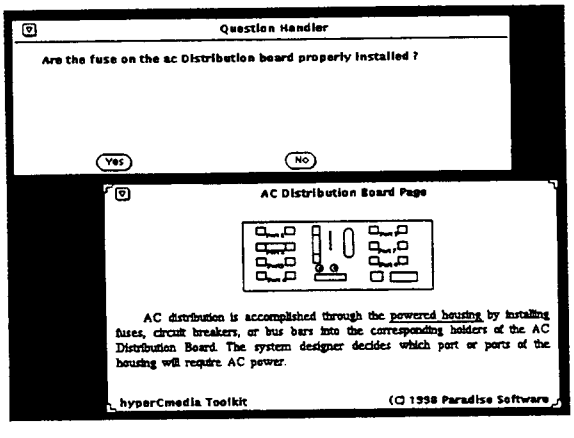

Figure 4: An inquiry presented during the operation of the expert system obtained from procedural knowledge capture by FLOWTOOL. Observe the associated hypermedia page which is also presented.

Once the user has answered all questions leading to a conclusion, another dialog appears to display the results of having navigated the knowledge base. If desired, the user can also view a trace of flowchart just processed. This trace details the decisions that were made in processing the flowchart and shows how the conclusions were arrived at. This is shown in Figure 5.

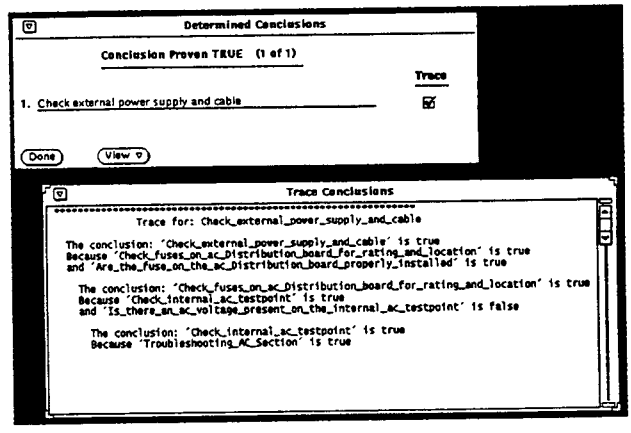

Figure 5: Typical Conclusion and Trace Screens

Network Diagnosis

Structure of the Network

A Cable Television Network incorporates a number of high frequency amplifiers forming (for conventional networks) a tree. In more advanced networks the structure incorporates a double ring from which subscriber drops emanate. In this work, we are focusing in conventionally structured tree networks. There are two categories of amplifiers, the ones belonging to the main trunk and the ones forming subscriber drops. Additionally, power supplies are located throughout the network, each one powering a limited number (typically three) of amplifiers. The majority of the main trunk amplifiers are equipped with a status monitor which uses a reverse channel to report the status of the amplifier to the head office. Subscriber drops and power supplies are not normally monitored.

Typical variables which are monitored include [1]

- Output pilot level
- Output data carrier level
- Raw DC voltage into the amplifier
- B+ voltage of amplifier power supply
- DC current into forward and reverse sections of the amplifier
- Temperature inside the trunk station
- Reverse Switch status
- Trunk lid status

The values of the monitored variables are allowed to vary within two intervals (warning and alarm) centered at typical values. If a value is outside these predefined intervals then a warning or an alarm is issued. Three consecutive alarms constitute a failure. Failure to raise a particular status monitor for three consecutive time intervals, also constitutes a failure.

Each status monitor apparatus has its own electronic address, which is used at the head end to poll it for a sta-

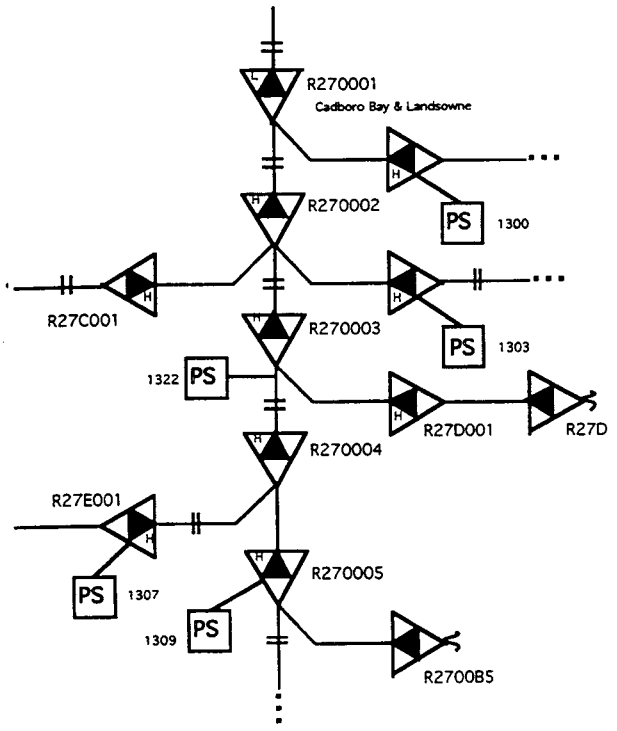

Figure 6: A typical Section of a Main Trunk (curtesy Roger's Victoria)

tus report.

Because of the number of amplifiers in the network, each amplifier is polled at fixed intervals (typically every few minutes).

The polling order may change if a particular section of the network needs closer attention.

A typical section of the main trunk is depicted in Figure 6. Each amplifier in the network has a name, as well as a location, connectivity and functionality attributes.

Diagnosis Problem

There are several modalities of failure. Some are discussed below.

A single amplifier may fail, whereupon the signal is fed through unamplified to the subsequent stage of the network. The subsequent amplifiers, equipped with automatic gain control, will boost the signal back to its normal level after two to three stages. Because of the failure, all amplifiers located between the failing amplifier and the stage at which the signal was boosted to its correct level, report alarms or failures. Because of the tree structure of the network, the reporting amplifiers are not typically polled sequentially and the reports appear at seemingly random locations in the report.

A power grid failure, will affect a number of amplifiers. The affected amplifiers fail to communicate, while the deteriorated signal, cause downstream nodes to report failure until the signal is boosted again to its typical values. Again, because of the structure of the network, these reports appear at random locations and are seemingly unrelated.

Another category of failures are due to the unmonitored subscriber drops. These are typically reported by the subscribers themselves as deterioration of service which range from increased noise levels for some channels to complete interruption of reception. These reports are due to failures in the main trunk, subscriber drops or even interruption of service due to the customer being at arrears in his/her payments.

In this work, we shall concentrate on the diagnosis problem of the monitored main trunk.

Diagnosis Approach

We follow a model-based approach in our reasoning. Because the structure of the network is known a-priori, and since the behavior of the amplifiers in the network is well understood, and because there is a large number of elements in the network, model-based reasoning is best suited for our domain. Model-based reasoning, as opposed to classification reasoning, bases its decisions on an underlying model of the system being diagnosed.

In our case, the structure, and operation of the elements in the network, provide us with a general and powerful model on which we base our diagnosis.

As we discussed previously, the failure is localized to at most a small number of amplifiers, but because of their interconnection, deteriorated levels of the signal propagate and affect a large number of neighboring nodes which themselves report failure. One of the main objectives of our diagnostic system is the identification of the *truly-failed* nodes from the ones which report failure because of their proximity to the failed nodes. Such agglomerations constitute *failure clusters.*

A *failure cluster* is defined to be a connected set of nodes, at least one of which exhibits a failure while the remaining register a warning, alarm or failure within an observation window which spans several observation cycles. For the remaining of the discussion, we shall use the term *failure cluster* to denote both a set of amplifiers as defined above, and its representation.

Failure clusters are important because they incorporate all the affected amplifiers within an observation window, and the reasoning engine can focus only within the cluster. Additionally, forming the failure clusters dynamically, we isolate the reasoning process from sources of information which cannot be made temporally constant. For example, the amplifier data-base together with the connectivity information changes in time as the cable-plant evolves.

Failure clusters act as filters, preventing the continuous formation of diagnoses of the same problem that has been previously diagnosed and is currently awaiting resolution.

Based on the above observations, we have constructed a system which automatically and in real time re-

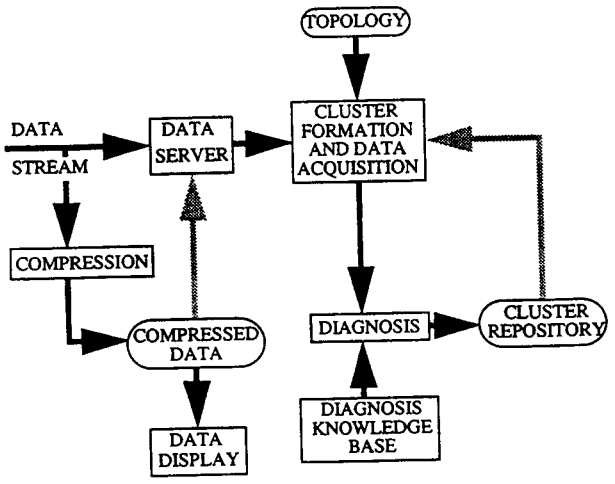

Figure 7: The structure of the Expert Network Analyzer

ceives the status monitor data from the amplifier network and produces a diagnoses in cases of failure. As depicted in Figure 7. above, our system comprises a data server, a compression and archival module, the cluster generator and the diagnosis module.

The stream of the status monitor data is captured, compressed and archived. The archiving is important, since this data is to be used in further analysis and failure prediction. A specialized display tool has been written which is capable of displaying arbitrary sequences of status monitor data. A typical display of such a sequence is depicted in Figure 8. below.

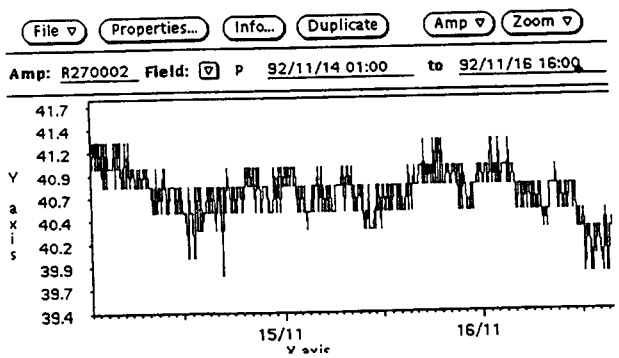

Figure 8: Display of a typical status monitor data sequence. This pertains to the pilot level.

A Data Server obtains data either directly from the status monitor stream or from the compressed and archived data and sends them to the cluster formation and data acquisition module. There, new clusters are formed according to the algorithm presented in Figure 9. The seed amplifier is defined as an amplifier that reports failure and does not belong to any of the currently active clusters. Once a cluster has been formed, it is sent to the diagnosis module which analyzes it and produces a diagnosis of the failure. This diagnosis is currently *mailed* electronically to the domain expert who provides a *critique* of the diagnosis conclusions reached. These critiques are saved and will be used to validate the diagnosis knowledge base.

Finally, the diagnosed cluster is deposited to the "active cluster" repository and remains there until the fault is repaired and all the amplifiers report normal operation.

Failure clusters, in addition to the amplifier names, incorporate connectivity information as well as observation information within the window of observation.

```
get next status monitor data
name the amplifier It
determine Its status

case (Its status is OK):
  if It is in queue[i] discard It from queue[i]

case (Its status is failing ORin alarm OR in warning):
  if It is in queue[i]
  then
        put It in newcluster[i]
        discard It from queue[i]
        put Its children in queue[i]

case (Its status is failing):
  if It is not in any of the queues
        if It is not in any of the clusters
            It is the seed of a new cluster
            make a new cluster
            make a new queue
            put It in the just formed newcluster
            put Its children in the just formed queue

if queue[i] is empty
  export newcluster[i] for data collection & diagnosis
  delete newcluster[i]
  delete queue[i]
```

Figure 9: Cluster Generation Algorithm

A failure cluster is translated into a set of NEXPERT objects which fully quantify the topology and behavior of the cluster within the window of observation. Each amplifier in the cluster comprises three objects, namely the topology_data object, the amp_data object and the log_data object. The topology data object incorporates static properties of the amplifier such as the location, connectivity, power grid, status monitor number, termination etc. The log data object incorporates the fault data of the amplifier such as the fault message, the fault type (warning, alarm, failure) the actual and nominal values of the variable that has caused the failure and the fault times. Finally the amplifier data object incorporates the observed data for all the monitored variables of the said amplifier within the window of observation.

Typical examples of objects are given in Figure 10., Figure 11. and Figure 12.

The set of objects, corresponding to a failure cluster and obtained as discussed above, is exported to the rule based system which produces a diagnosis. The rule based system, implemented in NEXPERT OBJECT,

currently comprises over 80 rules and is capable of diagnosing 12 different failure modalities. A preliminary evaluation of the diagnoses, has established an accuracy rate of approximately 80%.

```
{data_R20000D
 Classes: amp_data
 Properties:
    B     (23.8,23.8,23.8)          /* B+ voltage */
    C     (1143.3,1143.3,1143.3)    /* current */
    P     (-100.0,-100.0,-100.0)    /* forward pilot */
    R     (-6.0,-6.0,-6.0)          /* reverse pilot*/
    RD    (54.9,54.9,54.9)          /* Raw DC voltage */
    T     (-10.0,-9.0,-10.0)        /* Temperature */
    times (12140916,12140919,12140922)  /* Pol times */
}
```

Figure 10: Example of an amplifier data object

```
{topol_R20000D
 Classes:     topology_data
 Properties:  location      (Haultain@foulbay)
              parent        (R20000C)
              pilot         ()
              power_grid    (1209)
              SMT_number    ()
              subs          ()
              termination   (False)
              type          ()
}
```

Figure 11: Example of a topology_data object

```
{log_R20000D
 Classes:     log_data
 Properties:  fault_message  (Communication)
              fault_nominal  ()
              fault_times    (12140931)
              fault_type     (Alarm)
              fault_value    (NO REPLY)
}
```

Figure 12: Example of a log_data object describing an amplifier in an alarm condition because of failure in replying to a polling request.

Status and Conclusions

The work described here is part of a larger undertaking in collaboration with the Canadian Cable Labs Fund to investigate the use of AI and Knowledge Engineering techniques in the automation of the diagnosis procedures in a large cable TV plant. The development has proceeded in two phases.

During the first phase, we have developed FLOWTOOL™ a graphical knowledge acquisition tool which is capable of capturing knowledge expressed in the form of flowcharts and translating it into rules for NEXPERT OBJECT™. In addition, the resulting knowledge base is linked to hypermedia documentation incorporating information that cannot be expressed in the form of rules within the knowledge base. Appropriate sections of this documentation is being made available automatically during the operation of the inferencing and this can be used to assist and/or train the user of the knowledge base in providing appropriate responses and comprehending the inferencing involved.

The second phase includes the development of the Expert Network Analyzer.We have completed the beta version of our expert network analyzer on a SPARC platform in C and NEXPERT OBJECT. As discussed, this version of the expert network analyzer operates on line and in real-time. We are in the process of validating and refining the analyzer. This is done through the critiques of the diagnoses which are *electronically mailed* to the domain experts.

One more area that we plan to enhance our system, is its responsiveness. Currently, all the amplifiers in the network are polled sequentially, each complete polling sequence requiring approximately 90 sec. In order to form a failure cluster and accumulate enough data for a diagnosis to be possible, several polling cycles (currently 5) are required. This results to a delay of approximately 8 min from the onset of a failure to the time that a failure cluster is formed and a diagnosis becomes possible. In order to enhance the response time, we are currently implementing failure directed polling. This means that in addition to the standard sequential polling of all the amplifiers in the network, our system will be capable of polling specific amplifiers involved in the formation of failure clusters out of sequence and very rapidly. This would have as an effect the minimization of the delay in forming a failure cluster, and thus the generation of the diagnosis soon after the onset of a problem.

Lastly, we are currently focussing our attention in developing failure predicting models. For this part of the work, we utilize the archived status monitor data of the entire Roger's Victoria network, which we shall use in developing our predictive models. The status monitor data have revealed some interesting patterns of behavior, which we hope to correlate with and use as precursors of failure. Such patterns include the variation of the forward pilot level in close correlation with the temperature of the amplifier itself, as well as seemingly random variations of the current. Such patterns are observed only for data corresponding to failing amplifiers, while amplifiers operating normally exhibit stable readings for very long periods of time.

Acknowledgment

The support of the Canadian Cable Labs Fund is gratefully acknowledged.

References

1. *Instruction Manual and User Guide Status Monitor System*, C-COR, 5/84, rev 0, May, 1984.

2. Preliminary Instruction Manual for the B-507 Remote Bridger C-COR Inc. July 1983

3. S. Abu-Hakima "Visualization and Understanding Diagnoses" *Canadian Artificial Intelligence* No. 30 pp. 4-8 (Autumn 1992)

4. J. Conklin "Hypertext: An Introduction and Survey" *IEEE Computer* Vol 20. No. 9 pp.17-41 (Sep. 1987)

5. N. J. Dimopoulos, K. F. Li, A. Watkins, S. Neville, A. Rontogiannis "An Expert Network Analyzer" *Proceedings of the 35th Canadian Cable Television Association Annual Convention* pp.123-127 (Jun. 1992)

6. P. R. Frey, W. B. Rouse, R. D. Garris "Big Graphics and Little Screens: Designing Graphical Displays for Maintenance Tasks" *IEEE Trans. Systems Man Cybern.* VOL. 22, No. 1, pp. 10-20, Jan/Feb. 1992

7. Miyazaki, T., Fujimoto, H., Kim, M.W., and Wakamo, M., "Improving Operation and Maintenance for switching network," in *Proceedings of GLOBECOM '89,* IEEE, Nov. 1989, pp. 1149-1153.

8. M. A. Musem, L. M. Fagan, E. H. Shortliffe "Graphical Specification for an Expert System" in J. A. Hendler (Ed.) *Expert Systems: The User Interface* Ablex Publishing Corporation, Norwood, New Jersey.

9. Nuckolls, V., "Telecommunications diagnostic expert system," in *Proceedings of GLOBECOM '89,* IEEE, Nov. 1989, pp. 507-511.

10. Sugawara, T., "A Cooperative LAN Diagnostic and Observation Expert System," in *Proceedings Ninth Annual International Phoenix Conference on Computers and Communications,* IEEE, March 1990, pp. 667-674.

11. Yudkin, R.O., "On Testing Communications Networks," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 5, 805-812, 1988.

12. A. Watkins, N. J. Dimopoulos, S. Neville, K. F. Li "Flowtool: A Procedural–Knowledge Acquisition Tool" *Proceedings IEEE Pacific Rim Conference on Communications Computers and Signal Processing* pp. 31-34 (May 1993)