Flowtool: A Procedural-Knowledge Acquisition Tool

A. Watkins, Nikitas J. Dimopoulos, S. Neville, K.F. Li

Department of Electrical and Computer Engineering
University of Victoria
Victoria B.C.

Abstract— **In this work we present FLOWTOOL, a procedural- knowledge Acquisition tool developed specifically to acquire procedural knowledge associated with the diagnosis and calibration procedures of engineering systems specifically main trunk amplifiers used for communication and cable-television distribution networks.**

I. 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.

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 [5], information is organized nonlinearly and delivered using the most appropriate medium. This may include audio, video, text, graphics etc. The nonlinarity 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 procedural 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 had unless the patient has fasted for a certain period of time). Physicians use protocols for both treatment and diagnosis [3]. Engineers use similarly structured procedures in operating, maintaining and diagnosing systems [6]

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 [1].

In this work, we are presenting FLOWTOOL, a knowledge acquisition system specifically targeted towards the acquisition of procedural knowledge commonly found in such domains as diagnosis, calibration, safety etc. and its integration with descriptive knowledge of the target system. Thus section II gives a brief overview of FLOWTOOL, section III presents a detailed operational description, while we conclude with section IV.

II. 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.

III. 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 symbol'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

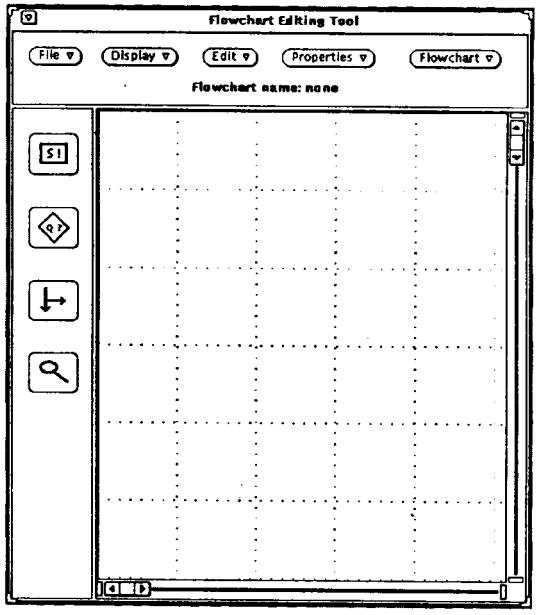

Fig. 1. FLOWTOOL's graphical user's interface

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).

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 positive 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 also 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 pertainng 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

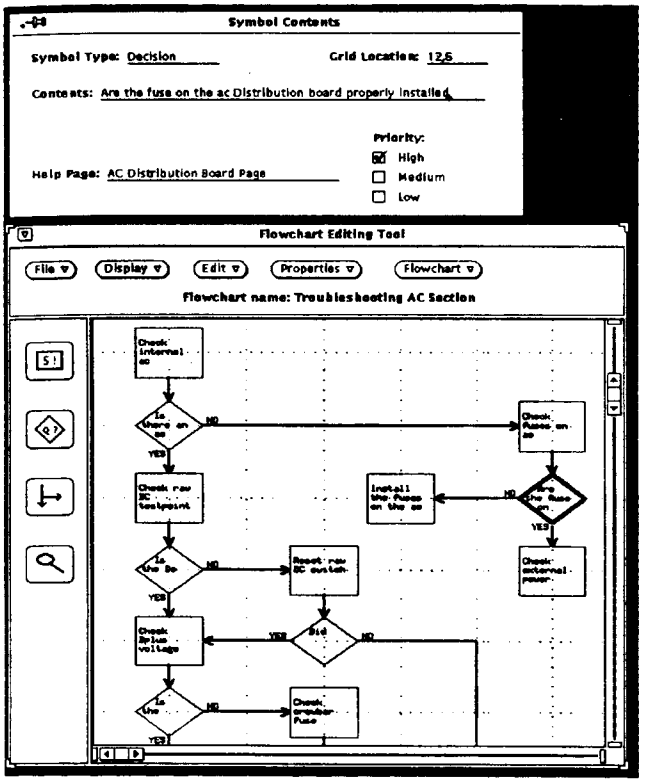

Fig. 2. Example of Procedural Knowledge Acquisition.

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. 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.

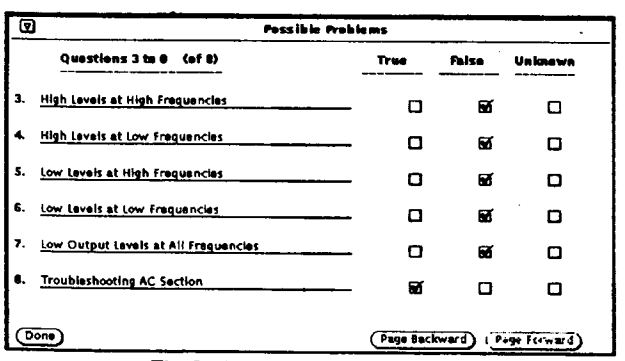

Fig. 3. A possible-Problems Screen

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.

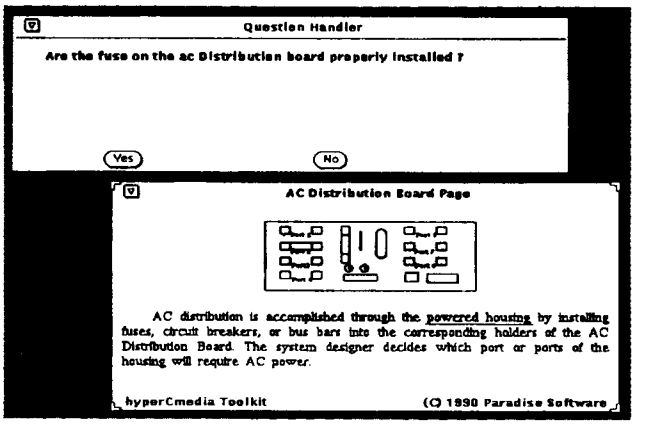

Fig. 4. An inquiry presented during the operation of the expert system obained from procedural knowledge capture by FLOWTOOL. Observe the associated hypermedia page which is also presented.

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.

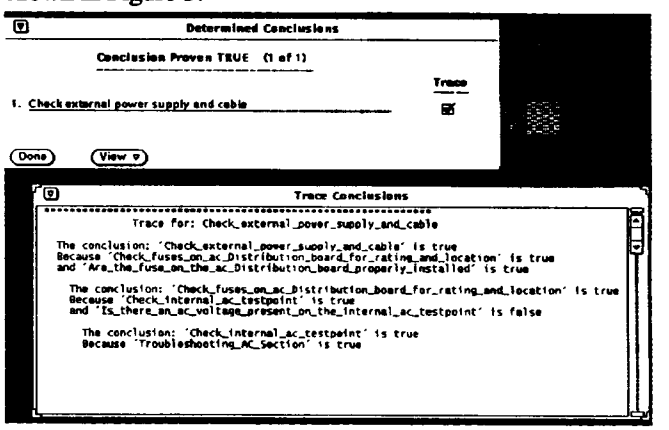

Fig. 5. Atypical conclusion and trace screen.

IV. Conclusions

In this work, we have presented FLOWTOOL, a procedural- knowledge acquisition tool. It uses a graphical user's interface to acquire procedural knowledge and delivers a knowledge-base complete with links to related hypermedia document(s).

FLOWTOOL has been implemented and operates in a UNIX / X-WINDOWS environment and creates rules for NEXPERT OBJECT and PROLOG.

FLOWTOOL has been used to capture diagnostic and calibration knowledge pertaining to C-COR main trunk amplifiers.

In the future, we plan to expand FLOWTOOL to incorpo-

rate fuzzy inferencing and optimal decision traversing [4].

ACKNOWLEDGEMENT

This work has been supported by a grant from the Canadian Cable Labs Fund.

REFERENCES

1. 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
2. S. Abu-Hakima "Visualization and Understanding Diagnoses" *Canadian Artificial Intelligence* No. 30 pp. 4-8 (Automn 1992)
3. 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.
4. A. Rontogiannis "Optimal Traversing of decision Trees" *M.A.Sc. Thesis* University of Victoria (April 1993).
5. J. Conklin "Hypertext: An Introduction and Survey" *IEEE Computer* Vol 20. No. 9 pp.17-41 (Sep. 1987)
6. *Preliminary Instruction Manual for the B-507 Remote Bridger* C-COR Inc. July 1983