AN EXPERT SYSTEM FOR ACCELERATOR FAULT DIAGNOSIS

E. Malandain, P. Skarek
CERN, 1211 Geneva 23, Switzerland

ABSTRACT

This paper discusses an expert system to assist operation and running of the CERN PS accelerators. The expert system aimed for in future will be an essential tool to ease maintenance, fault finding and eventually operation. It will reduce downtimes of the accelerators and should facilitate setting up beams. Two prototypes are discussed, one written with an EMYCIN-based shell, the other with a powerful hybrid frame, logic and production rule based development tool. The prototypes are based on deep-knowledge modelling of the structure and functionality of the accelerator and the computer control system. It will create its knowledge base directly from data bases describing the system. In addition it includes heuristics of experts' knowledge about the running of the accelerator and dealing with breakdowns.

Introduction to the control system

The CERN PS accelerator complex consists of several particle accelerators, accumulator rings and various beam transfer lines. They are controlled from a central control room through a network of about 20 minicomputers and about 150 microcomputers located in CAMAC, our interface to the process hardware. The accelerators serve many users in a recurring sequence of cycles of beams of different characteristics. Operators in the main control room can tune these beams concurrently (Fig.1).

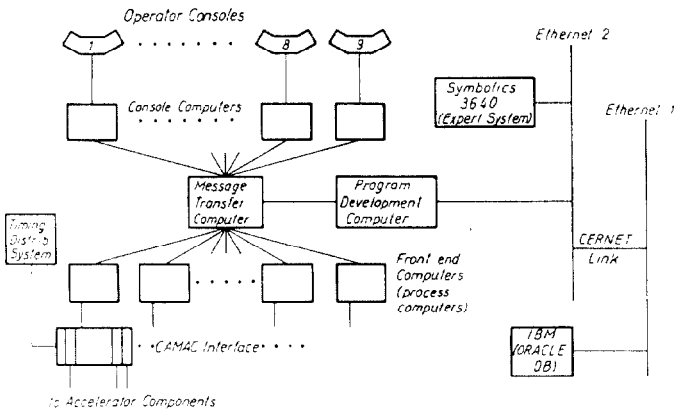

Fig 1: THE COMPUTER CONTROL NETWORK

There are about 4000 interface modules installed in about 250 CAMAC crates, and the application software behind that control system amounts to 1200 programs with about 400 displays representing 150 manyears of software.

The accelerators, their modes of operation and the control system has been described in numerous articles elsewhere, but a good impression and overall view of the complexity and size of the control system can be found in (1).

Background of the Project

For a system of the complexity described here, maintenance and fault diagnosis become a task demanding a high degree of competence and effort from the maintenance team. In parallel with a constant quest for improved reliability, methods had to be found to organize and optimize the procedures used in fault finding for efficient interventions. The use of expert system technology is considered as the most promising direction. To evaluate what that new technology can offer, a prototype has been implemented for fault-finding in a subdomain of the control system.

At the same time it can be said that most of the efforts going into the task of conceptualization and modelling the accelerator environment for fault diagnosis can be useful for a much wider range of applications: hardware and software maintenance and installation procedures, cold startup of the accelerator and finally operations like tuning beam lines and parts of the accelerator can very well profit from such a knowledge-based system.

Very few references can be found in the literature mentioning similar applications in the accelerator field (2), (3).

First Fault Finding Prototype

The Domain for our Prototype

The prototype is vertical: it covers a small part of the system but treats quite some details (4). The subdomain chosen is "an accelerator parameter attached to a console knob". This domain is representative of hardware faults and a great deal of software problems. The prototype treats possible problems when an operator attaches an accelerator parameter to a knob on the console, e.g current in a power supply or - on a higher level - a combination of timings and power supplies for controlling a closed orbit deformation. Examples of such problems are: the acquired value of the parameter is not identical to the intended control value or that the operator cannot control his parameter via the knob.

A call to corresponding software modules sets up conditions and sends out information via the CAMAC interface system to act on the desired element(s) in the accelerator. Mainly elements such as power supplies and timing generation modules are dealt with. General information is included from the operator's environment in the fault finding process (alarm information, error messages displayed to him etc.) which is entered by dialogue. Errors in the specific equipment itself are not treated.

Assuming for this prototype that the rest of the control system, i.e. consoles' and computers' hardware and software is working correctly, our domain is reduced to equipment drivers and their action chain through the interface to the accelerator components.

One of the difficulties is to deal with a mixture of hardware and software, a problem that becomes particularily apparent for the microprocessors and their software.

The prototype is not on-line. Therefore, the expert system will ask a lot of information during the consultation. It could get this information on-line directly from the control system and from the database describing the design parameters of the accelerator and the hardware and software of the control system. For a prototype of this modest kind, implementation of such links was not considered the main problem.

The TI Personal Consultant Expert System Shell

Shells are expert systems with an empty knowledge base. There exist now several commercial software packages or shells more or less adapted to different kinds of expert system applications and of any degree of sophistication. Some of those shells run on

CH2387-9/87/0000-0559 $1.00 © IEEE
PAC 1987

conventional computers some, very powerful, run on special harware.

We used a shell to be able to concentrate on the conceptualization and design problems and to get some experience in the field leaving out details about knowledge engineering. A shell permits to do so by providing mechanisms to treat the knowledge which we, as experts in the domain, program into the shell.

To gain experience in using such tools the use of a rather simple tool was started, providing a good user interface and some way of structuring the problem.

The shell at the disposal through courtesy of Texas Instruments is called Personal Consultant [5] and runs on a TI-PPC (Portable Professional Computer). It is based on a shell called EMYCIN [6], a decendent of MYCIN, the expert system for diagnosing infectious deseases.

To represent knowledge in the Personal Consultant one has to use rules with a condition and an action part. These rules can be grouped together to describe or treat objects which are called contexts here. Objects can be arranged hierarchically in a context tree.

Models of Diagnostic Problems

Diagnosis is one class of problems for which expert system technology has proven to be successful. There are two extremes in the way to look at a system to diagnose. One is to observe the system from the outside and say "if the system behaves like this, then that is the fault". To be able to state this, one has to use experience gained before. One sets up a list of symptom/fault pairs. This kind of statements about known facts and relations is called heuristics. The other extreme is to describe the system in detail and deduce the fault by deterministic methods. Then one has "deep knowledge" about the system.

In the heuristic case the fault is deduced from knowing how the system misbehaves for each specific fault. In the second case faults can be deduced by knowing about the system decomposition and the behaviour of the different subparts. The diagnostic reasoning in this case is based on some kind of simulation.

If a good model of the system behaviour is available, power is added to the simple heuristic approach. One is now able to diagnose faults that have never appeared before. Efficient expert systems use both ways. The main difficulty is to model the diagnostic problem and to represent the knowledge in a way so that it is easy to maintain.

The Personnal Consultant is suited for a conceptualization of the problem in terms of an observation-diagnosis-treatment-space using heuristic statements (rules). The shell permits some classification of these rules via the context tree. By using this possibility one can mirror the physical layout of the hardware and software modules representing the domain, and the reasoning chain used in diagnosis (Fig.2).

The contexts in the context tree represent substructures of the domain and the parts to be diagnosed and be checked one after the other. In the parent context on the next higher level, the findings on the lower levels are reasoned about by combining the information. The higher level contexts are capable of deciding if or not to enter the lower level contexts. The model consists of about 130 rules.

The shell allows for using certainty factors in the rules. They were used when the conclusion in the rule was not 100% sure or to increase the cetainty of a conclusion when several rules deduce the same fault.

Conclusions from the First Prototype

The prototype worked very well for problems in the domain covered. The possibility of quick prototyping

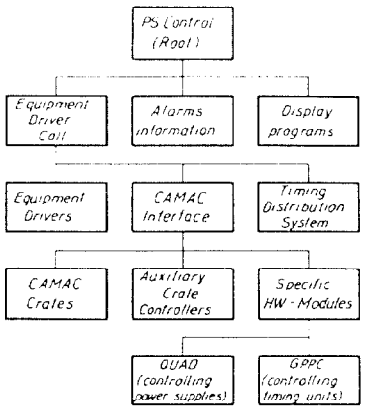

Fig 2 CONTEXT TREE FOR TI-PC PROTOTYPE REFLECTING A FAULT CLASSIFICATION

(3 months by two half time people) were also appreciated. which is one of the advantages offered by the exploratory programming style coming along with expert system technology. It was decided to continue in that direction and to apply these techniques to the full complexity of the control system and to the environment of accelerator operation.

Expert Systems in an Operational Environment

Models

For an expert system to be a good tool there are a certain number of issues to be solved. A deep model of the structure and behaviour of the components in the control system is necessary. This gives the possibility to separate the objects from the reasoning mechanism. This separation should finally allow to use more or less the same domain knowledge for different kinds of tasks: diagnosis, simulation and control (operations planning). It also gives the expert system the power of solving difficult problems. For this one needs a tool that gives the possibility of object oriented programming.

Use of Databases

Related to the problem of describing the physical structure is the whish to use a general source for our information of the software and hardware layout. The structure and the functionality of our control system is described in a relational database (ORACLE). There are essentially three different ways to use a database for the expert system. One can first create the complete knowledge base off-line. Secondly an access to the database can be done directly whenever an item is needed by the system. The third way is to let the expert system make an "intelligent guess" of what might be needed during the consultation. All this has to be seen in the light of efficiency and consistency considerations and the choice depends on the size of the database and the required access time.

Sensors and Testpoints for Diagnosis

Another important issue is the problem of sensors. Somehow the expert system has to be informed about the state of the control system. In this system parts of the diagnosed system itself are used when an on-line access is done. The control system has to be probed in using parts of the system itself. In diagnosing a breakdown, it has to be considered that the testpoints are no longer available, since that part of the system might be down in the first place or down as well. The expert system should be able to change access path whenever possible and necessary. Here again an object

oriented interface for the expert system to the existing software in the control system is vital.

User Interface

The user interface is essential and it should be based on graphics and adaptable to the skill of the user. Explanations of why the system comes up with suggestions and solutions should guide the user and give him confidence.

Extendability of the Domain

One advantage of expert systems is that they are more easily extendable than conventional software systems. A second prototype is now implemented covering the same subdomain as the first but in such a way that it can grow to cover more and more of the control system including the on-line requirements stated above. Diagnosis will be added on the beam level by including descriptions of beam entities (magnets, kickers, ...) and their behaviour and beam properties (radial displacement, chromaticity, ...). Beam diagnosis and the diagnosis of the control system can be seen as well separated tasks. The expert system should be able to couple them together and switch levels when appropriate. Qualitative reasoning methods will be included and commonsense models for beam diagnosis and optimisation to short cut complicated simulation models.

Development Environment

To be able to continue our work according to our requirements a Symbolics LISP machine is used with a powerful software development tool KEE (7). It is connected via an Ethernet link to the NORSK-DATA computer network controlling the accelerator and to an IBM mainframe as host machine for ORACLE. KEE is a tool which gives the possibility to develop an object oriented model. It has very powerful graphics tools and will allow for access to ORACLE. With KEE it is possible to design good reasoning structures adapted to the specific problem one wants to solve.

References

(1) G. Benincasa, A. Daneels, P. Heymans, Ch. Serre, p.242-250 in: Proc. Second Intnl. Workshop on Accelerator Control Systems, Los Alamos, NM, Oct.7-10,1985, North Holland.

(2) S. Clearwater, G. Papcun, D. Clark, p.193-196, in : same as (1).

(3) H.R. Brand, C.M. Wong, "Application of Knowledge Based Systems Technology to Triple Quadrupole Mass Spectrometry (TQMS)", Proc. AAAI-86, Fifth Natl. Conf. on AI, Aug.11-15, 1986, Philadelphia, PA.

(4) E. Malandain, P. Skarek, "A Knowledge Engineering Exercise for Accelerator Fault Diagnosis using the EMYCIN-based Personal Consultant (Texas Instruments)", CERN/PS (CO) 86-1.

(5) "Personal Consultant. Expert System Development Tool. User's Guide", Texas Instruments, Jan. 1985.

(6) W. van Melle, "A Domain-Independent System that aids in Construction Knowledge-Based Consultation Programs", Rep.No. STAN-CS-80-820, Stanford, CA.

(7) "KEE, the Knowledge Engineering Environment", IntelliCorp, Software Development System User's Manual, 3.0-U-1, 1986, Mountain View, CA.