KNOWLEDGE ENGINEERING METHODS FOR ACCELERATOR OPERATION

E. Malandain, S. Pasinelli, P. Skarek
PS Division, CERN, CH-1211 Geneva, Switzerland

Abstract: The availability of powerful tools for knowledge engineering opens a vast area of possible applications. Ideas for an expert system to be used for diagnosing the injection procedure of the CERN PS Booster will be given. The representation of the knowledge will be such that it can also be used as a reference document by the operator. This expert system should be integrated with the expert system developed in parallel for the diagnosis of the CERN PS control system.

Introduction

Diagnosing faults and finding the reasons for errors in a complex process is a task that demands experience and considerable knowledge about the process and the elements controlling it. This diagnostic task can be time-consuming and has to be done under stress. A computerized intelligent aid, providing reference values, recipes for testing and procedures for locating errors would be very valuable. The knowledge-based approach to these problems of diagnosis are usually called expert-systems and we consider this new technology useful in our domain.

Expert systems are special computer programs which perform logic reasoning about facts and situations. They come on all levels of expertise: from an essential help for a novice to a purely advisory function or analysis aid for the expert. To try out the principles and the usefulness of such methods as a help for accelerator operators to find errors, we are currently developing a small prototype treating the injection process from the Linac hadron injector to the PS Booster. Most of the software used has been developed for a prototype expert system to diagnose errors in the PS control system [1]. The prototype runs on a SYMBOLICS LISP machine.

Domain description

One of the advantages of using expert systems and knowledge-based prototyping techniques is that the domain covered can be gradually enlarged. The complexity of the problems to be solved can gradually be increased by adding more expertise and experience to the system. At present, our prototype deals with problems in the injection line and the injection procedure of one of the Booster rings. We consider the injection as successful if the beam can be captured by the RF system, the beam parameter we observe as the primary symptom is the intensity. Since, for the moment, the system has no on-line connection to the control system, quite a lot of information has to be supplied by the user, information which later on can be acquired automatically.

Knowledge representation

It is very important that the knowledge representation chosen fits well to the reasoning mechanism to be applied. We have to define the objects we want to reason about, but not only the physical objects like the elements of the accelerator. We need also concise and convenient definitions and representations of the concepts which we use when talking about a problem to be solved. Examples are the "intensity", "capture efficiency", "injection septum", etc.

The expert system development tool we use is KEE3 (from Intellicorp) which provides an excellent object-oriented frame-based knowledge representation scheme. KEE3 is not a shell in the sense that it provides a ready-made expert system where the user only has to fill the knowledge base with relevant domain knowledge. KEE3 provides you only with the building blocks. To tailor the system to an application quite a lot of some LISP programming has to be done. We have developed a kind of shell for another prototype for fault-finding in the PS control system [1]. Most of this software is now re-used for this second application.

In diagnosis one talks about symptoms, test points and faults. The faults we treat are in the controllers (referred to by a unique label, called UB name in our system). The reason for a fault is either a breakdown (i.e. a fault in the control system or in the specific equipment) or a wrong setting (this can come from the operator or again from the control system or the specific equipment).

A symptom is an observed deviation from an expected value or a deviation from some expected more complex information, like the form of a signal, representing a beam property.

Test points represent possibilities to get information about the state or the behaviour of the system. We distinguish between two kinds of tests. One type concerns the beam measurements, usually the source of our symptoms. The other kind of tests return control values or control signals like for function generators. It is highly desirable that the second kind of tests can be made automatically without user intervention.

The acquisition of beam properties is more difficult. Measurement devices are complex, the data often have to be interpreted by people knowing the equipment, etc. Judgement of this kind and reasoning with uncertain and incomplete information have to be investigated. This is related to fuzzy logic and fuzzy data for control.

Faults, symptoms and tests have to be defined as conceptual objects, likewise controllers, measurement devices, beams and beam properties. A beam can be seen as an assembly of beam properties. Conceptual objects are templates representing certain large classes of objects. We store them in a special knowledge base (a kind of object-oriented data base) which we call "CONCEPTS". For each basic type of object, there is a knowledge base where we describe, in a hierarchy, the objects from the most general features to the more specific. The reason for this is to be able to treat large numbers of objects producing a minimum of software. An example: in the knowledge base for controllers, we have a class of objects called dipoles. Most of their properties can be defined for all dipoles, on the class level, inherited by all members, and only specific properties of a special dipole are defined for that particular dipole. Part of a classification hierarchy and some representative properties of such objects are shown in Fig. 1.

The amount of objects in our system (several hundred) makes it necessary to have procedures creating these hierarchies automatically from simple files. Later on, we replace these files by accessing our database (ORACLE on an IBM mainframe) where the information about the accelerator should be kept to avoid duplication of data.

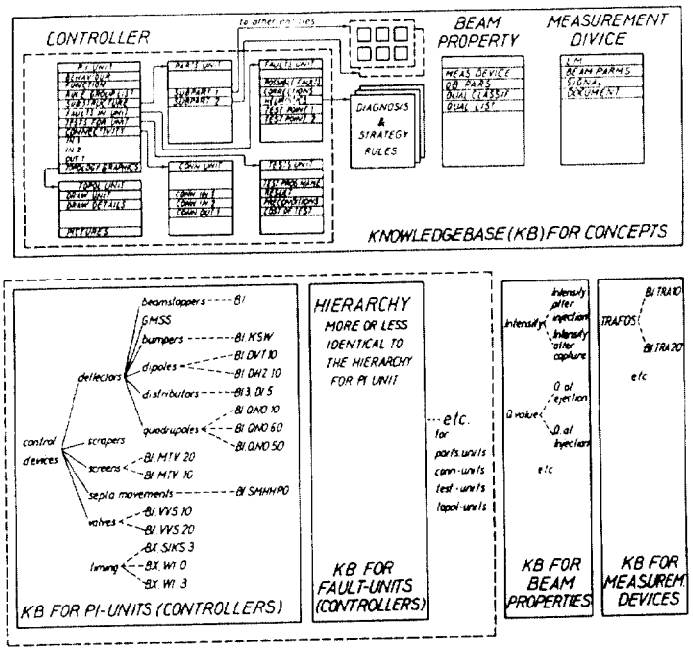

Fig. 1: Objects and knowledge bases

How do we represent the complex relations between parameters in an accelerator? If we find a symptom for one beam parameter we need to know the relations to other beam parameters and to the controllers. First, we must know that there is a relation at all and to define the type of relation. Then according to the type of relation, it can be qualified and quantified. This quantification we want to express to first order and constrain to boundaries in the parameter space (to avoid solving differential equations for instance). With this description, the symptoms can lead us to a fault. This so-called semantic net description can also be useful for control on a higher level (Fig. 2). The basic structure of the process is defined by a simple concept called connectivity where a separate frame unit (one of the "templates") describes which element sits in front of and behind which other element. This connectivity can be interpreted as data or information flow ordering of elements, similar to a simulation chain or any other ordering of what influences what in any respect. We use it for timing sequences for instance.

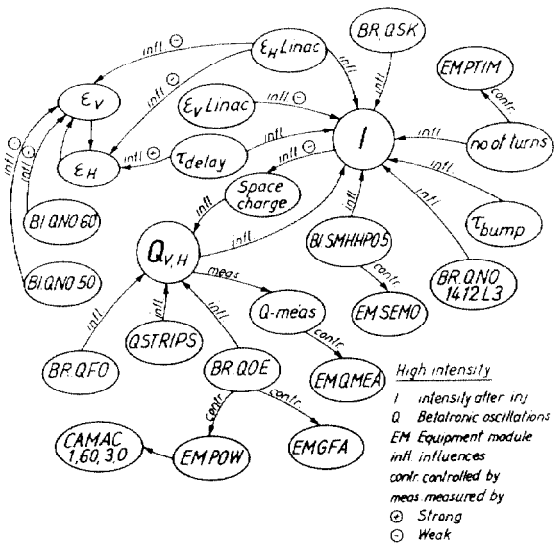

Fig. 2: Semantic net

The reasoning process

There are essentially two different ways of finding faults: one can sometimes deduce a fault or formulate a hypothesis directly from a symptom (shallow knowledge), or one has to know the structure and behaviour of the components of the system (deep knowledge). We use both methods. Shallow knowledge is used as much as possible; it is more direct and therefore gives a quicker result. When we lack shallow knowledge, we can always use descriptions of the component behaviour and their interconnections to deduce a possible fault.

The shallow knowledge we get from specialists telling us which symptoms indicate which faults. We also get information of this kind by inverting the problem: provoke a fault and observe the symptoms. In doing so we get so-called symptom/fault matrices, one matrix for each observed beam property, with the measurement devices on one axis and the possible faulty elements on the other. What we get as a symptom is a change in the observed beam property. We use this method only for getting the symptoms if elements fail completely. Otherwise several symptoms and more measurements have to be taken into account.

There are often several possible faults for one observed symptom. Then we have to create a number of hypotheses and treat them one after the other according to some ranking criteria (for example that the most likely one first or the one that has the least cost for testing). When we cannot find the fault by this simple approach, we have to know the layout of the process and how the process behaves. The classic way of doing this is simulation. But we do not have to simulate to find faults of the kind we are looking for here. We can define in/out behaviour between observation points from tests and observations or by very simple calculations valid for defined ranges of process parameters. By comparing the calculated values with observed values, one can determine if the error is between two test points. By simple rules one can then distinguish the different components or create hypotheses which have to be treated. We call this test point reasoning and it can be useful for an injection line.

The reasoning is implemented by using the rule system in KEE. To start the reasoning access to some simple observations is useful. We have chosen the intensity as the primary symptom. The intensity at a few points in the injection line, the intensity after injection and after capture are permanently displayed at the control console of the accelerator. The loss at those points is used to start the reasoning. Some obvious error sources are also eliminated by some simple questions in the beginning of the fault finding session. The system then starts to ask questions or to ask for more symptoms via measurements on the beam and to generate hypotheses. The hypotheses are confirmed or rejected by starting rules proposing tests for the controllers. The user is asked questions by menus, where a certain number of choices are presented or he is asked to choose a signal form that most resemble of what he observes. An example of a user interface to the reasoning process is shown in Fig. 3 and some corresponding rules in Fig. 4.

Documentation

To be able to propose tests, the system has to give information on how to perform tests on controllers and how to access reference values. When a fault is found, recipes and names of specialists, etc. are also available. Intelligent documentation should be a by-product of this expert system development. We are

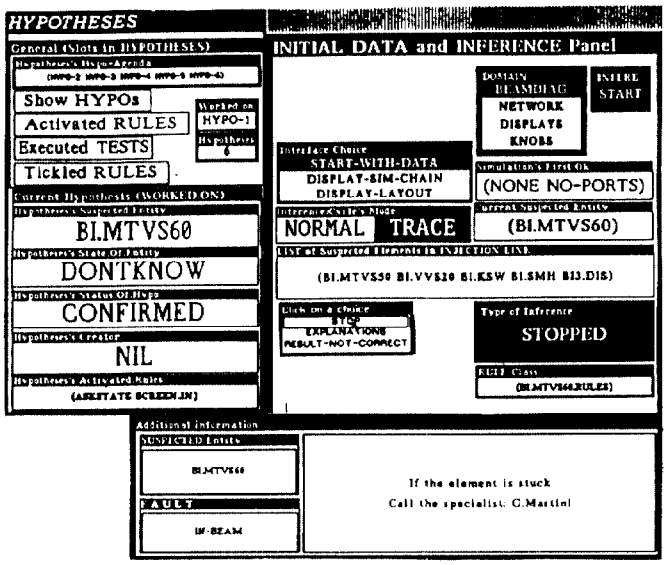

Fig. 3a: Example of user interface

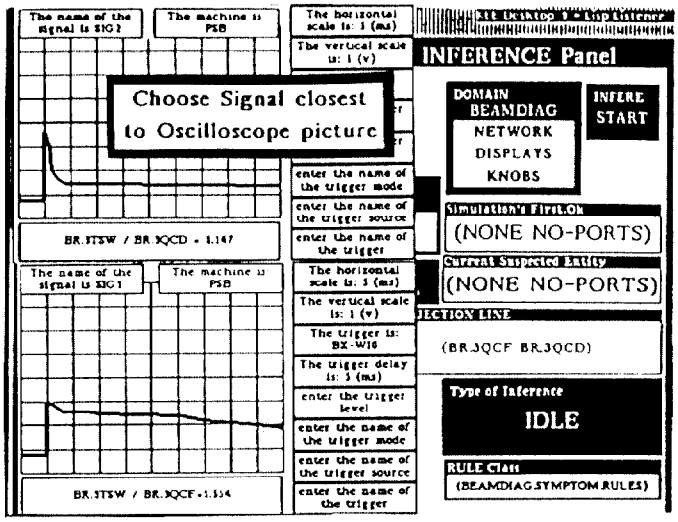

Fig. 3b: Examples of user interface

working on getting an easy access to our ORACLE database running of an IBM mainframe, where we want to put reference values for the process to be used not only by the expert system. A link to the control system is also of high priority. In this way many questions and manual checks can be avoided.

Extension to qualitative reasoning

Quantization of a continuous domain to a set of discrete symbols as quantity space can make reasoning about that domain quicker and more tractable. Instead of dealing with and reasoning about exact physical laws, usually expressed by conservation laws or differential equations, one can now reason about that symbol set, constraints between them and difference equations. This is one of the main aspects of qualitative reasoning about physical systems.

We have the feeling that the application of methods developed for qualitative physics, especially for qualitative process theory and qualitative measurement interpretation, and the research going on in these areas will have some impact on the use of knowledge-based methods in the accelerator domain. Although we use some kinds of quantity space in our rules (like the intensity is LOW MEDIUM HIGH, etc.) instead of the continuous range of real values, we are very well aware of the fact that - as Forbus [2] pointed out - such a simple symbolic vocabulary might be a usable representation for the specific problem at hand, but what one is really after is a quantity space that makes relevant distinctions for the general physical reasoning behind it.

Another aspect of qualitative reasoning is the automatic generation, of diagnostic rules from a qualitative model of the process [3]. In our case, one can imagine the automatic generation of the bulk of necessary rules, not only from the previously mentioned symptom/fault matrices, but also - on a more causal basis - by running simulations on a qualitative model to be developed. This can be done on the basis of the semantic net already shown.

The semantic net description we use for finding relations between parameters should later on be the basis to define optimal control sequences.

```
(INJECTION.RULES)

(EFF.INJECTION.HIGH1
 (IF (LISP (REMEMBER-TICKLE ?$RULE$))
     (THE QUAL.EFFICIENCY OF INTENSITY.AFTER.INJECTION IS HIGH)
     THEN
     (LISP (REMEMBER-RULE-INF.CYCLE ?$RULE$))
     (LISP (ASK-SIGNAL '((BR.QCF SIG1) (BR.QCO SIG2))))))


(EFF.INJECTION.ZERO
 (IF (LISP (REMEMBER-TICKLE ?$RULE$))
     (THE QUAL.EFFICIENCY OF INTENSITY.AFTER.INJECTION IS ZERO)
     THEN
     (LISP (REMEMBER-RULE-INF.CYCLE ?$RULE$))
     (LISP (CREATE-HYPO 'BI.MTVS60 'DONTKNOW))
     (LISP (CREATE-HYPO 'BI.MTVS50 'DONTKNOW))
     (LISP (CREATE-HYPO 'BI.VVS20 'DONTKNOW))
     (LISP (CREATE-HYPO 'BI.KSW 'DONTKNOW))
     (LISP (CREATE-HYPO 'BI.SMH 'DONTKNOW))
```

Fig. 4: Rule examples

Conclusion

We think that knowledge-based techniques are a promising way of providing operational help for running an accelerator. We are developing a small prototype for diagnosing errors in the injection process to the PS Booster and could partially re-use existing software, a sort of a diagnostic expert system shell. To make such a system really operational, it has to be linked on-line to the accelerator control system and integrated into the operator consoles environment. The application of qualitative reasoning to these kinds of problems might be promising as well, but it is far from being well understood and will still need much research.

References

[1] P. Skarek, E. Malandain, S. Pasinelli, I. Alarcon, A fault diagnosis expert system for CERN using KEE. Invited talk at the SHARE Eur. Assoc. Spring Meeting, April 18-22, 1988 at Davos, Switzerland, and CERN/PS 88-12 (CO).

[2] K.D. Forbus, Qualitative process theory, Artificial intelligence 24 (1984), 85-168

[3] Y. Ishida, An application of qualitative reasoning to process diagnosis: Automatic rule generation by qualitative simulation, Proc. 4th Conf. AI Applic. San Diego, Ca, March 14-18 1988, p. 124-129.