TOWARDS THE AUTOMATED COMMISSIONING OF AN ACCELERATOR

M.Plesko* and A.Wrulich,

Sincrotrone Trieste, Padriciano 99, I - 34012 Trieste, Italy

Abstract

Specifications for a high level software application, that will automatically aid the operator during the commissioning phase of an accelerator, are presented. The application, which is directly embedded into the control system, makes extended use of machine physics oriented programs, guided by an expert system. A prototype's performance, realizing a subset of the goals given in the specifications, is discussed.

1. Introduction

Today accelerators are operated under computer control, therefore many tasks in machine operation can be performed automatically by use of the computer. The algorithms of the computer programs that perform this automation are purely numerical, not allowing for special cases or general error finding, i.e. they include no expertise to deal with problems.

In order to prepare an automated operation and start up of the machine, more general approaches have to be taken. Not only the mathematical model of the accelerator has to be integrated into the programs, but also the knowledge and experience of an expert that would have normally used them. The modern approach to such a problem is to build an expert system (ES).

It has to be noted that - contrary to the impression induced by the notion of 'artificial intelligence' - an ES is not necessarily a sophisticated set of Prolog or LISP instructions. Simple heuristic rules may be far more efficiently executed as FORTRAN compiled statements, without the overhead generated by experts system building shells. In principle a single logical IF-THEN statement might be called an ES. The following definition is strictly followed here: "An expert system is a computer programme or a part of it, that encodes as closely as possible the heuristic reasoning and decision making of an expert, reproducing it step by step while working on a specific task."

In this paper we define a general paradigm of where and how to make use of an ES during the commissioning (or more generally: start up) of an accelerator and which other software is needed to support the ES. A prototype of such an application that is to be used for first turn steering of an electron storage ring is presented and discussed.

2. The Application

The system has to be built in a way that resembles the reasoning of a human operator. A classical ES has obviously not enough power for that. The operator reasons globally with objects and heuristic rules, but in addition, he uses mathematical expressions to derive exact values, once he has deduced the interrelation of physical quantities and the objects they describe.

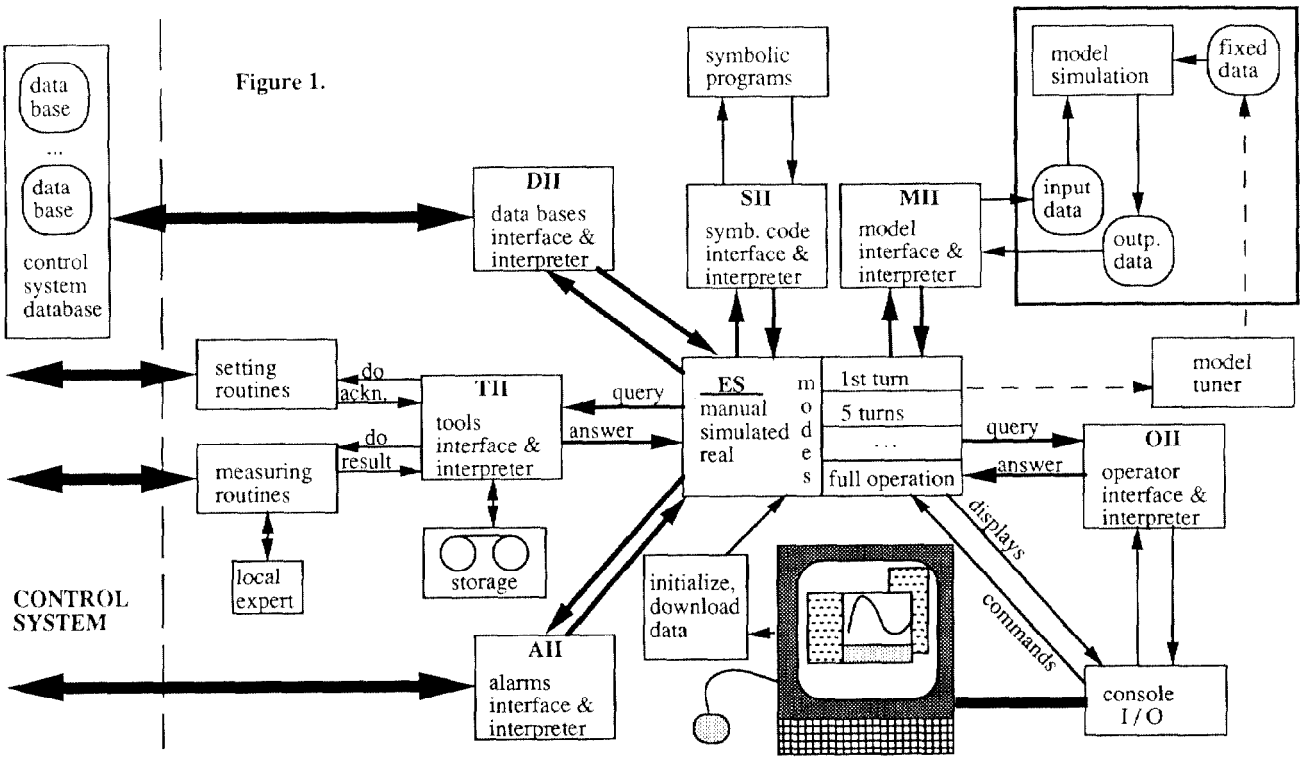
The application should be organized in the same way. It must make extensive use of the existing physical knowledge about the accelerator. Pure quantitative reasoning is tolerable only in domains, where the dynamics is not known, like in business, medicine, etc.

The basic features of the application are :

- a combination of an ES for symbolic reasoning with all the power available from a model simulation, measuring algorithms and their machine physics related programs.
- a modular system, open for future extensions of the accelerator and for improvements of the application itself, making it portable to other accelerator facilities.
- complete transparency of the performance to the user - no additional computer knowledge should be needed.

The software structure of the application is shown in figure 1. It has two basic layers - the ES in the centre and the application routines surrounding it. The ES performs symbolic reasoning, using a set of heuristic rules, just like an operator would. The application routines are usual numerical routines. Therefore the ES and the application routines are connected by an array of interface and interpreter modules. Their task is to translate the commands issued

Figure 1.



* On leave of absence from the University of Ljubljana, Yugoslavia

by the ES into the form that is acceptable as input for each particular application routine and then to interpret the output of them into statements of the ES.

On both ends of the application, there is a console I/O driver and an interface to the control system of the accelerator. The structure of the ES and the modules are briefly described in the following sections (see reference [1] for further details and references).

2.1 The Expert System:

There has been quite some effort related to the use of ES in the accelerator domain. Besides existing prototypes [2,3], there are some proposals for ES's [4] and even for the use of neural networks [5]. With the exception of the one reported in ref. [3], however, none of them has reached the operational stage as yet.

Past experience teaches us, that an ES cannot efficiently handle neither time dependent data nor large domains with many parameters. Furthermore, the effort over benefit ratio, by considering all possible cases, typically rises exponentially.

Therefore, the ES is split into several distinct ES's, one for each subgoal, which try to reach its subgoal by using a certain knowledge that is related to it. If the problem encountered can not be solved by using this knowledge, the ES will inform the operator about this, give a list of task it has performed and leave the operator to continue. It is clear, that such a situation may arise quite frequently, yet the ES is not useless. It performs the trivial checks that the operator would have to perform and might give, by means of its output, at least some hints about the problem.

To further ease the cooperation between the ES and the operator, the ES is operating in three modes. In the *manual* mode, a bypass of the ES is provided, to allow the operator to use the convenience of the tools in spite of a possible malfunctioning of the ES. In the *simulated* mode the ES acts on the parameters of the models. The operator may observe the choices made by the ES, before he enables it to perform the actual change. The complete automatism is realized in the *real* mode, where the ES acts on the real machine.

2.2 The Application Routines and Their Interfaces and Interpreters

While the ES may be written in some object oriented language, the surrounding programs, like the model, the measuring routines, etc. will be written by physicists, using FORTRAN. In order to keep the structure modular, the ES must not contain any code explicitly referring to one of these programs. Therefore the Interface and Interpreter (II) modules are designed. They act as 'hooks' of the ES to which the surrounding routines are attached.

For each type of application, there is one interface and interpreter module. The following II modules are necessary together with their relevant applications.

The *database II* loads machine specific data directly from the system data bases. The *operator II* is needed to allow the ES to use the same syntax for querying the operator, the database, or the measuring routines. The *alarm II* keeps the ES from tuning in vain as long as the hardware is down. Since the ES is not good at representing temporal knowledge, it will just wait for the alarm to disappear, before it resumes operation. The *symbolic code II* allows the application to make use of analytical expressions directly, using programs like Mathematica, SMP, etc. Some of these symbolic algebra programs also have powerful graphics, which might be used as well.

The *model II* connects the model to the ES giving all the relevant data of the lattice as needed. On the other hand, the model will not represent the true state of the real machine, with all its alignment errors and other mismatches. Therefore the ES will use the *Model Tuner* to adjust the data of the model (such as magnet strengths, positions, etc.) to make the model predictions match the measurements, as emphasized in ref. [3].

The *tools II* connects the ES to the other most important part of the application. The synonym 'tool' represents all application routines that measure or change the state of the machine.The tools are divided into two groups, the setting routines and the measuring routines. The former are straight forward, as they just set the control system variables. Measuring routines are considered all routines, that acquire data from the accelerator, even if they also change settings of the machine. The low level measuring routines may just read some values of the control system. The more physics oriented ones will use the beam as a measuring device. They will change some settings and observe their effects on the beam. Some routines might have their own local expert system (LE).

An example of such a LE is a routine to find and correct (by software) miscabled monitors or steerer magnets as described in the next chapter. Another possible LE could be an image processing and pattern recognition code for data obtained from a CCD monitor or for the comparison of digital oscilloscope signals.

3. A Prototype for Automated First Turn Steering

The requirements on the prototype were, that it is a true subset of the application described in chapter 2. As a promising domain, first turn steering of the beam in a storage ring has been chosen. This is a real subgoal of commissioning, where at least one action among the following fields is performed: model simulation, measuring beam parameters, comparing model and measurements, setting accelerator elements, model tuning, communicating with the operator.

From the machine physics point of view, the lattice used for starting first turn steering may be simpler than the one used for the final operation of the storage ring. Because there are less magnets used, it is easier to spot the errors. The complexity of the magnet structure can then be stepwise increased until finally all the elements are switched on and tested. It should be noted here that for a simple test of the magnetic elements no closed optics is needed, as first turn steering is equivalent to steering the beam through a transfer line.

It was assumed, that the injection into the storage ring had been properly commissioned and no possible sources of errors before injection can exist. Hence, the prototype deals only with transporting the beam from the injection point around the machine. It checks all magnets, their power supplies and the beam position monitors by using steerer magnets to sweep the beam transversely through the elements. It also uses the steerers to adjust the beam towards the centre in case some magnets are misaligned. Since during first turn steering the injected currents are usually too small to allow for precise measurements, only gross errors are searched for, such as wrongly connected power cables (exchanged + and -), or wrong connections between the button electrodes of the monitors and their preamplifiers, or corrupted power supply setting tables.

As automated beam position measurements are needed, nondestructive beam position monitors were considered in the prototype. Those monitors have four button electrodes to determine the horizontal and vertical position of the centre of the beam. It might happen that two cables are interchanged, giving a wrong beam position (figure 2). The prototype has a local expert system to scan each monitor with one steerer, using a simple algorithm to detect which cables are interchanged. Even if no cable is in its proper connector, the LE finds the real relations between the electrodes and the amplifier ports.The LE is sensitive to miscabled steerers, too.

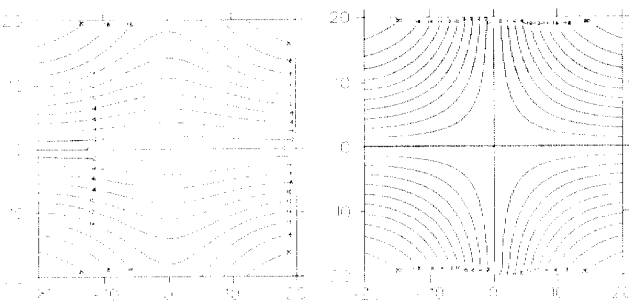


Figure 2. Contour plot of the measured vertical beam position as a function of true horizontal and vertical coordinates (all units in mm) for a correct monitor (left) and when the upper right and lower right button connections are exchanged (right). Note in the right figure, that if the beam is far left, the measurement is almost correct, while if the beam is to the right, even the sign of the measured value is wrong.

Since commissioning is not so much of an art as a step by step procedure, it can be quite handy encoded as an algorithm, without the use of special object oriented languages. We actually

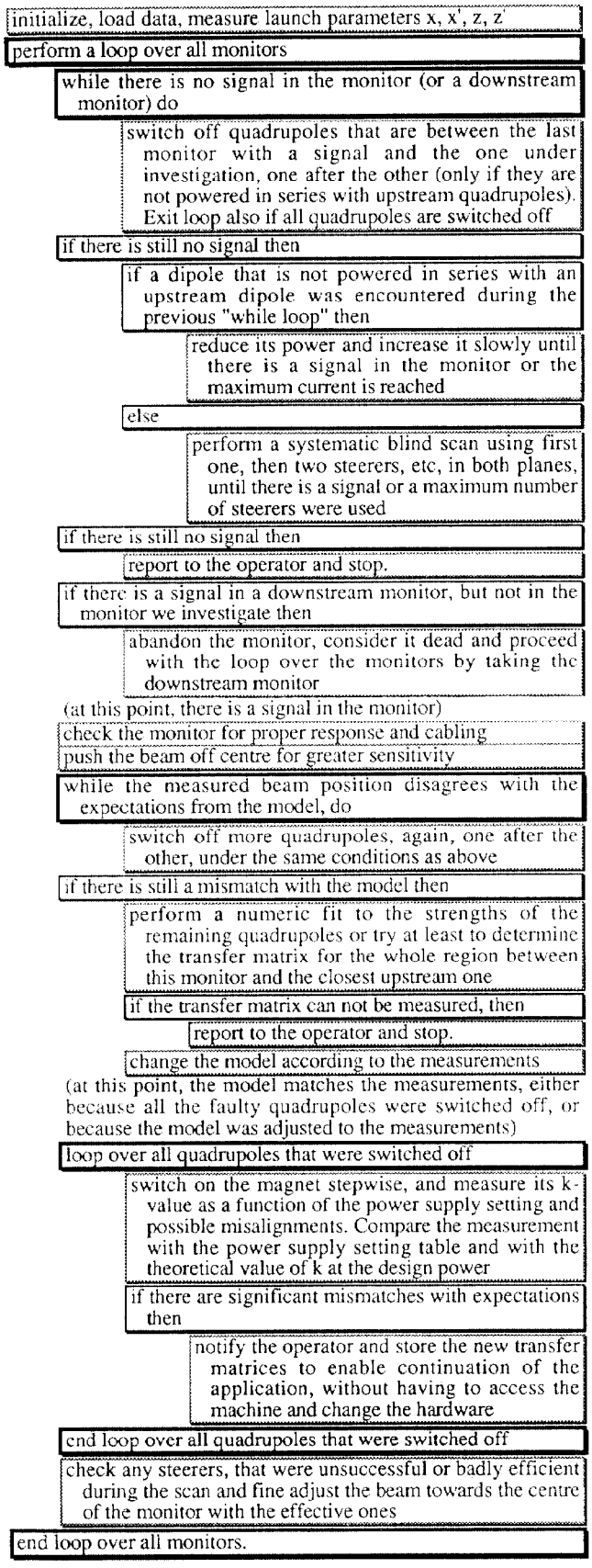


Figure 3. Flowchart diagram of the prototype's expert system.

tried to code the algorithm in Prolog[6], but with further elaboration of the tasks that had to be performed, the Prolog code had to be changed more and more to support the features of conventional procedural languages, so it was abandoned and rewritten using VAX FORTRAN.

The flowchart of the prototype is sketched in figure 3.The dipole magnets are checked by changing the power supply current. The quadrupoles are measured individually by displacing the beam in their vicinity and comparing it to the model. The quad current is varied, if the model cannot match reality. It is not important to measure the exact k-value but rather to see, whether k has a wrong sign or whether k varies with the power supply setting differently from expectations. Steerers are checked by how effectively they can displace the beam at a monitor.

The algorithm has been defined after extensive discussions with people that were involved in commissioning tasks. Each individual unit of the algorithm has been developed and tested separately. Testing and refinement of the complete prototype on a realistic simulation of the electron storage ring ELETTRA is under progress.

4. Conclusions

Judging from this experience, we believe that it is possible to incorporate enough expert knowledge, call it ES or not, for certain commissioning tasks even with FORTRAN. However, an appreciable amount of work does not go into the development of the software but rather in the refinement of it, by using it on the simulated machine. Unlike a numerical program, which can be either right or wrong, the ES is typically fault tolerant, working in certain cases even if it is inefficient in others.

Although the prototype is far from performing automated commissioning, it can be very helpful in commissioning or first turn steering, doing the basic checks instead of the operator enabling him to reason directly about less trivial problems.

The authors gratefully acknowledge discussions about commissioning issues with Dieter Einfeld, Rudolf Richter and Richard Walker.

References

[1] M. Plesko, *Towards the Automated Commissioning of ELETTRA*, ST/M-90/2,Sinctrone Trieste, February 1990.

[2] E. Malandain, P. Skarek, *An Expert System Project In the Accelerator Domain*, CERN/PS 87-54(CO), 1987.
E.Malandain, S. Pasinelli, P. Skarek, *Knowledge Engineering Methods for Accelerator Operation*, Proc. EPAC 88, p.1255, Rome, 1988.
D. Lager et al, *MAESTRO A Model and Expert System Tuning Resource for Operators*, Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Vancouver,1989.
H. Brand et al., *Coupling Models and Expert Systems On-Line to Accelerators*, Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Vancouver,1989.
D.P. Weygand, *Artificial Intelligence and Accelerator Control*, Proc. PAC 87,p.564, Washington D.C., 1987

[3] S.C. Clearwater, M.J. Lee, *Prototype Development of A Beam Line Expert System*, Proc. PAC 87, Washington D.C., 1987.
M. Lee et al., *Analysis of the Orbit Errors in the CERN Accelerators using Model Simulation*, SLAC-PUB-4411,1987.
M. Lee, S. Kleban, *SLC Beam Line Error Analysis Using A Model Based Expert System*, Proc. EPAC 88, p.767, Rome, 1988.

[4] N. Bongers et al., *Expertsystem for COSY Control*, Proc. EPAC 88, Rome, 1988.
L. Burczyk et al., *Ground Test Accelerator Control System Software*, Proc. EPAC 88, Rome, 1988.
P.A. Brown, D.E. Schulz, *The Development of an Expert System to Tune a Beam Line*, Proc. Int. Conf. on Accel. and Large Experimental Physics Control Systems, Vancouver,1989.

[5] J.A. Howell et al., *Control of a Negative-Ion Accelerator Source Using Neural Networks*, Proc. Int. Conf. on Accel. and Large Experimental Physics Control Systems, Vancouver,1989.

[6] M. Plesko, *Proposal for a Prototype for Automated First Turn Steering*, ST/M-TN-90/3, Sincrotrone Trieste, April 1990.