

EE



C1



CERN PS 91-15 (OP)

April 1991

CM-P00059457

Integration of Generalized KB-Systems in Process Control and Diagnosis

J. Fuchs
P. Skarek
L. Varga*)
E. Wildner-Malandain

Abstract

We present a novel method to create hypotheses centered diagnostic systems which integrate control systems, databases and expert systems. Several instances of this framework permit information sharing in a multi-domain environment, taking advantage of cooperation between knowledge based systems. The modelling techniques, the chosen solution and the implementation questions for the CERN PS accelerators will be presented. Issues of data acquisition and treatment in view of reasoning about qualitative values and the combination of traditional computational methods with knowledge-based techniques will be covered.

We discuss ideas how to make different subsystems cooperate to solve the common goal of diagnosis. These ideas have been partly developed within the framework of our ESPRIT-2 collaboration, ARCHON, P2256 on architectures for cooperating heterogeneous on-line systems.

Invited paper for the "SEAS" conference
Lausanne, April 8-13, 1991

Geneva, Switzerland

*) on leave from CRIP, Budapest, Hungary

Table of Contents

1. Introduction	1
2. The Expert system shell	1
Hypotheses treatment	1
3. Data acquisition and uncertainty	4
4. Cooperation between Expert Systems	6
5. Status Report of the CERN PS Application	8
BEDES - The beam diagnostic Expert System	8
Cooperation between BEDES and CODES	10
6. Conclusion	11
References	12

1. Introduction

New techniques are presented for monitoring, diagnosing and controlling industrial processes. They are centered around the notion of hypothesis creation and hypothesis treatment where a hypothesis is seen as a "package of knowledge" expressing a suspicion and presenting all necessary data to verify, deny or expand the suspicion. During the diagnosis process these "knowledge sources" are collected in an agenda and can also be moved around in a distributed system of cooperating expert systems.

These methods have been applied to the CERN Proton Synchrotron complex which comprises nine particle accelerators which are controlled through a large computer network of about 25 minicomputers and 150 microcomputers, interfaced by CAMAC to the physical components of the accelerators. Two expert system prototypes have been developed. One of them, the beam diagnostic ES (BEDES) [1] works on the process level and is meant to help the operators in running the accelerators. The other, the controls diagnostic ES (CODES) [2] should assist the fault finding team in trouble-shooting problems in hardware and software of the computer control system and its electronic interface.

To automate the two expert systems as much as possible and so as not to ask the users too many questions it was essential to link them directly to the computer network of the control system. They can now acquire all the necessary data and run simple tests directly.

The basic structural domain data on all modules of the control system (and later on operational archives for the setpoint values for the accelerators) are contained in a large database, ORACLE on a mainframe. For maintenance reasons and not to duplicate information we considered it essential to link the expert systems on-line to that database [3].

The paper is organized in the following way: In chapter 2 the shell - written in KEE - is discussed which provides the general framework for the creation and treatment of hypotheses, including some considerations on qualitative reasoning. Issues of data acquisition and uncertainty in measurements are dealt with in chapter 3.

Chapter 4 discusses ideas how to make different subsystems cooperate to solve the common goal of diagnosis. Chapter 5 is a status report on our implementation in the accelerator domain.

2. The Expert System Shell

2.1 Hypotheses Treatment

The expert system shell developed in KEE includes not only the reasoning mechanism but also automatic knowledge base creation, transparent access to the accelerator control system and to the ORACLE database and a general mechanism to convert measured values into qualitative terms etc. This has been described in details elsewhere [1],[2],[3].

Here we concentrate mainly on the central reasoning part of the diagnostic process: how to deduce a fault from apparent or suspected discrepancies between the real world and the model of the system.

The real world to be diagnosed may be described by observations on all kinds of levels of abstraction: from suspicions on a high - usually behavioural - level, often called symptoms, down to simple sensor readings of some measurement data, sometimes very well defined.

On the other side there is the knowledge represented which defines how the system should behave correctly and/or a representation of how faults manifest themselves. Any difference between the real world and the model - i.e. between observations and expectations - is a "discrepancy".

A diagnosis is usually seen in the following way (e.g. [4],[5]): The starting point is the presence of a symptom (a discrepancy). From these initial discrepancies one creates hypotheses about faulty states of the system. These generated hypotheses have now to be tested - verified, rejected or refined - by making some more tests on the system components, looking at other discrepancies and reasoning about them, i.e. deriving other statements (possibly hypotheses) from assumptions and facts.

In our work we see the concept of a hypothesis differently: not as something secondary, derived from the primary symptoms or discrepancies. We see a hypothesis as the primary concept for diagnosis and monitoring, namely as representing a "package of knowledge", as a knowledge source (in slight analogy to the term used in the blackboard paradigm).

A hypothesis consists essentially of:

- A statement that a system or system component might be in a wrong state or could go wrong.
- All necessary information about how to verify this statement (the necessary inference steps).

This may include - but very well separated from each other - procedural steps (e.g. data acquisition to be made from the control system) and declarative data, like rules. The procedural steps can be seen as a filter towards the process, transforming and preparing measured data for generic inferencing as far as possible.

The declarative data covers to a large extent the diagnostic expertise.

In our implementation these hypotheses are collected in an agenda and re-arranged and worked-on according to certain criteria. If one puts a hypothesis onto this agenda without having a discrepancy detected before, one is monitoring the system or a particular component; if discrepancies exist, one is automatically diagnosing. Surveillance (monitoring) is seen as a continuous diagnosis of possible faults, working with hypotheses. A hypothesis is just "any suspicion" to be checked, justified or not. And while there are hypotheses, i.e. knowledge sources active - either for the monitoring task or the diagnostic task - they might put other hypotheses onto the agenda, which all have to be verified, denied or refined.

The two diagnostic expert systems BEDES and CODES can run continuously. In the monitoring phase hypotheses about the possibility that something basic is wrong in the whole system are put repeatedly on the agenda. For BEDES that can be "injection efficiency for the beam has gone down", and for CODES "some alarm data have arrived from the control system". Both expert systems monitor the corresponding data and in the case of discrepancies a diagnosis is started automatically, using the same hypotheses mechanism.

This unified view of the notion of a hypothesis relieved us quite a lot from the conceptual difficulties to define and distinguish "initial data", "observations", "symptoms", "discrepancies" etc., and provides a straightforward initialization process.

The basic step of our inference procedure is an inference cycle, which corresponds to the complete treatment of one hypothesis taken from the agenda.

The inference procedure cycles over each hypothesis in the agenda consecutively. The hypothesis treated points to a "suspected entity" in the domain knowledge base and this frame unit contains in its slots the corresponding methods to be run and the rulegroups to be forward chained.

As already mentioned, the methods take care of the interface to the process. They deal with data acquisition and related problems of uncertain information, with filtering of discrepancies. They also read structural data from the ORACLE database and load archives as reference values when necessary etc. The rules contain the domain knowledge how to treat a particular hypothesis.

A hypothesis can be rejected by either simply classifying it as not confirmed and storing it for later consideration (when new facts have been asserted to the domain knowledge base), or completely different new hypotheses can be created (change of focus of attention). A hypothesis can be confirmed (accepted) and either conclude immediately a fault directly or suggest a refinement by creating more specific hypotheses.

The shell contains also the mechanisms for exchanging hypotheses between different cooperating expert systems (written with the same shell), e.g. between BEDES and CODES. This is described later on.

2.2 Integrating Physics into our Shell

To be able to describe the chain between the controllers and the the resulting effect on the beam parameters, we use an influence net to describe the parameters and the relations between parameters. There are simple pointers just to indicate which parameter is influenced by which parameter or controller. The controller, for our view of the system, is the beginning of the chain and thus influenced by nothing. For diagnosis we work "backwards" in this net, from the parameter showing a deviation, via the parameters influencing it, ending up with the controller.

This is done with our shell in the following way: A hypothesis is used to monitor a beam parameter. If a deviation is detected, the procedures and rules for this hypothesis will use the information about the interconnections in the net to create new hypotheses. New suspicious elements are those which are in a "influences" relation to the initial hypothesis. The new hypotheses are treated in

the same way; rules are fired, and if there are pointers to influencing elements, hypotheses for these elements are put on the agenda. This implies that basically all the information of the topology of the influence net is exploited by the rules.

A beam parameter is normally influenced by several controllers or beam parameters or related, in a circular way via some other parameters, to itself. To avoid pointing back to the same element and creating new hypotheses in a repeated way, the output for the pointer to other elements is marked and never activated again. This does not exclude the creation of a hypothesis through another path, but this is perfectly in order. This fact increases the probability of our hypothesis or suspicion.

How are now all these hypotheses, created from the net, organised to be handled in a reasonably efficient way? As already mentioned we have some measure of the plausibility of a suspicion. This is used to rearrange the order in which the hypotheses are treated. To be able to make this ranking, the physics between the parameters have to be taken into account. For instance, if two parameters influence the parameter for which a problem is detected, one has to see which of the two influencing parameters has the greatest effect for these specific operating conditions. If the symptom is a large deviation it is clear that the parameter with a little influence is less suspect. The same hypothesis coming from different paths in the net increase the probability for this hypothesis.

When a hypothesis is tested we can use the net in simulation mode to simulate effects of parameter changes.

3. Data Acquisition and Uncertainty

The data necessary for the diagnosis is not available locally on the same computers as the expert system. However, the expert system is linked via an Ethernet network and a special remote execution protocol to the front end computers of the control system.

The first step of a data acquisition is merely a remote execution of an acquisition command on a control front end computer (FEC), where the results are sent back through the network. This provides the system with a raw data stream which has to be further analyzed to be used by the diagnostic system. This access is implemented procedurally to reduce the amount of network traffic, whereas the reasoning is implemented declarative knowledge using rules. The procedures manage a "buffering" philosophy for on-line data, which means that per inference cycle data for an element is acquired once only, independently of how many times rules are fired.

The following figure shows the different possible fluctuations of the data acquisition until the value can be used for reasoning in the expert system.

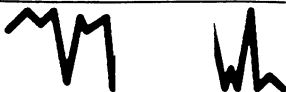


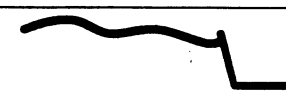

Level	Possible Fluctuations	Example
ES	(hopefully) none	
Data Transfer	Network Problem	
Instrumentation	Acquisition Errors	
	Sensor Variance	
Process	Minor fluctuations	
	Drifts and breakdowns	

Fig.1: Levels of data acquisition and possible faults

The example column shows a possible scenario of a process signal as a function of time.

At this point we have to state clearly that the interest of the expert system are the drifts and the breakdowns in the process (highlighted in figure 1). The aim of the acquisition process is to minimize the error in this chain.

Drifts in this context can be very slow (in the order of days) and have to be detected in order to optimize the process, whereas breakdowns are rather abrupt and are the starting point for a diagnosis.

Since the process of our application is a real (physical) process with many elements involved there are inevitably some fluctuations due to minor irregularities in those equipments.

The sensors used for the process observation are usually analog devices, so that the result obtained does not exactly correspond to the process value. We decided to model this behaviour with standard statistical methods, i.e. we performed an extensive statistical analysis on real data. This analysis was done off-line and the results are used more or less as constants in our data acquisition. This use of off-line statistics could imply as well the use of uncertainty reasoning, where a certain probability is associated to the data used which affect the reliability of the result.

The next possible source of wrong data can be a wrong acquisition. That means that single values are entirely wrong, without any relation to the process. This can have several reasons; one of them is that the instrumentation equipment is not exclusively dedicated to the measurement task and consequently did not entirely fulfill the real-time constraints.

Those errors are usually unique and there are two major possible ways to check on this data: For some equipment the range of these data is known. So there is a simple check if the acquired value is within these limits.

If this is not possible a simple consistency check is performed on the acquired data. This is done by a very simple on-line statistics, mainly by making usually two successive acquisitions and compare the two resulting vectors of values. If the difference is bigger than an admitted tolerance a third acquisition is made and the two closest vectors are taken to calculate the mean value (method of nearest neighbors). This method of having a simple on-line statistics has as major drawback a considerable increase of processing time.

The last (but unfortunately not least) source of problems may be a network breakdown, which disables the access to the control system entirely. This error is quite easily detectable (network time-out, empty return string...) and there are two basic reactions to this kind of error. In certain cases the reasoning process can continue on incomplete data, but in certain cases this is not possible and we try to reason on the error encountered, possibly concluding in a fault.

4. Cooperation between Expert Systems

This chapter is partly based on ideas developed in the framework of our ESPRIT collaboration ARCHON, P2256 [6]. The project consists of the following members: KRUPP ATLAS ELEKTRONIK GMBH, AMBER, CERN, CNRG-NTUA ATHENS, ELECTRICITY RESEARCH & DEVELOPMENT CENTRE, FRAMENTEC, FWI UNIVERSITY OF AMSTERDAM, IBERDUERO, IRIDIA UNIVERSITE LIBRE BRUXELLES, JRC ISPRA, LABEIN, VOLMAC, UNIVERSITY OF PORTO, and QUEEN MARY AND WESTFIELD COLLEGE.

Particularly ideas about the modelling of cooperating semi-autonomous agents [7] have been fruitful for the development of our implementation to make BEDES and CODES cooperating.

Expert systems designed to diagnose the same system from different point of view can improve their performance by communicating with each other and execute the diagnosis job in a distributed problem solving way. In our case the expert systems have different domain knowledge and they are responsible for different levels of the accelerator, but they can help each other by sending task requests and information.

The locally generated tasks of our agents are hypotheses. Since our expert systems use the same shell (Chapter 2), the hypotheses can be directly sent to other agents. In our system the format of the work request is a hypothesis of the shell. Important facts acquired from the control system or concluded during diagnostic reasoning are attached to the hypotheses, so the hypotheses received by the expert systems are not only work requests, but may also contain this information.

The knowledge representations in the expert systems are somewhat different, so hypotheses are transformed to the internal reasoning mechanism of the receiving agent. A diagnostic agent does not depend on other agents. Agents do

not wait for a result, however when a hypothesis is treated and a result is generated, the result is sent back to the agent, which issued the hypothesis.

The expert system agents in our application have the same structure: both of them consists of an expert system written with the same shell, a knowledge base for cooperation, and a monitor.

The cooperation knowledge base contains an agent acquaintance model and a self model. Acquaintances are modelled by the name, the address, and the simplified representation of the domain knowledge of the other agent. In the agent acquaintance model the domain knowledge is defined by the list of objects the other agent can reason about. A hypothesis can be sent to the other agent if the hypothesis refers to an object on this list.

The self model of an agent contains the name of the agent and the list of the hypotheses sent. Sent hypotheses are recorded in order to avoid repeated sending of the same hypothesis and to be able to distinguish if an arriving hypothesis is a new work request or an answer for a previously sent hypothesis. The agents do not negotiate, we assume, that the agent is able to treat the received hypotheses.

An agent makes decisions: when and which hypotheses to send to other agents. In our system the decision making of agents is independent of the diagnostic rule application. The hypotheses are not sent by the expert system, but by the monitor of the agent. The agenda of the expert system is examined at the start of every inference cycle by the monitor and if on the agenda there is a hypothesis, which can be of interest to the other agent, then the hypothesis is sent to it. However if the hypothesis can be treated by the sending agent as well, then the agent will keep the hypothesis on its agenda, otherwise the hypothesis is deleted from the agenda.

At the start of every inference cycle it is also examined if messages have arrived from other agents. If a hypothesis arrives, then the agent checks if it is a new hypothesis or it was sent by this agent before and it came back as an answer. If it is a new hypothesis, then a new hypothesis is created at this agent, otherwise the existing hypothesis is updated according to the newly arrived information and put back to the agenda.

The above described techniques allow the agents to exchange hypotheses and results. If an expert system is unable to continue its work, because a hypothesis is not in its domain knowledge or the expert system recognizes, that a hypothesis can be treated by another system, then it can send the hypothesis to another expert system. The other system treats the hypothesis and sends back the result, and thus a speedup can be achieved. This type of message exchange gives a subroutine like task sharing.

Expert systems can really cooperate if they do meta-reasoning on the hypotheses in their own agenda. There is real cooperation if one expert system can reduce its search space by dropping hypotheses from its agenda when a new hypothesis arrives from another agent or if one expert system can produce a better result by combining results from different agents working on different conceptual level. In our system the tool for meta-reasoning is a function to rearrange the agenda, which can change the order of hypothesis treatment and discard hypotheses from the agenda of the expert system.

5. Status Report of the CERN PS Application

In the CERN PS accelerator complex there are similar diagnostic expert systems BEDES and CODES for diagnosis and monitoring, but they have different domain knowledge and they are responsible for different levels of the accelerator. They communicate with each other by sending hypotheses and execute their tasks in a distributed problem solving way.

The domain of BEDES is the process of the injection line of the booster accelerator, whereas CODES looks at the control system of the same part of the machine.

The expert systems BEDES and CODES are implemented on two SYMBOLICS LISP machines using the expert system development tool KEE 3.1. The machines share the same file system. Currently there are only two agents, but the cooperation scheme was designed to be able to handle several agents.

5.1 BEDES - The Beam Diagnostic Expert System

The Expert system for the Beam Diagnostic (BEDES) has a set of production rules, which is complete for the injection line. It is capable of testing every element of the accelerator in question at least in a very basic way. This allows the user (operator) to concentrate more on specialized questions since it does the scanning of the system in a general way.

For some complicated elements specialized rules have been developed to integrate expert knowledge as far as possible into the system. There are as well complex routines to test some of the elements in a more detailed way by considering the information of different hypotheses. This part is the most probable candidate for further enhancements of the systems.

User Interface

The interface for the expert system is built with the facilities of the KEE software package, particularly active images and panels. KEE uses Common Windows as graphical environment.

There are two different types of interfaces to the BEDES expert system. Those interfaces have two different aims, namely

- supporting the development team in writing and testing the application and
- providing a 'foolproof' interface for the operational use of the system.

The development interface is very well described in [2] and uses the full power of the interfacing capabilities of KEE. It provides a maximum of information and a minimum of consistency checking for the interactions between the system and the user. This was the reason to design a second interface for the everyday use of the system.

Since the expert system is supposed to run more or less independently of any intervention - it is only to be used in case of trouble, for the rest of the time it should only monitor - one of the design principles was to reduce the interaction between the user (the operator) and the system as much as possible. The interface

should as well cope with possible errors from the user and problems encountered during run-time without needing the intervention of an expert.

The first point was to limit the access for the user: most of the capabilities of the KEE-interface have been disabled, and the only actions allowed for the end-user have to be executed through a menu. This menu allows basically three types of actions:

- Start and Stop of the Expert system, whereas the stop will only be effective after the complete execution of an inference cycle (i.e. no 'emergency stop' available).
- Making archives from the actual situation of the accelerator (see below).
- Starting separately the steering or the optimization procedures for the actual operation. (This option is disabled while the expert system is running to avoid interferences between the processes.)

The second aim - to provide enough information for the user - was obtained by several displays:

- An event history of all the actions undertaken by the expert system.
- Several windows indicating errors during run-time, which are usually not visible but are displayed for a fixed amount of time when such an error is encountered (e.g., Problem with network access, access to an equipment...).

If the monitoring triggers on a discrepancy between the desired operational values of the accelerator and the observed values this will be displayed and the different elements which are then considered and tested are shown in a separate window.

If the system concludes a fault a detailed information about this fault is displayed in a dedicated window.

The Steering and Optimization

The expert system is capable to optimize the injection of the accelerator automatically [8]. The injection is divided into two different sections, the 'Common Injection Line' and the injection process in the four accelerator rings.

The quality of the steering in the common injection line is measured by the deviation of the beam to the ideal trajectory. This deviation can be measured with the means of pickups, each one logically related to a dipole. Given this deviation it is possible to estimate a correction for the injection dipoles to approach the ideal line.

After a validation phase for the corrections (to allow the elements to stabilize on the new control values) the system tests if the correction had the desired effect and the deviation from the ideal trajectory is small enough. If not, corrections are initiated.

There are no devices easily accessible to measure the trajectory of the beam during the injection in the rings. For this reason a different approach had to be taken. For the injection a quality measure has been defined, which has to be optimized.

The optimization is done by a hill-climbing method, which effectuates controlled perturbations to the process around the operation point in order to maximize the performance. That means that the actual optimization is done in a step by step approach. The control values of the correction elements are changed by a certain step. After the change the influence of this change on the output is measured. Depending on the observed changes, compared to the initial quality, a decision is taken if the modification executed for these elements will be adapted or not.

Archives

Archives in the sense of our expert system are instantaneous and complete pictures of the state of the accelerator with all its elements and observations which are possible with an automatic acquisition.

Two types of archives are considered within the system:

- The complete set of values of all the elements which can be remotely controlled. These are the desired status of the element and the acquired status as well as the desired control value and the acquired value of the elements.
- The "observables" of the system, which are indications of the quality of the actual operation and are basically the properties of the output of the accelerator which have to fulfill certain conditions. (e.g., intensity of the beam).

There are eight basically different operational modes, and each of them is still divided into three different subclasses (for a low, middle and high intensity). If the user judges the actual situation as good, he can produce an archive, meaning that he will record all the above mentioned characteristics for the given moment.

During the monitoring and diagnosis phase these values are taken as references, whereas the classification and the assignment of the appropriate archive is done automatically. Any major deviation from the reference values is taken as the starting point for a further diagnosis.

5.2 Cooperation between BEDES and CODES

If the beam is not correct, then the expert system agent BEDES - responsible for the beam - tries to find the reason for the insufficient beam. If in the problem solving process BEDES finds that the problem might be in the control system, then BEDES activates the other expert system agent CODES responsible for the control system by sending hypotheses to it. CODES can also be activated manually.

All agents load the same cooperation knowledge base, but they initialize it using different initialization functions. The name of the initialization function is defined in a special slot for each agent [9].

The monitors of the agents are also the same. The monitor is implemented by LISP functions. Hypothesis sending and receiving is synchronized to the activities of the expert system, the LISP functions of the monitor are called at the beginning of each inference cycle.

Since the expert systems share the same file system, it was easy to implement message passing through files as mailboxes. Hypotheses sent from one agent to

the other are written into files. The format of a message in this file is a list similar to the format of the INIT files used to link the expert systems to the ORACLE database [3].

When hypotheses are received, they are translated by a LISP function. The name of the translation function is stored in a slot of the self model of the receiving agent. When a hypothesis arrives from BEDES to CODES, it is translated to a general monitoring hypothesis ("some alarm data have arrived") and the relevant information is put into this hypothesis. Later, when this general hypothesis is selected and treated by CODES, a part of the model of the control system is instantiated and new hypotheses are created based on the information stored in the general hypothesis.

6. Conclusion

Our general definition of what is an hypothesis makes it possible to unify the view and the implementation of tasks which have been treated by different systems up to now. The monitoring task and the diagnostic task are seen as basically the same action, both linked to a hypothesis which suspects an element of the observed system or the system as a whole. The monitoring is considered as repeated verification of a suspicion which may detect a discrepancy or not.

Control tasks are also part of the hypothesis checking, since our notion of hypothesis includes all the information necessary to verify or deny the hypothesis.

This representation of a hypothesis is also well adapted to be shared in a distributed system, where tasks are shared or exchanged within a community of agents.

References

- [1] E.Malandain, "An Expert System in the Accelerator Domain", paper invited at the Internatl. Workshop on Softw.Engin., AI and Expert Systems for High Energy and Nuclear Physics, March 19-24, 1990, Lyon, France and CERN/PS 90-45 (OP).
- [2] P.Skarek, E.Malandain, S. Pasinelli, I.Alarcon, A Fault Diagnosis Expert System for CERN using KEE, SEAS Spring Meeting, Davos, 18 - 22. 4. 1988 and CERN/PS 88-12 (CO).
- [3] E.Malandain, P.Skarek, Linking a Prototype Expert System to an ORACLE Database, IASTED Int. Conf. EXPERT SYSTEMS, Theory and Application, June 26-28, 1989, Zurich and CERN/PS 89-44 (CO/OP).
- [4] O.Raoult, Survey of diagnosis expert systems, in Knowledge Based Systems for Test and Diagnosis, pp.153-167, G.Saucier, A.Ambler, and M.A.Breauer (eds.), North Holland, IFIP, 1989.
- [5] R.Davis, W.Hamscher, Model-based Reasoning: Troubleshooting, pp.297-346, in Exploring Artificial Intelligence (AAAI Survey Talks), H.E.Strobe (ed.), Morgan Kaufmann, San Mateo, CA, 1988.
- [6] ARCHON - Architecture for Cooperating Heterogeneous On-Line Systems, Status Report (compiled by S.Becker, J.Ehlers), presented at ESPRIT CONFERENCE WEEK 1990, and Technical Report No.9 (ESPRIT-Project 2256 ARCHON), 9-1990.
- [7] N.R.Jennings, C.Roda, E.H.Mamdani, Cooperation in a Multi-Agent Environment, Technical Report No.3 (ESPRIT-Project 2256 ARCHON), 2-1990.
- [8] J.Fuchs, S.Pasinelli: Automatic Optimization for the Booster Injection Line, PS/OP/Note 90-91, CERN, Geneva
- [9] L.Varga: Cooperation between the Two Diagnostic Expert Systems BEDES and CODES, PS/CO/WP 91-02, CERN, Geneva