MULTI-AGENT COOPERATION FOR PARTICLE ACCELERATOR CONTROL

Paul Skarek, László Zsolt Varga[1]

CERN, CH-1211 Geneva 23, Switzerland
Phone: +41-22-767 3522, Fax: +41-22-767 9145,
email: paul.skarek@cern.ch

ABSTRACT

We present practical investigations in a real industrial controls environment for justifying theoretical DAI (Distributed Artificial Intelligence) results, and we discuss theoretical aspects of practical investigations for accelerator control and operation. A generalized hypothesis is introduced, based on a unified view of control, monitoring, diagnosis, maintenance and repair tasks leading to a general method of cooperation for expert systems by exchanging hypotheses. This has been tested for task and result sharing cooperation scenarios. Generalized hypotheses also allow us to treat the repetitive diagnosis-recovery cycle as task sharing cooperation. Problems with such a loop or even recursive calls between the different agents are discussed.

Keywords: Distributed AI, industrial control, diagnosis, cooperating expert systems, multi-agent systems.

1. INTRODUCTION

Quite exhaustive theoretical studies exist in DAI, but it seems that there is not enough feedback from practice. In this paper we present practical investigations for applying and justifying the theoretical DAI results in a real industrial controls environment, and, conversely, we discuss the theoretical aspects of practical findings in these applied investigations made for accelerator control and operation. The results presented here are partly based on the research carried out at CERN during the ESPRIT-II Project ARCHON™ [1].

The CERN Proton Synchrotron accelerator environment, the motivation to apply DAI methods and, related to it, the results of the ESPRIT-II Project ARCHON on ARchitecture for Cooperating Heterogeneous ON-line systems, are given in chapter 2. Chapter 3 describes different cooperation techniques applied to accelerator control and accelerator operation, some cooperation scenarios between diagnostic agents and examples taken from a control program called SETUP [2] to bring the accelerator into working condition after a shutdown period or after some component breakdowns. The analysis made in chapter 4 leads to a general method of exchanging hypotheses by cooperating expert systems which has been tested. This

1. on leave from KFKI-MSZKI, Budapest, POB 49, 1525 Hungary, and supported by OTKA F4064

result sharing cooperation amounts to exchanging diagnostic knowledge at different levels (partial results) and of different certainty. The definition of a generalized hypothesis allows for a unified view of control, monitoring, diagnosis, maintenance and repair tasks such that agents implementing these tasks simply exchange hypotheses. This permits us to treat the repetitive diagnosis-recovery cycle as task sharing cooperation. Problems with such a loop or even recursive calls between the different agents are discussed. From another area of accelerator control concerning the different aspects of diagnosing timing faults we show that feedback from one diagnostic agent to another can be considered as similar to machine learning in a multi-agent system by rule compilation from simulation runs in another agent.

2. BACKGROUND

CERN is a European research institute, financed by 19 member states and employing some 3000 staff members. In addition, 5000 visiting physicists, engineers and computer experts from nearly 400 research centres and universities in 40 countries use its facilities: accelerators and experimental areas. Particle accelerators are necessary to provide physicists with beams for their experiments. The CERN accelerator complex is one of the world's most sophisticated high energy research tools. The CERN Proton Synchrotron (PS) is the heart of CERN's accelerators and experimental facilities, and acts also as injector for CERN's bigger accelerators, the Super Proton Synchrotron and the huge LEP (Large Electron Positron rings). Accelerator operation and maintaining the underlying control system are complex tasks and difficult to survey. The necessary knowledge is naturally distributed, as is the control system which contains many systems to solve common problems. These facts suggest the application of DAI methods and in particular solutions emerging from multi-agent cooperation.

Given the need to apply DAI techniques, CERN joined the ARCHON project as an application partner providing a large accelerator control system and two expert systems as a test-bed for the development and evaluation of the methodologies and software produced by this collaboration. ARCHON [3], [4] can be considered as a cooperation shell for combining semi-autonomous systems to work on an implicitly defined common goal, in our case, the correctly operating accelerator. In ARCHON, and this was one of the very important design objectives, the different agents can be pre-existing systems. Problems, insights and experiences gained whilst deploying ARCHON technology in real applications with the two on-line running examples (respectively IBERDROLA, an electricity utility in Spain, and CERN) are described in [5]. Details of applying the ARCHON technology can also be found in [6] and [7].

The ARCHON system architecture [3] defines an agent as an independent system (Intelligent System) plus its "ARCHON Layer", the cooperation layer to be added. The Intelligent Systems are systems dealing with a certain field of application. They are first of all expert systems, but other systems like database and control systems are also included: so that one could talk about Industrial Systems in general.

The ARCHON Layers of each agent provide the functionality for cooperation, so that they can communicate to coordinate their tasks if necessary, similarly to human operators in a control room who decide when and how to cooperate and take the necessary actions. Via these ARCHON Layers the Intelligent Systems now control not only themselves but also the way they interact with each other: this is the essence of cooperation provided by an ARCHON Layer. It is not only the distribution of data and control, but rather the distribution of the control of all data available.

Each ARCHON Layer has two I/O channels according to its twofold functionality: controlling the Intelligent System and controlling cooperation, i.e. one connecting to the underlying domain system and the other to communicate to the other agents in the community. The ARCHON Layer consists of several modules like the one for the task control and status check of the Intelligent System and another one that reasons about other agents and decides when and how to cooperate, and supervises cooperation by situation assessment and planning. The acquaintance and self models are also part of the ARCHON Layer.

3. MULTI-AGENT COOPERATION SCENARIOS

Cooperating Diagnostic Expert Systems

The two diagnostic expert systems written at CERN and then used for the ARCHON test-bed were CODES [8], which stands for COntrol system Diagnosis Expert System and BEDES [9], [10] for BEam Diagnosis Expert System. The first aspect in running an accelerator is addressed by CODES and related to fault finding and repair tasks in the complex control system of these accelerators. The main problem lies in the complexity and the implicit connectivity of the processes to be diagnosed, with its numerous interconnected hardware and software modules. CODES was implemented using the expert system toolkit KEE™ from IntelliCorp to write a generic shell for diagnostic reasoning and includes device-centred model-based reasoning and on-line access to a database describing the parts and modules of the control system and their connectivity. BEDES addresses another aspect of the work done in the control room of an accelerator, the work of the operators to set up a certain particle beam, keep stable running conditions or change to a different kind of beam. BEDES was written to help the operators to detect and to treat malfunctions in the transfer line between LINAC II and the PS Booster Accelerator (PSB)[1], and in the process to inject the beam into the PSB.

The operators act upon the accelerator exclusively via the control system. Different sections of the accelerator are controlled from different general purpose consoles (work stations) in the same control room. From here stems the idea of introducing the notion of cooperation between expert systems and cooperating heterogeneous systems in general. Just as different operators coordinate their work on a common goal by cooperative communication, the software in the different ARCHON Layers provides this cooperation.

A general method has been developed at CERN for cooperation which was applied to expert systems that create hypothesis trees for their diagnosis. This method uses essentially result sharing cooperation, by which it speeds up the diagnostic process and gives more detailed results. The method works better if hypotheses are in a tree structure, because the additional information of relations between the hypotheses provides more information for cooperation.

The two expert systems use the same diagnostic shell: Hypotheses are created by some high level symptoms, error codes coming from the control system, by the verification method of a hypothesis just being evaluated, etc. They are kept in an agenda which is aware of a tree structure of the hypotheses and they are selected from this agenda by certain criteria related to the importance of a hypothesis to be able to lead to a result. Child hypotheses in the tree structure usually correspond to subparts and more specific fault suspicions. In more detail, a *hypothesis* is an object which comprises, among other attributes, the following principal information: *Suspected Entity*: which object in the control system is suspected (e.g. a focusing quadrupole); *State of the Entity*: what is the suspected erroneous state (e.g. switched off); and *Verification Method*: how to verify or abandon this particular hypothesis. The verification method includes both procedural data (e.g. to be collected from the database or from the process via the control system itself), and declarative data (i.e. the diagnostic rules).

The cooperation is based on the exchange of hypotheses. This has the advantage that it fits naturally into the diagnostic structure of the expert systems and can be applied to existing expert systems - as shown in our case - with only slight modifications [11]. Although the hypothesis exchange as a means for cooperation was developed mainly for the cooperation of diagnostic expert systems and a larger applicability was not among the design goals like in the design of the Knowledge Query and Manipulation Language (KQML) and the Knowledge Interchange Format (KIF), hypothesis exchange proves to be a generally applicable method as we will see later in this paper.

---

1. The LINAC II is a linear accelerator which injects into the PS Booster, a pre-accelerator boosting the energy and the beam qualities before injecting the beam into the main PS accelerator.

The two expert system may cooperate in several ways. In a simple task sharing cooperation when BEDES finds something that indicates a possible error in the control system, it sends this indication to CODES. In a result sharing cooperation CODES runs ahead of BEDES and makes sure that BEDES can complete its diagnosis. The result of CODES, sent to BEDES, enables BEDES to arrive at a better or more detailed diagnosis. The result sharing cooperation is also possible in the other direction: repeated messages from BEDES redirect the attention of CODES which is working in parallel to BEDES in its expanded hypothesis tree. Details of these scenarios are described in [5], [11], and [12]. These scenarios have been implemented and tested as part of the ARCHON project.

The basic elements of this cooperation method are the following: sending hypothesis to each other based on the acquaintance models to check if they are of interest at all, translating received hypothesis to be understood by the receiving agent if necessary, inserting hypotheses into the agenda, and influencing the reasoning mechanism inside the expert systems by changing the "importance" of this hypothesis. The cooperating expert systems can give more details and more accurate results than the stand-alone systems which means that information can be presented to the system's operators in a more coherent and accurate manner. In stand-alone operation, BEDES can tell that the beam is lost and suspects the operator having set wrong values or used wrong archived values and then CODES can be asked if the control system is not faulty. The DAI system, on the other hand, can immediately present the information that the efficiency of the beam went down because a control module is stopped and a control value cannot be set.

Cooperation Between Accelerator Setup And Diagnosis

The SETUP is a collection of programs to bring an equipment or interface module or a collection of them into a correct working state. It is quite natural to think of the SETUP as an integrated part of a general fault finding system. In relation to a CODES-agent (CODES stands here for any diagnostic expert system for analysing control system faults) the SETUP would represent a "recovery"-agent: on finding an error, CODES can cooperate with the SETUP for error recovery, and - in the other direction - the SETUP agent, in case of errors in its recovery actions, can ask CODES for more details. One could even think of an extended arbitration via an ARCHON Layer of finding the most efficient recovery procedure if there is a dynamic choice between several possibilities.

There are also good reasons to include BEDES, the beam diagnosis expert system, in such a cooperation: as we have seen, BEDES provides a different, additional view of the diagnosis and therefore a more powerful diagnosis for the SETUP than CODES alone, and in some cases, BEDES needs directly recovery actions provided by a SETUP.

The diagnostic modules can cooperate with the SETUP modules in other ways as well. Besides helping the diagnostic module with the capability of recovery, the SETUP module can contribute to the diagnosis process of the diagnostic module. The diagnosis requires some tests to be executed (in the verification methods of the hypotheses). In a simple case it reads an alarm code from the appropriate equipment driver, but by calling the SETUP the diagnostic module can execute complex tests and receive high level information. These scenarios have been proposed as improvements to the present SETUP system [13].

Cooperation Between Different Aspects Of Timing Diagnosis

The timing system in accelerator control has two main functions: to provide real-time control independent of the operator, and the time-sharing and sequencing of different operation modes, i.e. different beams through the chained accelerators. The timing diagnosis can be related to three levels of the timing system: (1) Programming the schedule of operation modes mentioned above, (2) generating the master timing, and (3) the derived timings on different levels and generated in the various timing modules of the control system, depending on the changing operation modes.

On the operation modes programming level the diagnostic and checking system (as described in the paper I. Campos, J. Lewis, P. Skarek, L.Z. Varga; Rule-Based Consultant for Accelerator Beam Scheduling, to be submitted to: ICALEPS'95, Chicago; 1995.) should detect if the schedule can be executed by the accelerators. On the master timing level the diagnostic system should help to detect why the master timing generator select certain operation modes, because this is not uniquely determined by the planned schedule, but also by the state of the equipment. On the detailed timing level the diagnostic system should be able to detect whether the correct signals are generated as it is requested by the master timing generator.

The different aspects of timing diagnosis are interrelated. Some of the restrictions expressed in the rules of the operation modes programming level are partially based on the results of the master timing and detailed timing level analysis. For example if the master timing generator can never select a certain combination in a schedule then this combination can be ruled out immediately at the programming level. Automating the feedback from the lower level diagnostic agents to the operation modes programming level is similar to machine learning.

The agent responsible for the programming level instructs the master timing level to execute a certain schedule and assigns to this schedule a credibility factor (in the sense: suitable for execution). This credibility factor is - let us say - 1 in the beginning. If the master timing level is unable to execute parts of the schedule, it reports this to the programming level which reduces the factor each time it detects a problem, and if it goes below a certain level, the programming level can take action to add a rule to the programming level diagnostic system to inhibit this type of schedule. The programming level must be intelligent enough to identify those characteristics of the schedule which cause the master timing level to fail. This identification can be done by comparing the schedule that failed several times with accepted schedules, and by finding those characteristics which are present only in a schedule which fails.

Currently the first two agents (operation modes programming and master timing simulation) are implemented and running; the proposed feedback and the detailed timing diagnosis are in the design stage.

The reason for this rule generation from numerous simulation runs is an argument for speed since the master timing level would detect the impossibility anyway, but the simulation can take a very long time. If the rules in the first agent are rather complete, the simulation run can become superfluous.

4. ANALYSIS AND GENERALIZATION

We have seen that diagnostic expert systems were made to cooperate by hypothesis exchange. Now we are going to extend our view of hypotheses in such a way that the major tasks in accelerator control can be related to these hypotheses to constitute a sound basis for cooperation.

Generalized Hypotheses

We define a generalized hypothesis as a triplet consisting of an *object*, the *suspected element*, with a certain fault attribute (what is suspected) and a *verification method* or a reference to it (what to do and how to proceed in the diagnosis). In diagnosis one usually talks about the Hypothesis-and-Test strategy. We have combined these activities into the generalized hypothesis, which is a knowledge package or a knowledge source, but in a wider sense than used in the blackboard paradigm.

The tasks behind the "verification-methods" need not be locally available but can be distributed in the agents' community. This conceptual unit called generalized hypothesis represents the "Diagnostic Knowledge" and exchanging hypotheses between knowledge based systems in the form of cooperation should be seen as the exchange of formalized diagnostic knowledge. It amounts to a Hypothesis-and-Test strategy in a distributed but conceptually very compact form.

Another attribute of a hypothesis refers to the “State of the Hypothesis”, which can have values like: “not yet evaluated”, “proved”, “denied”, etc. It corresponds to different types of diagnostic knowledge, and may also be seen as different steps towards a final result. The mechanism of our cooperation scheme is exchanging hypotheses based on the state of the hypothesis and possibly on the availability of the verification method.

We give some simple example of hypotheses from the accelerator domain in TABLE I. It is worthwhile noting that the methods can create other hypotheses, child hypotheses in the hypotheses tree. Hypotheses can be “at different levels”, e.g. talking about different objects in a part/subpart hierarchy. The different levels in this abstraction correspond to different levels of the partial result as present in the result sharing cooperation via a Functionally Accurate/Cooperative [14] type strategy.

TABLE I - Examples For Hypotheses.

Object	Attribute	Method
The particle beam	is lost	Some measurement procedures (including data acquisitions)
A crate controller	is in bypass	A rule set or/and test program to verify it

Usually one distinguishes in diagnostic reasoning between *symptoms* (observed, initial data to start the reasoning) from which one derives *hypotheses* on several levels. If one finally can identify the *fault*, then this corresponds to a final verified hypothesis, which in turn is now able to explain the *symptoms*. For us conceptually and in our implementation these are all hypotheses on different levels of details as the examples in TABLE II show.

TABLE II - Symptoms And Faults Seen As Hypotheses.

	Symptom	Hypothesis	Fault
Object:	a magnet	an electronic module	a microprocessor in that module
Attribute:	is uncontrollable	is faulty	has a corrupted software table

Our definition for a hypothesis even allows more: the unification of the concepts of control, diagnosis, maintenance and repair/recovery actions and represents therefore a handy way of implementing cooperation between agents executing these tasks.

We recall that the SETUP program system brings the accelerator into an initial state (normal working conditions) after a shutdown or simply recovers from a failure in one or several modules of the control system. It initializes the equipment one by one and sets the correct values in the correct order. The SETUP is just an example of a *controls* task. *Diagnosis* finds out why a certain control or accelerator components are not working as they are supposed to do and suggests possible recovery actions from the fault. Cooperation between the SETUP, i.e. a control and the corresponding diagnosis, is quite natural. If a control action cannot be executed, one needs to diagnose the reason, which in turn will suggest a repair/recovery action to correct the fault. This recovery action is again a control which might fail, and diagnosis should give as answer: why, and so on. This type of cooperation implements the task sharing operation. The interwoven play between control and diagnosis can be seen on FIGURE 1.

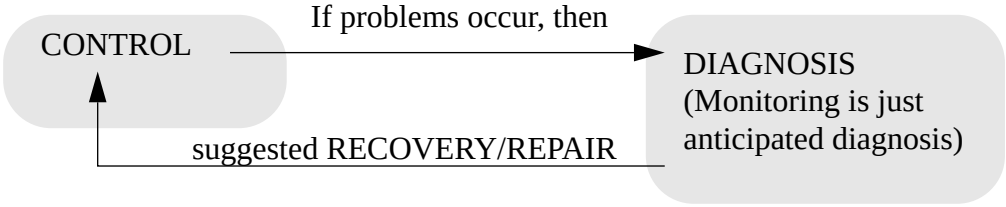

FIGURE 1. - Control And Diagnosis

Table III shows how the different control tasks are represented as hypotheses. The generalized hypotheses integrate the different activities in accelerator control. Any agent can create such hypotheses, and the hypotheses will be treated by those agents which have the capability to execute a verification method of these hypotheses. In this way the cooperative problem solving can be viewed as problem solving in a tree of these generalized hypotheses, where the hypotheses are evaluated by different agents, always by agents, who have the best knowledge base and capabilities for that. The hypotheses can be distributed to the appropriate agent by the ARCHON Layer. With that unified view for an hypothesis, cooperation is reduced to the standard operation of exchanging hypotheses for all kinds of control and diagnostic tasks.

TABLE III - The Generalization Of Hypotheses.

	Object	Attribute	Method
CONTROL in general:	Element to control (to set)	Control value (Set point value)	Equipment access
RECOVERY	Faulty element	Fault	Repair and recovery action
DIAGNOSIS	Faulty element	Suspected fault	Verification methods
MONITORING	Element that can go wrong	Possible fault	Checks (if it has really happened)

Handling Of Infinite Cycles

In the scenarios of control/recovery and diagnosis agents, repetitive mutual calls are possible and these calls may end up in infinite loops or even recursive calls of infinite depth. It may occur that the request from one agent to another agent may generate another request to the other agent which may regenerate the first request and these cycles are infinitely repeated.

In a loop, we have repeated calls from a diagnosis agent to a recovery agent (SETUP), but if the recovery is successful, the whole system is all right immediately, i.e it recovers immediately from the first, the original problem. In recursion any recovery is related just to the last diagnosed problem. This latest diagnosis can now therefore try the corresponding remedy proposed. Thus we have to work back through the different levels of depth to the moment where the original recovery works. This need not, however, to be the case: it is possible to think of a diagnosis at any level which has several possibilities for a recovery, and it would now try an alternative. This backtracking in a tree of recovery possibilities complicates the interaction between control and diagnosis agents even more.

The detection of an infinite loop or recursion is difficult, because an agent cannot reliably establish from its local information if the other agent's action is a response to a message sent to it, or if there is another reason. Therefore it cannot detect alone the presence of infinite cycles with certainty. Since infinite cycles must be avoided, they have to be detected and stopped with the help of a joint agreement of a group of agents. A seemingly easy way to stop the infinite cycle would be that if an agent detects that it

repeatedly executes the same action, it stops the repeated action. However this may not be appropriate in the case of a database type of agent for example, which has to answer to similar requests repeatedly, and these requests may not be part of an infinite cycle. An infinite cycle detection protocol is needed to stop such a cycle. Such a protocol must be activated when one of the agents suspects that an infinite cycle has been entered, i.e. when it sends out several requests and/or data of the same type, receives several requests and/or data of the same type and there is a causal relation between the incoming and outgoing request and/or data.

An infinite cycle detection protocol could work basically in the following way: If agent A_0 suspects that a request related to a message X_0 sent to agent A_1 ends in an infinite cycle, it sends a message "Agent A_0 suspects an infinite cycle related to request X_0" to agent A_1. If agent A_1 receives this message and agent A_1 also suspects an infinite cycle, agent A_1 finds out which messages are generated as a consequence of X_0. Let us assume that these messages are X_{10}, X_{11}, etc. and they are sent to agent A_{10}, A_{11}, etc., respectively. Agent A_1 then sends a message to each of agents A_{1i} with the content: "Agent A_0 and A_1 suspect an infinite cycle related to request X_0 and X_{1i}". These messages are propagated by the agents in the similar way as A_1 has done it. If in the end one of these messages gets back to one of the agents that has sent out such a message, there is a loop of agents where each agent suspects an infinite cycle and it indicates that there is an infinite cycle. Now the agents can stop this cycle.

The basic infinite cycle detection algorithm described above can be included in the concept of the generalized hypotheses. The generalized hypotheses can contain, as attributes, references to the requests and data exchanged in cooperation and the causal actions of the agents is represented by the creation of descendent hypotheses. If one keeps track of the creation dependencies of hypotheses, the information that is needed in the above infinite cycle detection protocol is immediately included in the generalized hypotheses. Each hypothesis should thus contain references to its direct and indirect ancestors. This was already included into the hypotheses used in the ARCHON test-bed scenarios. If an agent sees that a chain of descendent hypotheses returns to the agent several times, it can detect the infinite cycle.

5. SUMMARY

This paper has presented examples of applying the methods and results of DAI to an industrial controls environment, in particular in the accelerator field, and it has demonstrated the validity of several DAI concepts and techniques. These investigations are among the first experiences in creating operational DAI systems by transforming existing intelligent systems of a real industrial environment into members of a multi-agent community.

The generalization of the fault hypothesis concept has two major advantages: firstly, a unified view of control, diagnosis and recovery, which leads to an easy mapping of different tasks in accelerator control to cooperating agents, and secondly a straightforward method of transforming autonomous intelligent systems into a cooperating multi-agent system by exchanging hypotheses. Although some translation between the internal representation of the intelligent systems and the hypothesis representation may be needed, hypothesis exchange is a simple way to exchange knowledge in a multi-agent system.

These ideas have been illustrated by several cooperation scenarios and a suggestion to avoid the problem of infinite cycles of mutual agent calls and an example of relating knowledge exchange between agents to machine learning. The development of new intelligent systems related to timing and alarm information handling as well as their transformation into a multi-agent community are under study.

6. ACKNOWLEDGEMENTS

The work described here is partly based on the concepts of the ARCHON architecture, which was developed in the ESPRIT II project P-2256 (see Ref. [1]). We acknowledge the continuing support of the

former group leader of the PS controls group, Fabien Perriollat, during the ESPRIT collaboration and its follow up.

7. REFERENCES.

1. ARCHON (ESPRIT Project No.2256) whose partners were: Atlas Elektronik, JRC ISPRA, Framentec-Cognitech, Labein, Queen Mary & Westfield College, IRIDIA, Iberdrola, EA Technology, Amber, Technical University of Athens, University of Amsterdam, CAP VOLMAC, CERN and University of Porto.

2. G. Daems, V. Filimonov, V. Homutnikov, F. Perriollat, Yu. Riabov, P. Skarek; A Knowledge Based Control Method: Application to Accelerator Equipment Setup, presented at the Intern. Conf. on Accelerator and Large Experimental Physics Control Systems, ICALEPCS-93, Berlin, Germany, Oct. 18-22; 1993.

3. Th. Wittig, ed.; ARCHON: An Architecture for Multi-Agent Systems, Ellis Horwood, ISBN 0-13-044462- 6.; 1992.

4. N.R. Jennings, T. Wittig; ARCHON: Theory and Practice, In: N.M. Avouris and L. Gasser (eds): Distributed Artificial Intelligence: Theory and Praxis, Kluwer Academic Publishers, ISBN 0-7923-1585-5, Dordrecht; 1992.

5. N. R. Jennings, J. Corera, I. Laresgoiti, E. H. Mamdani, F. Perriollat, P. Skarek, L. Z. Varga; Using ARCHON™ to develop real-world DAI applications for electricity transportation management and particle accelerator control, to appear in IEEE Expert - Special Issue on Real World Applications of DAI; 1995.

6. D. Cockburn, N. R. Jennings; ARCHON: A Distributed Artificial Intelligence System for Industrial Applications, in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings) Wiley, 1995.

7. N. R. Jennings, J. M. Corera, I. Laresgoiti; Developing Industrial Multi-Agent Systems, (Invited Paper), First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, CA., June 12-14, 1995.

8. E. Malandain, S. Pasinelli, P. Skarek; A Fault Diagnosis Expert System for the CERN PS, Proc. Europhysics Conf. on Control Systems for Experimental Physics, Villar-sur-Ollon, Switzerland, Sept.28 - Oct.2, 1987, Proceedings in: CERN 90- 08 (ed. B. Kuiper), 217-220; 1987.

9. E. Malandain; An Expert System in the Accelerator Domain, 1st Intern. Workshop on Softw. Eng., AI and Expert Systems for High Energy and Nuclear Phys., March 19-24, 1990, IN2P3, Lyon Villurbanne, France, and CERN/PS 90- 45 (OP); 1990.

10. E. Malandain, P. Skarek; An Expert System in the Accelerator Domain, IASTED, Proc. of the Symposium EXPERT SYSTEMS Theory & Application, Switzerland, June 16-18, 1987, M.H. Hamza (Ed.), ACTA Press, Anaheim, CA, 100-105; 1987.

11. N.R. Jennings, L.Z. Varga, R.P. Aarnts, J. Fuchs and P. Skarek; Transforming Standalone Expert Systems into a Community of Cooperating Agents, Engineering Applications of Artificial Intelligence, Vol.6, No.4, Aug., 317-331; 1993, and ESPRIT ARCHON Technical Report No. 42/2-93, and CERN/PS 93-15 (CO); 1993.

12. F. Perriollat, P. Skarek, L.Z. Varga; Report on the CERN Application Study, ARCHON Public Deliverable 1060; 1993.

13. P. Skarek, L.Z. Varga; The Integration of the SETUP - Diagnosis and Recovery in Accelerator Control, CERN/PS/CO/Note 94-34, CERN, Geneva, 9 June; 1994.

14. V.R. Lesser, D.D. Corkill; Functionally Accurate, Cooperative Distributed Systems, IEEE Tr. on Systems, Man, and Cybernetics, Vol. SMC-11, No.1, January, 81-99; 1981.