LL

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN - PS DIVISION

01

✓ CERN/PS 94-03 (CO)

Sw 9410

**A KNOWLEDGE BASED CONTROL METHOD:
APPLICATION TO ACCELERATOR EQUIPMENT SETUP**

G. Daems, F. Perriollat, P. Skarek ¹⁾
V. Filiminov, V. Homutnikov, Y. Ryabov ²⁾

Abstract

An executable description language for defining the structure of large installations and their corresponding control algorithms is presented. This method of knowledge representation is based on software entities called "goal procedures" for real-time control in an object-oriented manner. The knowledge interpreter is driven by events coming from the control system and is capable of forward graph traversal and backtracking.

The accelerator equipment setup problem concerns a large collection of hardware and corresponding parameter descriptions, equipment initialization and testing, data loading and diagnosing these processes. The proposed method and tools are applied to the CERN/PS accelerator setup, i.e. cold start and recovery.

This system includes descriptions of the CAMAC hardware and accelerator equipment, and the possibility to control and to diagnose all stages of the setup process.

1) CERN 2) PNPI, Gatchina, Russia

International Conference on Accelerator and Large Experimental Physics Control
Systems, Berlin, Germany, October 18-22, 1993

Geneva, Switzerland
28/2/94

**Information to SIS about
CERN Publication or Divisional Report**

From : E. Durieu PS Division

Date: 1/3/94

To : Mme Dutour, Main Library

The following paper has been authorized by the Division Leader. Six copies are attached.

Title **A Knowledge Based Control Method:
Application to Accelerator Equipment Setup**

First author: **G. Daems**

Reference: **CERN/PS 94-03 (CO)**

No. of manuscript pages: 11

☒ This paper is **submitted to a conference** (*please give exact title, place and dates*):

International Conference on Accelerator and Large
Experimental Physics Control Systems, Berlin,
Germany, October 18-22, 1993

☐ It will be published in the **Proceedings**, as noted below :

☐ This paper is **to be published** (*please give details indicated as far as possible*) in a :

☐ **Periodical** (*title, vol.*) :

☐ **Book** (*title, editor, publisher, date*) :

☐ This paper is a Divisional report **not to be published**.

☐ No further copies are available for distribution.

Signature :

Signature Division Leader :

d. finon
1.3. 1994

A KNOWLEDGE BASED CONTROL METHOD: APPLICATION TO ACCELERATOR EQUIPMENT SETUP

G. Daems ^{a)}, V. Filimonov ^{b)}, V. Homutnikov ^{b)},

F. Perriollat ^{a)}, Yu. Ryabov ^{b)}, P. Skarek ^{a)}

a) CERN/PS, 1211 Geneva 23, Switzerland

b) PNPI, Gatchina, 188350, Russia

An executable description language for defining the structure of large installations and their corresponding control algorithms is presented. This method of knowledge representation is based on software entities called "goal procedures" for real-time control in an object-oriented manner. The knowledge interpreter is driven by events coming from the control system and is capable of forward graph traversal and backtracking.

The accelerator equipment setup problem concerns a large collection of hardware and corresponding parameter descriptions, equipment initialization and testing, data loading and diagnosing these processes. The proposed method and tools are applied to the CERN/PS accelerator setup, i.e. cold start and recovery.

This system includes descriptions of the CAMAC hardware and accelerator equipment, and the possibility to control and to diagnose all stages of the setup process.

1. Introduction

The problem of accelerator equipment setup includes at least two main tasks: the initialization of hardware and software components during a cold start of the control system or after a replacement of some element(s) and the execution of non-destructive tests and recovery actions during runtime.

These tasks should be available for any system element independently of an exhaustive diagnosis of faults. This means that the setup facility should be based on detailed system structure description, including status information and control algorithms.

The rejuvenation of the CERN Proton Synchrotron control system [1],[2] requests for a new SETUP facility. The evolving nature of the process to be controlled calls for easy and flexible changes of the corresponding SETUP software. (Possibility of working with "hot" system, i.e. on-line updated structure description would be ideal). One suitable way to fulfill the previously described requirements seems to be the use of methods and tools of knowledge based systems. The methods and software presented here are developed for real-time control of large technical installations [3],[4] and were improved while implementing the SETUP application.

The development is based on an object oriented approach of programming, combined with procedural knowledge representation. It results in a tool with the following main characteristics:

- description of system structure in very concise form, based on inheritance and encapsulation of objects;
- description of control rules and algorithms in a compact and understandable format, providing global control on different subsystems by the individual algorithms;
- flexibility to modify the existing control and structure descriptions.

The developed methods for knowledge representation and interpretation provide a possibility of complex control goals descriptions, taking into account on-line temporal state changes of the control process. These features, together with the classical knowledge interpreter capabilities make it possible to use this tool for a broad area of applications, from fundamental setup and diagnosis to automatic run-time control of complicated installations.

2. System description concepts

The knowledge description for a controlled system is composed of two parts: system structure description and control information (procedures). Any element of a system is an object that has a set of facts characterizing its current state (objects database). A fact is an assertion about an object. A fact can be established either as a result of some logical inference based on other facts (e.g., "this crate is able to work") or by means of accessing the environment (hardware or an operator) for measurement or control (e.g., "this crate is bypassed"). Any fact can be declared as a control goal when defining knowledge for system control.

Only two possible fact types have to be included into description of object as an entity. The first one is a fact which can be established via some connection to the environment exclusively. The second one is a fact established (reached by means of some control actions) under some predefined conditions.

The facts of these two types are called properties. Property definition must include specification of interface type for connections to environment (key word INTERFACE, see below) via external procedures or a condition (a logical statement following the key word IF). Properties can be set or removed by external procedures or goal procedures, but other facts - only by goal procedures.

An object can include other objects that are called its elements or members in such a case. An object can be the member of several parents. An object description can be based on one done before for other object. In such a case the latter object is called basic object. Derived objects inherit all properties and goal procedures of basic objects.

The elements of basic objects can be redefined and new elements, properties and goal procedures can be added to a derived object. There are no differences in the syntax of the descriptions of basic (abstract) and real objects (the differences appear after binding objects to the external environment during run-time). The possibility to define that several members of the same type can be included into some abstract object is provided by the concept of variables.

2.1. Semantics of goal procedure

A goal procedure is represented with a graph showing the possible ways to achieve a goal, i.e., to establish some facts for an object. Each arc of such graph represents some action or several parallel actions to achieve partial (intermediate) goals. Each procedure belongs to some object. A goal procedure of an object can refer to facts and can call procedures belonging to that object or its members.

Possibility of arc passing is determined by the associated conditions for the database of objects the procedure belongs to and possibly for its members databases. Procedure graph nodes are not more than logical decision points defining which actions can be taken next.

The knowledge interpreter is capable of searching a way of passing the procedure graphs (including backtracking to previous nodes in case of faults) taking into account asynchronous events arising during an inference. The goal procedure can include calls to other goal procedures or to external procedures (syntactically the same). The means proposed here provides the possibility to describe knowledge for backward chaining to reach a goal (goal procedures) as well as for forward chaining (properties to be the goal under prescribed conditions).

Each procedure is represented by a triplet $\langle p \rangle w \langle q \rangle$, where: $\langle p \rangle$ - precondition; the procedure is able to be run if and only if $\langle p \rangle$ is true; $\langle q \rangle$ - goal; $\langle q \rangle$ is a fact of the object the procedure belongs to, which becomes true after successful termination of the procedure; w - an oriented graph of possible paths; its arcs are marked with temporal goal assertions.

There are temporal assertions of three types: $!p$, $?p$ and $\#p$. Here:

- $!p$ means that after the arc is passed, p has to be true;
- $?p$ means that before passing the arc p has to be true;
- $\#p$ means that during passing the arc p has to be true.

There is a set of paths for executing the procedure. The set of paths consists of successful and faulty paths. The arc passed may finish successfully or faulty. A node and its incoming arcs is considered, if it has been successful. Otherwise the next arc is considered. If there are no successful arcs issuing from the current node the previous node is considered. Passing an arc may generate several parallel goals. If one goal

has been rejected, the arc does not finish successfully, there is the problem what to do with other parallel goals. For diagnosis applications, to our opinion, the best way is to permit goals to execute up to the end. For real-time applications it is necessary to stop all goals in order to find a way to achieve the main goal as soon as possible.

2.2. Syntax of system object description

A name of an object is a string of symbols. If the string contains spaces, it must be enclosed in double quotes. Facts are strings of symbols too, which are some statements characterizing an object. These strings need no quotation marks.

The names for describing objects and goal procedures are written by capital letters. The description of an object is terminated by a dot. Objects can be simple or composite: Simple objects do not include other ones. This is the syntax of simple object:

```
<object name 1> [, ... ,<object name J>] HAS PROPERTY
<definition of the property 1> [, ... ; <definition of the property N>] .
<definition of the property> includes the property name followed by
logical assertion with preceding key word 'IF' or interface definition:
<property name> [INTERFACE <interface name>] [IF <logical statement>];
Property definitions are separated by semicolons.
```

Objects can be created based on other ones in the following way: <object name 1> [, ... ,<object name J>] IS A <name of basic object>. Each of <object name 1>, ... ,<object name J> inherits all the properties and includes all the sub-objects of <name of basic object>.

Composite objects:

An object can include other ones. This is defined by an expression of the following syntax:

```
<object name>
INCLUDES
<description of included objects>;
HAS PROPERTY
<definition of properties>.
<description of included objects> is a set of descriptions separated by
semicolons.
```

Composite objects inherit properties and included objects from a basic object. An object can include other properties and objects in addition to the inherited ones.

2.3. Description of goal procedures

The head of a goal procedure consists of three strings:

- the name of the object this procedure belongs to;
- the fact which is established if the procedure finishes successfully;
- the precondition i.e., a logical statement defining conditions for making it possible to start a procedure.

A procedure body is an oriented graph description. The description of an arc consists of a number of nodes leaving the arc, a number of nodes entering arc and a temporal assertion. If the number of a node equals to zero it is an end node (it is a success).

The general view of an arc representation is $m\ n\ ?(C), \#(P), !(Q1), \dots, !(Qj);$

Here:

- m, n are start and end node numbers of this arc;
- Q_i is a string which consists of an object name and the fact to be achieved (proved); if the fact belongs to the object being this procedure host the name of the object may be omitted;
- C is a logical statement describing a condition which has to be true before passing the arc;
- P is a logical statement describing a condition that must be true during achieving the goals.

Full syntax of a goal procedure may be written as follows:

```
<name of object> ;  
<goal>;  
<precondition statement>;  
<number of node 1> <number of the next node> ?(c1), #(c2), !(g1), ..., !(gn); ...  
<number of node j> <number of the next node> ?(c1j), #(c2j), !(g1j), ... ,  
!(gnj).
```

As it was said above the <goal> is the fact that will be put to the object's database after the goal procedure would finish successfully. The goal can correspond to a set of procedures for achieving it and the precondition statements ensure it. If the precondition statement is false the procedure cannot be executed and it is necessary to find another one. A small procedure from the SETUP is shown on fig.1.

The system acts on the equipment by means of a set of executive commands. If the expert system controls the equipment directly, it must be able to call programs to operate on the devices. If the equipment is run under control of the operator, the system sends an advice to him.

Fig. 1: Example of the procedure graph

2.4. The Knowledge interpreter

The structure of the knowledge interpreter is represented by Fig. 2 as part of the control system. The main part of the interpreter is an object oriented knowledge base containing current databases of the objects, the tree of current goals (procedure tree) and the current list of protected statements.

Fig. 2: Structure of the knowledge interpreter

The interpreter obtains the new facts (properties with INTERFACE keyword) from the environment and stores it in the object database. Then the IF statements of the object are calculated. If a statement is true then the interpreter refers to the property corresponding to this statement as the goal and tries to achieve it.

After that, it tests the current goals and the protected statements. If some goal is achieved then the new goals are determined and the corresponding procedures are called.

Before executing any goal procedure the interpreter has to check (prognosis) whether the goal contradicts a current protected assertion. In this case the goal is rejected and the next one is searched. If some protected statement has become false then the arc marked by this one and the corresponding goal procedures are rejected.

The new way of achieving a goal should be searched as it was described in the previous chapters.

All the data about the installation including the available and current goals can be displayed for the operator and he can choose the new goals for execution. It is necessary to check this choice by means of the interpreter too, as it was mentioned above.

III. THE SETUP APPLICATION

3.1. Representation of the equipment structure

All the features of real system structure (technical or logical) can be described by means of the tool proposed. The structure description is based on two main objecttypes: Equipment and CAMAC. The corresponding inheritance can be showed asfollowing:

- 1) Equipment -> <type of equipment> -> <equipment unit>;
- 2) CAMAC -> "Dedicated module" -> <type of module> -> <concrete module>; CAMAC -> <standard crate facility> -> <facility of concrete crate>.

The first line reflects object types and their inheritance in accordance with logical interface description of equipment facilities adopted at CERN/PS for control [5],[6],[7]. The seconds correspond to inheritance of general properties from a basic object called CAMAC to any "Dedicated module" and standard crate facility, etc.

At the same time derived objects have some properties and/or procedures not belonging to its parent objects: an initialization procedure for the timing equipment will be different from that of the power supply equipment.

Hierarchies of objects are quite natural. They reflect technical structure and dependence (equipment unit < module < crate < front-end computer) or logical inclusions (equipment unit < working set), where sign "<" is used as inclusion symbol.

The concept of variable is widely used as notation for the structure of abstract generic objects:

```
"Dedicated module" IS A CAMAC
INCLUDES $EQP IS A Equipment.
DSC IS A SYSTEM_OBJECT
INCLUDES $Crate IS A CAMAC_Crate.
```

3.2. Capabilities of the SETUP facility

Two main tasks of SETUP facility: initialization and non-destructive testing with recovery can be carried out for any equipment unit, CAMAC module, CAMAC crate or a whole CAMAC loop. This possibility is shown schematically by the following figure (Fig.3).

Fig. 3: Scheme of SETUP command execution

The listed actions are available to the user as interactive control commands. They are described in the corresponding objects as properties with the interface type set to "user". A command issued for an object causes the same action to all included objects (vertical arrows between actions in the figure). If some action finishes unsuccessfully then a test is executed on the including object of the next upper level (arrow-arcs at the left side of the figure).

Other interactive tools are available to the user for a more detailed analysis of the current system state. He can display the contents of the database describing a system object or the setup protocol. Each action which was attempted to be executed by the setup facility leaves a logging trace of two records in a "protocol file". The records mark the path of traversals of the different actions and decisions in the graph.

3.3. Features of implementation

The SETUP facility has to be embedded into the running control system. A software package was developed in order to interface the knowledge interpreter according to the control and acquisition standards of the existing system and the network architecture (the SETUP itself runs on DEC Workstation under ULTRIX).

All the connections with equipment objects are performed via standard equipment interface accesses [5],[6],[7]. They provide also protection against simultaneous actions of the SETUP and control system (via RESERV/RELEAS properties) and hide the physical addresses of equipment by using a structure description from a real-time database derived from the general database under ORACLE.

Although direct CAMAC access for crate initialization and testing had to be implemented via remote procedure calls (RPC) based on the CERN Network Compiler [5], there are no restrictions for developing other types of interfaces e.g. for VME modules direct access if needed, by the same tool.

It should be noted here that with the help of some special utilities to be developed, the system structure data can be derived on-line automatically from the general database under ORACLE, making structure description changes immediately visible to the SETUP.

IV. CONCLUSIONS

The SETUP project was started at the end of 1992. The first step was the development of a prototype and the tests with some typical hardware assembly. This was successfully finished in July of 1993. The software was run in the real CERN/PS hardware and software environment and a serial CAMAC loop with two crates containing CAMAC modules of three types.

From these results we concluded the soundness of our approach and decided to implement the full system. The users' interaction requirements seem to be easily matched to the user interface facilities of control system. The SETUP can become a flexible and convenient tool for the control system maintenance team.

In addition, we started an investigation to use the cooperation shell ARCHON [9] for treating the SETUP as part of a general fault-finding and recovery system in an multi-agent environment [10].

V. REFERENCES

- [1] The PS and SL Controls groups, "PS/SL Controls consolidation project", CERN PS/CO 91-09.
- [2] G. Benincasa, G. Daems, B. Frammery, P. Heymans, F. Perriollat, Ch. Serre, C.H. Sicard, "Rejuvenation of the CPS Control Systems; the First Slice", CERN/PS/91-14(CO), April 1991.
- [3] V.M. Filimonov, A.P. Gulin, V.P. Homutnikov, S.B. Oleshko, Yu.F. Ryabov, "Data & Knowledge Technique for Software of Real-Time System" in: IEEE Seventh Conference REAL TIME '91 on Computer Application in Nuclear, Particle and Plasma Physics Conference Record, June 24-28, 1991, Juelich, Fed. Rep. of Germany, pp. 222-324.
- [4] V.M. Filimonov, V.P. Homutnikov, Yu.F. Ryabov, "An Approach for Description and Interpretation of Procedural Knowledge for Complex Physical Installation", Preprint PNPI 1828, St. Petersburg, September 1992.
- [5] L.Casalegno, J.Cuperus, C.H.Sicard, "Process equipment data organisation in CERN PS Controls", Proceedings of ICALEPS '89, Vancouver, 30 Oct. - 3 Nov.1989, pp. 412 - 415.
- [6] J.Cuperus, W.Heinze, C.H.Sicard, "Control Modules Handbook", PS/CO/ Note 91-25.
- [7] "Generalities on the Control Protocol", The working group for protocol (WOPRO), April 1991 - CERN/PS/CO/Note 91-01, Reported by G.Benincasa.
- [8] Georgeff M.P., Lansky A.L. Procedural knowledge. Proc. of IEEE, Vol. 74, 10, Oct 1986. pp 79-100.
- [9] ARCHON (Project No.2256) whose partners are: 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. Th.Wittig, ed., ARCHON: An Architecture for Multi-Agent Systems, Ellis Horwood, 1992, ISBN 0-13-044462-6.
- [10] F.Perriollat, P.Skarek, "Applications of Distributed Artificial Intelligence for Accelerator Control", IASTED Symposium on Applied Informatics, Annecy, 19- 21 May 1993, and CER/PS Report 93-20 (CO).