INTEGRATED TOOLS FOR CYCLOTRON OPERATION

W. Busse
Hahn-Meitner-Institut, Bereich Schwerionenphysik
Glienicker Straße 100, W-1000 Berlin 39
Germany

ABSTRACT

With reference to the generally accepted architecture of modern control systems a review of facilities as implemented for cyclotrons presently being commissioned will be compared to commercially available control packages and tools. An attempt is made to point out their qualification to integration, their interoperability and present drawbacks.

1. INTRODUCTION

Modern process control systems use embedded distributed intelligence which is based on well defined hardware standards and more or less standardized software. Data bases hold all the necessary configurational static or dynamic views. More modern systems even implement software standards and integrate tools. Future control systems will be generated and run by fully integrated tool kits. These are the highlights which you might take along from control conferences and working group meetings. But where are we today? What in fact is a control system in this context? What are integrated tools? Are there any such tools available for accelerator, or more specially for cyclotron controls?

An abstracted view of a machine control system is given by fig.1. The process in view and the product it has to deliver are part of the lowermost level. It may be called the SHAPING level where the manipulation of producing objects is carried out and where views of the resulting states or of the transitions between them are generated. The uppermost level, which could be called the REASONING level, comprises all the activities like DESIGN, MODELING, SIMULATION, OPERATION etc., which in general still involve human brains and human intervention. The area between these two levels is the TRANSACTION level, which transfers requests of the REASONING into appropriate actions of the SHAPING level and vice versa, according to well defined rules of interference. This intermediate level is the translator of functions and data according to their level-dependent meaning and representation; it has appropriately defined access interfaces to the upper and lower levels respectively. The process Control System is the vehicle for all the transactions. A good control system is completely transparent and offers the flexibility and expandability needed to adapt to changing functionality in either level.

2. TOOLS FOR DESIGN AND IMPLEMENTATION

The analysis of the functional requirements has presumably always been the basis of any control system design. However, the final implementation often had to decide on acceptable drawbacks because of technological limitations or even man-power or financial constraints. It must be admitted,however, that the need for more functionality has motivated and pushed technological development and that the advent of new technologies has opened the view for refinements in both, design and implementation.

Whilst the system architecture is based on the functional requirements through the question 'what do we want to achieve?', the implementation of the hardware and a software architectures is based on 'how do we want to achieve this aim?'. Again both have functional components which should reflect the overall system functionality. Although logically interrelated, the topology within the architectural schemes need not necessarily be the same, e.g. a given hardware architecture can be the basis of different software architectures and vice versa [1].

Presently all but the smallest systems use distributed computing. The underlying hardware and software architectures are surprisingly similar and can largely be divided into three logical layers as symbolized by fig. 1.

Hardware:

- an upper layer made up of the operator interface and the application computers,
- a middle layer of processors distributed around the accelerator dealing with sets of equipment,
- a lower layer of processors which interface to dif-

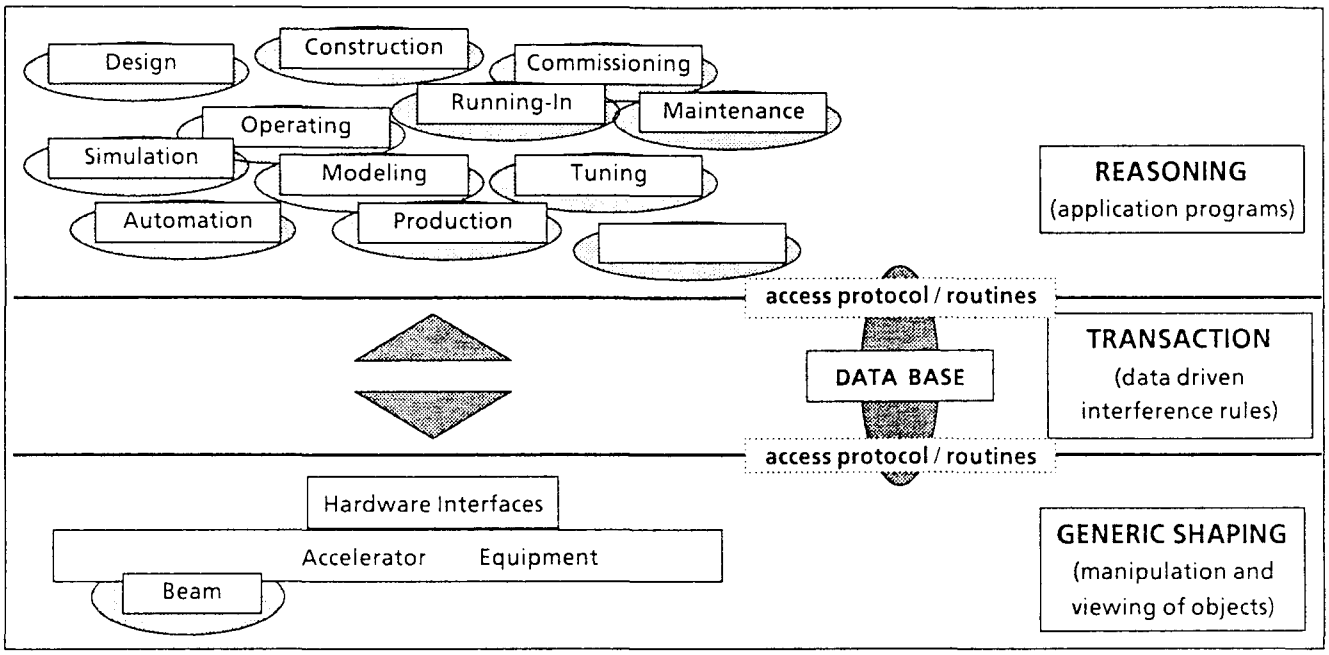

Fig.1: Schematic view of logical control system levels

ferent types of equipment.

The layers are interconnected by a communication network as the backbone. In most cases this is a local-area network for the upper and middle layers with multidrop or individual links connecting the middle layer processors to those of the lower level (cf fig. 2).

Software:

- an upper level comprising the application processing, including the operator interface,
- a middle level, transferring operational actions into hardware interface actions and vice versa (in general data driven),
- and a lower level of equipment driving and hardware access routines.

A (distributed) data base is the backbone, with its access routines from and to the upper and lower levels. It allows to separate the upper and lower levels from each other and to make them independent of each other by operational protocols which only deal with 'meaningful' operational parameters and which are transparent with regard to distribution and specific hardware.

The hardware architecture is generally supported by accepted standards with written specifications. The hardware is commercially available and may often be chosen from a variety of products according to the actual needs.

The same is, or is becoming, true for what I would like to call the control system firmware, i.e. the computer and microprocessor operating systems, the network communication software and the software development environment.

It goes without saying, that increased empha-

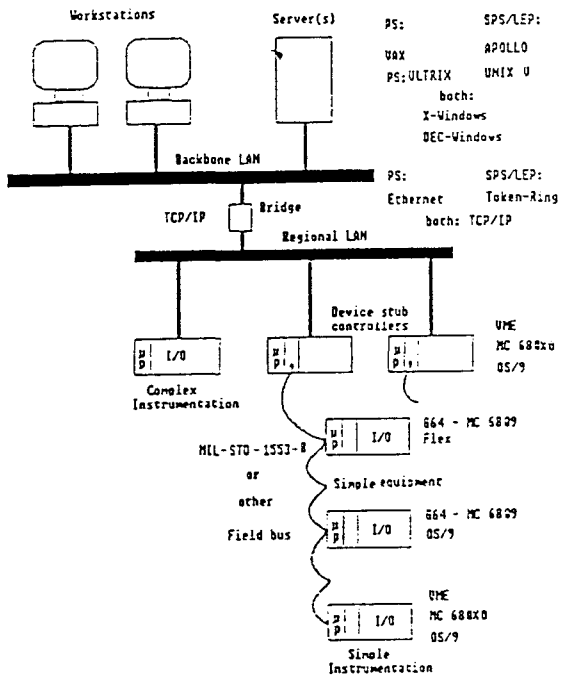

Fig.2: Simplified view of the hardware architecture for CERN accelerator control [1)]

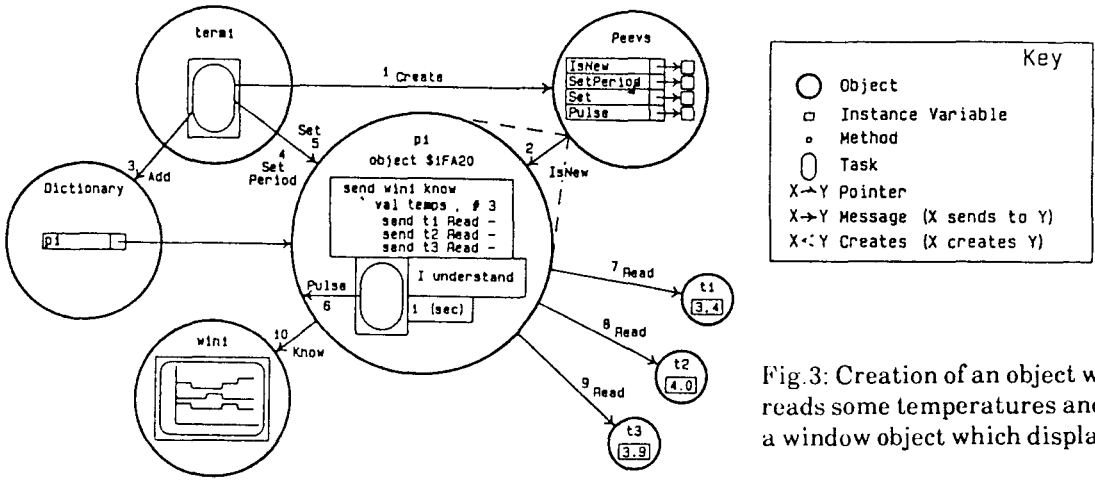

Fig.3: Creation of an object which periodically reads some temperatures and sends the values to a window object which displays them graphically

sis is now put on techniques for designing software in a more formalized way in order to reduce the increasing cost of ever more sophistication.

Given the fact of the commercially available hard- and firmware mentioned above, the idea is to base the underlying control system design on a higher level of abstraction. Initial discussions should not concern specific hard- and software architectures and implementations, but should regard the system as being primarily independent of such aspects.

A notion of L.Chapman[15] may serve as an example. His basic vision of a control system is a sea of objects spread over a computer network, organized logically and somewhat independently of the physical organization of the network. That is, any particular processor might contain one or more logical groups of objects, and an object group might be spread over multiple processors. These objects send each other messages in a simple, uniform way, regardless of whether these objects are nearby (e.g. in the same processor) or distant (e.g. on a different network node). Figure 3 illustrates a detail of an implementation which is based on these ideas

Tools are supposed to help their user achieving the aim in mind in a smooth and easy way. With wrong or badly adapted tools he will not succeed. With the aim of refinement good tools can be used to produce new more sophisticated tools intended for more sophisticated operations, and so on.

Integrated tools produce an output which can be directly used as input for the next step. They are plug-compatible to a common backbone like hardware modules to their standard bus. They are the basis of interoperability.

Assuming that the hard- and firmware for a control system implementation is commercially available as mentioned above, tools for their production, e.g. CAD/CAM-techniques, need not be looked at. It is still felt, however, that there still is little choice among tools helping with the design and implementation of the software architecture or even tool-kits representing a tailorable control system which can be tuned to the user's needs.

Commercially available tools exist e.g. to facilitate software implementation: language sensitive editors, version management and control, graphical editors, program development environment for programmable local controllers (development on host computer, down-loading, remote debugging) often even menu-driven via graphical user interfaces, test managers. These tools are in general part of the software delivered by the computer manufacturers.

The next higher level of commercial support is by Computer Aided Software Engineering (CASE) tools. Various companies offer packages which they like to call *workbenches* and which lead the user with the help of graphical desk user interfaces through analyzing, architecturing, designing and programming phases. Code generation is not necessarily available. The integrating backbone is a project database which often may be a different manufacturer's product. These packages are mostly model based and implement methodologies like *Structured Analysis* (SA)[2], *Real-time Structured Analysis* (RT/SA)[3] and *Structured Design* (SD)[4], the so-called SASD techniques. More modern implementations are based on *Object Oriented* principles, which seem to be more adequate to process control systems. However, *object orientation* is today where SASD was about ten years ago. An application of these techniques is discussed is discussed by G.Ludgate and E.Osberg[5]. In summary, these tools completely integrate their sub-tools, run on many commercially available platforms, but in general do not produce the code to let you directly use the product which is designed with their help.

The nest group of tools to be discussed consists

of products with embedded executable software for a specified range of applications. They offer graphical editors to let the user choose objects from given selections, choose their attributes and their interconnections. The next paragraph will describe tools of this group in more details.

3. SELECTED COMMERCIAL TOOLS AND APPLICATION EXAMPLES

The following examples were chosen to illustrate commercially available tools which can indeed help with implementing or generating an accelerator control system. Their common feature is the generation of animated synoptics for operator interaction, nevertheless their internal features and their range of completeness in the context of a process control system are totally different.

3.1 DATAVIEWS, SL-GMS

DATAVIEWS [6] and SL-GMS [7] (SL Graphical Modeling System) are two of the more complete development systems for building and managing graphics screens which can be (easily) embedded in user applications. Both products enable the user to generate sophisticated graphical user interfaces to any application, e.g. monitoring and control, modeling, simulation. Apart from a very powerful drawing tool the user is supplied with a prototyping package (no programming is needed) and a library of run-time routines for programming access to views, sub-views, graphical objects and their attributes, screen management etc. Programming access is provided for most standard languages such as C, Fortran, Pascal, Ada.

Figure 4 illustrates the positioning of these tools within an imagined control system as *application*. For *monitoring* the interface receives input from the application in form of application data, the interface then delivers output to the user in the form of graphical data displays. For *control* the interface receives input from the user in form of keystrokes, mouse picks or other locator events which is delivered as output to the application in form of control data. The data exchange can either be done by intermediate files or by direct delivery to or from the product internal data structures using calls to the respective library routines. In the first case a description of the file structure must be maintained in both, the application and the interface, the second needs a sort if integrating gateway between either data structures.

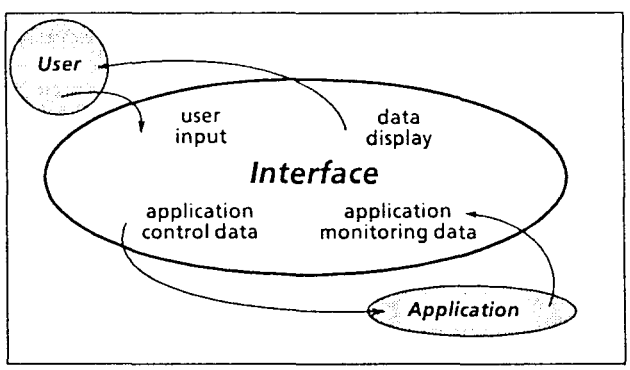

Fig.4: Illustration of the dialogues of a visualization and interaction interface with user and application

Both products support a large number of platforms, operating systems and graphical interfacing standards. DATAVIEWS seems to be the more industrial and stable product being based on classical design concepts, the design of SL-GMS follows the more modern object oriented principles but still is less ergonomic. [8]

Mind, however, that these products produce visualization and interaction interfaces, they do not generate a runnable process control system.

3.2 Vsystem

Vsystem [9], the Vista Control system software, is developed to control a given specifiable process through a CAMAC or VME hardware interface using Digital Equipments VAX-computers with the manufacturer's VMS and ELN operating systems. Devices to be connected to the system have to be described in a data base of text entries which is afterwards converted into binary format, the run-time data base.for use by the control software. A graphics interface based on window software provides control and monitoring functions, referred to as channels. The system incorporates a comprehensive toolkit to create operator displays, to connect device-channels to parts of the displays and to access information in the runtime data base.

Before the system can be put into service, the system manager must provide a *handler* for each hardware interface describing its characteristics (if not delivered by VISTA) and he must specify a data base entry for each channel in the system which also includes the handler name for the hardware channel.

Vsystem offers a variety of features which are known to be useful in process control systems, such as generic readers and reader banks, data logging, "strip-charts", alarms for specified channels. A sequencer package is available to the operator to perform sentence-like instructions on specified channels in a certain sequence.

As any other commercial company in the field VISTA is continuously implementing enhancements

and new features [10].

Various accelerator control systems have implemented partial systems generated by the Vsystem package or are evaluating for future use. Figure 5 is an operator interaction display of the TRIUMF Beam Line 2C vacuum system [11].

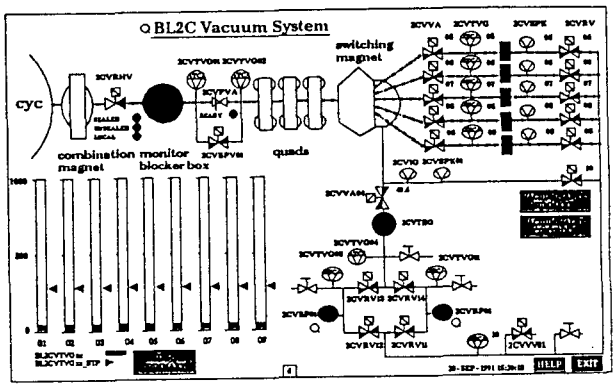

Fig.5: Vsystem operator display of TRIUMF beam line 2C vacuum system

Assuming that all the hardware handlers mentioned above are available, the user of this package need not write any software, unless he wants to integrate features which are not yet offered (e.g. model support, simulation). In this case, however, the user is either compelled to stay with the VAX hardware and VMS/ELN software and especially with the system data base or he has to undergo a considerable integration job where data integrity can be an issue.

3.3 SYSTEMS BASED ON INDUSTRIAL PROGRAMMABLE CONTROLLERS

Various industrial systems within this group provide the highest degree of commercially available tool integration. This is due to the fact that one manufacturer delivers proven hardware and software which can be configured to the costumer's requirements. Hardware and software configuration is achieved by putting together prefabricated modules with the aid of graphical tools, appropriately engineered for each level of the process control system hierarchy.

The outstanding advantages of such sysstems are proven and guaranteed reliability in an industrial environment, determined and stable behaviour even under extreme conditions, possibility of interconnection and point-to-point communication. Programming, which means logical assembly of appropriate building objects, is done by integrated 'programming tools' which can be used by 'programmers' of all levels. The packages also integrate 'programmable' user interfaces. Complete compatability between hardware and software is guaranteed on all levels, documentation is automatic and maintenance is assured by the manufacturer.

Generally speaking slower processing is the price to be paid for the reliability of such systems within industrial environment. Limited memory space which was an inconvenience of early implementations is being overcome.

Complete knowledge of the process to be monitored and/or controlled is another basic requirement for these systems to be appropriate. No modeling or basic research about the internal process behaviour is appreciated. Hence, these systems are favorable candidates for the implementation of turn-key systems.

The industrial example chosen here is the SIMATIC-S5[12] system with the STEP-5[12] and COROS[12] programming and production tools. Figure 6 is a pop-up of the base-level I/O-modules, function generators and signal processing modules, their

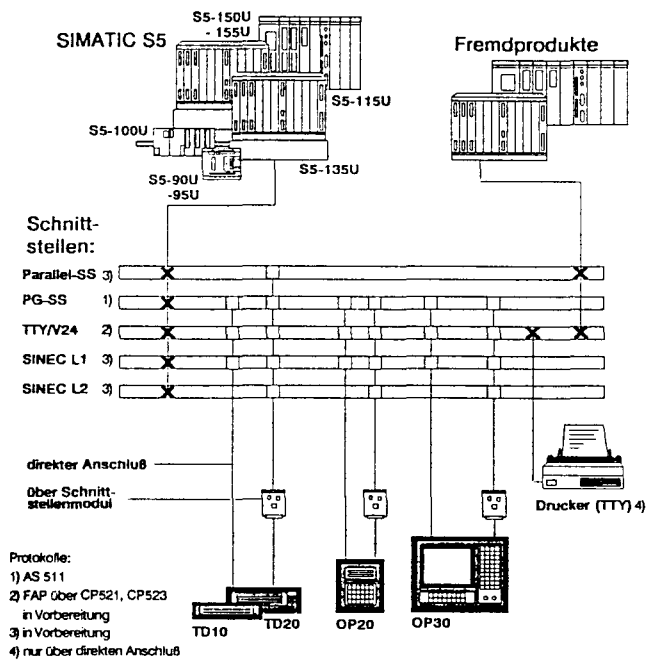

Fig.6: Base level hardware of the SIMATIC-S5 system

bus systems and interconnections as well as base-level local operator interfaces. Programming on this level is done with the tool package STEP-5 which offers a symbolic editor to define and name (globally uniquely) input- and output-signals or functions and to produce or to correct graphical views of their logical interconnection with the help of ladder logic. COROS provides the tool to interconnect base-level equipment of subsystems, subsystems to site-systems and sites to central monitoring and control.

Commands to exchange data between processors and process image as well as reactions to be carried out in case of data changes can be defined. Messages and reaction strategies on alarms, alarm analysis and automatic archiving can be set up to the user's requirements.

In the accelerator field the SIEMENS tool kit has been successfully applied to e.g. the control system of the compact cyclotron Cyclone 30 at the Catholic University of Louvain, Belgium[13]. Figure 7 is a schematic view of the system. The following functionality is offered to the user: automatic startup and rundown to a standby state, automatic change of source

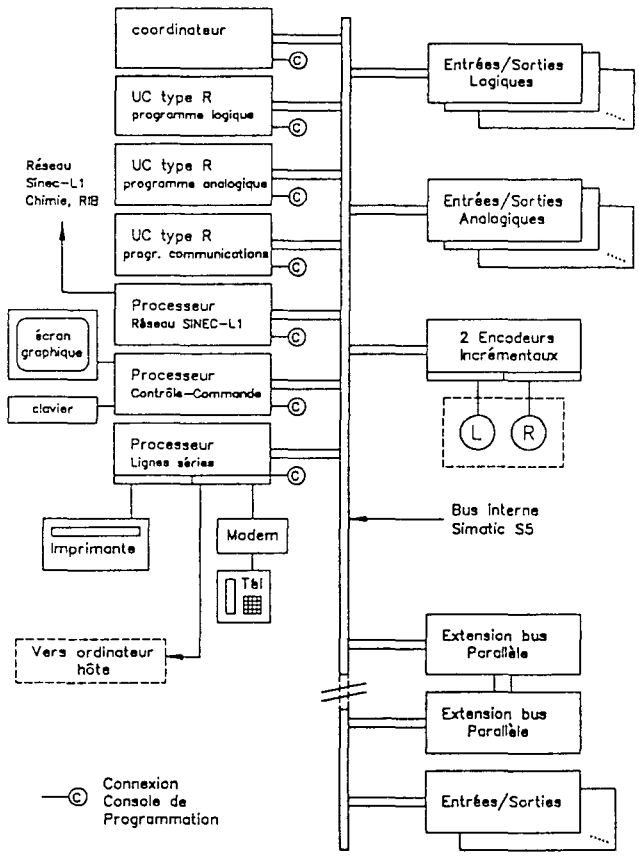

Fig.7: SIMATIC-S5 system as implemented for CYCLONE 30. The system also provides 2 shaft encoders to fine-tune parameter settings.

or target material on operator request, intensity monitoring, intensity reduction if preset limits are exceeded, calculation of extraction energy, automatic adjustment and shift compensation of the main magnetic field, calculation and proposal of setting values for extraction of new beams, automatic call of the person in charge in case of a failure. The setting values for different beams are held in an archive for downloading if required.

Keitel et al.[14] report on a similar system implemented for the TRIUMF TR30 cyclotron on the basis of Allen Bradley PLC's.

4. CONCLUSIONS

Highly integrated tool kit solutions comprising hardware and software exist for well defined processes. Due to their reliability and robustness they can be used in industrial environment and are applicable to turn-key systems. Some have been implemented for compact cyclotrons and other ion implanting machines.

Partial solutions exist for systems where some flexibility and extensibility to new requirements or research about the process itself is desirable. Often more computing power and fast signal processing is mandatory in such cases. These solutions have been evaluated or implemented for medium scale systems.

No complete example exists for the application of the above mentioned tools in large scale complex machines. On the contrary, while the complexity of these systems has increased several orders of magnitude over the last twenty years the ability to reliably specify them and manage their implementation has not kept pace.

5. REFERENCES

1) Rausch, R. et al., "Common Control System for the CERN PS and SPS Accelerators", ICALEPCS'91, Tsukuba, Japan Japan, Nov. 11-15, 1991 (Proceedings to be published)
2) De Marco, T., "Structured Analysis and System Specification", Yourdon Press, 1978, ISBN 0-13-854380-1
3) Ward, P.T., and Mellor, S.J., "Structured Development for Real-time Systems", Yourdon Press, 1985, ISBN 0-917072-51-0
4) Yourdon, E., and Constantine, L.L., "Structured Design", Prentice Hall, 1979, ISBN 0-13-854471-9
5) Ludgate, G.A., Osberg, E.A., "Contemporary Approaches to Control System Specification and Design Applied to KAON, contribution to this conference
6) DATAVIEWS, product and trademark of V.I. Corporation, Northampton, Ma 01060
7) SL-GMS, product and trademark of SL Corporation, Coste Madera, California 94925
8) Lugnet, J. et al., "Evaluation de Produits d'Animation de Synoptiques", Technical Report No. 91/012 of CEN SACLAY DPhN-STAS-STP

9) Vsystem, product and trademark of VISTA Control Systems Inc., Los Alamos, NM 87544
10) Clout, P. et al., "Past, Present and Future of a Commercial, Graphically Oriented Control System", ICALEPCS'89, Vancouver, Canada, Nucl.Instr. and Meth. A293, pp. 456-459, 1990
11) Wilkinson, N.A., Ludgate, G. A., "Experience with the Application of Object Oriented Techniques in the TRIUMF Beam Line 2C Control System", contribution to this conference
12) SIMATIC-S5, STEP-5, COROS are products and trademarks of the SIEMENS AG, Berlin, Germany
13) Lacroix, M., "Il est possible de contrôler un cyclotron compact avec des processeurs industriels, en vue d'en rendre le fonctionnement automatique", thesis submitted to the Catholic University of Louvain, Belgium, 1989
14) Keitel, R. and Dale, D., "The TR30 Control System a Case for Off-the-Shelf Software", ICALEPCS'91, Tsukuba, Japan, Nov. 11-15, 1991 (Proceedings to be published)