CURRENT TRENDS IN ACCELERATOR CONTROLS:
THE ISSUE OF APPLICATION SOFTWARE.

AXEL DANEELS
PS Division, CERN, CH-1211 Geneva 23, Switzerland

Abstract Operation of industrial processes is unthinkable without a fleet of computers. Under the impulse of the progress in various fields of informatics, it is anticipated that one of the major issues of the 90's will be the application software. Accelerators are not escaping this trend. Their increasing complexity requires more and more application software, reaching limits which tend to go beyond the production capacity of even large organizations. Still, the level of automation obtained is not following this trend and is stagnating. This paper suggests that in the light of current developments, the implementation of application software might be rendered more efficient, opening the way to higher levels of automation and hence better physics.

INTRODUCTION

The growing complexity of accelerators imposes escalating requirements on their controls. The fall in prices of electronics is offset by the large quantities of interface hardware and processors that are brought into play. In the 70's, controls amounted to 5 to 10 % of the cost of a machine; it now runs towards 20 %, contrasting with the climate of economy and reduction of resources in most laboratories. Application software is particularly affected as huge quantities need to be developed whenever a new machine is projected or its controls renewed: in the wake follows an equal amount of maintenance. The level of automation achieved stagnates and little progress has been made since the early days when computers were experimented for data acquisition and remote manual setting. Some 20 years later the software is supplying supervisory controls and requires hundreds of man-years.
CASE STUDY: THE CERN PS (CPS) CONTROLS

This escalation is illustrated by the CERN 800 MeV Booster synchrotron. In the 70's, 2500 control channels were controlled from a centralised computer system involving 1 IBM 1800 and 2 satellite computers (for console and analog function generation). Operation was helped by 30 application programs. A decade later, its controls involves at least 5 mini-computers and 30 micro-processors. The number of process devices (power converters, beam instrumentation, etc.) did not increase significantly, but the number of control channels rose to ~10,000. A further "explosion" came from Operation: requirements for short cycles, pulse to pulse modulation of the beam, general purpose consoles in the main control room, and local consoles. In the 80's, operation and exploitation are assisted by 150 dedicated applications, 120 in common with all machines of the CPS complex, and 50 for beam modulation (also in common) (1); in total more than 300 application programs. It should be no surprise that application software for large modern machines run well into the 100 man-years, as was confirmed in 1984 by a survey at different laboratories (2).

Though the functionality has grown by an order of magnitude, it does not provide much more than supervisory controls. Model support allowing to test proposed settings on a model before sending them to the hardware, are being experimented, but are not yet included in closed loop algorithms as e.g. in chemical and power generating plants. Further levels of automation are still academical exercises. This stagnation is partly due to the fact that, after having installed the controls, part of the staff is often directed to other activities, whereas the remaining staff is dragged into maintenance and exploitation which quickly run beyond their resources. The CPS package amounts to 150 man-years. It went into routine exploitation from 1986 onwards. The group is assisted in this task by other groups of the division. Despite the 40 man-years the division spent on applications (maintenance and new development), more than 10 man-years could not be delivered: this deficit accumulates with the years.

It could be argued that the application software is not of the quality wished for: hence its high maintenance cost. Quality is evaluated by the degree of userfriendliness of the interaction, of the tools for development and maintenance, robustness, reliability, etc. The
quality one obtains is often the quality one is willing to pay for. As an illustration the CPS controls is compared with an extreme case: the Safeguard project, a ballistic missile defense system designed to respond to attacks by intercontinental ballistic missiles. It involved 2 million instructions: ~ 1/3 real-time applications, ~ 2/3 support and utilities (3). We assume that its userfriendliness and reliability are much better than at CPS. If the CPS application software controls had been designed and implemented in the safeguard style, it might have amounted to 1.3 million instructions and around 3000 man-years.

DIFFERENCES WITH INDUSTRIAL CONTROLS

Industry pioneered computer controls 5 to 10 years before it was introduced in accelerators. Today numerous plants are fully automated incorporating control strategies based on models which take into account the current working condition of the plant and using expert systems to switch to the appropriate strategy as conditions change (4). Obviously there are differences between industrial processes and accelerators which do reflect on their controls. Accelerators are characterised by constantly changing modes of operations. The number of channels, data, high sampling rates and real time responses, put their control requirements at the higher end of the scale. Industry tends to farm out the design, implementation and often also maintenance of its controls to specialised companies. The latter are in general more eager to work for industrial processes which represent a market much larger and more stable than accelerators. From the managerial point of view, they apply more rigour throughout the development of controls, thus also application software. Applications are defined early in the project, are carefully estimated and planned, get sufficient and stable staffing, undergo rigorous review at every stage of their implementation. From an economics point of view, industry has long recognised the value of controls: case studies claim process economics benefits of 5 US$ for every US$ invested in controls (5). The overwhelming contribution of applications and their high cost that is forecasted to grow to 90% of the total control cost, are considered as reflecting the demand. Industry is thus seeking ways to produce more efficient software with high economical return.
TRENDS IN TECHNOLOGY

A number of standards with wide industrial support and going towards open architectures are now being followed by several laboratories.

Control Hardware Architecture

Controls is in general structured in levels of functional hierarchy. The device level, nearest to the hardware, interacts with I/O modules, needs speed, responsiveness of real-time multi-tasking executives and guaranteed response. The next level acts as data concentrator for combinations of equipment. Next comes the operator interface. Finally there are utilities: database, library, software development, modeling, etc. The hardware architectures range from purely hierarchical ones to flat ones. In the purely hierarchical ones every functional level resides in a computer and communication happens through the various processors in the hierarchy. In the flat one all processors are connected to a single backbone and communicate directly. The flat one is thus less sensitive to failures. Uniform network interfaces and communication protocols allow to connect any processor to the network. The disadvantage of concentrating all communication on a single network is circumvened by subdividing the network in segments communicating over bridges.

Networks and Communication

After having used non-standard networks for their controls, accelerators now tend towards the IEEE 802 standard. Ethernet runs at 10 Mbits/sec, spans distances of 2.5 km, is strongly supported by industry, has a wide range of reasonably cheap compatible products and most computers can be connected to it. Token bus is more robust in hostile environment. Its performances can be predicted making it attractive for real time controls requiring guaranteed responses. It is less supported and more expensive. Token ring also has predictable performances and spans longer distances than Ethernet. Reasonably priced products are appearing on the market. Some laboratories are adopting Ethernet; others prefer token rings because of their large distances. The transmission control protocol - internet protocol (TCP/IP), is generally accepted as a standard to communicate between different systems, and is available on many of operating systems.
Fieldbuses
Between the data concentrators and the equipment processors, the sophistication of a LAN cannot always be justified on basis of economy and because of the required noise immunity. There is no agreement yet as to what bus to choose and many different examples are found in laboratories: IEC 625-1, IEEE 488, Fastbus, MIL-STD 1553, Multibus II, the German PDV-Bus, ... and for historical reasons also CAMAC-serial.

Interface Hardware
Most controls have been implemented with CAMAC: their investment is such that they will live more years with CAMAC. For new developments several laboratories are adopting VME using MC 68000 family microprocessors at the process end.

User Interface
Consoles tend to be "invaded" by workstations. Their processing power and windowing techniques allow multiple applications to be active simultaneously on a screen. The user can control the window size, position and shape, and move data between them. A standard system, X window, allows to build effective graphics, is available on most workstations and allows applications to be ported on different hosts.

Operating Systems and Languages
At the level of workstations the world seems divided into VMS and UNIX. Under the impulse of the Open System Foundation (OSF), initiated by the larger computer manufacturers, UNIX seems to become a de facto standard outphasing VMS. Various laboratories in Europe have adopted UNIX.

For microprocessors the situation is not so clear. Many real time operating systems exist with more or less different size of kernel, range of services, speed, machine independency, program development facilities, documentation and cost. At the workshop organised by the European Physical Society, EPS, interdivisional group on Experimental Physics Control Systems, EPCS, not one single system emerged out of SOS, MTOS-UX, OS-9, VRTX, SRTX, VXWORKS, etc. (6). OS-9 is claimed to be like a mini UNIX and provides a wide range of facilities for program development in Basic, C, Fortran, Pascal.

C seems to appear as the successor of assembler languages. It combines assembler like facilities (e.g. access to low level hardware features) with high level language ones (e.g. data structuring and
control flow facilities) making C suitable for real-time applications. Some versions of C include object oriented concepts which seem to orient the more recent application software designs. Though ADA is being introduced as a high level language, no accelerator controls are using or planning to use it. Fortran is still used for modeling and other large calculations. Interpreters, e.g. NODAL, are used for test, commissioning, trouble shooting, diagnostics and experimentation.

Data Bases
Data bases intimately relate to applications whose unique source of data they are. The relational model in which data are organized into interrelated two dimensional tables, gains wide acceptance because of its flexibility. Data for real-time controls are extracted from the central data base and downloaded to target processors. Data for documentation and management stay in the central host. The standard query language, SQL, has been adopted for data base manipulation.

TRENDS IN APPLICATION SOFTWARE
Methodologies
In the quest for more efficient software, research has been undertaken to understand its development process and lead to methodologies (7). CASE (computer assisted software engineering) tools exist for analysing and designing applications following the concepts of structuration. They verify the coherency and consistency of applications, produce source code and automatic documentation. One of these, Teamwork, used at CERN for designing software for SPS/LEP controls and the large experiments ALEPH and DELPHI, is now widely adopted.

Structures
Software structures yielding efficient but more economical controls are being explored. They aim at good quality applications to be implemented by anyone (device specialists, operators, physicists) and at reduced maintenance effort. They hide all system, process and operational intricacies, and provide a standard frame for most application programs so that they execute properly with adequate performance.

CERN SPS pioneered the data modules which were seen as system variables in NODAL. Their standard interface makes them easy to call by any application hence providing the flexibility for so-called
people's programming. CERN PS introduced a layered structure throughout its application software (8). The data module was refined and extended to control CAMAC I/O, process devices and beam physics variables (bump, harmonic orbit corrections, etc.). The higher level structures concerned respectively control and data acquisition of entire subsets of the machine and interaction with operators. Finally there are central facilities: program selection, alarms, archives, etc. These layers correspond to the ones information passes through on its way from the process hardware to the operator. Each layer has its dedicated functionality and is structured in modules with high functional strength but low intermodule coupling. Those functions that should be performed as a rule by all modules in each layer are collected in templates. Templates enforce standardisation and are an effective means for producing more reliable, reusable, readable and maintainable code. Similar approaches were followed by LANL for the Fusion Material Irradiation Test Facility (9), and the Proton Storage Ring (10).

Structuring is further refined by introducing object oriented concepts (11). In that concept a physical entity, e.g. a power converter, is modeled by an "object", consisting of data (e.g. type, location, authorised functions) and procedures to manipulate that data. Procedural code and data are separated; for their protection, data are only accessible through routines. Code and data are encapsulated in a module, hidden from the user. New objects can be configured from existing ones inheriting all properties of their components (e.g. a bump composed of 4 power converters can be switched on or off through its 4 components). New objects are in turn available as building bricks for further constructions. This concept is powerful at achieving open ended applications by constructing higher levels of abstraction from existing components. Further rationalization is expected from generic software packages. These collect all functions the objects at each level are expected to perform (management function) and standard operation facilities which may operated on any object at any level (data acquisition, surveillance, alarms, logging, and setpoint control, sequences for initializations, setting-up, closing down, diagnostics). Such packages are complemented by tools to configure their functionality to the specific needs of the accelerator.
Error Handling and Diagnostics

Error handling and diagnostics are often the Cinderella of application software; the programmer concentrates primarily on the control problem he has to solve and pays little attention to properly treating error states. Still, efficient troubleshooting and diagnostics rely on accurate and consistent error information. This may partly explain the slow progress of expert systems for accelerator diagnostics. Noteworthy is LEP's attempt to systematic structuring its fault states (12).

Operational Control Protocols

Process devices of a same family, sometimes behave differently for similar functions. This confuses the operator and must be normalized by software. It is more effective to build this normalization into the devices themselves so that they all behave identically, are controlled through a single software and operated unambiguously and consistently. This requires the definition of a protocol that is based on an agreed model of their behaviour and a device specific message format. CPS controls have shown the value of a well conceived protocol for power converters (13). Under the auspices of the EPS interdivisional group on EPCS, a number of laboratories are working on a laboratory wide protocol for power converters; a similar activity is about to start for beam instrumentation also.

Management, Cost Estimation, Quality Control

Despite recent positive examples, management of application software is still considered with scepticism. The Europhysics Conference on Control Systems for Experimental Physics held in Villars-sur-Ollon in 1987 (14) stressed that success of a software project to a large extent depends on management also. A well managed project assures a more economical and efficient system by compromizing between the goals (performance, flexibility, ...) and the constraints of economy and technology.

Cost estimation is iterative and becomes more accurate as the design progresses and the constraints all become apparent. It is still empirical and needs a posteriori evaluation of the effort actually spent to verify the estimates. Crosschecking with those of similar projects, also in other fields, allows to improve one's estimating skill by taking into account the level of experience of the staff, use of high level programming tools, reliability, real time response, etc.
Industry or large government organization apply quality control as a rule to ensure that the applications are up to their design. The attempt to introduce some limited quality control at CERN PS, failed due to lack of resources. According to industrial software producers, not less than 15% of the resources should be invested in this activity; probably beyond the possibilities of individual CERN divisions, true quality control might be a real issue on an overall CERN scale.

RETURN ON INVESTMENT

Due to the lack of detailed cost figures elsewhere, reference is made to the CPS controls. Though the precepts of previous chapters were not all followed (as they were either not available or could not be applied), it can be shown that the development of a 44 man-years kernel package that is common to all CPS accelerators, saved in the 100 man-years on development and maintenance. LANL implemented a similar package for applications in workstations, using VMS and Xerox Park style interaction. That package was implemented first on the Ground Test Accelerator and next ported to the Argonne Telescope project, where it was made to work in a few weeks. Brown Boveri used a similar approach for a 131 man-years application package for a power plant (15) and claims now to be able to keep the maintenance down to 2% per year of the development effort because of higher reliability, more efficient maintenance, and by the opportunity such package gives to incorporate new requirements, normally included in maintenance, in a next release.

CONCLUSION

Following the tendency in other fields, accelerators are trying to organize their controls application software to be more efficient by designing packages which can be configured to different types of machines, adapted to their specific requirements and expanded to higher levels of automation. Such packages are more economical if they are applicable to more processes. Several laboratories are speculating that it might be conceivable to develop packages that might be used in common. Auspicious is the agreement on the use of workstations, UNIX, X windows, TCP/IP, VME based interfaces, the endeavour towards standard operational control protocols, introduction of CASE tools, etc. Some divergence still remains, primarily as to operating systems at the
process end. Fortunately the agreement towards C as a common language might render the portability of the process end of such packages less problematic. CASE tools should enhance their quality and facilitate collaboration between laboratories. In view of the hundreds of manyears of experience that are floating around in all laboratories one should be able to install the basic supervisory controls that are now re-invented at every new accelerator as a standard package "as easily" as operating systems.

REFERENCES