© 1985 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

IEEE Transactions on Nuclear Science, Vol. NS-32, No. 5, October 1985

ENGINEERING A LARGE APPLICATION SOFTWARE PROJECT: THE CONTROLS OF THE CERN PS ACCELERATOR COMPLEX

G.P. Benincasa, A. Daneels, P. Heymans, Ch. Serre
CERN, 1211 Geneva 23, Switzerland

Abstract

The CERN PS accelerator complex has been progressively converted to full computer controls without interrupting its full-time operation (more than 6000 hours per year with on average not more than 1% of the total down-time due to controls). The application software amounts to 120 man-years and 450'000 instructions: it compares with other large software projects, also outside the accelerator world: e.g. Skylab's ground support software.[1] This paper outlines the application software structure which takes into account technical requirements and constraints (resulting from the complexity of the process and its operation) and economical and managerial ones. It presents the engineering and management techniques used to promote implementation, testing and commissioning within budget, manpower and time constraints and concludes with experience gained.

Introduction

The PS accelerator complex comprises 2 linear accelerators, a 4 ring synchrotron booster (PSB), a proton synchrotron (PS), an antiproton accumulator (AA) and various beam transfer lines. It provides various high-energy physics experiments and other accelerators on the site with beams of different particles in pulses of 1.2 sec. The beam characteristics are modified from one pulse to the other ("pulse-to-pulse modulation") in a repeated sequence called "supercycle" which is changed daily, if not hourly, by the operators. There may be up to 24 beams of 8 different types in a supercycle.[2] Any beam of the supercycle in any accelerator is operated concurrently and in a similar fashion from any of 7 identical general purpose consoles in the main control room.[3] Similar consoles, of reduced format, are located in remote areas for local experiments. The effort invested in the controls amounts to 200 man-years: 60 for hardware interfacing, 10 for consoles, 10 for system software and 120 in application software. The hardware budget was 14 MSfr. (in 1976 money).

The new controls is based on a network of 20 minicomputers and 100 microcomputers interfaced to the process hardware through serial CAMAC. The minis are Norsk Data machines with a multi-program operating system SINTRAN III. There are three categories: the console driving computers, one per console; front-end computers, one per accelerator; central activity computers for beam sequencing and synchronization, message handling, etc.. The micros are TMS9900 located in CAMAC in CAMAC crates as auxiliary crate controllers. They execute hard real-time activities: pulse-to-pulse modulation and local data buffering. Programming languages are: ASM for the micros; the manufacturer provided intermediate level language NPL and the home-made P+, an extension of PASCAL, for the minis; and the interpreter Nodal for tests, experiments and not time-cricital applications.[4]

Application Software

Definition: System software embraces operating systems, network software, programming languages, etc. Application software covers all aspects of process controls ranging from operator interaction and displays to process and equipment control algorithms and data processing.

Requirements: The process requirements deal primarily with performance: 1.2sec cycle, pulse-to-pulse modulation requiring 100ths of set points to be modified from one cycle to another within a window of 30msec. Operators at the console, either in the main control room or in remote areas, require safe and "friendly" interaction with responses within 0.5sec, comprehensive error reporting, up-to-date information as to the status of the machines. In addition to selecting analog and video signals, interacting with the alarms system, they run up to 8 different repetitive displays (200 refreshed data per second) from program selection trees. The physicist relies on the usual 95% availability of the beam for his experiment: the controls availability should thus be around 99%. From the economics point of view, the controls should have a low production and life cycle cost: the application software should provide a frame which is versatile, extendable and maintainable for future expansion and new applications.

Constraints: First, it was imposed to use SPS developments. A pilot project was set up in 1977 to evaluate the performance of the SPS controls in the PS specific environment.[5] It was found too slow for the fast cycling PS. The same brand of computers, the same network system and Nodal could be used, but the SPS operating system was not adequate and the application software needed redesigning. Next, the conversion had to be completed without interrupting the full-time operation of the accelerator complex, and preserving its 95% availability. The project suffered from personnel shortage, as permanent staff was drained by higher priority accelerator and physics projects. One had to fall back upon Summer and graduate students, fellows, operators, temporary programmers. They had a high turnover, as their contracts ran from a few months to 2 years. Around 90 people of all types took part in the application programming !

Application Morphology and Ancilliary Tools

The application software is structured in hierarchical layers of modules. Starting from the process hardware, the layers are defined as follows:[6]

- RT : Real-time Tasks, servicing real-time events,
- IM : Interface Modules, hiding the protocols of the CAMAC hardware modules,
- EM : Equipment Modules, hiding the details for controlling the equipment,
- CVM : Composite Variable Modules which handle beam physics parameters.
- PM : Process controls Modules,
- OM : Operator's interaction Modules, dialoging with the operator.

The RT run in the micros, IM, EM, CVM and PM in the front-end minis; OM in the Console computers.

Except for the low level RT, all modules are passive and run under supervision of managerial tasks whose collection is called the skeleton. Further, there are general facilities for operation (knobs, analog and video signal observation, trees to call programs, alarms, etc.) and hardware controls (beam synchronization, pulse-to-pulse modulation, timing and power supply controls, etc.). They are "general" because they apply to all accelerators. Their degree of generality depends on the standardization which is achieved for the operation, and the controls protocol of devices. The bulk of the software involves applications which are specific to every accelerator. For cost effectiveness they should have a standard format, and production tools (software templates, editors, etc.) have been developed as part of the application software project

0018-9499/85/1000-2029$01.00© 1985 IEEE

Finally, the project size and constraints required appropriate management tools for planning and progress monitoring.

Engineering Techniques

Participative design means that "customers" (operators, machine physicists), "suppliers" (hardware, console and computer specialists) and all of the permanent application staff took part in the design. This was achieved by a proper organization of the software team and the design process.

Structured, top-down design was enforced by the morphology of the application software which lends itself naturally to the use of lower level modules (CVM, EM, IM) and by the convention of using program structure diagrams as design tool,[7] or P+ as design language. The design resulted in a formal document reviewed with customers and suppliers, and their agreement was obtained before starting the programming.

Structured coding was quite automatic in case of P+; for other languages one attempted to ban non-structured statements (e.g. "GO-TO").

A prototype was developed to evaluate early in the implementation cycle the performance of the control system's most critical functions: i.e. those performed by the skeleton and some general facilities.

Management Techniques

The project was implemented by a medium size team averaging 20 people with a high turnover. Most of the 90 participants were not familiar with accelerator controls and needed training for at least 3 months before becoming productive. The team was therefore organized so as to keep tight control over its activities.

The structure of the Application Software Group reflects the structure of the application software itself. To stimulate motivation, the layers of modules were grouped into independent packages and assigned to teams as individual projects. The teams were thus fairly autonomous. There were 3 teams each headed by a team leader with many years of experience in controls:

- the Equipment Controls team concentrated on low level controls: EMs, IMs and RTs for pulse-to-pulse modulation and data buffering.
- the Process Controls team covered OMs, PMs, CVMs.
- the General Facilities team had the responsibility of the beam sequencing system, the trees, etc.

During the design phase these teams were kept small so as to be efficient study teams. They included the more junior engineers of the permanent staff. When it came to implementation, they evolved to programming teams by hiring additional programmers.

Participative design and implementation: the application software project was conducted collectively by the project leader and the team leaders: they formed the "layout team". From the user requirements and specification, the project leader made an initial design and submitted it to the layout team. Together with users and suppliers, the layout teams analysed the requirements, refined the structure, defined inter-module interfaces, documentation standards, and discussed priorities and implementation strategies. It coordinated the activity of study and programming teams, and ensured cross flow of information between them. The study teams each designed a software layer in detail, proposed interfaces, identified and designed the various applications for which they also defined programming standards that were enforced by templates. Results and problems were discussed with all members of the study and layout team. The programming teams were responsible for the implementation of each module.

Interdependence with other controls project teams: the Application Software group had connections with other teams respectively involved in operational aspects, system software, consoles, interfaces and process equipment. The Operational Aspects team,machine engineers and operators, specified the operational requirements: the general purpose character of the consoles and their components. This was input for the Console team to construct the console hardware and software. They also defined the pulse-to-pulse modulation, operational accelerator subsets, the various control functions and the trees to call them. An application software engineer participated to evaluate the feasibility and cost of some requests. The General Facilities team then designed the beam sequencing system and the trees. The list of all control functions, and the definition of the console was input for the Process Controls team to design the PMs and OMs. Meanwhile, the CAMAC Interface team defined standard control protocols and designed and built prototypes of CAMAC modules not commercially available. The Equipment specialists adapted the equipment specific interfaces to CAMAC standards. This allowed the Equipment Controls team to design all low level software: EM, IM and RTs.

Planning and Progress monitoring: used a data base containing the list of all software modules, and programs to compute their progress. With the old control system's experience, the pilot project, cost estimation techniques,[8] every module was estimated in man-months depending on complexity, real-time requirements, language, etc. They were grouped in packages of more or less equal load and distributed among the programming teams. These packages were further distributed between the programmers taking into account their preference for individual or team work, experience, length of contract, vertical or horizontal development. "Vertical" is when a programmer produces an entire suite of modules from the OM to the RT; "horizontal" is when he produces modules of one type only (e.g. EMs). The programmer's activities were planned according to schedules and priorities defined with the users. Milestones were defined in the implementation of every module and estimated as percentage of the total effort:

- design : system engineering and detailed design, 40%,
- coding and unit test : 20%
- testing: simulation test on a special purpose computer, 10%; on-line tests without beam production, 10%; and finally, with beam, 10%.
- documentation: supposedly done all along the implementation, with a 10% provision for a final update after commissioning.

The progress was monitored every month. Team leader and programmer discussed the status of every module in terms of milestones that had been reached and adherence to schedule. This data was entered into the management data base, and the layout team discussed the overall status of the project, the progress and production efficiency. Changes of implementation strategy were decided to accomodate changes of priorities, schedules or personnel, and integration tests forecasted. Requests for addition or modifications to modules under development were closely evaluated. The technical impact, cost and drawback on the schedule was examined before the request was entered in the planning.

The Design and Implementation History

The project was implemented in 3 main phases:

Overall design: was performed by the layout team from Spring 1978 until end 1978 and took around 5 man-years. It involved: survey of the various processes and equipment: analysis and synthesis of user requirements; definition of skeleton, general facilities and production tools; definition of management style and tools.

Detailed design and prototyping: all skeleton modules, first priority general facilities and development tools were designed into details by the study

teams and a prototype constructed. This 10 man-years job ran from early 1979 until early 1980. Prototype tests started in October 1979. The skeleton, general facilities and an entire vertical suite of application modules, were tested on spare off-line power supplies; first in non pulse-to-pulse modulation mode, next in pulse-to-pulse modulation with simulated beam cycles.

Implementation in slices: the PSB controls was converted first together with all AA low-level software and some general facilities. This was a major package of 40 man-years: 35 for PSB and 5 for AA. The PSB is a small accelerator, but sufficiently significant for the controls to cover all aspects. It was a new machine at that time and almost fully computer-controlled by the old system. The application group had sufficient expertise of this machine, so as to concentrate more on the novel control software aspects, rather than on the intricacies of the machine itself. This slice took from mid-1979 to end-1980. Next followed the PS, with some follow-up of the previous slice and performance improvement. It ran from beginning 1981 to end-1982 and took 36 man-years. The 3rd slice, the PS-Ejection and AA high-level application, went from early 1983 to mid-1984, and took 22 man-years, including some follow-up of both previous slices. The last slice was terminated in February 1985 when the PS was started with its RF under computer control, at the cost of another 7 man-years.

Experience Gained

This paper emphasizes the spirit which prevailed throughout the project: standardization, structuring, modularization and management. These features were novel at CERN for accelerator controls. It changed the traditional work style and had to be introduced against prevailing scepticism. It is now generally agreed that engineering and management are just as essential for large software projects as for large hardware ones.

Standardization of operational procedures, equipment controls protocols and hardware interfaces, result in single data driven programs, applicable to any accelerator. It is further enhanced by the structuring into skeleton and specific applications, and use of appropriate production tools. It reduces the cost of implementation and also of maintenance through improved reliability. Modularization saves on production by cutting down the overall project into small ones and eases management. This is best illustrated by the later phases of the project where a small team was able to put all of the PS-RF on-line in less than a year. Except for the PSB, all subsequent slices were converted to full computer control in a more stepwise fashion. Standardization and modularization had reached a point where in shutdowns of a few days only, significant packages could be installed, tested and commissioned.

Because of the hierarchical and modular structure, the implementation could be vertical or horizontal. Vertical was preferred on the basis of people's motivation, clearer definition of responsibilities and better quality programs. However, as modules are programmed in sequence, the elapsed delivery time is rather long. Horizontal implementation implies parallel development of all modules belonging to a system: the delivery time is short but responsibilities are diluted leading to maintenance problems. Unfortunately, the tight schedule and the short stay of temporary programmers, imposed often horizontal implementation.

Management was another issue. A strict implementation plan was defined, and no modifications to the originally agreed design were accepted during the implementation. They were executed after completion of the original plan so as not to disorientate programmers. Progress monitoring tools were essential to assess progress, forecast the evolution, plan tests and commissioning, and to give confidence in meeting deadlines.

The problems relate to the high personnel turnover and the tight schedule. Follow-up and maintenance were difficult when a programmer left the organization. It was difficult to motivate someone else to take over software, without having him redesigning it all. Application software is last in the implementation sequence of a controls system and is continuously squeezed between lagging deliveries from other parties and hard operational deadlines. This puts the emphasis on quick delivery, rather than on high quality software, and proper documentation. Test time in early slices was insufficient and requested round the clock work.

Participative design was introduced to enhance professional stimulation, motivation and commitment of participants, including the users. It left scope for creativity of the programmers who later were involved in dull coding, but appeared to be less satisfactory than expected. Design of intermediate layers made them loose oversight of the entire structure and was found too abstract. Vertical implementation was preferred. The rigidity of the skeleton was not considered an obstacle.

Technical problems resulted from insufficient design of data structures. The design concentrated on the functional logic and the corresponding data was derived from it more or less ad hoc. Similarly the system was designed for steady state operation of the accelerators and controls, overlooking transitory states such as start-up and shutdown. The depth of design was never clearly defined: if the design was not sufficiently detailed, the initial cost estimates could not be corrected properly and the product overran the schedule.

References

[1] F.Terry Baker, Structured Programming in a Production Programming Environment, IEEE Transaction on Software Engineering, June 1975.

[2] P.P. Heymans et al, "Concurrent control of interacting accelerators with particle beams of varying format and kind," presented at the EPS Europhysics Conference on Computer in Accelerator Design and Operation, Berlin, West-Germany, 1983.

[3] F. Perriollat, et al, "The Main Operator Console of the PS Accelerator Complex," presented at the Particle Accelerator Conference, Vancouver, May 13-16, 1985.

[4] B. Carpenter et al, "System Software for the CERN Proton Synchrotron Control System," CERN 84-16, 20 December, 1984.

[5] G.P. Benincasa et al, "Structured Design Benefits to a Process Control Software Project," presented at the ACM Sigmini Symposium on Small Systems, New York, August 2-3, 1978.

[6] G.P. Benincasa et al, "Design Goals and Application Software Structure for the CERN 28 GeV Accelerator Complex," presented at the 2nd IFAC/IFIP Symposium on Software for Computer Control, Prague, 1979.

[7] R. Cailliau, "Program Structure Diagrams," PS/CO Cookbook, 23 November, 1976.

[8] B.W. Boehm, "Software Engineering Economics," Prentice Hall Inc., Eaglewood Cliffs, New Jersey, USA, 1981.