Why Physicists do not Stop Writing Their Own Accelerator Control Programs?

Yu.I. Eidelman
The Budker Institute of Nuclear Physics, Russian Academy of Sciences
Prospect Lavrenteva 11, Novosibirsk 630090, Russia

Abstract

The connection between four hardware and software components of the control system an accelerator and/or any type of large physical facilities is analyzed. Software structures used in the control systems are considered. A thesis about the software products "authorship" distribution between professional programmers and physicists working at accelerators is formulated.

1 Introduction

This report was already written, when the author came across a CD-copy of the Proceedings of the ICALEPCS'95 Conference (Chicago). It turned out, that the theme of this report was very actively discussed during the conference. Many authors expressed their views on this subject. The summary of the lunchtime discussion "The interface between operations and controls" can be found at [1]. This report should be treated as the belated contribution to the discussion.

The accelerators, or, better to say, large accelerator complexes, have existed and developed for about fifty years. During this time several generations of such facilities changed. Their control systems evolutioned correspondingly. Now they are represented by control rooms, filled with terminals and displays, large number of computers, interconnected in some odd manner, and hundreds (or maybe thousands) of microprocessors, which are literally "padded" into the control hardware. But control hasn't become easier. Just the amount of routine work decreased. And one must has to put himself to more and more strain to solve the new complicated tasks, arising during the operation of modern accelerators. At the same time one type of job remains constant. This job is performed by physicists [1]. They deal with the development of software control systems. Let's e the reasons for this phenomenon and try to understand what will we encounter in future.

2 "Oceans" and the triads of "continents" of control systems

The "Standard Model"[2] of the control system hardware in each accelerator complex includes the following three "continents"[3]:

- the low level electronics and maybe microprocessors, located the closest to the accelerator hardware (power systems, the systems of beam diagnostics etc.);
- front end controllers and computers, serving this electronics;
- workstations, which are used to control the accelerator and its subsystems.

These "continents" are surrounded with an "ocean", under the name "networks". The specific implementation of these components can be very different. The author analysed attentively the proceedings of the ICALEPCS'91 (Berlin), to find the layouts of the hardware part of control systems, which will be "good" and a "bad" illustrations of this idea. The layouts of 26 accelerator complexes, of several control nuclear fusion facilities, and even of a pair of large telescopes were investigated. In all the cases it was easy to extract all the three "continents" and the "ocean", connecting (but not separating!) them. So, it became useless to demonstrate the "best" and the "worst" schemes.

Correspondingly, we can speek about four software components for four hardware components:

- programmed logic on the hardware level (in some blocks and/or controllers);
- a special server program between the hardware and workstations;
- the control programs themselves, running in the workstations;
- the code to handle the user data, being transferred via the networks by the standard (and/or non-standard) protocols.

The meaning of the first and the third components is clear. But the second and the fourth should be commented. Of course, control systems are very different and strongly depend on the size of the facilities, on a large number of various factors, and especially on a precise implementation of the complex as a whole. But when we speak about modern accelarator complexes and even about their separate subsystems, the next dangerous situation appear sooner or later. Two *different* control programs try to work with the same physical element simultaneously using the *same* control channel. If the software is designed in such a way that each control program interacts with the hardware itself, the conflict is unavoidable – these programs will interfere with each other.

[1] It should be determined, that the work specifics of the BINP accelerator complexes is that the operations personnel, "the people, who do the 24hr a day running of the machine for physics production", are the same physicists, "who come to do machine development session" [1]. By the way, the same author writes that "some operations personnel are engineers or physicists with several years of hand-on experience of running the machine(s)" [2]. From here on the term "physicist" will be used in both meanings.

[2] This term was introduced successfully in [3].

[3] Of course, there was a temptation to call these hardware components "whales", swimming in the ocean. But this image is inappropriate in describing the connections between these components.

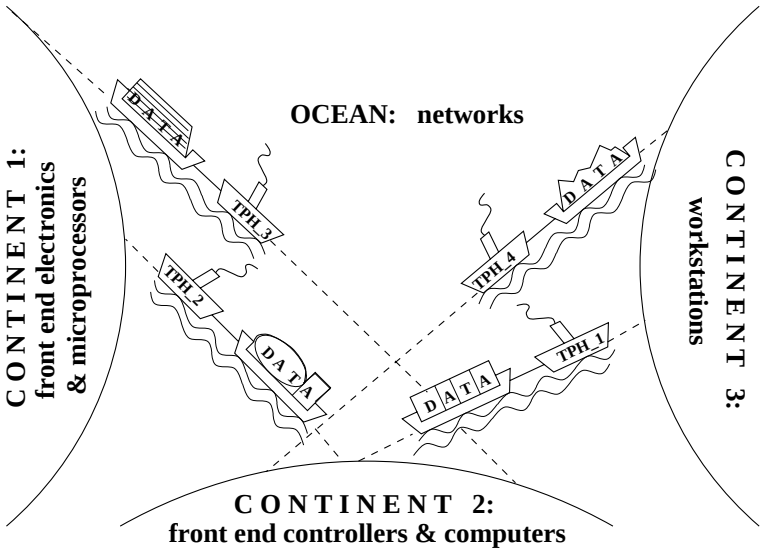

Figure. 1. The map of the system control world.

Even a harmless case, when two programs make duplicating requests to the hardware is unpleasant. It results in extra time, spent on multiple execution of the same actions, which could be done only once.

This conflict can be resolved by a special server program, which gathers requests from different control programs. It is one and the only program interacting with hardware and distributes the results to the "customer" programs. These results are taken from the dynamic database, which is maintained by this server.

Recalling more than twenty years of experience in the VEPP-4 complex operation, it is necessary to mention the nice results of this approach to the software design. Many of the eleven home-made computers Odrenok [4] run the programs called "banks" which act just like servers. They eliminate the possibility of such situations. Attentive analysis of the software of other accelerator complexes confirms the validity of this approach.

It is worth saying some words about the user data handling sent via the networks between "continents". Even when the existing protocol types (TCP/IP, MIL STD etc.) are utilized, some private protocols must be developed. They operate over the standard protocols and describe the structure of the data being sent. It provides, for example, understanding of the server's requests by the controllers, and understanding their responses by the server. All of the above can be illustrated by Fig.1, which shows three hardware "continents" in the "ocean" of networks. Possible different types of standard protocols are depicted as different tug names (TPH is the Transport Protocol Header), and diverse organization of data is presented by diverse cargo types on the barges. A very important point is that there are no direct links between the "continent" 3 (client programs in workstations) and the "continent" 1 (the front end electronics and/or microprocessors). Such links are typical for small facilities, where the "continent" 2 is excessive and the data are towed directly between the "continents" 1 and 3.

3 Software structure

It should be mentioned that four software components are very unequivalent. The first "continent", which directly serves the hardware, is just a set of very specialized drivers. Depending on the level of the block's intelligence, these drivers can be even embedded into the block ROM. Less clever hardware is operated by a program/process in the corresponding intellectual controller. This process must also have a code supporting the data interchange with the next "continent" of software.

As was mentioned above, this "continent" is a specialized server, connecting the hardware with client programs. Depending on the distribution of intellect among the hardware microprocessors, controllers and workstations, it will run immediately in the controller(s), or in a special computer, or just in one of the workstations. The design of this server depends, to a great extent, on the organization of the hardware and the network(s). The server will probably work under some OS and will use standard communication protocols. In this case many of its components (handling block's interrupts, queues dispatching, maintaining the connections with workstations etc.) can be implemented via the standard services of this OS. But there is a server part which reflects the exact hardware configuration, logic of the facility operation. This part doesn't readily exist anywhere, and must always be created together with the facility.

And the third software "continent" is constituted of a set of control programs running in the workstations. These programs always work under the control of some OS. It obviously makes the creation of such programs easier. But noth-

ing more. All the tools and utilities (EPICS, LabVIEW etc.), supplied with the OS or separately, will never cover all the demand tasks, arising during the process of accelerator complex operation. So, this level of software also can't be filled only with standard programs.

4 Who writes the software?

Now a question arises: who fills the "holes" in the control system software? The answer depends on which part of the software we are talking about.

Lower level. The drivers for individual blocks are relative simple. Of course, they can be written not only by professional programmers, but by physicists or engineers as well, and even by equipment specialists ("who come to debug and fix equipment" [1]). This level also contains a dispatcher. Its task is not only to receive/return the data, but to maintain the organization of these data. The structure of the dispatcher is a function of the hardware implementation and the logic of the facility's work. This part of the dispatcher (and, of course, the reciprocal part of the server) can't be made in such a way as to satisfy all the control system's requirements without close cooperation between a professional programmer and a physicist, who "feels" how his facility functions. And it will be even better if this part will be written by the physicists themselves – some of them will for sure have the appropriate programming qualification (or want to gain it). The experience of BINP is a good proof of this statement.

Middle level. One part of the server program was described above. But it also contains other not mentioned components. These include the database support[4], logging the history of all operations, access control and security, and many others. Undoubtedly, this part of the server is sphere of action for the professional programmer.

Top level. All of the programs of this level will never be completed. The facility lives, new tasks appear, as new solutions of already solved problems do. So, the existing programs are always updated and replaced with new versions, and brand new software appears. The facility is condemned to failure if physicists and engineers don't take a hand in this component of software. Every problem with existing programs (the old but non-revealed error arises, the program must be urgently modified for a new situation etc.) becomes absolutely unsolvable timely (during the routine operation). Long and tormentful is the waiting of a programmer, that is why it is very difficult to quickly explain what is desired. As a rule, it can be clearly understood and formulated just in the process of modification of the program or while writing a new one. So, a physicist should better pass through himself. And he alone will be to blame.

But even standard tools and utilities are not always satisfactory while being universal. It is so because of their universality. But self-made programs are always problem-oriented and solve given tasks optimally. Just a simple example. How long will the program, working with a hierarchically-organized data with a big depth of nesting and branching, "ascend" from the most distant data element to the "root"? More than likely it will require clicking the mouse as many times as many levels exist. It is okay everywhere, except the facility, controlled in the real time – here this move should be made in one step.

So, the top level software includes not only standard, but also the "natural" product. And here the programming environment being used is very important. It is clear, that it must provide simplicity of modification of existing software, quickness of new programs development and their integration into the already working system. It all has to be done by physicists and engineers in close cooperation with programmers.

Let's end this section with a citation from W.McDowell [5]: "The APS physicists were dissatisfied with aspects of all these solutions (the porting of several scripting languages to EPICS – *Yu.E.*) and asked for a simple, interactive language without the baggage of a commercial package or the remnants of solutions from other accelerators. To this end **two APS physicists** (*emphasized by me – Yu.E.*) have developed an interactive language called GUS (General-purpose data acquisition UNIX Shell). This language has been used extensively at the APS magnet measuring facility running on IBM-PCs and has been ported to UNIX". There is nothing to add.

5 Conclusion

So, what will we find in future? Work, of course. Work on inventing new experiments, also designing new accelerators, their commissioning, etc. And between this all – writing more and more programs to control our facilities. Without them we'll always feel that the accelerator doesn't work as we want. It to so, a physicist, as always, is more than just a physicist!

References

[1] R. Bailey. "The interface between operations and controls: brief summary of the lunchtime discussion", Proc. ICALEPCS'95 (Chicago). CD-copy of the Proc., pp. 333-334.

[2] R. Bailey. "How can operations get the applications software that they want?", Proc. ICALEPCS'95 (Chicago). CD-copy of the Proc., pp. 1-7.

[3] A. Götz et al. "Experience with a standard model'91 based control system at the ESRF". Proc. ICALEPCS'93 (Berlin). NIM **352A** (1994) 22-27.

[4] G. Piskunov. "CAMAC-embedded 24-bit computer", Autometriya, N4 (1986) 32-38 (in Russian).

[5] W. McDowell et al. "Status of the Advanced Photon Source and its accelerator control system". Proc. ICALEPCS'93 (Berlin). NIM **352A** (1994) 13-15.

[4] It is absolutely clear that the level of satisfaction by the quality of software is influenced to a great extent by the database organization. The complex just wouldn't function with a badly though out and, as a result, negligent database.