MODERN ACCELERATOR CONTROL SYSTEMS

K. Furukawa*
High Energy Accelerator Research Organization (KEK)
Tsukuba, Ibaraki, 305-0801, Japan

Abstract

Modern approaches to accelerator control systems are discussed including software and hardware implications, in view of maintaining reliability under changing requirements.

INTRODUCTION

Recent particle accelerators becomes much advanced and it tend to require well-arranged operational tools. One of the major parts of such preparations is the control system. We have made several trials in building the control systems for the present and past accelerator projects. And we have learned practical solutions and also are evaluating possible additions.

At first KEKB and Linac control systems are described as examples of accelerator control systems and operational environments. Then, accelerator controls are discussed more in general in practical views. Typical available technologies for controls are described in the next section. And argument for the reliability is also presented for the practical situations.

KEKB AND LINAC CONTROLS

There have been several control systems in KEK. KEKB and Linac control systems are briefly described as typical examples of the control systems.

KEKB Control System

KEKB is an asymmetric collider B-factory for the CP-violation study with 8-GeV electron and 3.5-GeV positron rings. In order to achieve higher luminosity and greater experimental results many active parameters are tuned and daily improvements are carried so that requirements to control system have been changed.

The KEKB control system is a standard EPICS system (experimental physics and industrial control system) with about 100 VME based IOCs (I/O controller) [1]. For the field interfaces 200 VXI mainframes through MXI interfaces, 50 CAMAC crates through serial highways and 200 ARCnet segments are installed besides VME frames. Many GPIB, RS232C devices, PLCs are employed as well. Control services are carried by different flavors of Unix computers including HP-UIX, Tru64 unix, Solaris, Linux and MacOSX. MacOSX computers manage most of the data processing and graphic displays.

Since there were five-year shutdown time between the previous project, TRISTAN, and the KEKB project, major part of the software and hardware resources was reconstructed while some part of CAMAC resources was reused.

Software development for EPICS R3.13 and R3.14 applications are carried with Capfast and MEDM(DM2k), and others. The number of EPICS records is about 250 thousand spreading over those 100 IOCs. While some of the hardware related algorithms are programmed in dedicated record types or as record links, however, most of the control and operational algorithms are carried with scripting languages including SADscript/Tk and Python/Tk. The reasons for the scripting languages are rapid development and wider coverage of physicists and operators. It is also affected by the fact that IOCs have to be rebooted when new records and record links are implemented.

Among more than 200 operation programs the KEKBlog archiver and viewer, the KEKB-alarm alarm handler, the Zlog operational logbook system and KEKB optics panels are some of the most often accessed operation tools. KEKBlog archiver stores data just more than 2 GB per day.

Linac Control System

Electron/positron Linac has been operated since 1982, and was upgraded for KEKB injection during 1994 and 1997 with 8-GeV electron and 3.5-GeV positron. The length is about 600 m and it has 60 high-power rf stations and 400 magnets. Since it continued the operation for the PF injection during the upgrade, it utilized many components from the previous projects.

It provides beams with different characteristics to KEKB, PF and PF-AR rings, and it switches those beam modes more than 300 times a day. Since those rings are factory machines, upstream Linac is required to carry a reliable and stable beam operation and to control precisely the beam characteristics like Twiss parameters, timing and charge while new operational beam modes are added almost each year [2].

The Linac control system was rejuvenated between 1991 and 1993 just before the KEKB approval, and minor and gradual modifications are made during the upgrade for KEKB. It consists of 30 VMEs, 150 PLCs, 15 CAMACs, 30 VXIs, many Unix computers, and redundant Ethernet/IP networks. Some of the VMEs will be replaced by 20 oscilloscopes with built-in 3-GHz computers.

Its design concept was the use of de-facto standards like Unix, VME, TCP/IP, and the use of optical Ethernet/IP networks for all device controllers without any special field

* < kazuro.furukawa @ kek.jp >

networks. The concept was inherited by J-PARC controls while it inherited EPICS from KEKB controls [3].

Most of the communication in the control system is carried by home-grown RPC (remote procedure call). The overall system is multi-tier, and lower level is controlled by UDP-RPC or simple UDP protocols in order to recover failures promptly. The upper level is connected by TCP-RPC and network-wide shared memory system is also provided for read-only information. The Linac API (application program interface) provides transparent access to UDP-RPC, TCP-RPC and shared memory.

It provides three views of the Linac accelerator and beam, namely an engineering view to devices, an operational view of overall accelerator and a scientific view for the beam improvement. It also has communication links and gateways to console systems, utility facilities, and downstream accelerators including EPICS gateways.

EPICS gateways are implemented in several ways like soft-IOC, portable CAS (channel access server), and dedicated IOC with gateway programs. EPICS gateways are utilized for most of the data archiving with EPICS channel archiver and KEKBlog now, and also for operational alarms with KEKB-alarm.

Operation of KEKB and Linac

Beam operation of the KEKB and Linac is carried by KCG (KEKB commissioning group), which is composed of about 40 staffs from KEKB and Linac groups. Most of the operational panels are developed with scripting languages, and ones for beam operation are written with SADscript [4].

SADscript is a Mathematica-like language whose processor is written in Fortran and has built-in interfaces to EPICS channel access (asynchronous and synchronous), Tk X11 widgets, CanvasDraw, Plotting, KBFrame graphic libraries on top of Tk, numerical data processing like fitting, FFT, etc., inter-process communication, and of course SAD, full accelerator modeling including symplectic beam tracking and beam envelope capabilities [5].

SAD and SADscript are designed to carry almost everything for accelerator and beam operation. Its Mathematica-like list-processing functions enable the rapid development of the online operational software. Many novel ideas were tested using such rapid prototyping just after the proposals. Many of them were found not to be practical, however, some of them were proved to be inevitable to improve the accelerator. Rapidness is very important because the time is anyway limited and the circumstances are changing. Virtual accelerators are also built with SAD to help understanding behavior of new operational parameters.

Several methods are provided to define the accelerator beam-line lattice in SAD. However, it does not read a standard input format. Some hope is there to develop a new format to cover beam-line geometry and beam optics.

ACCELERATOR CONTROLS

The definition and goal of the control systems are normally delayed until the technical details of the accelerator are decided. And it is often re-defined during the operation. Thus, it requires flexibility as well as robustness.

History

At the beginning NODAL interpreter language environment made success at SPS/CERN. Then, people needed more general software tools, and ICALEPCS (international conference for accelerator and large experimental physics control systems) was started to discuss on those topics[6]. In the meantime, NODAL was chosen at TRISTAN/KEK, and SLC/SLAC built advanced control system with many micro-computers and VMS computer to realize a linear collider.

Around that time the standard model of the control systems was said to be the combination of field-network, VME, Unix (or VMS) and X-Window. Linac/KEK followed such trends with Ethernet-only field network and scripting languages. Also more software sharing between accelerator projects was hoped, but the discussion went to the definition of a class to represent a whole accelerator, which was never achieved. More general tools appeared like ncRPC/CERN, TACL/CEBAF, ACNET/Fermilab, etc. One of them, EPICS/LANL got popular maybe because its simplicity and SSC's selection, and many institute including KEK joined in the development.

Recent direction seems to be more object-oriented approaches, which is natural after RPC and can receive more software engineering benefits. And CORBA-based (common object request broker architecture) tools have been developed like CICERO/CERN, TANGO/ESRF and CORBA+Java/CERN. The environment provided by Microsoft may be used more in the future since it becomes more open and many instruments began to have it embedded.

Balance

In designing accelerator control systems, we may have to balance between many considerations. Most of the cases we cannot choose one of them, because the system has to interface with many different sub-systems. We consider implementations optimizing between strong points of different ideas under different criteria. Some people think that it was solved that what we should have. However, there are reasons that we don't have common accelerator control system yet.

Object- and Channel-Oriented

Object-oriented technology surely helps software development and maintainability based on the benefits from software engineering, which were accumulated for a long time. The system should become extensible in software and naturally it would have cleaner definitions. However, dif-

ferent people may have different ideas on control objects. CORBA and object-oriented languages provide those features.

Channel-oriented technology is simple and flat, because of that the system becomes scalable and easy to understand for many people. It would not receive the advantages much from software engineering unfortunately. However, there is a way to overcome it to develop some more software tools, and application software layers can be a combination of several technologies. Since it is simple, it can have gateway links to any other architectures.

Compiled- and Interpretive Languages

Normally two-level languages are preferable; compiled language for established algorithms and interpretive language for rapid prototyping for new ideas. The NODAL language made much success at SPS/CERN, LEP did not use it since it was considered that software in a compiled language should be more policy-driven and manageable. However, interpretive or scripting languages can handle lists very well and many of them now are object-oriented, so that they enable more attendants from physicists. Another level of management may solve the maintenance issues.

Aggressive and Conservative

A new and aggressive technology is attractive but it can be only a fashion or fad. We need to watch those and the assimilation of their essence makes the project more active. But for the full deployment it is important that whether it has good quality but also whether it becomes a de-facto standard.

The life of an accelerator is often very long compared with the user facilities and the commercially available technologies. Accumulation of operational knowledge base is stored mainly as software and database in the control system. Even well-known beam stabilization algorithms need practical methodologies. They are rather valuable, and the knowledge and the container, the control system, should be kept longer.

International and de-facto Standards

Since international organizations tend to pursue ideal solutions, some of them don't become industrial standards. Then there is not much benefit to choose them. While finding a de-facto standard is difficult, if it is found, it bring us several benefits.

Products can be selected out of market and man-power can be saved avoiding proprietary development. Also, a standard at one time would be provided with solutions for the next generation. At CERN some projects select products based on market shares. As a whole, it is better for long life-span.

AVAILABLE TECHNOLOGIES

There are many good technologies that are employed or are being evaluated. Here some of them are discussed why they got interested.

PLC

PLCs (programmable logic controller) are utilized at relatively slow controls, since its rule-based algorithms can be well-adopted in simple controls. Recent PLCs provide IP network connectivity for the both controls and management such as program development and healthiness monitoring. Some of them provide socket-based communication with response time of one millisecond over the network. Because of those features about 150 PLCs are installed at Linac/KEK for magnet, vacuum and microwave controls.

It also enables isolated controller development, which makes outsourcing of the device and the controller possible. Local panels can be attached and many local maintenance functions can be implemented, that improve the reliability of the devices.

For the software development, an international standard, IEC61131-3, defines five programming languages and their use with emphasis on naming. The standard itself contains many good insights for the control systems. It was not so popular in Japan, but recently many vendors pay more efforts to make common development environment with XML representation of resources. It should enable international shared development as EPICS.

Ethernet/IP-only Networks

As described before, a policy to use Ethernet/IP for the field network was chosen more than ten years ago. The decision was partially based on the preference to save manpower avoiding proprietary development. It was enforced by the later inclusion of TCP/IP software stack into Windows95 by Microsoft, which promote the wide acceptance of TCP/IP technology in the industry, and later it was inherited by J-PARC controls.

The policy enabled the network management simpler with commercially available network component, routing methods, network booting, failure analysis tools, etc. It also brought the cost reduction even with optical network components, which are inevitable in the Linac gallery with high-power modulators. The IP technology continuously makes the gradual transitions, that promises the longer life-span.

Recently many measurement instruments began to embed fast computers, which enable embedded EPICS IOC or MATLAB software. Unfortunately security issues arose when Windows was employed.

FPGA

FPGA (field programmable gate array) is another technology that can be found everywhere recently. The design

of a digital circuit board became very quick even with embedded software. It is flexible and robust, thus it is a wonderful platform for local controllers. Downsides are that a lazy design because of its flexibility may lead to a source of bugs and that it drove good-old TTL ICs away. More and more gates, memory, pins will be available with faster clocks. And better software supports are expected.

ATCA and MicroTCA

ATCA (advanced telecommunications computing architecture) is the next-generation networked computer standard that puts emphasis on reliability. A board can be connected with several 2.5-Gbps interconnects including GbE, PCI-express and 10GbE. The standard defines possibly redundant shelf managers that manage healthiness of the system through IPMI (intelligent platform management interface) over I2C connections.

Many reliability improving facilities can be utilized like hot-swap, redundant CPUs, switches, fans, etc. While the cost reduction is expected in the future, it is slightly expensive for now. It is selected for now as the main platform of the future ILC.

MicroTCA was defined in 2006 based on AdvancedMC (ATCA mezzanine card). Although the original standard was limited, it began to have many facilities from ATCA. It may not replace VME immediately, but it can be a good candidate for middle or small facilities, since it is not directly connected with PC industries unlike CompactPCI.

EPICS

EPICS has made success with international collaborations. It is a de-facto standard in this field. However, still there is list of features to be implemented and some of them are being sorted out.

Currently, a naming scheme and/or design of new records add some object-oriented design supports, however, more software-engineering support is preferable. To that end several different efforts are being carried such as, JavaIOC, Control system studio, and data access layer in the world.

While user and host based security mechanisms are available, for the larger installations more protection features like dynamic controls of protection and access logging are preferable.

The dynamic configuration of runtime database is one of the classic wishes. Static database had partially led to an intensive use of scripting languages at client side. The implementation of the feature may be shared with redundant-IOC project.

Magnet Controls

It is classic and typical control issues but still there should be many things to consider. The complicated part comes partially from non one-to-one correspondence between magnets and power supplies. The conversions of several units in engineering and physics applications, like current, field, kick or momentum are sometimes complicated and are dependent on situations. And timing synchronous operation is often required for ramp-up, tune-change, etc. Another difficulty is standardization.

Timing and Event Controls

Since accelerators are valuable, more and more time-sharing operations may be required. Thus, tighter coupling may be required between control and timing/event systems. Recently, an event system designed for SLS and DIAMOND is shared between several projects, and it could be a good reference[7].

RELIABILITY

The end user may require a robust operation of the accelerator, however, the accelerator itself should be flexible to enable continuous improvements. Thus, we may have to compromise between practical or ideal solutions under restrictions of safety, man-power, time, etc. The condition also changes during the operation. Here, we consider about adaptive and practical reliability.

Possible Right Solutions

There should be right solutions if we consider carefully. If we have well-defined software classes for accelerator controls, mistakes in the design, coding and usage of software can be reduced. We hope to have it someday.

Even a well-arranged naming conventions reduces human errors. They also enable more computer-aided tools. We have partial realizations depending on the accelerators. However, the real situation is not that simple and there are many exceptions, which may lead to exceptions.

Well-specified deployment procedures are employed at some accelerators, and some of them can be automated to reduce the errors. However, still many of us are lazy, and there are often reasons not to do so because of other criteria.

Surveillance for Everything

We have written too many pieces of software, which unfortunately assume certain circumstances, and they will eventually fail. We manage too many computers and too many network devices, which may have assigned wrong parameters or may have equipped with inadequate resources.

Thus, we have to find out the most important feature of those installations and the simplest tests for them. And routine tests should be carried automatically. If an anomaly is found, an alarm is issued, e-Mail is sent to the author or manager and an error log is recorded. If it is not critical the related software or hardware is restarted, otherwise the event is reported to the human operator.

This is not ideal, but is effective and practical under limited human resources.

Testing Frameworks

We often move operational environment for the better resource performances including computers, development software, etc. However, the event may lead to malfunctions because of incomplete compatibilities. Although some tests are applied beforehand, it is required to carry thoroughly prepared tests. Recently many free software projects, such as language systems and operating systems, began to have test cases. We may follow them.

We need a framework to implement them, which may cover following different kinds of tests.

The simplest test is the unit test that confirms the elementary features one by one. EPICS software now has "make runtests" to carry some collections of existent test cases. Users can provide tests in Perl/Test framework.

In order to find combined anomalies that cannot be found in unit tests, the regression test should be carried. While it is difficult to collect efficient test cases, some of the real running applications with faked data may serve as regression tests.

It also may be necessary to try stress tests, because some of the components like network sometimes show unusual behaviors including slow responses and real failures. We may have to install failure-recovery features or failover facilities.

We often find failures just after the shutdown period, and also it is difficult to make tests since hardware and software components are enabled one by one. During the shutdown, new software or hardware could have been installed, restoration of hardware or software could have failed, or power-stop may have brought another annoyance.

Thus, we may need intelligent procedure to start up the accelerator testing every feature of hardware and software. At KEKB and Linac we do it by human operator observations based on written procedures, which can be partially automated.

Redundancy

Redundancy may be the last-resort measure to improve the reliability, since it costs. However, many installations become complicated because it is easier for the users, and thus, nobody can guarantee the correctness. Redundancy is useful not only for failure recovery but also especially for the maintenance without any break.

We anyway may have to prepare backup installations, then automatic failover is close. There are several possible cases where redundancy is useful.

RAID and mirror-disks are used everywhere now. And redundant file servers can be employed for the network wide file services. At KEKB and Linac we have employed several different solutions for this purpose since even before the KEKB project. We sometimes had bad experiences, but it helped us several times during the operation. And it made the scheduling and maintenance much easier.

The redundancy of networks is also established technology. Before the KEKB project, we employed redundant transceivers for more than 40 optical Ethernet links, and had good experiences. Nowadays, the combination of the rapid spanning tree and HSRP or VRRP protocols enable reliable switches and routers.

Redundancy in PLCs is becoming mature as well. Several vendors implement CPU built-in memory redundancy. There are several possibilities what should be redundant, such as power-supplies, CPUs, network interfaces, backplanes, and even I/O.

The EPICS redundant IOC is the on-going project at DESY for XFEL. It is composed of RMT (redundancy monitor task) which monitors healthiness of controllers and manages PRRs (primary redundancy resource) and CCE (continuous control executive) which synchronizes internal states between two IOCs. It is necessary to modify PRRs, which includes scan tasks, channel access server tasks, sequencers and drivers. Users may modify user-supplied drivers and users tasks. KEK recently joined in the project for wider applications with OSI support. ATCA implementation may be possible.

Most of the upper-layer servers at Linac are redundant and sometimes it is useful. There should be more places where software redundancy and replication are useful.

SUMMARY

EPICS and SAD/SADscript made KEKB a great success although other accelerators have different criteria. Accelerator control design needs a balance between many aspects. Some of them are investigated. There are many good technologies waiting to be utilized. Some of them again are examined. Under practical situations reliability features are needed and many possible solutions are existent. The control system is located between all other facilities in the accelerator, which means it is enjoyable. If we have some "phronesis" or practical wisdom, we can solve our problems.

REFERENCES

[1] N. Akasaka *et al.*, "KEKB Accelerator Control System", Nucl. Instrum. Meth. **A 499**, 2003, p. 138.

[2] I. Abe *et al.*, "The KEKB Injector Linac", Nucl. Instrum. Meth. **A 499**, 2003, p. 167.

[3] K. Furukawa *et al.*, "Accelerator Controls in KEKB Linac Commissioning", *Proc. of ICALEPCS99*, Trieste, Italy, 1999, p. 98.

[4] K. Akai *et al.*, "Commissioning of KEKB", Nucl. Instrum. Meth. **A 499**, 2003, p. 191.

[5] `<http://acc-physics.kek.jp/SAD/sad.html>`.

[6] `<http://www.icalepcs.org>`.

[7] K. Furukawa *et al.*, "Timing System Upgrade for Top-up Injection at KEK Linac", *Proc of EPAC2006*, Edinburgh, UK, 2006, p. 3071.

