STATUS AND FUTURE DEVELOPMENTS IN LARGE ACCELERATOR CONTROL SYSTEMS*

Karen S. White, Jefferson Lab, Newport News , VA 23606, U.S.A

Abstract

Over the years, accelerator control systems have evolved from small hardwired systems to complex computer controlled systems with many types of graphical user interfaces and electronic data processing. Today's control systems often include multiple software layers, hundreds of distributed processors, and hundreds of thousands of lines of code. While it is clear that the next generation of accelerators will require much bigger control systems, they will also need better systems. Advances in technology will be needed to ensure the network bandwidth and CPU power can provide reasonable update rates and support the requisite timing systems. Beyond the scaling problem, next generation systems face additional challenges due to growing cyber security threats and the likelihood that some degree of remote development and operation will be required. With a large number of components, the need for high reliability increases and commercial solutions can play a key role towards this goal. Future control systems will operate more complex machines and need to present a well integrated, interoperable set of tools with a high degree of automation. Consistency of data presentation and exception handling will contribute to efficient operations. From the development perspective, engineers will need to provide integrated data management in the beginning of the project and build adaptive software components around a central data repository. This will make the system maintainable and ensure consistency throughout the inevitable changes during the machine lifetime. Additionally, such a large project will require professional project management and disciplined use of well-defined engineering processes. Distributed project teams will make the use of standards, formal requirements and design and configuration control vital. Success in building the control system of the future may hinge on how well we integrate commercial components and learn from best practices used in other industries.

HISTORY

The origin of computerized accelerator control systems coincides with the advent of the first affordable, commercially available digital stored program computers in the early 1960's. Prior to the availability of these relatively inexpensive and portable computers, accelerators were controlled by arrays of hardwired knobs, dials and push buttons attached to specialized hardware. In an era of much smaller machines, racks full of these dedicated devices provided a useful interface. In 1964, DEC introduced the PDP-8, the first minicomputer, effectively putting computer technology within reach for single purpose applications. A variety of minicomputers soon followed, offering an affordable combination of computing power, speed and memory size. By 1965, this technology, coupled with desires for more sophisticated control of existing machines and plans for larger machines, led scientists to explore the possibility of computerized accelerator controls. Proponents pointed to early success with computer controls of industrial systems and pushed forward with plans and prototypes leading to the first computerized accelerator control systems.

One of the earliest machines to be designed with computer based controls was the Los Alamos Meson Physics Facility (LAMPF). [1] Plans for this system began to take shape in 1963 and pointed to the flexibility and expandability to be gained through the use of cutting edge computer controls. Scientists at LANL proceeded to develop this system, beginning with a 16-bit computer with 8K words of memory and programming in assembler language. Other than the computer, nearly everything about this early computer based accelerator control system was custom, including the operating system. Operators used consoles featuring color CRTs and knobs. Displays were cryptic, but useable and the system accumulated a variety of data in a centralized fashion, making it available to application programs which could perform more complex operations, ensuring consistency and delivering results in a prompt fashion.

Being one of the first accelerator control systems to use computers provided ample opportunity for learning, and the system rapidly evolved, eventually incorporating more powerful computers, a switch to the FORTRAN programming language and the CAMAC fieldbus. A key element of the LAMPF system was a central data repository for static machine information such as standard device names and hardware addresses. Despite the availability of this database, hardware addresses made their way into application code which led to inevitable maintenance problems. Because such a system had never been demonstrated, a backup manual control console was provided, but never used.

In parallel with this successful demonstration of computerized accelerator control, many existing machines began to adopt computers for use in their control systems throughout the 1970s and all new machines followed suit. In fact, no one would suggest today, about forty years later, that any planned accelerator could be made useable without computer controls.

CURRENT CONTROL SYSTEMS

Today's control systems are much more complex than their early predecessors. Older control systems have undergone upgrades, incorporating advances in

* This work was supported by DOE Contract No. DE-AC05-06OR23177, under which Jefferson Science Associates, LLC operates Jefferson Lab.

technology to meet ever increasing requirements for speed, accuracy and automation. Architecturally, control systems have followed the progression of software systems in general, evolving from two-tiered architectures popular in the 1980s to the three tiered structures favoured today and incorporating more advanced communications models. Control systems have successfully adopted technology advances to add functionality and enhance performance. Also, like most software projects, control systems seem to grow to fully utilize the available technology.

Older control systems tend to feature two tiers: the client (back-end) tier, consisting of consoles running operator interfaces and other application programs, and the real-time (front-end) tier, consisting of distributed processors, directly connected to hardware and running programs to acquire data from and apply control methods to devices. Older systems often utilized a source/destination communication model where data is sent multiple times to different destinations, wasting bandwidth and making synchronization difficult.

Over time, these systems evolved, becoming more widely distributed, splitting the work over a far greater number of processors. It is not uncommon to see new machines come on line utilizing over 500 processors in the control system, not including local programmable logic controllers (PLCs). While the greatest increases have been seen in the number of processors used in the front-end, increasing the number of consoles machines available for operations has accompanied steady decreases in the prices of workstation and personal computers. The inclusion of so many additional computers, communicating on the controls network, required more network bandwidth to maintain acceptable update rates. Additionally, data has become more complex, migrating from simple time stamped values to vectors of values packaged with relevant attributes.

Mirroring trends in industry, control systems moved towards replacing the older point-to-point style communication characteristic of the source/destination model with more efficient producer/consumer or event-based publish/subscribe paradigms allowing data from a single producer to be consumed by multiple clients, providing a more efficient use of network bandwidth and allowing for easier synchronization. Further refinements such as send-on-change protocols, the use of deadbands and client selected update rates were also introduced in some accelerator control systems, effectively reducing the amount of data sent over the network without significant impact on operations. Gateways, used as data concentrators, have also been effectively used to throttle network traffic but likely have limitations in scalability.

Another step in control system evolution is reflected in the prominent emergence of a third tier. While new control systems are typically designed with three tiers, older systems have been able to easily introduce a third tier by adding services or middleware between their existing client and device tiers. The move towards three tiers reflects recognition of the need for improved coordination and consistency between processes and increased software reuse. Middleware consolidates functions, previously performed in individual applications, in servers for use by multiple clients, ensuring the function is performed consistently and allowing all applications to operate with the same data. The introduction of such middleware has also provided a convenient way to provide a common control system appearance while really running two or more different control systems at the device level as is common during the evolution of older machines. This model is also a much better fit to an object-oriented approach which is increasingly common in controls.

Control system user interfaces have also benefited from advances in technology. The line based CRTs of old have been uniformly replaced with animated graphical displays featuring multiple windows and often multiple physical screens per console. Today's control rooms are populated with dozens of monitors, some even prominently featuring wall sized displays. Numerous graphics software packages are used to create a wide variety of graphical user interfaces including synoptic displays and graphs. While all this technology makes is easy to display information as dynamic graphical symbols, it appears that little effort has gone into optimizing the information displayed for use by humans. Despite the technological advances, it remains easy to overwhelm an operator with too much less relevant data and allow information about an important alarm condition to be missed. Some alarms systems themselves now provide the functionality to filter out of context alarms and collapse related alarms into trees, however there has been limited use of such features due to the analysis required to structure the conditions properly.

Beyond these fundamental structural changes, a number of trends have emerged in today's control systems. Accelerator control systems were once almost exclusively custom creations. With the exception of commercial computers, virtually every piece, both hardware and software, of early systems was created by laboratory scientists and engineers. Today, with the notably exception of the National Ignition Facility (NIF) under construction at LLNL, the controls community has increasingly moved towards the use of commercial and shared components in an effort to reduce development costs and improve reliability. Some of the earliest significant movement in this direction came in 1989, when developers at ANL agreed to work with their counterparts LANL to adapt the Ground Test Accelerator Control System (GTACS) for use at the Advanced Photon Source (APS) which was under construction.

GTACS utilized a toolkit approach to control system development. This software was significantly enhanced for use at the APS and soon became known as the Experiment Physics Industrial Control System (EPICS). [2] The last fifteen years have seen well over 100 scientific projects worldwide adopt EPICS and successfully build control systems of varying sizes, some quite large. EPICS provides robust and flexible front-end software for device control and the channel access protocol to allow communication between the front-ends and client programs. The EPICS collaboration has experienced great success at sharing the EPICS device control and communications software with well over 300 different device support modules currently available from the EPICS website. Additionally, a large number of client programs have been developed by EPICS collaborators; however, it seems there are large differences in how the various EPICS sites provide user interfaces and high level applications. There is no standard EPICS middleware and each site seemingly invents their own client framework. Additionally, EPICS lacks a central device definition database, with this information distributed amongst the front-end processors, leaving each site to develop their own data management solution or go without. Initially hampered by its dependence on VxWorks, a costly commercial real-time operating system, the EPICS front end software was made platform independent by the release of version 3.14 in 2000. This made EPICS useable for smaller projects looking for shared software as a cost saving measure by allowing a single PC to run all functions of both the device and client tiers.

By 1999, TANGO, an object oriented control system framework based on open source software, including COBRA, was under development at ESRF. [3] TANGO was subsequently adopted by three other European laboratories and the objected oriented approach of TANGO provides a natural fit to controlling accelerator devices. TANGO holds static device information in a MySQL database. The TANGO collaborators also use different tools at the client level.

Along the same lines as the toolkit approach, many European laboratories currently favour the use of commercial Supervisory Control and Data Acquisition (SCADA) systems as a basis for control system development. The control system for the Large Hadron Collider (LHC), under construction at CERN, features a three tier architecture with an emphasis on hardware and software standards, industrial solutions, including a SCADA system and a central data repository for static data. [4]

The advantages of toolkit and SCADA approaches include reduced development costs and timeline, and increased reliability due to the high degree of software reuse. Reusing software components eventually results in code that has been tested far more extensively than is possible with individual custom developments. It is interesting to note, that at NIF, where the CORBA based control system is almost entirely a custom development, significantly more manpower has been allocated for software testing than in other projects of similar size.

TANGO, EPICS and SCADA based systems are examples of collaboratively and commercially developed components that have successfully become integral building blocks for accelerator control systems. Similar success has not been achieved with the beam applications that are increasingly important as accelerator complexity increases. Despite a desire to benefit from shared software, there seems to be little consensus among different laboratories on the requirements for beam applications, or for toolkits that could facilitate collaborative application development.

However, the recent use of a common Accelerator Markup Language (AML) by several laboratories is a positive move in this direction. Collaborative adoption of a standard lattice description such as the one provided by AML can be an important first step towards increased software sharing for beam application development across laboratories.

The movement away from completely custom control systems is also evident in the hardware arena where status reports for control systems of new projects prominently feature a laundry list of the best components and even systems adopted from other laboratories. The challenge of this approach is the difficultly of knitting together diverse parts into a cohesive system with a common look, feel and behaviour presented to users. This method has been successful in providing needed system functionality, but often falls short in the aforementioned integration area. Increased use of frameworks and middleware is helping to improve this situation. Many of today's newer machines have successfully outsourced entire machine subsystems, completely integrated with the chosen machine control system.

As the amount of processing power and memory continues to increase in rapidly shrinking packages, the line between hardware and software is blurring with traditional front-end computers, running data acquisition and control software, and once attached to hardware devices, are now being embedded into hardware devices. Processors are so small and cheap, future projects plan to have a processor built into each device, performing extensive local control, programmed in firmware and reporting to control system middleware programs via the Ethernet. There is also increased use of network attached devices as many vendors now sell almost anything with an Ethernet port. While this option can be appealing, existing control systems have been known to suffer difficult to solve network problems as the result of poor behaviour from one of these devices. Integrating thousands of such devices on a controls network which, needs to be rock solid, may prove challenging.

Today's machines exhibit varying degrees of data management. While most employ efficient means to transmit and archive dynamic machine data such as device readbacks, there seems to be widespread recognition of the need for improved comprehensive static data management. Some accelerator controls systems completely lack such a database and even labs that report having a central data repository admit it is not all inclusive. In many systems, machine modifications require multiple hand edits in the software which is known to be error prone and risky. Oracle and MySQL are popular choices for static data management, but these are only tools to minimize programming effort by providing standard relational database functions. The far more difficult task is to setup and populate the proper tables and make data efficiently available to applications. Furthermore, success depends on the uniform use of the central database by all applications. Another issue in the area of data management is archived data. Running machines accumulate volumes of data for future analysis, but often lack planning for long term storage and access needs.

Overall, today's control systems are doing a reasonable job of operating their respective machines, and these systems continue to evolve, often following trends and incorporating new technology from industry. However, the next generation of planned experimental physics machines will present more challenges and require a shift in emphasis.

FUTURE

Scientists and engineers have proven many times their ability to productively leverage emerging technology to meet the increasing technical challenges of new machines. Control systems for future large machine will undoubtedly continue to depend on the increases in CPU and network speed and memory size that have become routine. Future systems will employ a substantially larger number of devices, and with the movement towards front end processors becoming embedded in each device, the number of processors communicating on the network stands to grow by two orders of magnitude. Some machines will also be physically quite large, causing the control system to span tens of kilometres. Additionally, timing requirements will be tighter and the amount of data to be archived increased. Fortunately, demands for improved network bandwidth, processor speed and data storage required by the commercial sector exceed that of control systems so we can expect to continue to benefit from commercial technology developments to provide essential equipment for new machines. Given the rapid pace of technology development, selection of the exact technology to be used should be amongst the last decisions made when planning a new control system. Rather than the incorporation of new technologies, the real challenges for future lie in how to make these large control systems more robust, maintainable and operable in a global environment.

With future systems envisioned to be more than an order of magnitude larger than those we use today, the sheer number of devices will mandate much higher reliability requirements in order to ensure these expensive machines can be operated a useful number of hours per year and on a predictable schedule. Increases in availability can be most economically made through use of high availability commercial components, and good engineering practices, including careful change management. It is also quite natural to consider the use of redundant components with automatic failover as a way to increase reliability. It will likely be cost prohibitive to make every component redundant, therefore, engineers will need to evaluate the characteristics of each element in order to use redundancy to maximum advantage.

System availability is a function of both the frequency of failures and the amount of time to complete a repair. In addition to building a system with very infrequent failures, it is critical to ensure repairs can be accomplished quickly. The repair time consists of the time taken to diagnose the problem and the time to replace the broken component. Assuming spares are easily accessible and calibrated most components can be quickly replaced. There is opportunity for improvement in the time spent diagnosing problems which normally dominates this equation. To minimize the diagnostic time, all machine components must be designed with integrated diagnostics features, making it possible for the control system to constantly verify, in a uniform fashion, that all devices are functioning correctly, or, in the event of an error, enabling quick identification for repair. Availability may also be enhanced by designing application programs to utilize alternative hardware in the event of a failure. This strategy depends on the some level of extra instrumentation.

In planning for the overall reliability of the control system, one must also consider the reliability of the software where redundancy is of much more limited use. In order to achieve the very high reliability expected of future control systems, the software development process must be disciplined and rigorous throughout the life of the project. Studies have shown the rate of defects introduced in the software development process can be significantly reduced by following industry best practices. [5] The higher development cost pays for itself many times over during commissioning, operations and maintenance, with dividends in the form of much higher software reliability. Proven software engineering methods must be employed along with strict configuration management and extensive testing. Careful requirements and design development and review have been proven to reduce the overall development time of large software projects by significantly reducing rework. Such processes also minimize the number of defects in the delivered code and therefore time spent debugging and repairing code. To ensure high reliability software, controls engineers must learn from the professional software engineering industry and employ proven methodologies.

While many engineers already design with a reliability goal in mind, it seems less attention is currently paid to maintainability. Most hardware components already employ a modular design to facilitate future maintenance and repair. The software trend towards increased use of middleware also facilitates changing components in one software layer without impacting the rest of the control system. However, experience from many experimental physics projects tells us that the physical machine configuration is likely to be changed many times over the life of the project, necessitating changes in the underlying controls software.

Unfortunately, there is a common assumption that software is easy to modify, leading to a lack of advance planning for how to support configuration driven software changes. Understanding and tracking dependencies in large, complex software systems is difficult, and it is important to design the control system keeping the need for modifications to a minimum. It is not unusual for insertion or deletion of devices in an accelerator to necessitate multiple hand edited software changes in different areas of the control system. The need for such risky and error prone changes stems from hard coded configuration information. In order to prevent such maintenance problems in future machines, the control system must provide a single authoritative source of all configuration data – from the very beginning of the project, and all software must automatically adapt based on this data source. Ideally, a machine change entered into the master database will trigger all affected software to automatically update to match the new configuration. As an added benefit, the existence of a central repository for static data provides the infrastructure needed to supply a host of useful information about the machine in an integrated fashion.

An interesting idea in the area of data management is the use of commercial Geographical Information Systems as the basis for the central machine data repository**.** [6] This type of system, commonly used in municipal applications, lends itself well to an accelerator where the location of each device is part of the mandatory information needed to effectively provide an accurate model of the machine, which in turn is needed by many beam applications.

The rigorous use of a central data repository is a necessary improvement over many of today's machines where maintenance and repair efforts are hindered by the distributed and often undocumented storage of critical data. Successfully implementing integrated machine data management is difficult because it requires agreement and cooperation from all groups. Achieving such conformity may prove more difficult than any technical challenge.

Given the enormous size envisioned for the control system of a machine like a superconducting linear collider, and increasing pressure to keep costs under control, it is unlikely the next generation control system designers will have the luxury to entertain a built from scratch solution. This is not to say that a system exists today that could be easily adapted for the planned linear collider, however, success will hinge upon widespread use of commercial solutions, both hardware and software, and the reuse of components from existing accelerators. While this approach can help contain costs, it presents a large challenge in the form of creating a well integrated system. Middleware plays an important role in helping to integrate different controls at the device level with a cohesive operations layer. More importantly, standards must be established early in the project. Standards for the look, feel and behaviour of all user interfaces are critical to creating an integrated system rather than a collection of control system pieces. Additionally, the sheer size of the data space for a future machine provides ample opportunity for data overload. Interface designers should consider carefully what data is needed by each type of user. It is not enough to present such large volumes of data and expect humans to make efficient interpretations and responses. The control system must provide a high degree of automation, with interfaces becoming more for information than for control. Providing the needed level of automated setup and control will rely on having an accurate machine model available to all applications and having software adapt to empirical data.

On top of everything else, security challenges will continue to increase as determined hackers search for every possible weakness to exploit. In addition to the classic security vulnerabilities we face today, it is expected the next large accelerator will require a greater level of remote development, operations, diagnosis and repair, inevitable for machines born of large international collaborations. This requirement is completely orthogonal to the need for tighter security and will require the development of very strict access and authentication processes. Systems will need to be protected both from hackers and from unintentional or unauthorized modifications. By designing this in from the outset, and using new development methodologies that attempt to eliminate, for example, buffer overflow attacks, the next generation control system can be made robust in an increasingly hostile security environment.

SUMMARY

Major challenges face the developers of control systems for future large experimental physics machines. The current pace of development of commercial technology will likely meet the basic requirements for the building blocks of future control systems. More challenging will be providing a reliable, maintainable, secure and operable control system. Due to the international nature of future machines and the likelihood of distributed development, standards need to be developed first and enforced throughout the project. The requirements for the control system need to consider all stages of the project, including maintenance and the inevitable upgrades, rather than just making the control system work for the initial machine configuration and commissioning. Incorporating commercial solutions and the best features of existing systems will be important to our future success as will the use of vigilant engineering practices and central data services.

REFERENCES

[1] A LAMPF Controls Retrospective: the good, the bad and "it seemed like a good idea at the time", S.C. Schaller, Nuclear Instruments and Methods in Physics Research Section A, Volume 352, Issues 1-2, 15 December 1994, p. 293-295.

[2] Experience with EPICS in a Wide Variety of Applications, M. R. Kraimer, M. Clausen, W. Lupton, W. A. Watson, PAC'97, Vancouver, B.C, May 1997, p. 2403.

[3] TANGO – An Object Oriented Control System Based on CORBA, J-M. Chaize, W-D. Klotz, J. Meyer, M.

Perez, E. Taurel., ICALEPCS'99, Trieste, Italy, October 1999, p.475 – 479.

[4] The LHC Control System, B. Frammery, ICALEPCS'05, Geneva, Switzerland, October, 2005.

[5] http://www.cebase.org/www/AboutCebase/News/top-10-defects.html

[6] Evaluating the Potential of Commercial GIS for Accelerator Configuration Management, T. Larrieu, Y. R. Roblin, K. White, R Slominski, ICALEPCS'05, Geneva, Switzerland, October, 2005