LESSONS LEARNED ON THE GROUND TEST ACCELERATOR CONTROL SYSTEM*

Andrew J. Kozubal and Robert E. Weiss, Mail Stop H820
Accelerator Operations and Technology Division
Los Alamos National Laboratory, Los Alamos, New Mexico 87544, USA

Abstract

When we initiated the control system design for the Ground Test Accelerator (GTA), we envisioned a system that would be flexible enough to handle the changing requirements of an experimental project. This control system would use a developers' toolkit to reduce the cost and time to develop applications for GTA, and through the use of open standards, the system would accommodate unforeseen requirements as they arose. Furthermore, we would attempt to demonstrate on GTA a level of automation far beyond that achieved by existing accelerator control systems. How well did we achieve these goals? What were the stumbling blocks to deploying the control system, and what assumptions did we make about requirements that turned out to be incorrect? In this paper we look at the process of developing a control system that evolved into what is now the "Experimental Physics and Industrial Control System" (EPICS). Also, we assess the impact of this system on the GTA project, as well as the impact of GTA on EPICS. The lessons learned on GTA will be valuable for future projects.

Introduction

For the designers of a control system, the Ground Test Accelerator presented a major challenge. The GTA was primarily an experiment to demonstrate advanced technologies for producing a bright H^- beam, expanding the beam through a telescope, and producing a highly collimated H^0 beam. All parts of GTA would use a common control system, including LINAC (injector, RFQ, etc.), diagnostics (microstrip, beam stop, etc.,), and facilities (vacuum, temperature, state of health, interlocks, etc.). The system would be designed, fabricated, and tested in stages. One major goal was to demonstrate space traceability of the major GTA components. This goal included demonstration of automatic start-up, shutdown, and fault detection and recovery. Furthermore, the control system would demonstrate the ability to automatically produce optimal operation under a variety of operating modes.

Control System Design

Like the rest of GTA, the control system requirements would be developed in increments. For this reason the designers decided to make the control system architecture as general as possible. We developed a set of tools to facilitate applications development and rapid prototypes. With this approach we were able to meet the near-term support requirements (test stands and injector development) while adding additional capabilities as we assessed operations and discovered more requirements.

Strategic Design Decisions

At the outset we made several strategic design decisions that would have long-lasting effect. We chose to use a core set of run-time software throughout and a common software developers toolkit to generate applications. Software and hardware standards and well-defined interfaces would be used throughout. This core and toolkit software we call the *Ground Test Accelerator Control System* or GTACS.

Distributed architecture. We knew that high performance and the ability to expand the controls in increments were two important attributes that we needed. Therefore, we chose a distributed architecture based on multiple input-output controllers (IOCs) and operator interface consoles all connected via a local area network[1]. In addition we chose to base the all controls on a distributed run-time database.

The resulting architecture can easily be scaled as additional controls are added without impacting existing installed controls. Also, since the performance of existing controls was well-defined, the applications developer could accurately predict performance and specify hardware needs, including cost and time to delivery.

Developers toolkit. Because of the experimental nature of GTA, we expected a short lead time between requests for functions (requirements) and expected implementations. Also, some functions would need rapid prototypes. With a developers toolkit we knew we could meet these delivery constraints. This toolkit initially provided a tool to configure the run-time databases, a graphical display builder, and a tool to simplify programming sequential control applications. Later additions to the toolkit included an alarm manager and an archiver.

Common hardware. To simplify maintenance, keep spares to a minimum, and reduce repair time, we tried to use common hardware throughout. We selected Sun workstations for the operator interface consoles and for applications development and 68020 microprocessors for the IOCs. Also selected were standard input-output modules and signal conditioning equipment.

Common software. The use of common run-time software greatly reduced the learning curve for applications developers and allowed them to go from developing and supporting one GTA subsystem to another in a very short time. Basic IOC functions (database structure, startup files, external sequences, utilities, etc.) would be similar for all machines and all applications. Further, all subsystems would have similar interfaces to the operators, thus minimizing training time.

* Work supported by the US Department of Defense under the auspices of the US Department of Energy.

Standards. Although we found many choices of hardware and software available to us, we chose to stay with industry-wide standards such as UNIX, TCP/IP, VME/VXIbus, and Ethernet wherever possible. Where standards were not available or were not appropriate we chose approaches that would be well supported over the life of the GTA project. For instance, we decided on VxWorks for the real-time operating system on the IOCs.

Lessons Learned

GTA was cancelled before all goals were met, so we were not able to implement controls beyond the first DTL stage. However, the extensive controls we did implement met our expectations. In the following discussion we look critically at some of the lessons we learned. We first look at how well GTACS met the needs of the applications developers, and then we look at some of the lessons we learned in developing applications.

The GTACS Experience

As we anticipated, a large portion of the GTA controls software development effort (at least 50%) would go into developing GTACS. We were fortunate in being able to apply early versions of GTACS to test stands and off-line experiments, so that by the time we started on GTA most of the software and hardware had been tested under realistic conditions, and we had received extensive feedback from the users.

GTACS Requirements. During the initial design of GTACS little was known about the actual accelerator that would be built. Hence, no formal GTA requirements document was written and many assumptions were made that affected its fundamental design. Most of these assumptions led to over design. However, a few led to a lack of needed capabilities. For example, we were well into implementing applications on GTA when we discovered that there was no mechanism for correlating data by time from diagnostics that were on separate IOCs. Fortunately, we found solution that did not require redesigning GTACS, although this solution caused some delay in implementation and probably was not optimum. On the other hand functions were added to GTACS that were either not needed for some time or were never needed.

Upgrades and Support. We underestimated the effort involved to support the multiple versions of GTACS that we delivered to the applications developers. We eventually learned how to control the releases while keeping up with the request for new features. Because GTA was sometimes used as a development test bed for GTACS, much initial down time was caused by the development effort. Users eventually became aware that there is often an initial reliability issue between commercial (mature) software (and hardware) and custom developed ones. As GTACS matured, software reliability improved dramatically.

Applications integration into GTACS. Often we found it difficult to determining whether a particular function should be included in GTACS or developed as a GTA application. For instance, an image processing tool for beam diagnostics[2] made extensive use of the GTACS capabilities. Because of its generality we eventually integrated this application into the GTACS toolkit and run-time environment.

On the other hand, RF power applications were developed separately from GTACS, although GTACS had to allow the users to control and monitor RF power equipment. We demonstrated the flexibility of the GTACS architecture by installing GTACS in VXI-based RF control equipment. To the users, our goal of having the standard GTACS VME systems and the specialized RF VXI system operate as one common system was accomplished. This required development of specialized drivers for GTACS to communicate with this unique RF equipment. A close working relationship between the GTACS designers, RF power designers, and application engineers was crucial for this to be successful.

Automation. Although the project was cancelled before our original goals of automating GTA were fully met, there were some successes, such as the automatic beam steering in the low energy beam transport[4]. The GTACS toolkit and the common run-time software facilitated this automation effort.

Hardware. We paid close attention to the IOC hardware design for GTA. Design standards for cooling, EMI shielding, grounding, and cabling aided controlling and isolating hardware problems. We observed that most problems occurred in cases where these standards were not followed, particularly at interfaces with the users' instrumentation.

Documentation. By establishing and maintaining hardware and software documentation standards, we were able to update design and user documentation as the project matured. We also learned that work-around solutions to software or hardware problems had to be well documented and that all affected parties had to be informed.

Integration and Testing. Re-integration of new versions of GTACS with existing applications was a major challenge. Installation of a new release on the control system network was not always straight forward, and upward compatibility was not always achieved. We often tested GTACS on a separate network from the GTA, but we quickly learned that re-integration must be a careful, meticulous process. Also, we found that providing a central error logging capability was valuable in troubleshooting new software.

The GTA Applications Experience

Using the GTACS developers toolkit and run-time environment allowed us to develop GTA controls on time without a large programming staff. Programmer training time was minimal and more effort could be put into implementing applications rather than being concerned with low-level software and hardware problems.

Requirements. With the GTA controls system approach (GTACS separated from applications) we have three parties involved—the end users, the applications developers, and the GTACS developers. GTA being an experiment, the users often perceived their requirements in small increments over time. Also, the users often did not have a firm distinction between GTACS and the applications.

Implementation of a living requirements documents for the GTA subsystems eliminated a lot of misunderstandings between the control system developers and users. It established a feedback mechanism whereby users could provide comments on the

control system. In addition, frequent control system reviews were necessary. All assumptions, major risk area, and incomplete requirements were stated at the start, thus establishing a baseline for future development work.

Recovery. Occasionally IOCs would fail due to a lose of power, a hardware problem, or a software error. With multiple IOCs interacting we often found it necessary to reboot other IOCs to get them back in synchronization after fixing the problem When this became sufficiently annoying to the users we developed automatic recovery software.

Upgrades. As the control system became larger and more important to the operation of GTA, we found that we had to control software and hardware upgrades. The timing of these upgrades had to be coordinated with the users. We could not make changes just to try the latest enhancements; we had to plan upgrades well in advance. We also found it necessary to use detailed check-out procedures to ensure a new version was performing as desired.

Hardware Checkout. When checking out IOC hardware we found that GTACS provided most of the tools that the hardware engineers and technicians needed. Many input-output modules could be tested by creating a database for each channel and exercising the device through a custom operator display. GTACS was used to build both the database and the display.

GTA Operations. The flexibility of GTACS gave us the opportunity to simplify the operators' tasks by providing quick, convenient access to relevant operational and experimental data. In time this allowed us to reduce the number of operators without overloading their tasks.

The GTACS Legacy

The successes of GTACS on GTA led to its continued development, including a collaboration with other laboratories[5] and licensing to three commercial organizations. Eventually, GTACS evolved into the "Experimental Physics and Industrial Controls System" or EPICS.

Conclusions

Our up-front investment in a common environment and a comprehensive developers toolkit paid back in more rapid and less costly development of applications on GTA. Although we found the early experiences quite painful, we learned how to handle controls for an experimental facility. Furthermore, as GTACS (and later EPICS) is reused on other projects, the payback becomes more significant.

References

[1] Dalesio, L.R., M.R. Kraimer, A.J. Kozubal, "EPICS Architecture," in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, C.O. Pak, S. Kurokawa, and T. Katoh, Eds. (ICALEPCS, Tsukuba, Japan, 1991), pp. 278—281.

[2] Zander, M.E., R.M. Wright, "Image Processing & Computer Controls for Video Profile Diagnostic System in the GTA," *Proceedings of the LINAC Conference*, Ontario, Canada, August, 1992.

[3] Jachim, S.P., C. Ziomek, E.F. Natter, A.H Regan, J. Hill, L. Eaton, W.D. Gutscher, M. Curtin, P. Denney, E. Hansberry, and T. Brooks, "The Los Alamos VXI-Based Modular RF Control System," *Proceedings of the IEEE Particle Accelerator Conference*, 1993.

[4] Brown, S.K., W.H. Atkins, "Beam Steering in the Ground Test Accelerator Low Energy Beam Transport," Los Alamos National Laboratory Report LA-UR-93-358, 1993.

[5] Knott, M., M.E. Thuot, D. Gurd, S.A. Lewis, "EPICS: A Control System Software Co-Development Success Story," submitted to *International Conference on Accelerator and Large Experimental Physics Control Systems* (ICALEPCS), Berlin, Germany October 18-22, 1993.