THE CONTROL ARCHITECTURE OF THE DØ EXPERIMENT

J. Frederick Bartlett, Stanislaw Krzywdzinski, Geoffrey Savage, Vladimir I. Sirotenko, Dehong Zhang,
FNAL, Batavia, IL60510, USA

Abstract

From a controls viewpoint, contemporary high energy physics collider detectors are comparable in complexity to small to medium size accelerators: however, their controls requirements often differ significantly. D0, one of two collider experiments at Fermilab, has recently started a second, extended running period that will continue for the next five years. EPICS [1], an integrated set of software building blocks for implementing a distributed control system, has been adapted to satisfy the slow controls needs of the D0 detector by (1) extending the support for new device types and an additional field bus, (2) by the addition of a global event reporting system that augments the existing EPICS alarm support, and (3) by the addition of a centralized database with supporting tools for defining the configuration of the control system. This paper discusses the control architecture of the current D0 experiment, how the EPICS system was extended to meet the control requirements of a large, high-energy physics detector, and how a formal control system contributes to the management of detector operations.

1 THE EXPERIMENT

DØ is a high-energy physics experiment located at one of the two collision points of the 1 TeV proton/anti-proton beam of the Fermilab accelerator. The detector is constructed from multiple layers of sensors: (1) a precision inner tracking section consisting of silicon microstrip cylinders and disks, (2) eight cylinders of longitudinal and stereo scintillating fibers, (3) a super-conducting solenoid providing a magnetic field for the inner tracking layers, (4) an electromagnetic preshower section, (5) a liquid argon calorimeter, and (6) an outer muon spectrometer. The experiment has just commenced its second, extended running period that is expected to last until 2007.

2 EPICS AND ITS EXTENSIONS

Following the first running period, which ended in 1995, the computing policy of the laboratory decreed that future experiment software must be developed from platform-independent components. Since the DØ control group was small and the period before the beginning of the next running period was short, recasting the existing slow-controls system in the new formalism was not practical.

After a brief survey of the field, we selected EPICS (Experimental Physics and Industrial Control System) [1] to provide the building blocks for our new controls effort. The principal reasons for selecting EPICS were (1) the availability of device interfaces that matched or were similar to our hardware, (2) the ease with which the system could be extended to include our experiment-specific devices, and (3) the existence of a large and enthusiastic user community that understood our problems and were willing to offer advice and guidance.

One of the unique properties of the DØ detector interface is the use of the MIL/STD1553B serial bus for all control and monitoring operations with the detector and with the electronics components located in the remote collision hall. Since the detector is inaccessible for extended periods of time, a robust, high-reliability communication field bus is essential. We extended EPICS by providing a queuing driver for MIL/SRD1553B controllers and a set of device support routines that provided the adaptive interface between the driver and the standard EPICS process variable (PV) support records. Once these elements were in place, all of the features of EPICS were available for use with our remote devices.

High voltage channel control is an example of extending the basic PV record support. In this case, building a compound device from individual PV records was not feasible because of the complexity of the HV device and the speed requirements. A generic high-voltage record support module was developed based upon the extended, Harel state machine model [2]. The record support module provides the required, high-level behavior with (1) linear ramping with retries, (2) trip condition recovery, and (3) limits control. Device support modules then adapt the HV record to specific HV devices. Although developed for a specific device, the record support is non-device specific and may be used for other types of voltage generators that require a similar behavior.

Using the EPICS portable channel access server, we have constructed gateways to two other control systems: (1) the SCADA-based DMACS system that manages the DØ cryogenic and gas systems and (2) the accelerator ACNET control system. These links are used for exchange of information only.

3 GLOBAL EVENT REPORTING

Although the EPICS system provides an operator alarm display, alarms from slow controls are not the only, nor, necessarily, the most important events. To address this problem we have developed a separate facility, the Significant Event System (SES) [3], to collect and distribute all changes of state of the detector and the data acquisition system.

Unlike the EPICS alarm facility, in which the operator display explicitly connects to each PV, the SES has a central server that collects event messages from sender clients and filters them for receiving clients. Each EPICS IOC connects to the server via a TCP/IP link and all state changes on that IOC, including alarm transitions, are sent to the server. For large physics detectors with hundreds of thousands of PVs, the savings in connect time at startup can be significant.

The alarm class of SES messages receives special handling in the server. The SES server maintains the current alarm state of the entire detector so that receiving clients are able to obtain the current state when they first connect to the server.

In addition to specialized receiving clients that may connect to the server, there are two standard clients: the SES logger and the operator display GUI. The logger has a pass-all filter and writes all messages to a disk file.

In addition its monitoring and logging functions, the SES system provides the means for distributing synchronizing messages to other components of the online software system. For example, global control of the high-voltage system is accomplished by having the individual detector high-voltage programs connect to the SES server for messages that signal changes in the run state of the data acquisition system.

The SES server and most of the receiving clients have been coded in the Python scripting language, while many of the sending clients are coded in C or C++. We anticipate that, for efficiency considerations, the server may require recoding in C++ at some later stage in the development cycle. API's for SES clients are available in all three languages.

4 CENTRALIZED DEVICE DATABASE

The EPICS databases that configure the individual Input/Output Controllers (IOC) are flat, ASCII files that are read by the IOC's during startup. The EPICS system additionally provides a higher-level construct, called a template, which is a parameterized collection of record definitions. Generator files, which reference the templates, supply the parameter values to produce instances of these templated devices. While these collections of files are adequate for EPICS initialization, they are not easily accessible to host-level processes that may require the same information.

To address this problem, the DØ experiment centralized the relevant device information in a relational database (Oracle) [4] and provided a family of scripts, written in the Python language, to manage the transformation between the relational database and the EPICS, ASCII-format files.

By providing scripts for bi-directional conversions, it is possible to edit collections of devices (instances of templated devices) by extracting the parameterized devices to a generator file, modifying the generator file with a text editor, and re-inserting the generator file into the relational database. For large collections of devices, this three-stage process is often simpler and faster than using a database editor directly.

In addition to the database management scripts, a WWW browser interface to the relational database is available for initial definition and modification of the relational database entries.

With control system device specifications now centralized in the relational database, they are easily accessible to other host-level processes. This, in turn, has led to a series of extensions to the original database schema to support the needs of other, controls-related processes. An example is the SES operator alarm display that accesses the central device database for obtaining guidance text and action scripts related to specific EPICS devices.

5 DETECTOR CONFIGURATION MANAGEMENT

One of the most complex tasks performed by the control system is the configuration of the detector for specific run conditions. The set of distinct configurations, both for normal, data-taking and for calibration runs, is very large; and, so, the usual technique of uploading a specific detector configuration, once the required conditions are established, and saving it as a file for subsequent downloading is impractical.

For purposes of configuring the detector, it is structured as a tree with nodes at successively deeper levels corresponding to smaller, more specialized organizational units of the detector. The terminal nodes of the tree are, in nearly all cases, single instances of the high-level (templated) devices discussed in the preceding database section. The intermediate nodes of the tree primarily serve to organize the traversal order of the subordinate nodes since the detector is, in general, sensitive to the order in which devices are initialized. The terminal nodes, called action nodes, manage the configuration of a specific, high-level device.

One level of intermediate node, the geographical sector, has a particular significance. These nodes, in most cases, represent the individual read-out crates of the data-acquisition system and are the lowest level in the tree hierarchy in which the nodes are guaranteed to be

functionally independent. The load function for these nodes may be executed in parallel, significantly reducing the total time required to configure the detector.

A single program, COMICS [5], coded in the Python language, manages configuration of the EPICS-accessible part of the detector. The tree nodes, both intermediate and action, are specialized instances of a base node class, which defines most of the methods that characterize node behavior. The detector tree structure is defined by a set of configuration files that are Python program segments which instantiate instances of nodes.

6 CONCLUSIONS

Faced with the task of completely rebuilding the slow-controls system of a complex, high-energy physics detector in a limited time, the DØ collaboration selected the EPICS system to provide the component parts from which the system would be constructed. EPICS, itself, has been extended to support a new field bus, and numerous experiment-specific devices. Our experience with EPICS in building the control system has been an overwhelmingly positive one, although, as with many distributed development projects, we found that the user documentation was often incomplete.

By providing the Python scripting language with an interface to the EPICS channel access API, members of the DØ collaboration have been able to write nearly all of the operator interfaces to the experiment in a high-level, object-oriented language. Because Python is fundamentally object oriented and provides a number of high-level language constructs and because programs written in scripting languages tend to be significantly easier to debug, the development time for building the DØ online system was significantly reduced compared with what would have been required had the C++ language been used instead.

REFERENCES

[1] L. Dalesio et al., 'The Experimental Physics and Industrial Control System Architecture: Past, Present, and Future', Proc. ICALEPCS, Berlin, Germany, 1993, pp 179-184

[2] D. Harel, ‘Statecharts: A Visual Formalism for Complex Systems’, Science of Computer Programming’, 8 (1987) 231-274

[3] G. Savage, ‘Significant Event System Tutorial’, Internal DØ document

[4] S. Krzywdzinski, ‘EPICS Oracle Database Tutorial’, Internal DØ document

[5] J. F. Bartlett, ‘COMICS: DØ Detector Download Tutorial, Internal DØ document