APPLICATIONS TOOLKIT FOR ACCELERATOR CONTROL AND ANALYSIS

M. Borland
Advanced Photon Source, Argonne National Laboratory
9700 South Cass Avenue, Argonne, Illinois 60439 USA

Abstract

The Advanced Photon Source (APS) has taken a unique approach to creating high-level software applications for accelerator operation and analysis. The approach is based on self-describing data, modular program toolkits, and scripts. Self-describing data provide a communication standard that aids the creation of modular program toolkits by allowing compliant programs to be used in essentially arbitrary combinations. These modular programs can be used as part of an arbitrary number of high-level applications. At APS, a group of about 70 data analysis, manipulation, and display tools is used in concert with about 20 control-system-specific tools to implement applications for commissioning and operations. High-level applications are created using scripts, which are relatively simple interpreted programs. The Tcl/Tk script language is used, allowing creating of graphical user interfaces (GUIs) and a library of algorithms that are separate from the interface. This last factor allows greater automation of control by making it easy to take the human out of the loop. Applications of this methodology to operational tasks such as orbit correction, configuration management, and data review will be discussed.

1 INTRODUCTION

Every accelerator facility faces the challenge of creating high-level applications for controlling and diagnosing the operation of the accelerators. Traditionally, this has taken the form of compiled programs specifically coded for each application and accelerator. A more efficient approach is to solve one's problems using generic and configurable programs. This reduces programming effort and speeds the production of usable applications. It takes advantage of the fact that many accelerator control problems involve similar operations, e.g., orbit correction is really no different from other slow feedback problems. Similarly, many graphical display tasks (e.g., plotting an orbit or plotting vacuum pressure around a ring) involve identical operations.

While one need not use stand-alone generic programs and a scripting environment in order to have generic, multi-use code, it is very convenient to do so. Having algorithms available in the form of ready-made programs rather than as a library of subroutines is far more convenient and allows more rapid development. Individual compiled programs can conveniently be used in stand-alone fashion or coordinated by a script. This coordination creates a high-level application, typically with a GUI, out of reusable low-level components. In the rush to create GUI applications, we must not forget that sophisticated users frequently require more power and flexibility than is easily provided using GUIs. The use of scriptable, command-line-oriented tools satisfies this need in addition to aiding development of GUI scripts. Indeed, at APS many GUIs have their genesis in work done by experts using command-line-oriented tools.

2 SDDS TOOLKIT

The advantage of configurable programs is amplified when many such programs use a common data file protocol. These programs can then properly be called a program "toolkit." At APS, the Self-Describing Data Sets (SDDS) file protocol [1,2,3,4] is used not only for many program configuration tasks, but also for storage of experimental, archival, and simulation data. Briefly, self-describing data gives access to data by name and class only, without reference to position or concern about the presence of data items that are not of interest.

The SDDS Toolkit, comprising about 70 generic programs, provides the muscle for many script applications. The disadvantage of scripts is that they are slower than compiled programs. We mitigate this problem by using the SDDS Toolkit for computationally intensive operations. Operations available include graphics, fitting, winnowing, Fourier analysis, filtering, smoothing, interpolation, cross-referencing, sorting, histogramming, statistics, equation evaluation, and others.

3 INTERFACES VS. ALGORITHMS

A goal of our high-level applications effort is a high level of automation. This is often at odds with the desire to create GUI applications, since these tend to be written assuming the presence of a button-pressing user. We have found that separating the algorithm from the interface is most difficult when it comes to error and status reporting. Interactive applications need to report errors and status differently than noninteractive applications.

A workable way to handle status messages is the use of callback routines passed to an algorithm by a calling routine. In an interactive context, a procedure can be passed the name of a status updating procedure that it can use to provide operator status reports. In a noninteractive context, the same procedure can be passed, for example, the name of an empty callback routine that does nothing other than return.

The Tcl/Tk error catching mechanism is convenient for handling errors in a multi-level procedure context. When an error is encountered in a Tcl/Tk procedure, a return can be made with an error code that is 'caught' by the calling procedure. This procedure can either display an error message or return a catchable error itself. Most of our procedures do the latter only. Only the highest-level, GUI-related procedures display error messages when they catch an error.

In a separate paper in this conference [5], we discuss the Procedure Execution Manager (PEM), a Tcl/Tk envi-

0-7803-4376-X/98/$10.00

ronment for procedure execution. PEM was developed to allow procedures to easily operate with varying levels of operator involvement. For example, operator interaction with a PEM procedure is (normally) automatically confined to the highest-level procedure. In different contexts, the same code will automatically accept operator input or input from a calling procedure. Error and status messages are posted to a single area regardless of the level from which they originate.

4 OTHER TOOLS

APS uses the EPICS [6] controls system. In addition to the SDDS Toolkit, there are EPICS-specific programs that are used in creating high-level applications. Among these are 20-odd EPICS-specific SDDS-compliant programs [2], which include data-collection tools, experiment execution tools, save/restore tools, a general-purpose feedback program (discussed below), and others. These tools are all used by multiple GUI applications.

For displaying live data that is a function of position around a storage ring or in a transport line, the program ADT (Array Display Tool [7]) is used. ADT is configured by an SDDS file that tells it what process variables to display and an optional additional file that supplies information on how to draw the accelerator lattice. ADT allows saving and recalling of data to memory or files, tracking of minimum and maximum values, computation of statistics, differencing of current and saved data, and other features. It is most commonly used to view the orbit in the APS ring. In several cases, scripts use SDDS tools to create ADT configuration files on the fly to display context-dependent data.

5 EXAMPLES

In this section, we present applications of the above methods, all drawn from routine APS operations. All of these applications rely solely on the SDDS-compliant programs and Tcl/Tk scripts, unless otherwise noted. These examples illustrate how much may be accomplished with a set of tools like those in use at APS.

5.1 General-Purpose Workstation-Based Feedback

As mentioned, one of the EPICS-specific SDDS tools is a general-purpose feedback program that is configured by SDDS files. This compiled C program is used for controlling rf gap voltages under beamloading conditions, for providing specialized power supply regulation, for correcting the global orbit, for local beamline steering, and for correcting transport line beam trajectories. The program itself has no GUI, but is run in its various instances by different GUIs. Some of these are simply configurable instances of the same GUI, while others are custom-made for specific applications. One GUI allows on-the-fly creation of a simple one-readback, one-actuator feedback loop.

The orbit correction system for the APS storage ring [8] is broken into several parts. One script is used to edit configuration files specifying which correctors and beam position monitors to use. This script also performs singular value decomposition (using a compiled program) to create the inverse matrix file, starting with a forward matrix from either simulation or experiment. The job of using the inverse matrix to perform orbit correction is performed by another, independent script. The advantage of this arrangement is that either of these functions may be performed by any means desired, i.e., one could have several scripts that perform the actual correction or several that prepare input data. This is convenient during development of new algorithms, as it allows one to work on only that part of the system that is of interest. In addition, it allows use of interfaces for multiple applications; for example, the same script that prepares input for the workstation-based global correction scripts also prepares input for the real-time feedback system [9].

5.2 Configuration Management

Accelerator configuration or settings management is a problem that all accelerator facilities face. At APS, the "save/compare/restore" or SCR system is built using SDDS files, SDDS tools, and Tcl/Tk. For each of 15 major subsystems, a "request file" is available that lists and describes the process variables (PVs) that are important for that system. In addition to being used to acquire PV values from the controls system, the request file gives tolerance data, protection status, categorization, and other information that determines how each PV is treated in compare, restore, and review operations. Adding new PVs or a new major subsystem is largely a matter of adding entries to a request file or creating a new request file.

Partial review, restore, and comparison operations are supported via several mechanisms. Operators may choose categories and subcategories from those defined in the request file. Since these may change with time, the actual categorizations are extracted from the saveset and used to configure the GUI. Operators may also choose to select PVs based on "filter files," which are lists of PVs with a specific functional relationship (e.g., all PVs relevant to steering a certain beamline). For some subsystems, graphical display of comparison operations is available; addition of more such displays is planned.

In addition to the main SCR script, several other applications make savesets using the same procedure that the main SCR script uses. This procedure permits any application to make "backup" data giving the state of the accelerator prior to making a change. If the operator needs to recover from the change, he then uses the main SCR script rather than another, redundant interface. If the GUI SCR script were not designed with separation of interface and algorithm in mind, this feature would not be as simple to implement in as many contexts.

5.3 Data Review

Another function required at accelerator facilities is review of archival data. All archival data collection for accelerator operation is performed using SDDS-compliant EPICS programs. This includes simple time-series data collection, alarm logging, and glitch- or beam-dump-triggered data collection. The data collection processes are managed using UNIX cron jobs. This includes daily starting of jobs, automatic restarting of jobs following a crash,

and daily postprocessing of data. Postprocessing includes recovery of any corrupted files that might result from a system crash, compression of the data, and in some cases analysis of the data.

Normally, users access this data through Tcl/Tk GUIs. In most cases, a common GUI is used for accessing all the time-series data, another for all the alarm log data, and so on. In several cases, customized Tcl/Tk GUIs have been provided for certain systems that have unusual or specific display requirements (often involving preprocessing of data prior to display). Users requiring even more access can make use of the same Tcl procedures used by the GUIs to find the data of interest (typically by specifying the group the data belongs to and the time interval of interest). There is also a mechanism for exporting data to user files in various formats, such as SDDS or spreadsheet format.

Another application of this data is the automatic creation of GIF images for display on the World Wide Web. About 150 different plots are created at intervals of one to 15 minutes, displaying data from the APS storage ring vacuum, absorber, and rf systems. These plots are created using the same SDDS graphics program that is used to create plots from Tcl/Tk scripts. The Experimental Facilities Division at APS has created a Web interface that permits on-demand plotting of SDDS data from their systems using SDDS tools and Perl scripts [10].

Several types of glitch-based data collection jobs are used, again based on SDDS-compliant EPICS tools. These collect data at a relatively high rate in a circular buffer, until one or more glitch or trigger events are seen. Following the event, data collection continues for a specified period, after which the data is added to the log file. A single interface is used for all of the glitch data, with customized plotting and analysis available for some data sets. For example, orbit glitches are logged for the APS storage ring (a glitch is defined as a change in a beam position reading over a defined threshold at one or more locations). The user may select events based on cause (e.g., x orbit motion, beam loss), then plot the raw data, plot difference orbits, or perform analysis to find the most likely corrector to have caused the motion.

A third type of data collected in large quantities at APS is alarm information. The most typical use of this data is to determine when a certain piece of equipment tripped. The GUI provided as an interface to this data provides simple printouts of events and the times they occur; it also creates histograms of alarm density and total alarm counts per channel within a defined period. A related tool provides probability analysis of alarm data, indicating, for example, whether an observed alarm rate during one period is improbable compared to the alarm rate in another period.

5.4 Digital Scope Control and Data Acquisition

APS has a number of remotely-controlled digital oscilloscopes used for monitoring pulsed magnet waveforms, rf signals, current monitor signals, and the like. At present, two of these instruments are controlled using Tcl/Tk scripts and SDDS tools. The scripts perform configuration save and restore operations for the scope and essentially arbitrary associated equipment (e.g., multiplexers). Recently, we have implemented waveform documentation software that sets up equipment (e.g., a pulsed power supply) in a standard way, then acquires and archives waveform data. This permits tracking of changes in the response of such equipment.

5.5 Accelerator Measurements

A number of accelerator measurements have been implemented as GUI scripts that were originally developed by physicists using SDDS tools. To a large extent, we simply wrapped a GUI around the algorithm created by the physicist. Among the applications developed for the APS storage ring are measurement of small orbit and tune changes due to undulator gap changes, measurement of BPM offsets using orbit bumps and variation of quadrupole strength, measurement of the beam motion spectrum, and automated timing of BPMs.

6 CONCLUSIONS

The combination of the SDDS toolkit, SDDS-compliant EPICS tools, and Tcl/Tk has proven very effective in the development of high-level applications for APS operations. True reusability of programs is made possible through the use of self-describing data and proper generalization of program functions, saving programmer effort and speeding the development of new applications. Separation of the user interface and the algorithm has made it possible to reuse code and automate operations. The command-line orientation of the SDDS tools enhances use by experts, provides easy integration into scripts, and permits rapid use of expert-developed algorithms in GUIs.

7 ACKNOWLEDGMENTS

Contributions to the above work have been made by a number of individuals at APS, including J. Carwardine, L. Emery, K. Evans, C. Saunders, and N. Sereno. Work is supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

8 REFERENCES

[1] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," Proc. of the 1995 Particle Accelerator Conference, Dallas, Texas, pp 2184-2186 (1996).

[2] L. Emery, "Commissioning Software Tools and the Advanced Photon Source," ibid, pp 2238-2240.

[3] M. Borland et al., "Doing Accelerator Physics Using SDDS, UNIX, and EPICS," *Proc. of ICALEPCS* '95, pp 382-391 (1997).

[4] M. Borland et al., "The Self-Describing Data Sets File Protocol and Toolkit," ibid, pp 653-662.

[5] M. Borland, "The Procedure Execution Manager and Its Application to Advanced Photon Source Accelerator Operation," these proceedings.

[6] L. R. Dalesio, et. al., "EPICS Architecture," *Proc. of ICALEPCS* '91, KEK Proceedings 92-15, pp 278-282 (1992).

[7] K. Evans, private communication.

[8] L. Emery et. al., "Advances in Orbit Drift Correction in the Advanced Photon Source Storage Ring," these proceedings.

[9] J. Carwardine et. al., "Commissioning and Performance of the APS Real-Time Orbit Feedback System," these proceedings.

[10] M. Ramanathan, private communication.