OPERATIONAL TOOLS AT THE STANFORD LINEAR ACCELERATOR CENTER*

G.White#, S. Chevtsov, P. Chu, D. Fairley, C. Larrieu, D. Rogind, H. Shoaee, M. Woodley, M. Zelazny, Stanford Linear Accelerator Center, Menlo Park, CA 94025, USA

Abstract

Operational tools, for the purposes of this paper, are specifically those software applications which are used in the context of operations, to analyze or optimize a large scientific instrument. At SLAC, such tools for online accelerator physics, have been in continuous development for 20 years. This paper first reviews those tools from the perspectives of their scientific functionality and implementation. Present operational software developments are then introduced, together with the infrastructure created to enable migration from the host computers and operating systems on which those tools have run in past, to the new systems. Some techniques and experiences in bridging EPICS, CORBA, Matlab, and Eclipse are included.

INTRODUCTION

The first part of this paper presents a review of the scientific software applications that have been developed for use in the particle accelerator complex at SLAC. Such tools are concerned with the diagnostics and optimization of accelerators, typically by applying accelerator modeling and numerical methods to online instrument data, and applying the results to the control system. Examples of such systems are “Correlation Plots”, which is a facility for conducting small ad-hoc experiments, orbit correction, beam-path modeling, lattice matching, feedback, model diagnostics, beamline diagnostics, calibration, and beam-based alignment, and other tools for online experiments. For many years at SLAC, these have all been collocated in a single executable, which enables very tight application integration. Those mature, and very successful, applications are described first. Then we describe present work to transition those existing applications, plus new tools and methods that have been enabled by advances in technology, to new architectures, in both hardware and software. This includes a description of the foundational infrastructure that we are putting in place to support both the migration, and scientific scripting directly from tools like Matlab, Excel, gnuplot, and so on. Lastly, we introduce the long term application infrastructure we are developing for our vision of the architecture required for online accelerator science over the next decade.

THE SLAC CONTROL PROGRAM

The central online operational tool used at SLAC has been the "SLAC Control Program" (SCP). This single executable incorporates all scientific applications that we initially wrote for the Stanford Linear Collider, and all SLAC accelerators and beamline experiments since. The SCP is composed of >2M lines of high level code, mostly Fortran, some C/C++, implemented in 48 dynamically linked libraries, running on Alpha/VMS hosts. It mostly uses a proprietary control network (slcnet) to communicate with iRMX front-end processors and, notably, EPICS (Experimental Physics and Industrial Controls System) front-end processors using a specially crafted control network message-code bridge system called the SLC-aware-IOC, implemented itself in EPICS Input/Output Controllers (IOCs). All the scientific tools and all other control displays, such as magnet, BPM, and RF setup and operation, co-reside in a single instance of the SCP executable. The user interface is shown in Figure 2. In total there are ~3500 panels in the SCP, only some relatively small fraction of which are involved with operational scientific applications.

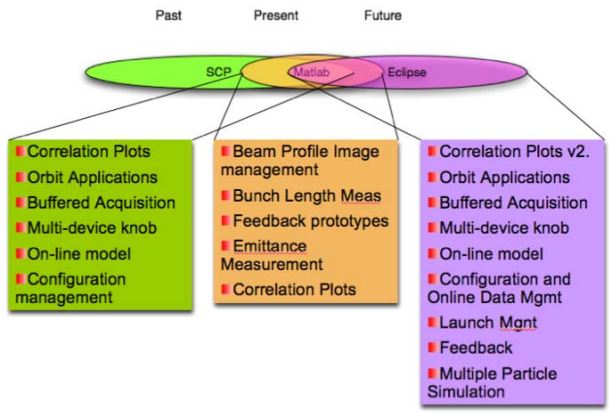

Figure 1: Transitionary phase of SLAC online physics software tools. Physics tools are being moved from a legacy Fortran/C VMS system, to Java/Eclipse on Linux; matlab is helping to bridge the gap.

An advantage of implementing applications in this monolithic suite of dynamically linked libraries, is that they are interoperationally very tightly integrated. All persistent state data of every application is available to every other application, without, in fact, a strict API or layered software architecture, or formal state sharing mechanism, and every application can directly call functions of any other. Although informal, this has proven very effective in the specific lifetime of this system, ~1985 to the present.

SELECTED SCIENTIFIC APPLICATIONS OF THE SCP

This section reviews some of the scientific applications of the SCP. These are now very mature and include significant scientific utilities and usability features.

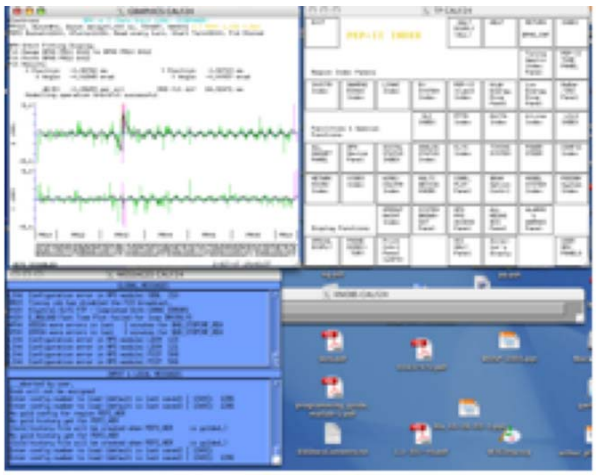

Figure 2: The user interface is composed of 3 main screens: i) a "panel" screen housing all the “buttons” of the Graphical User Interface (there are no pull-down menus, sliders, dials etc), ii) a graphics screen on which all data is presented, mostly in tabular or graphical form, and iii) a continuously updating message window that presents all messages relating to what the individual SCP user is doing in one half, and all global messages of the accelerator in general in the other half.

Beamline Plotting

A utility application that can plot an arbitrary function given by coordinate pairs along a beamline, is used to plot beam position data, dispersion, any twiss parameter, calculated orbit correction etc, by Z location. A notable feature is that it can augment the ordinate axis with meaningful information, such as lattice description and region names; and it can estimate the number of device names that can meaningfully be inscribed onto the axis.

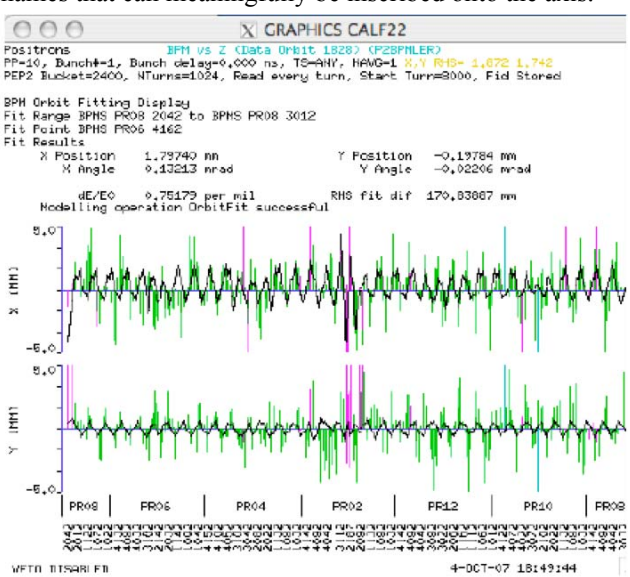

Figure 3: Orbit fitting in the SLAC Control Program. This brings together orbit acquisition, save/restore configurations, and beamline orbit plotting.

The beamline plotting system is used by many applications of course; among them orbit plotting and orbit fitting (see figure Figure 3). Two very heavily used applications are Orbit Correction and Correlation Plots.

Orbit Correction

The mechanism of Orbit Correction has been described many times elsewhere, but for the purposes of describing some interesting specializations of it developed for the B-factory at SLAC, it is first outlined briefly here. The role of orbit correction (sometimes called “Steering”) is to reduce the RMS of the beam orbit and hence overall emittance. In its simplest form, it reads the value of beam position measurement devices (BPMs) in the beamline section to be corrected, and calculates new energization levels for corrective lattice devices (typically “corrector” dipoles), by solving the system of linear equations which relate the effect of each the corrective device on the beam position at each BPM, for the present orbit offset values, and then changing the energization level in the correctors by the sign reversed solution value. That is, it solves the matrix equation:

$$\min \| Ax - b \|_2$$

$$\text{subject to } x_j < x_{j\,\max}$$

where A is the matrix of coefficients of the system of equations relating the effect of each corrector x_j on each BPM b_i. Typically, these coefficients are precomputed from the beam lattice model (though they may be found experimentally). For X plane orbit correction using only X-plane correctors the elements of A would be the so-called R12s. The “orbit correction problem,” is then formed by the unknown x, being the vector of desired corrector changes (sign reversed), and b being the vector of present BPM readings*. At SLAC, two main numerical methods are used to compute the minimization: Micado, and a specially formulated Singular Value Decomposition (SVD).

Some interesting facilities have been developed in the context of SVD for orbit correction at SLAC. The first is an extension of the SVD numerical method itself. It is to include interval constraints on the solution elements x_j and so include the practical limits of corrector maximum settings directly in the numerical problem. This is done by casting the eigenvalue decomposition done by the SVD, which eliminates the basic problem of matrix degeneracy inherent in the lattice of the B-factory accelerator, into a space in which the solution elements’ interval limits can be posed in a minimization problem which can be given to a solver that can accept solution intervals, but not singular objective matrices (A). These two numerical solvers were formerly mutually exclusive, since the SVD itself does not include solution intervals, and even robust

* Don’t be confused by the use of the letter x above to designate both the unknown vector in the matrix form of a system of linear equations, and to designate the X plane of an orbit

Chi Sq methods, which can include intervals, do not work well to solve near-singular systems. This hybrid numerical method is described fully in [1].

Both the beam offset from the main axis (in X, Y, or in a plane coupled region X and Y together) may be minimized, together with dispersion simultaneously. This is all done by constructing the objective matrix A as a block matrix of appropriate sub-matrices, and weighting the blocks appropriately [2]. The block matrix characterizing dispersion, which would be rows of R126s added under the R12s of the orbit block matrix in A, is nominally weighted 0.05 (so the orbit block would be weighted 0.95), though this can be changed from the user interface.

Other additions include:

1. Either the model transport "R" (or sometimes called "T"), or, for a ring, the closed obit "C" elements, can be used to form the orbit block of the objective matrix A
2. If numerically underconstrained, the problem can optionally be posed such that the particular solution will be that which would result in the smallest *absolute* corrector values
3. Residuals that are at high variance compared to the others, may well correspond to BPMs which are simply reporting incorrect measurements. The equations for such BPMs can be removed according to a user specifiable residual value "sigma-cut". Failing BPMs are reported, then removed from the problem, and the minimization is repeated. Reported BPMs can be interpreted as a prompt to check the hardware
4. The orbit position and angle can be held constant at any location, for instance ring injection, again by augmenting the objective matrix A with highly weighted equations.

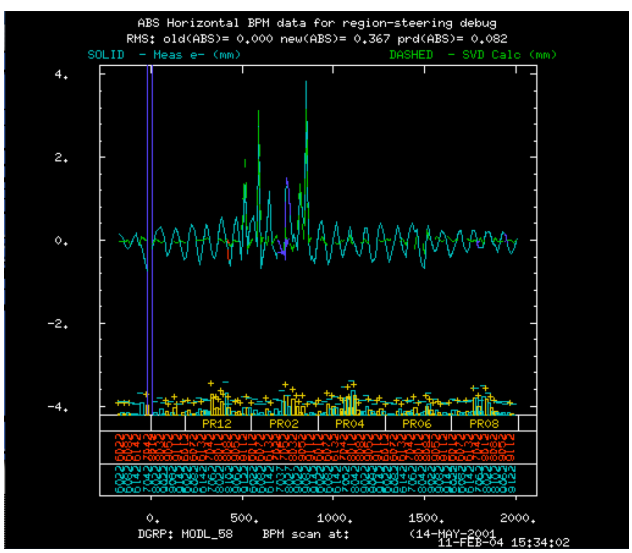

Figure 4: Orbit Correction example from the SLAC B-factory, showing an acquired beam orbit and calculated correction and predicted resulting orbit.

Multiknobs and Bump calculation

The "multiknob" facility, for simultaneously changing the value of many devices, each by some designated coefficient, is heavily used at SLAC. The "Bump" application, which builds on multiknobs, can be used to calculate corrector coefficients whose effect will be to sweep the beam though some range of orbit offset (a "position" bump) or angle, using 3 or 4 correctors, while holding the orbit outside the bump constant.

Correlation Plots

One key contribution of software to the analysis and optimization of the SLAC accelerators, has been the Correlation Plots (CP) application. CP enables a user to conduct simple accelerator experiments online, through its ability to scan any "stepable" control system variable, and read back many other "sampled" variables on each step. The stepped and sampled variable values are recorded, and results can be plotted and fitted. Expressions in the variables themselves can evaluated in CP's spreadsheet-like GUI. Both 1D and 2D scans are supported. An important aspect of Correlation Plots, which probably gives rise to much of its popularity, is that the variables it understands how to step and measure need not be simply control system process variables, but rather richer scientific quantities computed by special purpose code elsewhere in the SCP. For instance, CP knows the procedure to step the energy of the B-factory, or drive a multiknob, and it can sample such things as the fit parameters of an orbit fit computed by the application described above, as well as many other similar "macro" variables.

The above overview gives some flavor of the applications in the SLAC Control Program, which have been used to optimize the accelerators in the SLAC complex for 20 years. However, with the commissioning of the Linac Coherent Light Source (LCLS) accelerator, the three complexities of any such significant software reimplementation arose. Firstly, the rich and mature legacy applications, on VMS, should be usable with the process variables of new controls system, which was based on EPICS. Secondly, we wanted the legacy proprietary controls system, modeling and data processing of the legacy system to be available to unix based commissioning tools like Matab, and to the new scientific applications. Thirdy, the new scientific applications themselves to be rethought. The new applications and required infrastructure for this transition is where the remainder of this paper concentrates.

DATA AND CONTROLS INTEGRATION

To connect the legacy applications to the new EPICS based control system, we developed an "SLC control system aware IOC". This is a control system message code bridge, which translates messages such as "trim this list of magnets" or "acquire beam position data for this list of beam monitors" in the protocol of SLC control

system, to Process Variable operations and sequences that can be acted on by EPICS Input/Output Controllers.

To connect modern unix based Java or Matlab based applications to the legacy VMS control system, a CORBA based internetworking system called AIDA (Accelerator Independent Data Access [4]) was used. AIDA was first developed as a design study for the International Linear Collider (ILC). AIDA provides data location and semantics transparency to client applications. That is, a client asks for data (or to set it), and AIDA works out which control system, data storage or modeling system etc and host computer to ask for that data, and most importantly, makes the translation of how to ask.

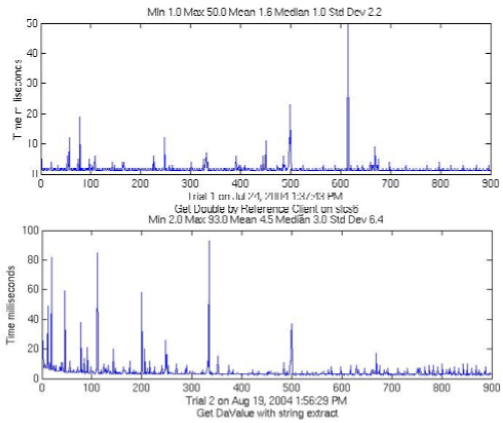

Figure 5: AIDA is implemented in CORBA. The graphs show AIDA's Java CORBA (Orbacus) performance over a 100Mbit Ethernet for acquisition of a single Double value (top) and for a dynamically constructed structured data object of an array of 11 doubles and an array of 11 strings (bottom), showing 900 roundtrip times. Note the Virtual Machine warmup evident at the beginning of the test of the structured object. In practice, AIDA's median performance of 2ms for simple data, and less than 10ms for structured, has been easily fast enough for high level applications software.

MATLAB APPLICATIONS

Matlab has been used both for scripting *ad hoc* solutions during the commissioning of LCLS, and as an interactive GUI application platform. Since new aspects of the SLAC accelerator controls specifically developed for LCLS, were done in EPICS we use the labCa system to connect Matlab to EPICS Process Variables (PVs) [3]. To enable the body of our legacy control system and applications to be used from within Matlab for LCLS commissioning, we employed the Java API of the Accelerator Independent Data Access (AIDA) system described above, since Matlab allows java calls to be made directly from scripts with no wrapping. We shall also be using this feature of Matlab to script XAL.

Given the infrastructure tools above, our experience of Matlab has been that it is an effective rapid development tool for applications that used the numerical analysis of Matlab significantly. Among these applications were a framework for prototyping and testing feedback algorithms, and GUI applications for beam profile image acquisition and analysis, beam bunch-length measurement, a Matlab based system for conducting simple correlation finding experiments through the control system, emittance measurement, and many other smaller applications.

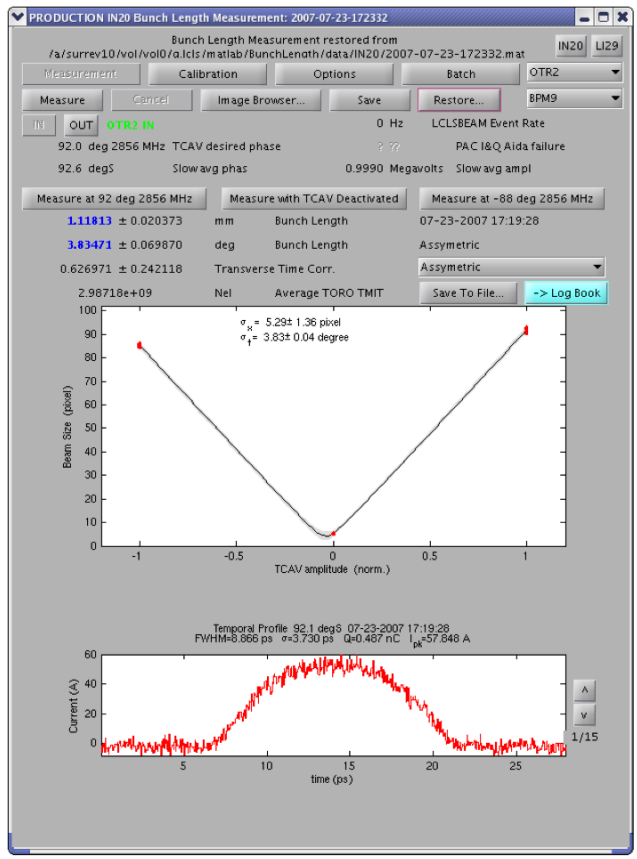

Figure 6: Screenshot of the Bunch Length Measurement application scipted in Matlab, using in Mathwork's GLIDE tool for constructing graphical user interfaces for Matlab programs.

Use of Matlab as an online tool also proved very popular among scientists who were already familiar with Matlab for data analysis. When they were provided methods to interface to the control system and data acquisition from directly within Matlab's scripting language, they could script applications and ad hoc analysis for themselves.

In these contexts, it very successfully satisfied our commissioning schedule. However, Matlab GUIs are slow, particularly if used in some thin client remoting context like over X11, and since it's oriented toward scripting it lacks programming constructs required for large, integrated, error-tolerant control system user interfaces. Additionally, some programmers who were accustomed to coding tools available in modern Integrated Development Environments (IDEs), like "refactoring" (where all instances of the use of a method or object, can be changed by the editor automatically) found the process of programming itself clumsy. In the light of these

drawbacks, we shall develop the production quality scientific applications necessary for LCLS operations (as opposed to commissioning), using a purpose built Java GUI framework, and use Matlab within that GUI framework for numerical and graphical tasks.

SLAC ECLIPSE ACCELERATOR LAB

Physics software applications for the Linac Coherent Light Source, and some other large experimental facilities of SLAC, will be developed in a new software framework. This platform for applications, is implemented as a Java Eclipse RCP (Rich Client Program) [6]. It system integrates code from a number of sources; XAL, a Java package for control and optimization of particle accelerators [7][8]; Java code emitted by the Matlab Builder for Java [9]; together with Aida and Java Channel Access data interfaces, and applications of the Control system Studio (CSS) [10].

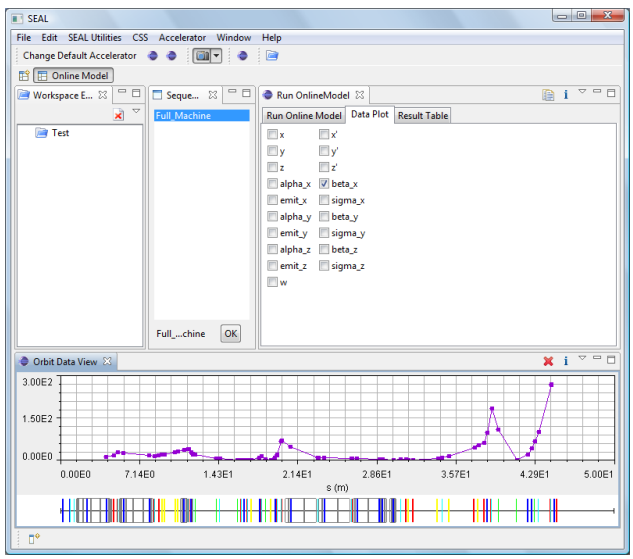

Figure 7: SEAL with the *Online Model* perspective. The perspective contains a workspace navigator displaying the workspace file system (upper left), a beam line sequence selector (upper middle), an online model run control (upper right) and an orbit data plot panel for showing the model run result (bottom).

CONCLUSIONS

Historically at SLAC, scientific software developed for accelerator operations, have been very successful. This has been, in part, due to the great interoperation of the applications themselves, together with their very specific concentration on the physics problems of the machines on which they were used. These joint considerations are guiding us in new developments when using modern software tools and techniques.

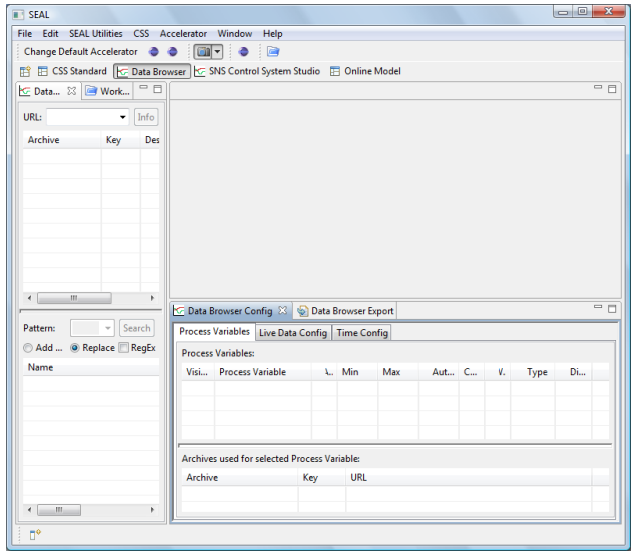

Figure 8: SEAL with the *Data Browser* perspective. The perspective includes Data Browser Archives view (left), Data Browser Config view (bottom right) and a blank editor area (upper right).

REFERENCES

[1] G. White, T. Himel, H. Shoaee, Hybrid Numerical Method for Orbit Correction, SLAC-PUB-7653, presented at 17TH IEEE Particle Accelerator Conference (PAC97), http://www.slac.stanford.edu/pubs/slacpubs/7000/slac-pub-7653.html.

[2] Gene H. Golub and Charles F. Van Loan, Matrix Computations, The Johns Hopkins University Press, 3rd Ed 1996.

[3] G.White et al., AIDA: ACCELERATOR INTEGRATED DATA ACCESS, presented at ICALEPCS'01. http://www.slac.stanford.edu/econf/C011127/THAP011.pdf.

[4] Till Staumann, SLAC, labCa.

[5] High-level Application Framework for LCLS, Paul Chu Chungming et al., ICALEPCS'07.

[6] http://www.eclipse.org.

[7] T. Pelaia et al., XAL Status, ICALEPCS'07.

[8] J. Galambos et al., XAL Application Programming Framework, ICALEPCS'03.

[9] The Mathworks, http://www.mathworks.com/products/javabuilder/

[10] J. Hatje, Control System Studio (CSS), ICALEPCS'07.