XAL APPLICATION PROGRAMMING STRUCTURE*

J. Galambos, C. Chu, S. Cousineau, V. Danilov, J. Patton, T. Pelaia, A. Shishlo, Spallation Neutron Source, Oak Ridge National Lab, Oak Ridge, TN, U.S.A.
C.K. Allen, Los Alamos National Lab, Los Alamos, NM, U.S.A.

Abstract

XAL is an application programming framework used at the Spallation Neutron Source (SNS) project in Oak Ridge. It is written in Java, and provides users with a hierarchal view of the accelerator. Features include database configuration of the accelerator structure, an online envelope model that is configurable from design or live machine values, an application framework for quick-start GUI development, a scripting interface for algorithm development, and a common toolkit for shared resources. To date, about 25 applications have been written, many of which are used extensively in the SNS beam commissioning activities. The XAL framework and example applications will be discussed.

INTRODUCTION

XAL [1] is a high level programming infrastructure developed to provide application programs for beam commissioning at the SNS. It consists of several components, which can be used together in many ways. These components include a hierarchal class representation of the accelerator, a model of the accelerator, a high level wrapper to the control system, a framework for writing GUI applications and a set of tools. These components provide a simplified interface for the accelerator physicist to the SNS machine. The entire framework is written in Java, and uses a database for storing the accelerator configuration. XAL applications have been used now for about 2 years of the initial beam commissioning.

An overall view of the XAL structure, relative to the control system is shown in Fig. 1. Communication with the diagnostics and beamline devices is through the EPICS [2] control system, or more specifically via the Java Channel Access (JCA) interface. Many of the components shown here are discussed in the following sections.

ACCELERATOR VIEW

A primary feature of XAL is the class structure representing the accelerator. By now, this approach is used widely for both modeling and machine applications, for example see Refs [3-5]. The accelerator classes include sequences, and nodes. There is a one-to-one mapping between nodes and "real" pieces of beamline hardware. Example nodes are magnets and RF cavities. Magnets are further subdivided into quads and dipoles, and so forth. The sequences are defined in terms of the SNS primary systems (transfer lines, ring sections, linac sections etc.). These sequences can be pasted together to form "combination" sequences. Applications are often written to operate on any sequence, which allows reuse of the applications. For instance, the orbit display or orbit correction applications needs to be written only once, and the user can select the region of the accelerator of interest.

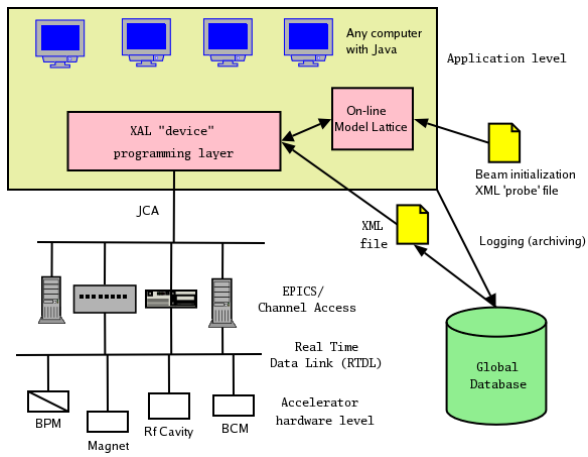

Figure 1: Application software infrastructure.

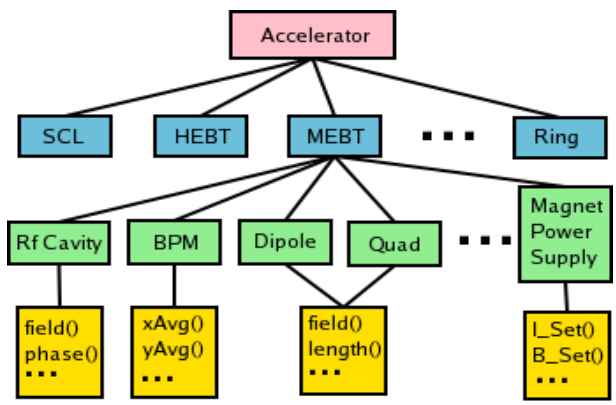

Figure 2: Schematic of the XAL accelerator class hierarchy.

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy. SNS is a partnership of six national laboratories: Argonne, Brookhaven, Jefferson, Lawrence Berkeley, Los Alamos, and Oak Ridge.

User knowledge of the details of the interface between the nodes and the control system is not necessary. Simplified interfaces are provided for commonly used actions. For instance, magnets have a "getField()" method that instantiates a connection to the control system, determines which signal to query, and returns the readback value of the magnetic field.

Database Configuration

The accelerator structure described above is configured from an Oracle database [6]. The tables used for defining the beamline structure represent a small fraction of a much larger database effort [7]. While the configuration can be setup directly from the database, we more commonly use an intermediate XML file representation of the accelerator that is also configured from the database. The XML accelerator setup is faster than the direct database query, and the accelerator configuration does not change often enough that this is a problem.

Use of a common database configuration for all applications has several advantages. When a new beamline device is added or removed, all applications have knowledge of this update. Also, as new portions of the accelerator are brought online, for many applications the only task is to populate the database for the new accelerator sequence(s). However, the initial database population effort should not be minimized. This is a tedious, yet important task, but thankfully must only be done once.

ONLINE MODEL

A critical component of many beam commissioning activities involves comparing measured quantities with model predicted values. To facilitate this, XAL includes an "online" model [8,9], which follows the probe-algorithm, element pattern from Ref. 3. The model has both envelope and single particle tracking capability, and has capability for single pass (linac + transfer line) and closed orbit (i.e. ring). Since SNS is a high intensity proton device, space charge effects are incorporated in the model. Four data sources for magnet and RF settings are accommodated: 1) design values from the database, 2) live values from the real machine, 3) user "what-if" input, and 4) archived "snapshot" values.

Importantly, the online model elements are distinct from the accelerator nodes described above. The model uses a separate "lattice view" that is constructed from the accelerator sequences and nodes. Whereas a quadrupole accelerator node is a single element in the database configuration, it is typically split into several sections in the model's lattice view. Also, drifts (critical for modeling) are included in the lattice view but not as nodes.

TOOLS AND SCRIPTING

Tools

Many general purpose tools have been written, which are shared between applications, some of which are described here.

The SNS is a 60 Hz pulsed device, and each signal has a timestamp attached, corresponding to the start of the pulse. A correlation tool allows collection of groups of signals with comparable timestamps. It has many features including triggered acquisition to allow sets of data to be grabbed only when some specified external criteria is met.

A data table structure provides simple database like functionality. This structure provides more capability than the built in Java collections (e.g. database like querying), yet is much simpler to implement than construction of a real set of database tables.

A communication layer is available for client-service needs. It uses XML-RPC for inter-process data exchange and uses Rendezvous for discovering subscribers and publishers. Knowledge of the details of Rendezvous and XML-RPC are hidden from the user.

A common need for application programs is the need to scan one quantity while monitoring others. Scanning tools have been written, which handle many of the tedious tasks involved here. These tools are implemented in general purpose scanning applications, as well as more specific applications with complicated data analysis needs.

Additional tools include plotting, database connection, XML data parsing, and optimization packages.

Scripting Interface

A common first implementation of many of the tools and XAL packages is via a scripting interface. We use Jython [10] as the primary scripting language with XAL. Jython is a Java version of the more common Python scripting language. Importantly, any Java classes can be used directly with jython, without the need to write the usual software interface glue code needed with more traditional languages like C++. Jython is used to write quick tests and example usage for new XAL packages, perform "post-facto" data analysis, do tedious one-off machine communication, and even to write simple applications. The scripts are typically much faster to develop than GUI applications, and if fruitful, can subsequently be converted into a traditional GUI application.

We have also tested XAL interfaced directly with Matlab. As Matlab now supports importation of Java classes into it's scripting language this is possible. However, most XAL developers prefer the Jython language.

APPLICATION GUI FRAMEWORK

Initially in the XAL development, each application developer created their own GUI interface from Java

swing components. This approach had drawbacks of duplicated effort, a different look and feel for each application, and maintenance difficulties. As such, a common GUI application framework base class was created to serve as an application template. This gives application developers a jump-start for laying out the application and also provides a common look and feel to the users. Also, features can be added to the base framework, and appear in all the applications. This approach has the additional advantage of easing the first time application development for "newbie" application developers.

The application framework panel and main menus is shown in Fig. 3. This framework incorporates some service features which help with administrative tasks. For example all applications broadcast information, and a special viewer application allows a user to see which applications are running, how long they have been running, memory usage, garbage collection, and can force an application to quit.

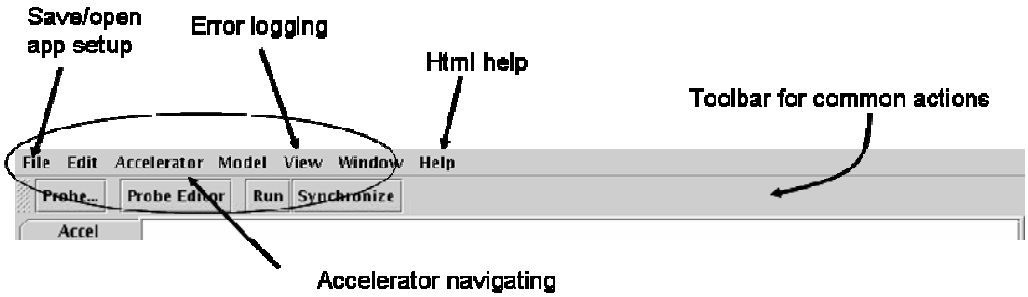

Figure 3: The XAL GUI application framework view, showing common features inheritable by all applications.

SERVICES

As mentioned above, one of the XAL tools is a mechanism to support client server communication. A few services have been written that utilize this feature. An example is the administrative application viewer tool that can determine the existence and status of other XAL applications. Another example is a machine protection system service that continually monitors machine trips, performs post mortem analysis and keeps machine trip statistics, which are stored in a database table. A logger service is also available for taking snapshots of prescribed sets of signals, and storing them in a database. The main use for this service is to log the machine settings needed to set up the online model, but a web application is also available for anyone to configure a set of signals to log. We expect the use of services to expand as XAL matures.

APPLICATIONS

The raison detra for XAL is for commissioning the SNS accelerator. To this end, over 25 client applications have been written, a few of which are described here. For the most part, these applications involve to accelerator physics, i.e. comparison of measured quantities with model predicted values, but there are also applications that automate tedious tasks. A few of each of these sorts of applications are briefly described below, to indicate activities that XAL is supporting.

Physics Applications

A screen snapshot of the online model application is shown in Fig. 4. The user can select the portion of the accelerator to analyze as well as a data source (either the machine, design values or user specified what-if values). Output includes beam Twiss parameters as well as centroid motion. Other features include a probe editor (to modify the beam initial conditions, comparison with diagnostic output (BPMs and wire scanners) and the ability to view differences from previous calculations, after modifying a magnet setting. The latter feature is useful for example, in performing orbit difference studies to identify polarity problems for BPMs and dipole correctors.

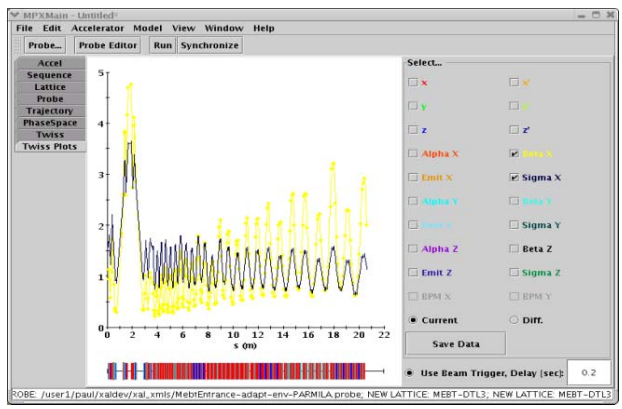

Figure 4: The online model application, with the results plot tab showing.

Several applications have been written to set RF structure phase and amplitudes. One example, PASTA (Phase and Amplitude Scan and Tuning Analysis) shown in Figure 5, is used for tuning the Drift Tube Linac (DTL) and Coupled Cavity Linac (CCL) structures. This application uses the online model for longitudinal dynamic calculations. The application also uses scanning, plotting and solving tools, in addition to the online model XAL packages. It has provisions for performing phase

and amplitude scans on the RF structures, and monitoring downstream BPM phases. By matching the measured "signature" of the measured scans, with model predictions under different conditions, the RF phase and amplitudes can be determined.

A common procedure in understanding the beam characteristics, is to calculate the beam Twiss parameters at some point in the accelerator, based on a set of profile measurements and knowledge of the transport line between the elements. An "Auxiliary Optics Control (AOC)" application has been developed to do this. It allows the user to select the beamline, and beamline elements to use in the analysis. This application uses the online model, plotting and optimization packages, as well as the database logger package to reconstruct the state of the beamline at the time the wire data was taken. A screen snapshot of this application is shown in Figure 6.

A number of applications are under development to assist in the SNS storage ring commissioning. One important aspect in ring commissioning is the insuring the correct beam properties (size and position) at the injection spot. The application displayed in Figure 7 will assist in this task. The user can select Beam Position Monitors (BPMs) in the ring to use to get the turn-by-turn positions as the beam circulates. The magnitude and frequency of the beam oscillations are used to calculate both the Ring tune as well as the beam coordinates at the injection point. Since the Ring is not available yet, the signals displayed in the Fig. 7 screen snapshot are generated artificially for testing. We also have a virtual accelerator application that uses the online model to produce realistic signals for common beamline elements (e.g. BPMs) and accepts signals to modify magnets and RF. Using artificially generated signals is a common practice for XAL application development, and greatly decreases the debugging time needed in the control room.

General Purpose Applications

In addition to applications that are primarily physics oriented (i.e. rely heavily on the online model), many other applications have been written to assist in commissioning. These applications typically involve automation of tedious tasks, and or some processing of the signals provided by the control system. Several of these are described below.

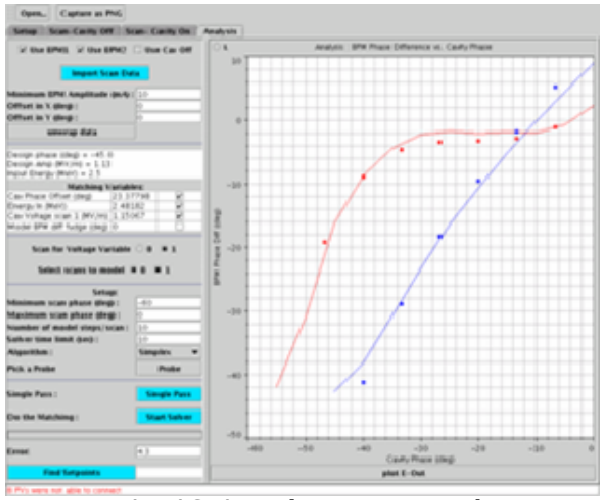

Figure 5: A PASTA application screen shot, showing a comparison of model (dots) and measured lines (BPM) phase differences.

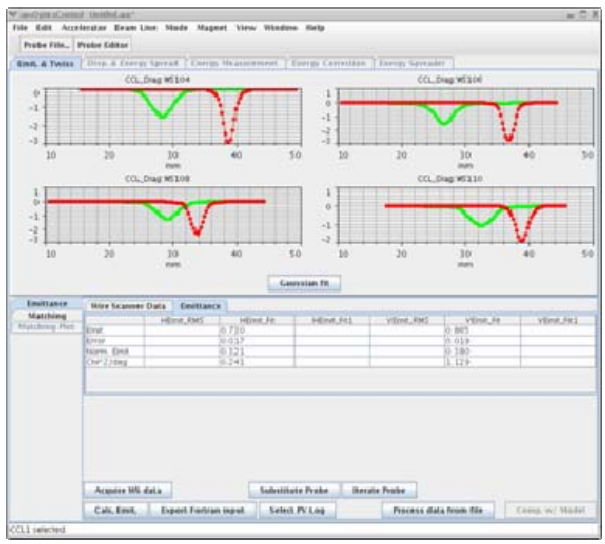

Figure 6: An AOC application screen shot, showing imported wire profile data used in the calculation of upstream Twiss characteristics.

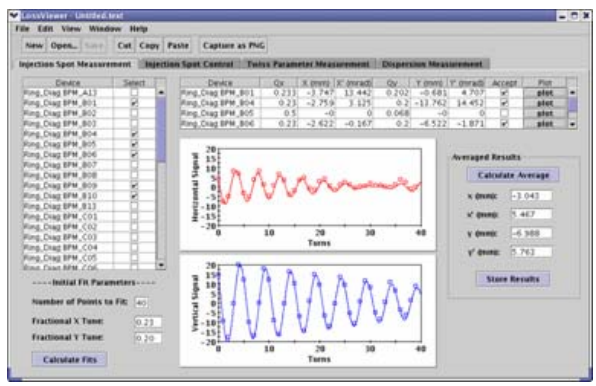

Figure 7: The Ring injection application, being developed using simulated signals from the virtual accelerator.

Several applications have been prepared to do "on-the-fly" experiments. One of these is the 1-D scan application, which varies one user specified parameter in a prescribed manner, and monitors a separate set of user specified measurable quantities. Since this methodology is basic for many "experiments", this application is widely used. Capabilities exist for displaying, analyzing and editing measured quantities, as well as exporting data. For commonly used scan procedures and analysis, the setup is saved and can easily be repeated. Other features include averaging over a specified number of pulses during the scan and use of a validation signal (i.e. do not take data if the beam is not present). A snapshot of a 2-D scan version this application is shown in Fig. 8. The 2-D scan version allows parametric variation of a second quantity.

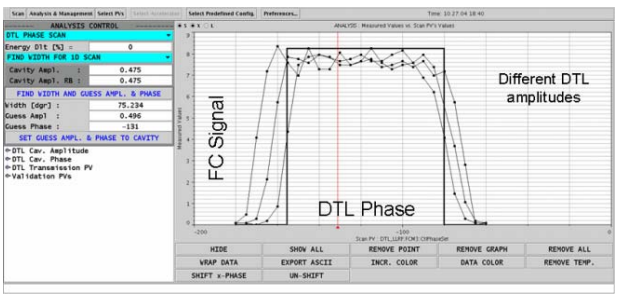

Figure 8: The scanning application screen shot, showing an example where RF phase was scanned at different amplitudes and a Faraday Cup signal was measured.

The Save Compare Restore (SCORE) application is used to diagnose problems and help restore beam to previous good states. This application saves signals and compares the saved values to live values. Also, restoration of selected sets of signals to the saved values is possible. Signals are sorted by the beamline, device type (e.g. magnets, RF, …) as well as setpoint signals vs. readback signals. All data storage is done in a database. Different operational groups can create their own save set groups. Figure 9 shows a screen snapshot from SCORE. When the live values are more than a prescribed fraction different from the saved values, they are displayed in red, facilitating identification of changed quantities.

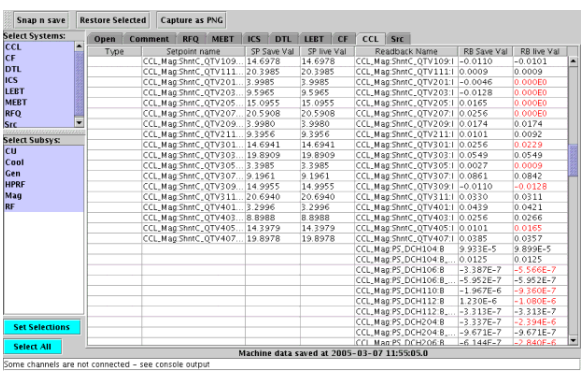

Figure 9: A save-compare-restore (SCORE) application screenshot.

SUMMARY

The XAL application development framework has being used extensively in the SNS warm linac beam commissioning activities. Preparations are underway to support the Superconducting linac and Ring commissioning. XAL is written in Java and uses a database for both configuration and measurement data storage. In retrospect, we are extremely happy with the decision to adopt these technologies. The SNS is a new accelerator project, with a programming layer that uses modern software technology.

ACKNOWLEDGEMENTS

Many people have contributed to the development of XAL, including A. Aleksandrov, S. Bunch, I. Campisi, S. Chevtsov, R. Dalesio, K. Danilova, A. Feshenko, D. Gurd, S. Henderson, J. Holmes, D. Jeon, R. Kennedy, W.D. Klotz, Y. Kiselev, I. Kriznar, A. Leahman, C. McChesney, N. Malitsky, D. Ottavio, N. Pattengale, M. Plesko, M. Plum, A. Pucelj, C. Sibley, E. Tanke, J. Wei, E. Williams, Y. Zhang , and A. Zupanc, and help from the SNS controls, diagnostics, magnet measurement, RF and operations groups. Everyone's contributions are greatly appreciated. Also the support of SNS Management for allowing us the luxury of developing a system from scratch is appreciated.

REFERENCES

[1] http://www.sns.gov/APGroup/appProg/xal/xal.htm
[2] http://aps.anl.gov/epics
[3] N. Malitsky and R. Talman, "The Framework of the Unified Accelerator Libraries", ICAP 1998.
[4] http://www.jlab.org/cdev/
[5] J. Holmes, "ORBIT, Beam Dynamics Calculations for High Intensity Rings", ICFA Beam Dynamics newsletter, **30**, April 2003, p. 100
[6] J. Galambos, et.al., "SNS Global Database use in Application Programming", http://accelconf.web.cern.ch/accelconf/p03/PAPERS/WPPE016.PDF
[7] http://shawnee.sns.ornl.gov/mis/EnterpriseModel/index.htm
[8] C.K. Allen, C. McChesney C.M. Chu, J. Galambos, W.-D. Klotz, T. Pelaia, A. Shishlo, "A Novel Online Simulator for Applications Requiring a Model Reference", ICALEPS'03, Gyeongju, Korea, http://www.jacow.org.
[9] C.K. Allen and N.D. Pattengale, LA-UR-02-4979 http://lib-www.lanl.gov/cgi-in/getfile?00796950.pdf
[10] http://www.jython.org/