SNS APPLICATION PROGRAMMING INFRASTRUCTURE AND PHYSICS APPLICATIONS*

P. Chu[†1], S. Cousineau[1], V. Danilov[1], J. Galambos[1], T. Pelaia[1], A. Shishlo[1], C.K. Allen[2]
[1]*SNS, Oak Ridge National Laboratory, Oak Ridge, TN, USA,* [2]*Los Alamos National Laboratory, Los Alamos, NM, USA*

ABSTRACT

A Java based hierarchal framework for application programming is developed for the Spallation Neutron Source (SNS). The framework, called XAL, is designed to provide an accelerator physics programming interface to the accelerator. Much of the underlying interface to the EPICS control system is hidden from the user. Also, since the accelerator structure is initiated from a database, introduction of new beam-line devices or signal modifications are immediately available to any generic-beam-line XAL application. An on-line model is included in this framework for quick beam tracking. Utility tools such as interfaces to other external modeling software and optimization solver are also available. A standard graphical user interface (GUI) is provided within the framework to minimize repeated coding work. Many beam-tuning applications, general purpose diagnostic tools and a physics data logger have been developed and tested during the SNS warm linac commissioning runs.

INTRODUCTION

At the SNS, the majority of the accelerator applications are written with a Java based programming framework, XAL [1, 2]. The key features of XAL include an object-oriented hierarchal representation of the accelerator, an interface to the control system, an envelope based online model, a standard GUI framework for applications, GUI applications, non-GUI service daemons and utility tools such as data plot, optimization, database wrapper and online logging service. There are over fifty XAL based applications being used for SNS beam commissioning.

Fig. 1 shows the relationship between the XAL application level and other control system components. A small subset of the global database is extracted to an XML formatted file which is for initializing beam-line related applications. An initialization "probe" file contains beam parameters is necessary for online model calculation. XAL applications communicate with the EPICS control system via Java Channel Access interface. PV (Process Variable) logging to the global database can be done periodically by an XAL service daemon or by application's request. More details of the XAL components and application examples will be discussed in the following sections.

XAL FRAMEWORK

The core part of the XAL infrastructure is a set of classes describing an accelerator hierarchy as shown in Fig. 2. An accelerator is at the top of the hierarchy; the accelerator contains sequence(s); each sequence is composed by hardware devices called *nodes*. In most cases, for each unique type of devices, there is a corresponding class for its property (attribute) and another class for its usage (implementation).

Database Configuration

The SNS global database [3] shown in Fig. 1 has schema design suitable for all the static information needed by various XAL applications. Although the global database is not part of the XAL framework, it is configured based on the XAL requirements. A database query application can generate an accelerator file whenever there is some beam-line device change in the database. At the commissioning stage, the database may not be suitable as the only static data source for the

*SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy. SNS is a collaboration of six US National Laboratories: Argonne National Laboratory (ANL), Brookhaven National Laboratory (BNL), Thomas Jefferson National Accelerator Facility (TJNAF), Los Alamos National Laboratory (LANL), Lawrence Berkeley National Laboratory (LBNL), and Oak Ridge National Laboratory (ORNL).
[†]E-mail address: chuc@ornl.gov.

application. Therefore, we use an accelerator file for application initialization. This accelerator file (shown in Fig. 1) also has similar XML structure as shown in Fig. 2. Other essential parts of the framework will be discussed in detail here.

Customized online data including physics related machine snapshot are also stored in the database via PV logger. Many applications can also take PV logger data for offline analysis.

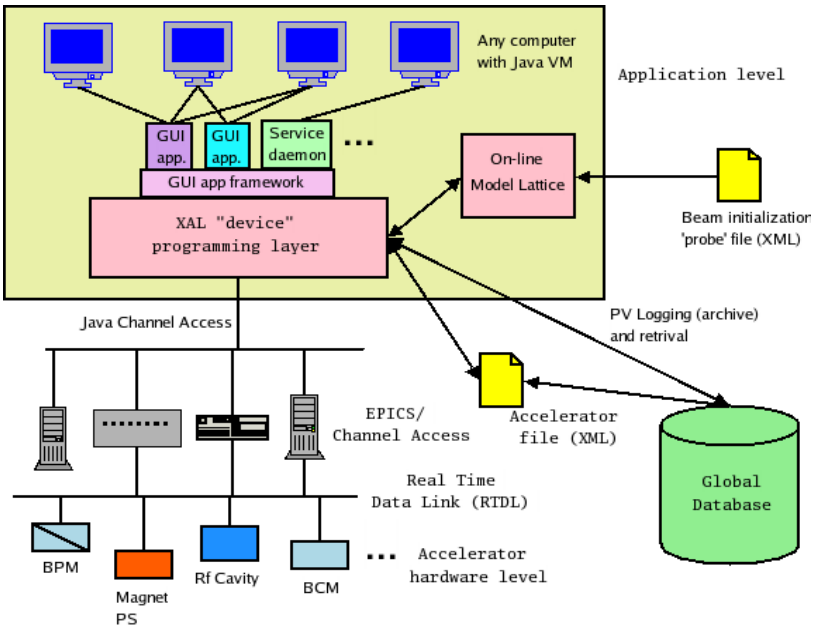

Figure 1: Application software infrastructure.

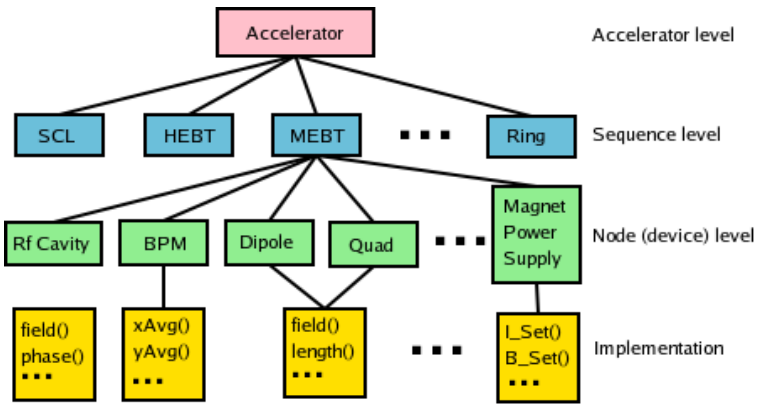

Figure 2: Schematic drawing of the core XAL accelerator class hierarchy.

Online Model

An important XAL feature is the online accelerator model [4], which allows for on-the-fly calculation of beam parameters, based on machine settings. The online model is loosely coupled with the other part of the XAL. The main components are a lattice, and a probe and its associated algorithm (describing the beam, and how it is to be modeled). The lattice is generated via a set of rules, from the accelerator node device information. In the transformation to the lattice view, devices may be split into more than one piece, and drift spaces are added (note that no drift information is stored in the XAL initialization database, only actual device information). The entries in the lattice view are called *elements*. Also, a visitor pattern scheme is used to facilitate synchronization of the lattice view parameters, with updates from different sources. The online model supports data sources from design lattice, live machine lattice, PV logger machine snapshot and user defined ‘what-if’ magnet and RF

cavity settings. It is possible to run the online model outside of XAL, as long as an online model lattice and a probe are supplied. Fig 3 shows an application based on the online model.

The online 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.

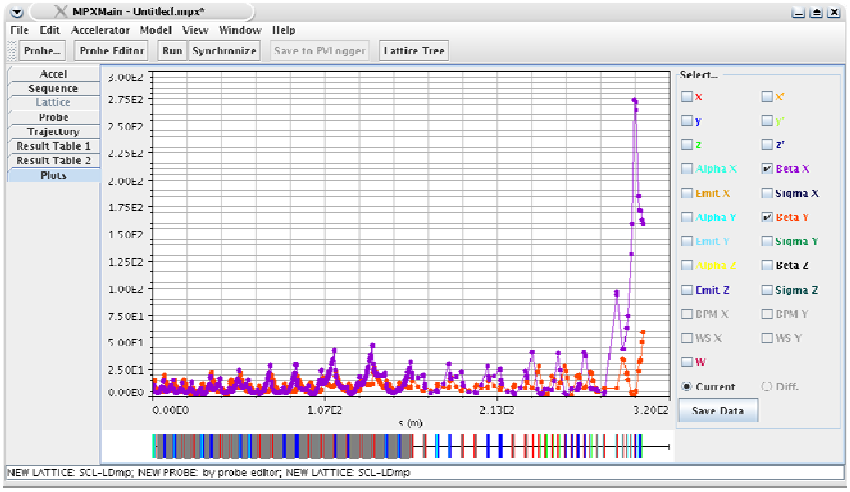

Figure 3: Online model application.

Connection to the Control System

XAL uses EPICS as the underlying control system to communicate with accelerator hardware (see Fig. 1). EPICS communication uses a single PV as the fundamental unit for communication with higher level programs via a protocol called Channel Access (CA). XAL has a Channel class that encapsulates the communication with a PV. As shown in the bottom row of Fig. 2, XAL provides a simple interface to the hardware and hides all the Channel complication from application writers.

GUI Framework for Applications

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. 4. 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.

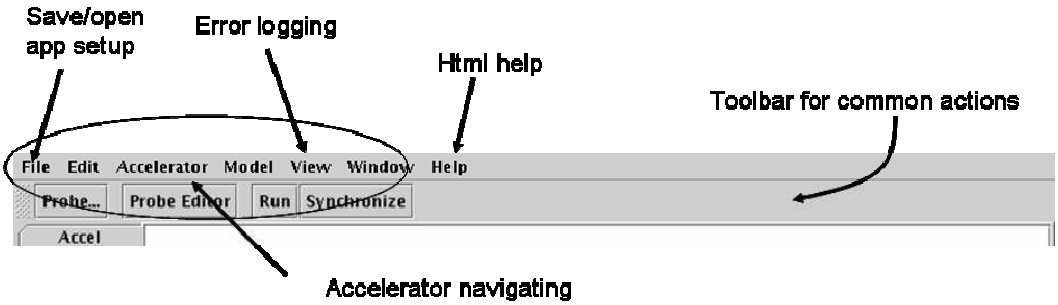

Figure 4: The XAL GUI application framework view, showing common features inheritable by all applications.

Tools

Many general purpose tools have been written, which are shared between applications, a small portion of all the available tools 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.

Additional tools include plotting, database connection, XML data parsing, external modelling tool support 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 [5] 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.

XAL APPLICATIONS AND SERVICES

General purpose applications

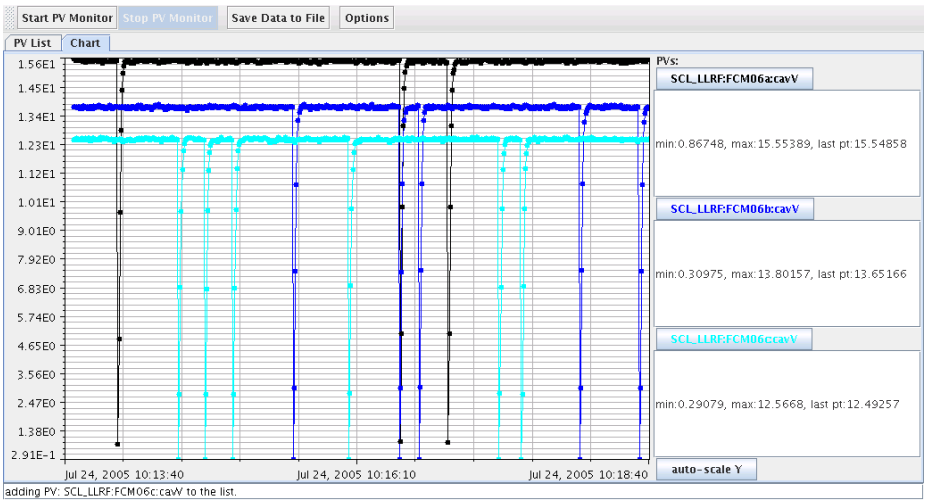

Figure 5: Timestamp test application shows with PV timestamps as horizontal coordinate.

There are many general purpose XAL applications which can be used for monitoring or diagnostic purposes. Besides the online model application shown above, some other examples are:

- Xio – monitoring and plotting any beam-line device non-array PVs.

- PV correlator – displaying two or three PVs from the same beam pulse in X-Y plot.
- Beam loss viewer – displaying beam loss monitor readings along the beam line.
- Scan application – scanning one or two PVs and monitoring some other PV(s).
- Scope – a digital scope-like application.
- PV timing test – displaying PV timestamps and live trace plot as shown in Fig. 5.
- Diagnostics device timing – display and set timing for diagnostic devices as shown in Fig. 6.
- Virtual accelerator – providing a simulation environment for application test.

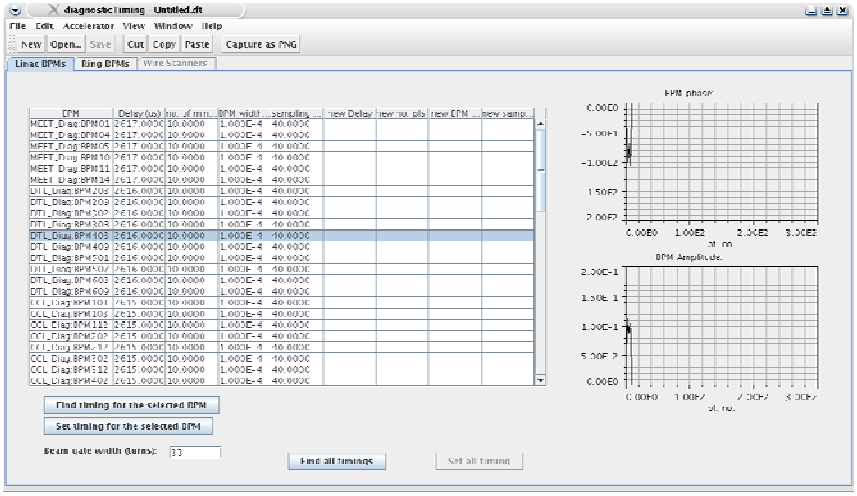

Figure 6: Diagnostics timing utility application.

Accelerator physics applications

Another group of applications is mainly for accelerator tuning or physics experiments. Highlights of physics applications are:

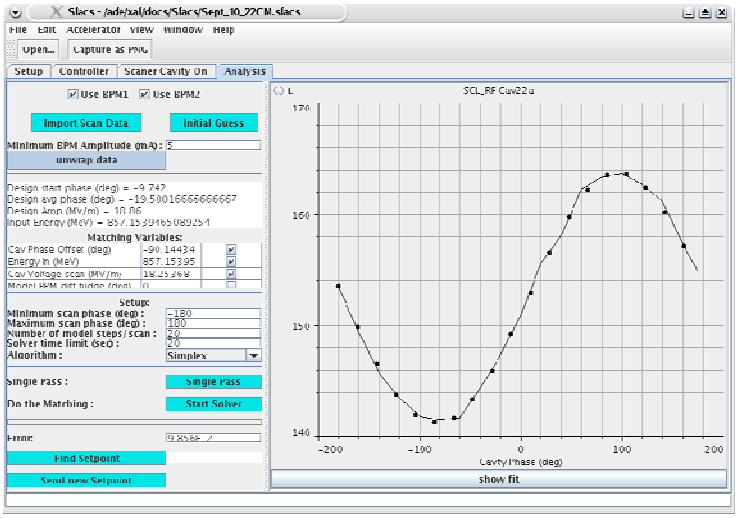

Figure 7: Superconducting RF cavity tuning application (SLACS).

- Orbit difference – checking steer and BPM polarity with the online model.
- Orbit correction – automated trajectory flattening tool.
- Energy manager – calculating a reasonable lattice when energy changed.
- PASTA – setting phase and amplitude of RF cavities.

- SLACS (Superconducting Linac Automatic Cavity Setter) – setting phase and amplitude for superconducting RF cavities as shown in Fig. 7.
- Emittance analysis – analyzing emittance scan data.
- Ring injection – providing ring injection tuning.
- Ring optics – providing ring optics optimization.
- Ring measurement – perform various ring related measurements.

Services

One of the XAL tools is a mechanism to support client/server communication. As multiple applications may access the same online resources, a dedicated service can serve as an agent for these applications and minimize the system resource requirement. 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.

CONCLUSION

The XAL structure is matured with many facilities for application programming. More than 50 applications are produced and used during the SNS commissioning. Many other accelerator laboratories are interested in adopting the XAL. The next development stages will focus on ring related applications. 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] J. Galambos, *et al*., "XAL Application Programming Structure", Particle Accelerator Conference, Knoxville, Tennessee, USA, May 2005.
[3] J. Galambos, *et al.*, "SNS Global Database Use in Application Programming", Particle Accelerator Conference, Portland, Oregon, USA, May 2003.
[4] C. Allen, *et al.*, "A Novel Online Simulator for High-Level Control Applications Requiring A Model Reference", ICALEPCS'2003, Gyeongiu, Korea, October 2003.
[5] http://www.jython.org.