APPLICATION PROGRAMMING STRUCTURE AND PHYSICS APPLICATIONS*

C.M. Chu, J. Galambos, W.-D. Klotz§, T. Pelaia, A. Shishlo, ORNL, Oak Ridge, TN, USA
C.K. Allen, C. McChesney, N. Pattengale, LANL, Los Alamos, NM, USA
D. Ottavio, BNL, Upton, NY, USA

Abstract

The Spallation Neutron Source (SNS) is using a Java based hierarchal framework for application program development. The framework is designed to provide an accelerator physics programming interface to the accelerator, called XAL. Much of the underlying interface to the EPICS control system is hidden from the user. Use of this framework allows writing of general-purpose applications that can be applied to various parts of the accelerator. Also, since the accelerator structure is initiated from a database, introduction of new beamline devices or signal modifications are immediately available for all XAL applications. Direct scripting interfaces are available for both Jython and Matlab, for rapid prototyping uses. Initial applications such as orbit difference, orbit correction and a general purpose diagnostic tool have been developed and tested with the SNS front end. The overall framework is described, and example applications are shown.

INTRODUCTION

The SNS is an accelerator for pulsed, high-intensity neutron production. For general-purpose, high-level accelerator physics applications for SNS commissioning and operation, a Java-based software infrastructure called XAL [1] is designed and implemented. The XAL is a programming framework providing an object-oriented model of an accelerator, interfaces to the SNS control systems for dynamic data and to the SNS global database [2] for static information, interfaces to various external modeling software packages, and a built-in lattice tool [3] mainly for quick, on-line calculation. The entire application software infrastructure is shown schematically in Fig. 1. A subset of the global database is extracted into portable extensible markup language (XML) formatted files which can be any part of the entire accelerator. The communication between the applications and the control hardware is through an EPICS Java Channel Access layer embedded in the XAL. Also, an optional data correlation engine in the XAL ensures the event data collected all occurred within a specified time window, usually the beam pulse width. The framework also provides standard user interface design such as common look-and-feel, Java logging and user preferences, and on-line help in HTML form. To test the framework and applications without a real accelerator running most of the time, we rely on an accelerator simulator called the virtual accelerator [4] with Trace-3D and PARMILA as model engines and portable channel access server as EPICS data provider. The advantages of this simulator are model-based simulation, and the same data acquisition interface and the same EPICS process variable (PV) settings as been used on the real machine.

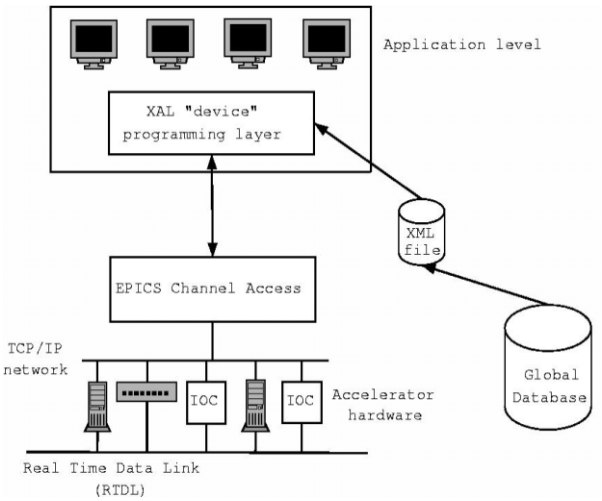

Figure 1: Application software infrastructure.

APPLICATION FRAMEWORK

In Fig. 1, the application framework is represented in the top box plus the connection to the database. A more detail diagram of the framework can be found in Fig. 2. The figure is also a simplified SNS application software design layout. The major parts of this software infrastructure can be categorized in the following subsections.

XAL toolkits

This is the core part of the framework. The framework design is based on hierarchal view of the machine, namely, a tree-like structure with accelerator sequences as branches and accelerator nodes as leaves. However, this physical view of the machine is not quite suitable for accelerator lattice which also contains drift space, split elements for various physics reasons and devices sitting on top of each other. Therefore, a conversion between the machine view and lattice view is introduced in the XAL. The XAL toolkits contain the followings:

* 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). SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy.
§ Also at ESRF, Genoble, France.

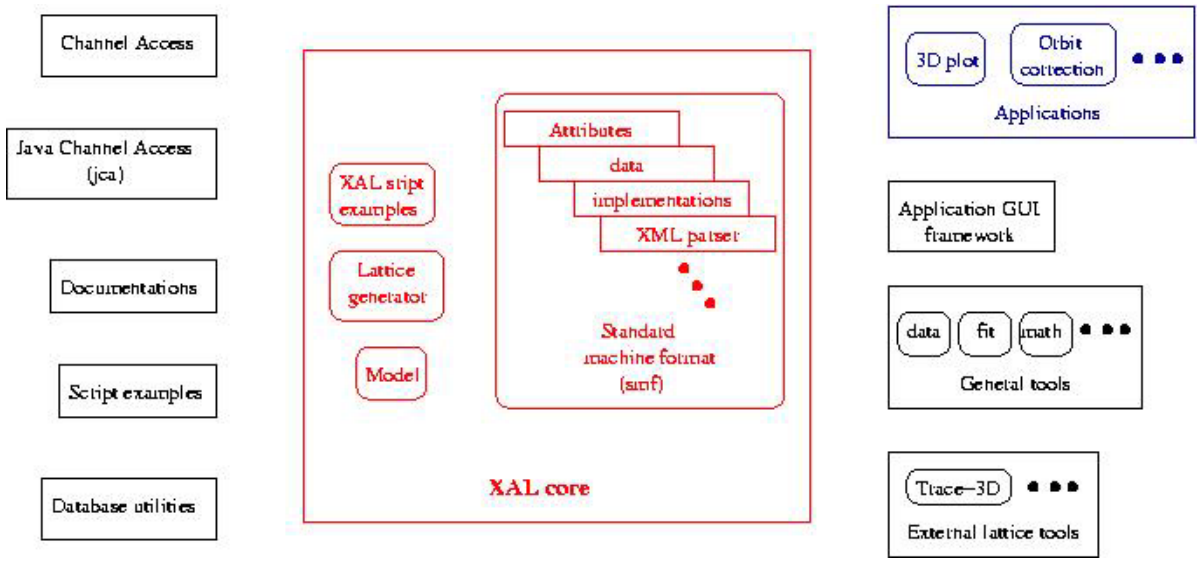

Figure 2: Application framework overview.

- On-line modeling tool providing a quick particle trajectory or envelope calculation. Currently we use external Trace-3D routine as the modeling tool. The on-line modeling tool will replace this external call and increase the flexibility and robustness.
- Standard machine format (SMF) defining an hierarchal view of the accelerator, implementations for various beam-line devices, sets of attributes for both static and dynamic data, and XML parser for file input/output of the accelerator object in XML format.
- Lattice generator providing the conversion between hardware view of the machine (a list of accelerator nodes) and physics view of the machine (lattice elements).
- XAL specific tools such as generating input files with XAL lattice generator for external modeling tools, setting or gathering multiple EPICS PVs.

Channel Access

The communication between XAL applications and accelerator hardware is through channel access. A Java wrapper layer over the native C language libraries provides extra exception handling, thread management and convenient methods for Channel Access. We plan to replace both the wrapper and the Java-to-C library by a pure Java communication protocol in the future.

General Tools

There is a collection of general-purpose tools for applications. These tools can be used independent of XAL. Here is a brief list of the tools:

- Mathematical tools such as elliptical integration.
- Graphical User Interface (GUI) components.
- General polynomial data fitting routine.
- Time correlation for gathering data events within a specified time window. This is a very important requirement for a pulsed machine.
- Standard messaging for saving and passing Java messages.

Database Utilities

As shown in Fig.1, a subset of the global database in XML format is parsed in by applications for initializing accelerator objects. The advantage of this intermediate XML file is portability and providing a hierarchal view of the database. So far we have demonstrated that the lattice generated from the database designed values via the XML file produces the same model result as the original design. Alternatively, direct access from applications to the global database is also possible. The database query and automatic XML generation, and several data validation-check routines are provided.

Scripts and Examples

Scripting language such as Jython can be a quick development or software debugging tool. Also, the Matlab scripting language provides many data analysis tools. Both Jython and Matlab can access XAL by importing their own Java interface libraries. General examples are also provided for quick tutorial purpose. Once a script is proven useful and needs to be more user-friendly, we convert the script to a real Java application.

Application GUI Framework

The idea of using a standard application GUI framework is to try to give every application share the same look-and-feel, so a general user can learn how to use an application more quickly. Also, many GUI components can be re-used in different applications, such as logging service, user-defined preferences, file input and output, etc. This GUI framework can greatly reduce the development time for user interfaces.

As shown in Fig. 3, all the XAL applications will use this GUI framework with common pull-down menus, editing buttons, console output panel, window control and on-line HTML help feature. Application writers can

easily replace or add to any pre-defined feature with customized ones.

Applications

There have been more than a dozen of applications using Java/XAL framework. Examples are orbit display, orbit correction, general-purpose EPICS process variable display, 2-dimensional and 3-dimensional correlation plots, and many other scripts. We also converted some applications originally written in Matlab to XAL-based Java applications for maintenance and improvement purposes. Several applications as examples are discussed in the next section.

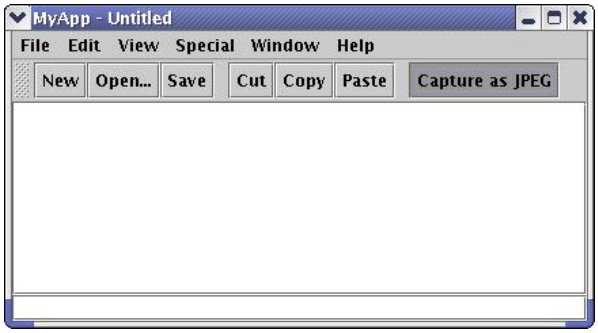

Figure 3: XAL application GUI common look.

APPLICATION EXAMPLES

General Purpose Diagnostic Display

A general purpose beam-line device display application is written for quick device diagnosis purpose. As shown in Fig. 4, the application can display same type of beam-line devices' signals along a given accelerator section as 2-D scattered plot or the same type of PVs versus time as water-fall plot. Plot updating at 5 Hz still maintains reasonable performance. The data plotting package used here and many other applications is JClass version 6.1 [5].

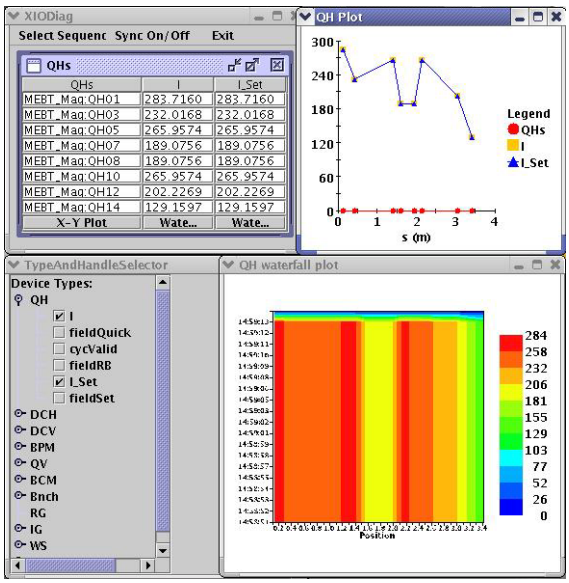

Figure 4: General purpose device display screen snapshot.

Orbit Display

The Orbit display application compares the model-predicted beam trajectory and the beam position monitor (BPM) measured average beam positions. In Fig. 5, the horizontal trajectory differences between before and after a dipole corrector strength change are plotted. The filled circles are model prediction and the squares are BPM measurement. In this particular application, the modeling tool is external Trace-3D Fortran code compiled as a shared library. The initial beam coordinates and momentums for the model are manually settable via user interface.

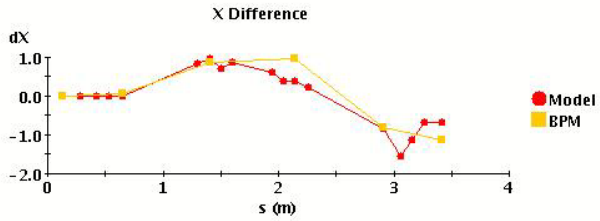

Figure 5: Partial screen snapshot of Orbit Difference Display application.

CONCLUSION

The SNS application framework is in production mode. Many applications using this framework have been tested during the SNS front end commissioning. As the XAL framework becomes more and more stable, we will speed up new application development and convert existing Matlab applications to use this framework. With a common look-and-feel for each application, the end-user will not suffer a steep learning curve. Also, the on-line lattice tool will be tested in the next commissioning period.

ACKNOWLEDGEMENTS

The authors would like to thank the SNS Controls, Diagnostics and Database groups for their great efforts of providing us various hardware and software support. All the other members in the SNS Accelerator Physics group deserve the credit for their valuable suggestions and help. We also thank Dr. N. Malitsky for the original XAL design work.

REFERENCES

[1] http://www.sns.gov/APGroup/appProg/xal/xal.htm.
[2] J. Galambos, *et al.*, "SNS Global Database Use in Application Programming", these proceedings.
[3] C. Allen, *et al.*, "A Modular, Online Simulator for Model Reference Control of Charged Particle Beams", these proceedings.
[4] A. Shishlo, *et al.*, "The EPICS Based Virtual Accelerator – Concept and Implementation", these proceedings.
[5] http://java.quest.com/jclass/jclass.shtml.