OPEN XAL STATUS REPORT 2013*

T. Pelaia II, Oak Ridge National Lab, Oak Ridge, TN 37831, USA

Abstract

XAL is the well established, accelerator physics high level application programming framework developed for and used at the Spallation Neutron Source in Oak Ridge National Lab. Due to interest from other accelerator labs, the Open XAL project was formed in 2010 to port XAL to be more suitable for collaboration. The Open XAL architecture along with the objectives, status and roadmap of this effort are presented in this paper.

INTRODUCTION

The Open XAL [1] international collaboration has formed out of the XAL [2] accelerator physics software effort at the Spallation Neutron Source (SNS) in Oak Ridge National Lab (ORNL). This collaboration currently includes SNS, Chinese Spallation Neutron Source (CSNS), European Spallation Source (ESS), Grand Accélérateur National d'Ions Lourds (GANIL), TRIUMF and Facility for Rare Isotope Beams (FRIB). The goal of this project is to create a common software platform for accelerator physics modeling, control and analysis. Since XAL is well established at SNS for accelerator physics software, it has attracted interest from other accelerator facilities.

Development of XAL began at SNS around the year 2000 before commissioning the accelerator. Development began at an intense pace, and now the source code is relatively stable. Currently there are over five dozen applications, four services and numerous scripts based on XAL. Interest from collaborators at other facilities has created an opportunity to make significant code improvements while introducing flexibility to support other accelerators.

The initial major milestones of the Open XAL project are to create a simpler and more powerful project architecture, implement an improved online model [3] (not discussed here), fix all compiler warnings with the strictest checks, develop an improved services architecture, support the latest third party libraries and port XAL code to Open XAL. To date, the only remaining action item from the initial major milestones is to complete the porting of XAL code to Open XAL.

COLLABORATION

Having been developed for SNS, XAL is very SNS centric. Others began to see the value in XAL, and it was adopted at other labs such as SLAC, CSNS and GANIL. However, these were mostly based on an old 2007 branch from SNS, and the source code rapidly diverged among the various sites. In late 2009, ESS renewed interest in a formal XAL collaboration with the goal of maintaining a common accelerator software platform in the spirit of the successful Experimental Physics and Industrial Control System (EPICS) [4] collaboration.

The first Open XAL workshop [5] was held in May 2010 at SNS. A task list was generated and development started. The old project named "xaldev" on Source Forge was reconfigured for Open XAL. Since then, the new project architecture was implemented, the new online model was developed, the XAL core was migrated to Open XAL with all compiler warnings addressed and the new services architecture was implemented. Some applications and services were ported.

The second Open XAL workshop [6] was held in December 2012 at FRIB. This workshop formalized commitments for Open XAL from the participating labs. Action items were identified, assigned to tickets and grouped into milestones. The project architecture was reviewed and tweaked based on discussions. Monthly online meetings are held to review the project status and discuss issues.

PROJECT ARCHITECTURE

Open XAL introduces a cleaner, more powerful project architecture. Like XAL, it is based on Apache Ant [7], but the new directory layout allows for zero configuration build rules and easier project navigation.

The new architecture has a hierarchical build system (see Fig. 1) [8]. Build configuration files reference others along a chain up to the top of the tree. Every configuration setting has a default, and several support customization. Build files reference the build configuration files. Also, batch build targets (e.g. building all applications) point down the tree.

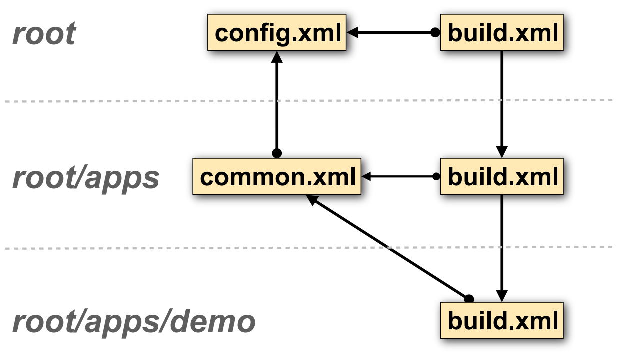

Figure 1: Build hierarchy flow.

Clean source code separation has been adopted which allows for easier project navigation, maintenance free build rules, leaner deployment and consistent organization. Directories for applications, services, scripts and the core are each rooted at the top of the project.

The core contains packages shared in common among applications, services and scripts. The core directory

* This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

contains a build file, lib directory for external jars, resources directory, source code directory and a test directory. The test directory contains its own build file, lib directory, resources directory and source code directory and contains all unit test source code and supporting files for testing the core. The test case builds are archived separately from the core so the core is free of test case code.

The applications directory contains a common application build file, a batch build file and a subdirectory for each application. An application's directory contains the application's build file, resources directory, source code directory and an optional lib directory for external jars against which the application should be built. The build file specifies the application ID and an import statement pointing to the common application build file. Optionally, it may also specify flags indicating whether the application should be excluded from batch builds (e.g. an application under development) or from installation (e.g. an application not meant for deployment such as the Virtual Accelerator). Applications are built against the Open XAL application framework within the core and their package is rooted at xal.app.appid where "appid" is replaced with the application's ID.

Services are organized the same way as for applications, but get deployed to the services directory instead of the apps directory. Services are built against the services framework within the core. Additionally, each service defines an interface and other supporting source code within the core so that applications and other core utilities have access to services without having to build directly against the services.

The scripts directory includes a single batch build file and a directory for each script. Since scripts don't require any processing, the build file simply allows for deployment options.

SERVICES ARCHITECTURE

Open XAL introduces a new services framework. Like its predecessor, it maintains a common services API that abstracts the internal details of the registration, discovery and messaging system. Registration and discovery are based on the latest version of JmDNS (version 3.4.1) [9] which involved a significant rewrite since its interface had changed. The Apache XML-RPC [10] messaging protocol of the predecessor was replaced by a homegrown JSON-RPC [11] messaging implementation which is significantly more powerful. Additional utilities make it easier to implement robust client-server synchronization.

The JSON-RPC implementation supports a large number of default types plus is easily extensible to support additional custom types. Unlike XML-RPC, JSON-RPC allows for methods that have no return type and our implementation also supports oneway calls flagged by a Java annotation. Oneway calls allow for methods that are dispatched without waiting for or expecting a return such as when shutting down a remote service.

PORTING CODE

Porting code from XAL to Open XAL [12] is relatively straight forward with most time spent on fixing compiler warnings. Since XAL development began, several versions of Java have been released introducing significant changes to the language, some of which cause compiler warnings when compiling the older code with the strictest warnings enabled. The most common compiler warnings are for use of raw types, unchecked conversion, cast and lack of serialization ID for serializable classes. Furthermore, Open XAL introduced new package names, classes, programming interfaces and eliminated obsolete features.

Table 1 shows package mappings that are commonly encountered.

Table 1: XAL to Open XAL Package Mappings

XAL	Open XAL
gov.sns.xal	xal
gov.sns	xal
gov.sns.apps	xal.app
gov.sns.xal.model.scenario	xal.sim.scenario

Table 2 shows class mappings that are commonly encountered.

Table 2: XAL to Open XAL Class Mappings

XAL	Open XAL
gov.sns.tools.apputils.iconlib .IconLib	xal.tools.IconLib
gov.sns.ca.BatchGetRequest	xal.ca.BatchGetValueRequest

Programming interface changes of note are in the Data Adaptor and Model Scenario. In the Data Adaptor, the *childAdaptorIterator* methods have been replaced by corresponding *childAdaptors* methods which are easier to use. The Model Scenario introduces an *AlgorithmFactory* class to create algorithm instances of various types. This factory mechanism replaces the XAL mechanism of directly instantiating algorithms through their associated constructors.

The application framework uses a menu definition mechanism for specifying an application's menu bar. The original delimiter was an underscore, but XAL later introduced an alternate dot notation. Open XAL drops support for the older underscore notation and accepts only the newer dot notation.

ROADMAP

Open XAL development began in 2010 following the first Open XAL workshop. The initial work focused on redesigning the project architecture based on lessons learned and then forming a practical development plan to allow migration from XAL to Open XAL. Several core packages were identified for significant modification and those changes are now complete. Remaining tasks are primarily for porting common applications and services. Table 3 shows the roadmap of Open XAL activities.

Table 3: Roadmap

Due Date	Progress	Task
Oct 31, 2010	100%	Project Creation and Architecture
Dec 31, 2010	100%	Website Development
Feb 15, 2011	100%	Application Framework Migration
Apr 30, 2011	100%	Design and Implement New Online Model
Sep 30, 2011	100%	Fix Compiler Lint Warnings
Feb 28, 2012	100%	JSON Framework Development
Feb 28, 2012	100%	Common Package Migration
Dec 31, 2012	100%	Services Migration
Jun 30, 2013	50%	Milestone 1 Tickets
Dec 31, 2013	5%	Milestone 2 Tickets

After the December 2012 workshop, we embraced a tracking system with tickets organized by milestones. Milestone 1 tickets involve migration to Java 7, online model additions and scan bug fixes. Milestone 2 tickets include porting of common applications from XAL and adding localization support.

SUMMARY

Open XAL has gained traction at several labs as a standard for accelerator physics applications. It has a new architecture and was designed for collaboration. Work began in 2010 following the first workshop and significant progress has been made. Porting of XAL applications to Open XAL is a major task to be completed.

XAL is actively used at SNS for production, and we as well as other labs are committed to migrating to Open XAL when it becomes production ready. We believe this collaboration will benefit all parties.

ACKNOWLEDGEMENT

The Open XAL collaboration is thankful for the contributions to these projects from many XAL and Open XAL developers representing several labs over several years.

REFERENCES

[1] Open XAL Project; http://xaldev.sourceforge.net/

[2] J. Galambos et al., "XAL Application Programming Structure," Proceedings of PAC 2005, Knoxville, TN 2005

[3] C. Allen, "Physics Improvements in SNS XAL," Open XAL Workshop, December 2012; http://xaldev.sourceforge.net/meetings/2012/12_XalPhysicsImprovementsVer3.pdf

[4] EPICS; http://www.aps.anl.gov/epics/

[5] First Open XAL Workshop, May 2010; http://xaldev.sourceforge.net/meetings/2010/index.html

[6] Second Open XAL Workshop, December 2012; http://xaldev.sourceforge.net/meetings/2012/index.html

[7] Apache Ant; http://ant.apache.org

[8] T. Pelaia, "Open XAL Status Report - 2012," Open XAL Workshop, December 2012; http://xaldev.sourceforge.net/meetings/2012/10_Open%20XAL%20Status.pdf

[9] JmDNS Project; http://jmdns.sourceforge.net

[10] Apache XML-RPC; http://ws.apache.org/xmlrpc/

[11] JSON-RPC; http://json-rpc.org

[12] T. Pelaia, "Migration from XAL to Open XAL," Open XAL Workshop, December 2012; http://xaldev.sourceforge.net/meetings/2012/17_Migration%20to%20Open%20XAL.pdf