ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPWI027

OPEN XAL CONTROL ROOM EXPERIENCE*

P. Chu#, D. Maxwell, Y. Zhang, MSU, East Lansing, MI 48824, USA
C.K. Allen, T. Pelaia, A.P. Shishlo, ORNL, Oak Ridge, TN 37831, USA

Abstract

This paper reports the control room experience, lessons learned, and quick deployment approach for the Open XAL application environment. Open XAL is a java-based framework for building high-level accelerator applications, it is a major revision of the XAL framework which was developed at the Spallation Neutron Source (SNS). Open XAL is site neutral and may be deployed at multiple accelerator facilities. Currently, Open XAL is installed at SNS and at the Re-Accelerator facility of Michigan State University. At SNS we are in the final process of replacing the old XAL environment with Open XAL; we describe the upgrade process and our accelerator operations experience using Open XAL. At Michigan State University (MSU), Open XAL has been tested during a cryomodule commissioning and results will be shown.

INTRODUCTION

The XAL [1] based applications have been successfully applied to SNS commissioning and early operation. Open XAL [2], on the other hand, is the updated version of XAL with international collaboration effort. Besides any common software practice, it is particularly important for software usability and customer satisfaction, especially for a new version of an already-successful software. Control room tests for Open XAL also can uncover issues which cannot be found with any offline tests. Unique challenges such as co-existence of operation for XAL and the new Open XAL applications are described below. New applications based on the Open XAL library are also demonstrated.

SNS CONTROL ROOM EXPERIENCE

Given that SNS is fully operational, it provides both challenges and opportunities for migrating from the mature XAL platform to the new Open XAL platform. The primary opportunity is to verify the code in a real operational environment while the primary risk is failure of the software affecting machine performance for production. To mitigate this risk as well as to facilitate software verification, we have chosen to deploy both XAL and Open XAL side by side which itself presents a challenge.

Following the summer of 2014 maintenance period, we have positioned Open XAL as the default accelerator physics toolset, and the old XAL code has been frozen and kept as a fallback and a benchmark source. All new applications and scripts are written against Open XAL.

*Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661 and DE-AC05-00OR22725
#chu@nscl.msu.edu

XAL and Open XAL Coexistence

Running Open XAL alongside of XAL presents several challenges due to requirements and incompatibilities. XAL was built to run against Java 6, while Open XAL was built against Java 7. Due to a performance issue discovered in Java 7, it is not feasible to run XAL under Java 7. To support Java 6 for XAL and Java 7 for Open XAL, separate launch environments for XAL and Open XAL were configured using BASH scripts.

Services presented a challenge since XAL uses XML-RPC and Open XAL uses JSON-RPC over WebSockets. Services and their associated clients must both be running on the same software platform (either XAL or Open XAL) to communicate with each other. A service such as the PV Logger which both logs continuously and on demand from a client creates a problem since both XAL and Open XAL clients must continue to work and be able to request the PV Logger to log on demand. To address this issue, the PV Logger service was modified to allow an optional on-demand-only mode. The Open XAL variant is configured to log both on demand and continuously (normal mode) while the XAL variant logs just on demand to support XAL client requests without generating duplicate logs.

Both XAL and Open XAL can read and write the same documents, but the accelerator optics input is different due to a new format for the online model configuration in Open XAL and ongoing device changes in Open XAL. Since it is common for documents to reference the associated accelerator optics, we needed to address this problem of a user opening a document referencing an XAL or Open XAL accelerator optics in an application running under the wrong platform. To address this issue, the Open XAL optics file includes a new version attribute which is set to 2.0 and the absence of this version or version 1.0 implies XAL. Both XAL and Open XAL code was modified to verify the accelerator optics version, alert the user and load the compatible optics file.

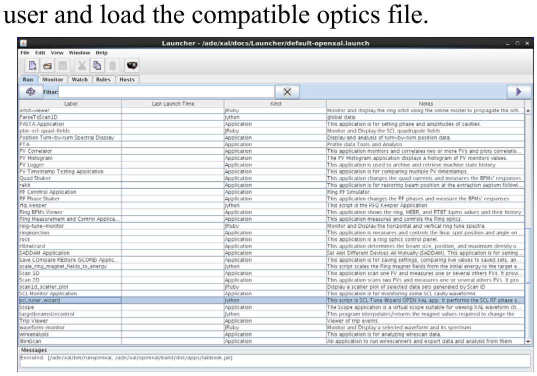

Figure 1: Open XAL Application Launcher.

With these changes XAL and Open XAL can coexist without issue and both versions can open documents

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPWI027

created by the other allowing for a seamless transition for users.

Remote X11 Issue

Our normal XAL deployment model is to configure the Launcher to run applications round robin on one of five dedicated physics servers via ssh (secure shell) using a physics account common across these servers. Unfortunately, Java 7 and 8 both have serious performance issues with the display of Swing menus for remote X11 applications. A menu may take many seconds to render after a mouse click which is unacceptable. Ultimately, our best solution (found and implemented by our Controls group) to this problem was to upgrade the console hardware to handle the load of running the applications locally instead of on the physics servers. To best manage the runtime environment, applications still run under the physics account, but this is done using UNIX user substitution instead of ssh.

Software Verification

To gain user acceptance, the software must be verified to offer a user experience as good as or better than XAL and give the correct answer. While we tested code offline, many issues can only be identified when running on the live machine in the hands of expert users. Most of the issues encountered involved either new code in Open XAL or imperfect code substitution for obsolete code that was not ported to Open XAL. For applications most critical to machine production, we spent machine studies verifying Open XAL applications against their XAL counterparts. This effort is ongoing. Figs. 2 and 3 show some Open XAL applications running in the SNS Control Room.

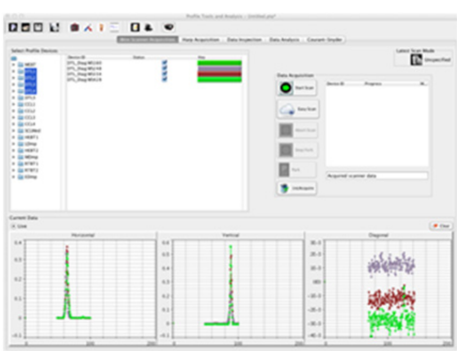

Figure 2: Profile Tools and Analysis Application.

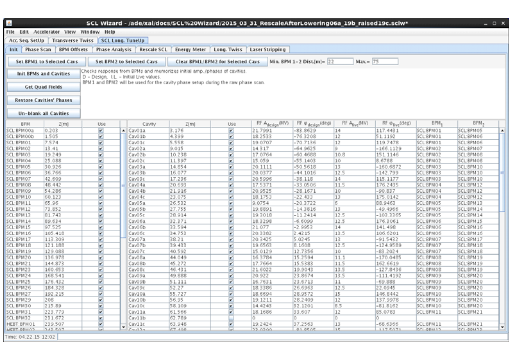

Figure 3: SCL (Superconducting Linac) Wizard.

New Application

A new application was developed exclusively in Open XAL for quickly tuning the cold linac. It was developed in the Jython scripting language and takes advantage of the new support for script based applications in the Open XAL version of the application framework.

Summary

XAL code has been frozen and all new development is happening with Open XAL. Open XAL applications are now the default SNS Control Room applications. Software verification work is ongoing.

MSU REA CONTROL ROOM EXPERIENCE

Open XAL based applications have been deployed and tested during a Re-accelerator (ReA) cryomodule commissioning at the National Superconducting Cyclotron Laboratory (NSCL). Prior to the control room test for the applications, the software has to be built and deployed. Additionally, data needed for running the applications are prepared. The entire process completes a demonstration for setting up Open XAL with a small machine. Details for the ReA Open XAL tests are described below.

Data Preparation

Because there is only one cryomodule for the test, it is not necessary to prepare a full set of lattice data with all detail information. The only data files needed are the Open XAL optics file and the online model initial condition file. With only a total of 17 devices, it is trivial to manually compile the files. Note that the model initial condition is solely for satisfying the Open XAL application initialization purpose and for running the model-based virtual accelerator.

Besides the application initialization data files, control system signal logging service also needs to be populated with proper signal names or EPICS Process Variable (PV) names.

Software Deployment Preparation

There has not been a well-established, rigorous software deployment procedure developed for NSCL; therefore, a compromised quick solution was selected. The Open XAL base library was built into a Java JAR file while individual Java applications were also built into their own JAR files. Launch scripts for each application then contains Java class paths to the proper JAR files. Similar to SNS approach, applications were running on a physics server with Windows Remote Desktop utility for local console display.

Virtual Accelerator

Before the software could be tested in the control room, it is preferable to test as much against a machine simulator, or Virtual Accelerator (VA), as possible. Open XAL provides a VA application which allows applications

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPWI027

to test for at least control system connectivity and certain physics tuning procedures.

PVLogger and Save/Restore Service

A combined functions of general purpose PV logging service, and machine settings save and restore service along with Open XAL Online Model replay capability was developed at MSU. The service can take machine snapshots periodically or on-demand. Over 5,000 snapshots were taken for a collection of 50 PVs during the cryomodule test period with no failures. Fig 4 shows a snapshot of logged PV values from the service's backend relational database.

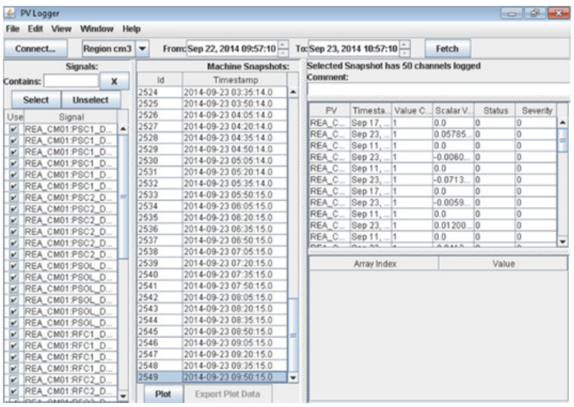

Figure 4: Screen snapshot for logged PVs.

Scan Application

A general signal scan or correlation plot application was tested with simple set-point and read-back signal correlation. Shown in Fig. 5 is an RF cavity phase set-point versus its read-back from 0 to 360°. During the phase scan tests, we found that the default phase range at NSCL is from 0 to 360° as opposed to the XAL/Open XAL's phase convention of -180° to 180°. For future operation, the phase convention might be part of the site specific configuration.

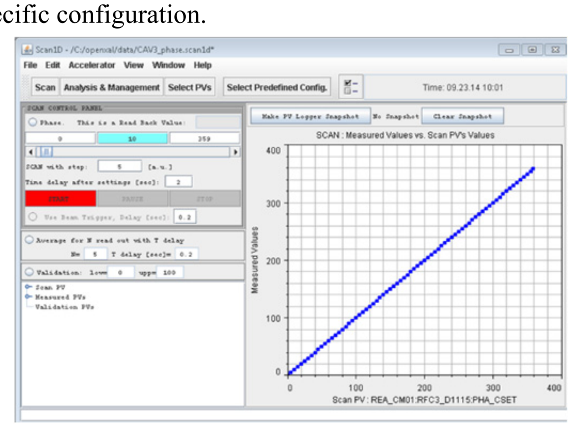

Figure 5: Screen snapshot for the Scan Application.

Degauss Application

An application, Degauss, was developed for quickly removing hysteresis in magnets by synchronously cycling the solenoid and corrector magnets. Fig.6 shows the magnetic field level quickly damped down with the application (the set-point and read-back curves are nearly identical, the two distinct curves are for a solenoid and its associated dipole corrector).

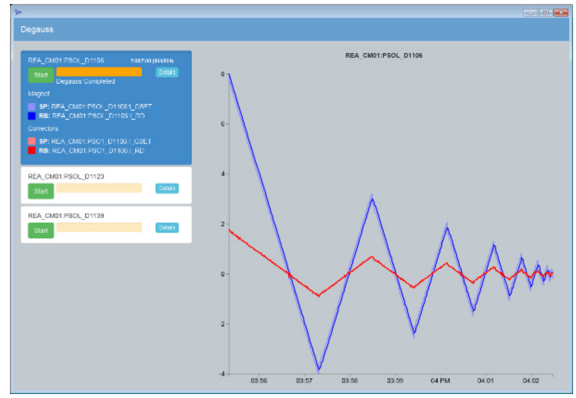

Figure 6: Screen snapshot for the Degauss Application.

Scripting Environment

Often in the control room, a quick program is needed to perform certain urgent tasks. It is much easier to code up a quick program with scripting languages than conventional programming languages. Because Open XAL is written in Java, it is trivial to use MATLAB or JYTHON as the scripting language. UNIX Shell Scripts, Windows Batch or Power Shell is also convenient for automating certain tasks such as application launching scripts with proper configurations set. Because Windows OS is the traditional NSCL control room computing platform, we prepared scripts in Windows Batch files.

CONCLUSION

The control room experience for both SNS and ReA is quite positive. Many applications and services were tested with no significant functional issues. On the other hand, other issues were uncovered and resolved. It is worth to mention that the original computing approach of running XAL applications on server computers and displaying back to operator consoles might not be a good solution due to performance and security concerns; instead, applications can run directly on newer console computers with much more computing power and the data can be saved on shared file systems. The Open XAL Control Room experience also provides us immediate future improvement plans.

ACKNOWLEDGMENT

The authors would like to thank both SNS Operation and Controls Groups, and NSCL ReA and SRF Teams for their full support on all the needs during the preparation and control room tests.

REFERENCES

[1] J. Galambos et al., "XAL Application Programming Structure", Proceedings of PAC 2005, Knoxville, TN 2005
[2] Open XAL project web site: http://xaldev.sourceforge.net