

HIGH-LEVEL APPLICATION FRAMEWORK FOR LCLS*

P. Chu[#], S. Chevtsov, D. Fairley, C. Larrieu, J. Rock, D. Rogind, G. White, M. Zelazny, Stanford Linear Accelerator Center, Menlo Park, CA 94025, USA

Abstract

A framework for high level accelerator application software is being developed for the Linac Coherent Light Source (LCLS). The framework is based on plug-in technology developed by an open source project, Eclipse. Many existing functionalities provided by Eclipse are available to high-level applications written within this framework. The framework also contains static data storage configuration and dynamic data connectivity. Because the framework is Eclipse-based, it is highly compatible with any other Eclipse plug-ins. The entire infrastructure of the software framework will be presented. Planned applications and plug-ins based on the framework are also presented.

each other easily. Also, many components such as *Help*, *Search*, *Cut*, *Copy* and *Paste* are seamlessly integrated through the Eclipse framework.

INTRODUCTION

LCLS is a hard x-ray laser light source which will use the last kilometer of the existing SLAC linear accelerator. A software suite, *SEAL* (SLAC Eclipse Accelerator Lab), is under development for the LCLS accelerator physics modeling and integration operation. The technology used for the framework is based on a Java open source project, Eclipse [1]. Eclipse is known as a Java IDE (Integrated Development Environment) but can also be a platform for hosting other functions. Eclipse is built with a plug-in based architecture which can easily integrate any compatible tools or widgets within the platform. The Eclipse component plug-in model provides the robustness and portability for applications built on “*Rich Client Platform*” (RCP). Also, there are many Java based accelerator physics packages available, such as XAL [2][3][4], which can be easily converted to an Eclipse plug-in. In addition to the SEAL suite, available XAL applications and Matlab scripts are also included in the LCLS high-level application framework.

FRAMEWORK OVERVIEW

An overview for the SEAL suite structure is shown in Fig.1. The three dashed line blocks within an Eclipse “product” represent the major component categories for the SEAL suite. Within the framework, physics modeling capability is provided by XAL online model, general-purpose EPICS tools are provided by the Control System Studio (CSS) [5][6][7], and any useful utility tools can be included from third party providers. Note that there may be overlaps among the three categories. The software framework is a desktop suite type of application which packages many plug-ins within an Eclipse product. One of the many advantages of a desktop suite over individual window applications is that the components within the desktop applications reside within the same Java virtual machine (JVM) and, therefore, can communicate with

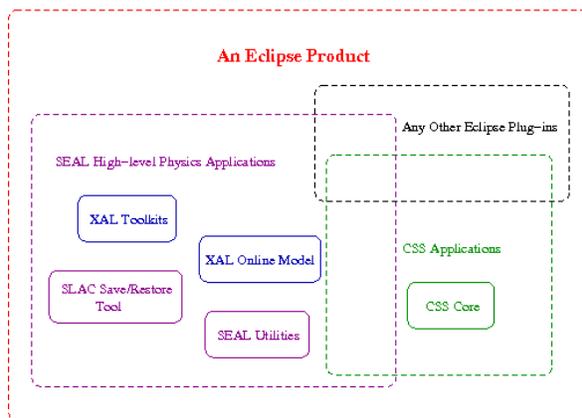


Figure 1: Schematic view of SEAL suite.

FRAMEWORK COMPONENTS

The SEAL suite can basically integrate any Eclipse plug-ins as well as any executables. Details for various categories of the SEAL suite components are explained in the following subsections. There will be minimal tweaks for certain components to better fit within the suite.

XAL Toolkit

Accelerator information such as default accelerator optics and signal names is parsed into SEAL from an XAL XML file. The XML format follows the XAL *Standard Machine Format* (SMF) which is a hierarchical view of an accelerator.

Because the graphical user interface (GUI) for Eclipse is built with SWT (*Standard Widget Toolkit*), the XAL Swing-based GUI framework is not compatible within the SEAL framework. In order to use XAL non-GUI tools, a custom built jar file without the GUI frame is used for creating an Eclipse plug-in. Each plug-in that needs to access any XAL class can then include this XAL plug-in in its dependency list.

Besides the SMF and online model, there are many other XAL tools such as data plot package, optimizer and mathematical utility are also included in the XAL plug-in.

Control System Studio

CSS is another Eclipse based application suite for control systems. The major difference between CSS and SEAL is that the focus for SEAL is mainly on physics applications while CSS is aiming toward general control system support. Nevertheless, some CSS plug-ins can be

*Work supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515.

[#]pchu@slac.stanford.edu

included in the SEAL suite and vice versa.

Plug-in Design and Organization

One of the design goals for the SEAL plug-ins is to re-use some plug-ins in multiple applications. There will be a set of base plug-ins which can simply be grouped together to form an *application*. For instance, an online model application can be composed of an accelerator optics selector plug-in, an online model run control plug-in and a special data display panel with distance along the beam line as horizontal axis, while an orbit display application can use the same data display panel plug-in and a beam position monitor (BPM) data collection engine plug-in.

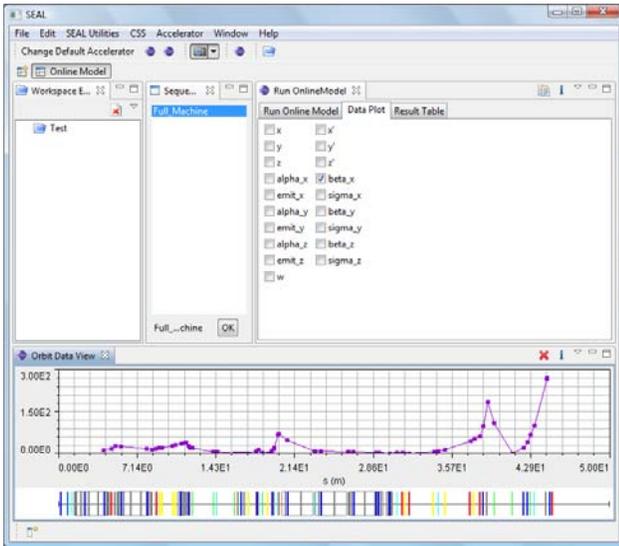


Figure 2: SEAL prototype with the *Online Model* perspective. The perspective contains a workspace navigator displaying the workspace file system (upper left), a beam line sequence selector (upper middle), an online model run control (upper right) and an orbit data plot panel for showing the model run result (bottom).

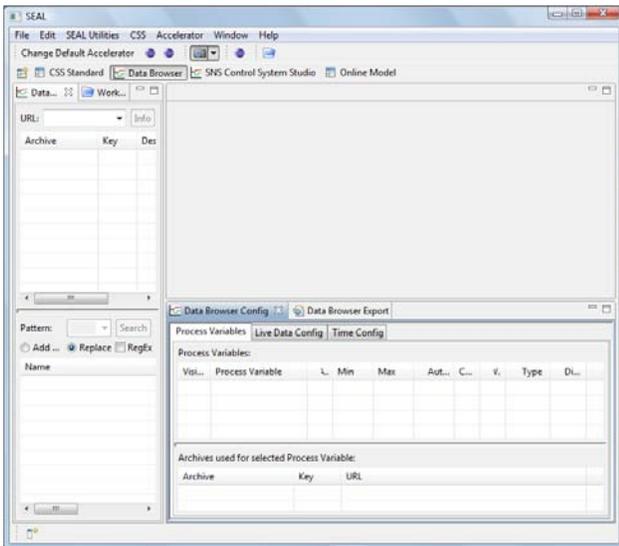


Figure 3: SEAL with the CSS *Data Browser* perspective. The perspective includes Data Browser Archives view

(left), Data Browser Config view (bottom right) and a blank editor area (upper right).

Application Displayed as Eclipse Perspective

Because the SEAL suite is a desktop application suite that contains many plug-ins, it is necessary to group relevant Eclipse *Views* (panels) in a reasonably arranged layout to act as an application. Eclipse provides such predefined layout mechanism called *perspective*. Figs. 2 and 3 show the same SEAL suite but in different perspectives, Fig.2 is of the online model perspective and Fig. 3 is of a CSS perspective for the Spallation Neutron Source (SNS). With any given perspective, user can still open or close any view at wish. A “Reset Perspective” click will bring back the predefined layout.

Data Plotting Package

There are three possibilities under consideration for the data plot within SEAL:

- XAL plotting package [8] – this is a Swing based graphical environment which needs the SWT_AWT bridge in order to work within SEAL. The drawbacks are a) it loses the native look-and-feel for Eclipse, and b) it is not 100% compatible for various operating systems. The existing orbit data plot plug-in is using this package.
- Matlab data plot utility – this gives the advantage of using Matlab mathematics tools with data plots. In particular, we plan to use the Matlab Java Builder and curve fitting toolbox for math and plotting/fitting, respectively. However, the performance and license cost may discourage this approach.
- Possible new plotting package – an SWT based scientific data plotting package would be the best solution.

SEAL Utility Tools

Present SEAL utility tools include the following utilities:

- Textual display – a general purpose data tabulating and data displaying tool.
- Java CM Log viewer – Java version of the CM Log in Eclipse plug-in form.
- Screen snapshot viewer – A screen capture plug-in which can also be a simple image editor.

Save/Restore Tool

A machine configuration save and restore utility tool is under development. The first version of the tool is based on the XAL *SCORE* application with partial support for the SLAC non-EPICS control system. Beyond the phase 1 of the project, the tool will be integrated into the SEAL suite.

Eclipse Workspace

Typically, Eclipse RCP applications preserve some preferences in a directory structure called *workspace*. The

workspace concept is similar to the standard Java preferences. For SEAL, in the *workspace* there will be information such as EPICS Channel Access address list, path or URL to the default XAL accelerator file and the default layout. The workspace for control room environment might be different from the one for office users, so SEAL includes workspace management such that when used in the control room all features are identical, but when used in offices SEAL allows persistent customization.

Software Deployment and Update

Any collection of Eclipse plug-ins can be bundled and shipped as a *product*.

There are many ways to update Eclipse based software. A popular way for software update is to use Eclipse's update utility by creating feature plug-ins with update site information. Alternatively, a standalone plug-in jar file can simply be copied to the plug-in folder in the SEAL release package.

Non-Eclipse Executables

Here the non-Eclipse executables are defined as any programs which are not written within the Eclipse RCP framework. There are many existing Matlab scripts for commissioning which can be launched within the SEAL suite for convenience. An Eclipse *workspace* is set up for caching information such as where the executables and how to run the executables. To run an executable within the SEAL is simply a mouse click. In addition, SEAL can also provide possible logging and messaging services for the executables.

Planned Plug-ins

In addition to the plug-ins already mentioned above, there is a long list of plug-ins to be developed in the near future. Highlights of the planned physics plug-ins include:

- Profile monitor data collection
- Emittance analysis
- Correlation plot (process variable scan)
- Orbit/trajectory correction
- Linac energy management

XAL ADOPTION EXPERIENCE

Because SEAL applications adopt the XAL online model, it is necessary to follow the XAL SMF for the LCLS lattice configuration. Typically, an initialization file in XML format is parsed to construct an accelerator object for applications to use. A database infrastructure is set to store necessary data for this XML file and an SQL

script queries the database to generate the XML file on demand.

LCLS specific beam line devices were added to a customized XAL SMF collection. LCLS rf cavities with end focusing effects are one example of such devices. Even for those LCLS devices already defined in the XAL SMF, there are some LCLS specific attributes which have to be included. For the short term plan, an XAL code snapshot taken from SNS repository in August 2007, is branched in the LCLS repository and all the LCLS specific modifications are saved locally.

A number of SNS XAL applications can be adopted with minimal or even no modification required. Such applications include knobs, save/compare/restore, one and two-dimensional scan and general purpose EPICS PV display applications.

For online model benchmark comparison, we compare with the LCLS nominal MAD lattice run results.

CONCLUSION

A small set of plug-ins have been created for the SEAL suite. A simple online model application is built on top of the available plug-ins. Many Eclipse framework features are being evaluated.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. K. Kasemir and the CSS collaboration for valuable Eclipse discussions and source code access. We would also like to thank the SNS XAL team for their effort to make XAL more generic. We appreciate A. Chan and her database group for their database efforts for the XAL configuration. Many thanks go to the LCLS Physics, Operation and Controls for providing support and feedback.

REFERENCES

- [1] <http://www.eclipse.org>.
- [2] <http://neutrons.ornl.gov/APGroup/appProg/xal/xal.htm>.
- [3] J. Galambos *et al.*, "XAL Application Programming Framework", ICALEPCS'03.
- [4] T. Pelaia *et al.*, "XAL Status", ICALEPCS'07.
- [5] J. Hatje, "Control System Studio (CSS)", ICALEPCS'07.
- [6] K. Kasemir, "Control System Studio Applications", ICALEPCS'07.
- [7] M. R. Clausen, "EPICS - Future Plans", ICALEPCS'07.
- [8] A. Shishlo *et al.*, "Java Swing-Based Plotting Package Residing within XAL", ICALEPCS'07.