BOY, A MODERN GRAPHICAL OPERATOR INTERFACE EDITOR AND RUNTIME*

Xihui Chen#, Kay Kasemir, ORNL, Oak Ridge, TN 37831, U.S.A

Abstract

Taking advantage of modern graphical editor software technology, a new Operator Interface (OPI) editor and runtime - Best OPI, Yet (BOY) [1] - has been developed by the Control System Studio (CSS) [2] collaboration. It uses the Eclipse Graphical Editing Framework (GEF) [3] to provide modern graphical editor functions, which makes it easy and intuitive to edit OPIs. With JavaScript and configurable rules, it allows users to create OPIs with powerful client-side logic. The graphical layer is decoupled from the data connection layer, conceptually allowing BOY to connect to arbitrary data sources. Current support includes the Channel Access protocol [4] and local process variables (PVs). BOY is integrated in the CSS platform, which provides inter-operability with other CSS tools. Fundamentally, it could also be integrated with other Eclipse Rich Client Platform (RCP) [5] applications due to its plug-in mechanism. We have several screens deployed at the Spallation Neutron Source (SNS), where BOY has proven to be stable in support of SNS operation.

INTRODUCTION

The Operator Interface is one of the three basic components of the standard control system model [6]. It provides not only operators but also scientists and engineers with rich graphical interfaces to view or operate the accelerator locally or remotely. With the increasing scale and complexity of modern accelerators and detectors, more and more requirements are imposed on a modern OPI editor and runtime:

Ease of Use: A modern accelerator control system generally has hundreds if not thousands of deployed OPI panels. Generation of these panels should not be limited to computer scientists. Operators and equipment specialists require a visualized, easy to use and productive OPI editing environment for this purpose. It should also support the automated generation of OPI panels, for example by scripts.

High Flexibility: As the complexity of modern accelerators increases, so does the OPI. In addition to reading or setting individual values, there is a need for dynamic displays with client-side logic. In the Experimental Physics and Industrial Control System (EPICS) [4], this allows moving display-related logic from the front-end controller into the OPI.

Performance: Performance is always important to control systems, referring to smooth display updates, delay-free operation combined with high reliability and stability.

Manageability: To manage a large amount of OPI panels, the tool must support a consistent use of colors, fonts and images.

Portability: Modern distributed control systems require access to OPI panels everywhere, at least on a local area network. The OPI should be able to run on different operating systems.

To meet these requirements, a modern graphical OPI editor and runtime, the Best OPI, Yet (BOY), has been developed at the SNS within the CSS collaboration. BOY is an integrated environment, comprised of a web-browser like runtime, a What You See Is What You Get (WYSIWYG) graphical editor, and several Views to support the editing of OPIs. Its design is based on the experience with two existing display tools, the Synoptic Display Studio (SDS) [7] and the Extensible Display Manager (EDM) [8], as well as adopting many new ideas from the EPICS community.

ARCHITECTURE

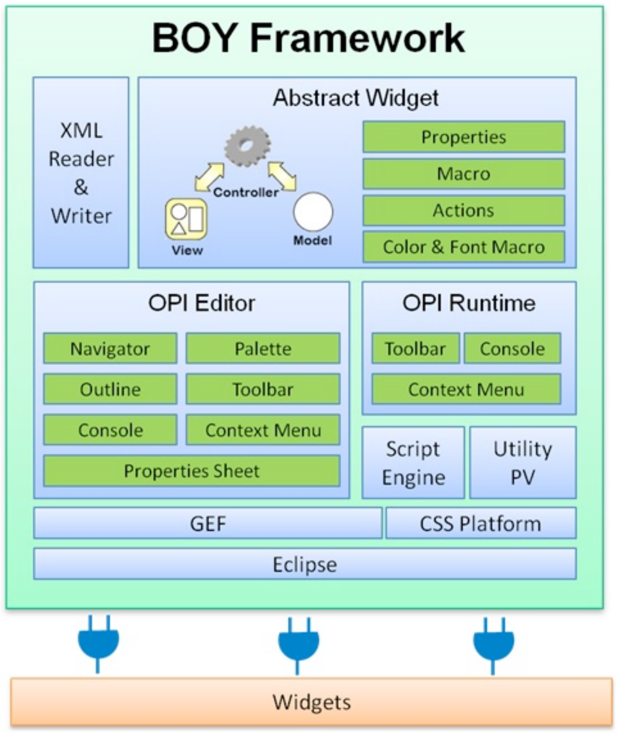


Figure 1: Architecture of BOY.

As shown in Fig. 1, BOY is a set of Eclipse plug-ins built on the CSS platform [2] using Eclipse and Java technology. With Java and Eclipse technology, BOY is

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy
#chenx1@ornl.gov

able to run on different operating systems without complicated installation. The Eclipse Graphical Editing Framework (GEF) [3] provides the basis of a modern graphical editor. BOY combines GEF with control-system specific code from SDS to implement an editor for OPIs.

In BOY, "Widgets" are the bricks for constructing an OPI. Widgets have properties like color and text. Internally, BOY widgets follow the GEF Model-View-Controller (MVC) architecture. Widgets are included in BOY via the Eclipse plug-in mechanism, allowing accelerator sites can add their customized widgets.

The graphical layer of BOY uses a data connection layer, the CSS "Utility PV" library, to connect to control system data sources. Most widgets can dynamically change their properties based on the value of a PV. A "Text Update" widget for example will display the textual value of a PV, and the border around the widget can change depending on the alarm state of the PV. The PV layer in turn uses the Eclipse plug-in mechanism to connect to the actual control system. Currently, there is support for EPICS Channel Access and local, in-memory PVs. The latter allow for data communications between BOY widgets or OPI panels.

To meet the high flexibility requirement, BOY can attach rules and scripts to widgets. Based on the Rhino JavaScript Engine [10], scripts can dynamically change any widget property.

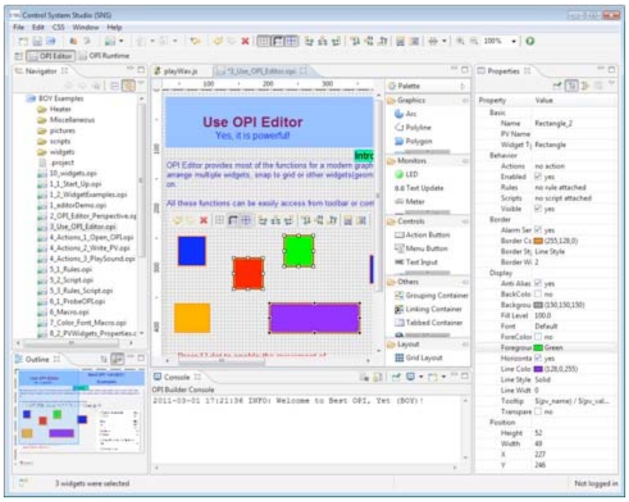
OPI files are stored in a well-defined XML format, giving the possibility to automatically generate OPIs from scripts. BOY uses JDOM [11] as XML parser.

In addition the BOY editor and runtime itself, there is growing ecosystem. For example, the Data Browser that interactively displays live and historic data has been developed outside of BOY [12], but was then wrapped into a BOY widget to offer its functionality within an OPI. Collaborators at the Advanced Photon Source (APS) developed an MEDM [13] to BOY converter. An EDM to BOY converter is under development at the SNS.

BOY can be deployed as part of any Eclipse RCP product, for example within the SNS version of CSS [14]. When integrated with CSS, it can provide inter-operability with other CSS tools.

OPI EDITING WORKBENCH

To meet the requirement of ease of use in editing and managing OPI files, BOY has an integrated OPI editing workbench as shown in Fig. 2. It is comprised of a graphical OPI editor, a JavaScript editor, a navigator view, a properties sheet, an outline view and a console view. These parts are arranged in a predefined perspective, but users can customize it by dragging, resizing or closing any of its parts.



```
2011-03-01 17:21:36 INFO: Welcome to Best OPI, Yet (BOY)!
```

Figure 2: OPI editing workbench.

A palette lists all widgets that can be placed on the central OPI editor panel. The graphical editor, its toolbar and context menus offer comprehensive editing functions such as copy, paste, zoom in/out, group, ungroup, alignment and snap to grid and so on. There is multi-level undo and redo support.

Creating and editing an OPI is fairly straightforward. For example, one can drag a "Text Update" widget from the palette onto the editor panel, and then enter the desired PV name in the Properties View, and that's it. Later, a click on a widget to select it allows changes to its other properties in the properties sheet, such as the foreground color, font and tooltip. One can change the size or position of a widget by dragging and dropping. It is possible to select multiple widgets simultaneously and then change their shared properties at once.

The Navigator View is the place to manage OPI files. Eclipse offers support for several software versioning systems, and the SNS version of CSS includes the Concurrent Versioning System (CVS) to allow storage and retrieval of OPIs from a CVS repository.

The Outline View gives a thumbnail picture as well as a widget tree structure of the currently edited OPI. The console view can display warnings or error messages.

OPI RUNTIME

To the user, the runtime environment that executes BOY OPIs is similar to a web browser (Fig. 3). OPIs can appear in tabs, panels within one window or as separate windows. Opening related OPIs is similar to following a hyperlink in web browser, and users can control if the new page will replace the current one, or appear in a new tab or window. A page history allows navigation 'back' to previously viewed pages, then to got 'forward' again to the most recent page.

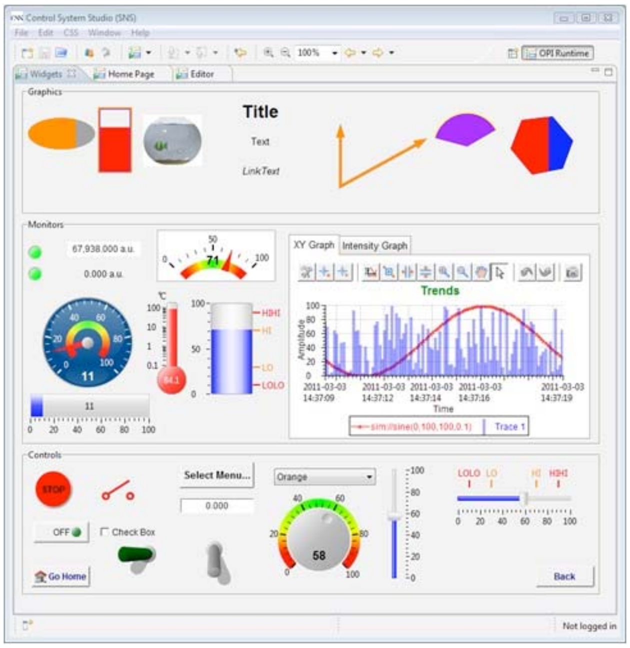


Figure 3: OPI runtime and widgets.

PV Connectivity

Based on the PV name property of a widget, the OPI Runtime connects to the control system and updates associated widget properties. The Meter widget for example will by default reflect the PV's value via its needle, display the PV's range and alarm limits on its scale and ramp, and update its border based on the PV's alarm states and connection status.

Macros

The OPI Runtime supports macros on many levels. Any text-base property like the PV name, tooltip and rules can include macros. The runtime expands macros based on parameters passed into the OPI from the command-line or parent displays. By using macros, the same panel can be reused for different PVs.

BOY also supports color and font macros, which can be used to provide a consistent look by for example defining a common "Title" font.

Rules and Scripts

Rules are configured in an interactive dialog without requiring programming knowledge, for example to change the background color of a widget based on a PV value.

Script follows JavaScript grammar, but also allows calling Java code. It offers full access to widgets, widget properties and input PVs. For example, a script can fit a waveform, write data to file, dynamically move a widget or popup a warning dialog in case a value exceeds a limit.

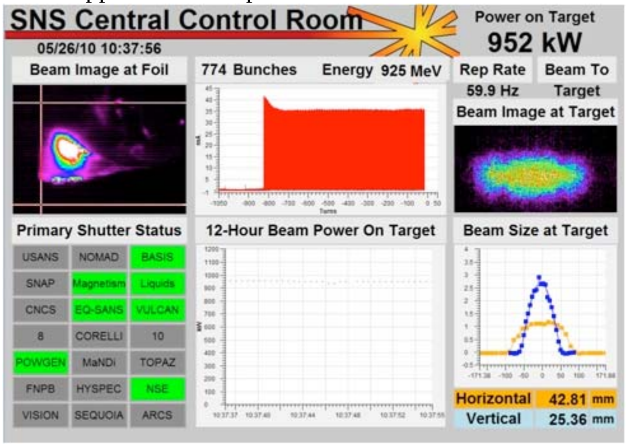
WIDGETS

BOY currently offers 38 (Fig. 3) widgets. There are basic graphics such as rectangle, ellipse, polyline, polygon and image; control system widgets such as meter, knob and XY graph; container widgets such as grouping container, linking container, tabbed container and web browser; as well as layout widgets that can help to arrange widgets in a particular way.

Eclipse/Java programmers can also add customized widgets via an extension point.

BOY AT SNS

SNS has been using several OPIs for about eight months (Fig. 4). BOY has proven to meet the performance requirements. It has been stable and reliable in its support of SNS operation.


SNS Central Control Room
05/26/10 10:37:56
Power on Target 952 kW
Beam Image at Foil
774 Bunches
Energy 925 MeV
Rep Rate 59.9 Hz
Beam To Target
Beam Image at Target
Primary Shutter Status
12-Hour Beam Power On Target
Beam Size at Target

USANS	NOMAD	BASIS
SNAP	Magnetism	Liquids
CNCS	EQ-SANS	VULCAN
8	CORELLI	10
POWGEN	MaNDi	TOPAZ
FNPB	HYSPEC	NSE
VISION	SEQUOIA	ARCS

Horizontal 42.81 mm
Vertical 25.36 mm

Figure 4: Screenshot of BOY OPI at SNS

ACKNOWLEDGEMENTS

We thank the original developer of EDM, John Sinclair, and SDS developers at DESY for their inspiration. John Hammonds developed the MEDM to BOY converter and byte monitor widget. We thank the ITER CODAC team, Ralph Lange and other BOY users for their active feedback and suggestions, Matthias Clausen for shepherding CSS, and Gabriele Carcassi, Kunal Shroff for their effort in managing CSS on SourceForge.

REFERENCES

[1] http://sourceforge.net/apps/trac/cs-studio/wiki/BOY.
[2] http://cs-studio.sourceforge.net/.
[3] http://www.eclipse.org/gef/.
[4] http://www.aps.anl.gov/epics/.
[5] http://wiki.eclipse.org/Rich_Client_Platform.
[6] M. E. Thuot, L. R. Dalesio, Control System Architecture: The Standard and Non-Standard Models. IEEE, Proc. of the 1993 PAC, Volume 3, p. 1806-1810
[7] https://sourceforge.net/apps/trac/cs-studio/wiki/Synoptic
[8] http://ics-web.sns.ornl.gov/edm/
[9] http://www.eclipse.org/gef/draw2d/index.php
[10] http://www.mozilla.org/rhino/
[11] http://www.jdom.org/
[12] K. U. Kasemir, Control System Studio (CSS) Data Browser, PCaPAC08, Ljubljana, Slovenia
[13] http://www.aps.anl.gov/epics/extensions/medm
[14] http://ics-web.sns.ornl.gov/css/products.html