CONTROL SYSTEM STUDIO APPLICATIONS*

K. Kasemir, ORNL, Oak Ridge, TN 37831, U.S.A.

Abstract

Control System Studio (CSS) is an effort to implement control-system related applications, primarily targeting the operator interface. It is based on current software technologies (Java, Eclipse), with a special emphasis on interoperability. We present initial versions of several CSS applications, their features, and how the Eclipse and CSS frameworks helped or complicated their development.

PURPOSE

Many control system (CS) user interface (UI) tools share the same problems: Implemented at different sites, their look and feel varies. Use of menus, configuration panels, or online help is inconsistent. Data exchange is limited to manual copy and paste of process variable (PV) names, often via gestures unique to the application.

They are often restricted to the X11 window system, and suffer from dwindling support, because programmers now prefer portable UI frameworks with better development environments. Finally, most UI tools are locked to a particular control system, while many sites now need multi-protocol support.

CSS is an effort to create applications with a consistent, modern look and feel in an integrated environment, and interfacing to multiple control systems [1].

ECLIPSE

Instead of starting from scratch, CSS builds on Eclipse, a modern framework for UI applications [2]. Eclipse offers graphical UI elements and also defines their behavior. Based on Java technology [3], it supports multiple operating systems. The Eclipse "Plugin" and "Extension Point" mechanisms aid in the organization and deployment of software modules.

The CSS "core" plugins define control-system data types for front-end computers (FEC), PV names, and live or archived data samples. There are APIs for accessing multiple CS network protocols, and guidelines for integrating the menu, online help, and preferences pages of control-system related applications [4].

PROBE

The CSS Probe tool displays the current value of a PV. It runs on Microsoft Windows, Mac OS X, and Linux, using their native widgets for buttons, text boxes, etc., and has several usability advantages over its namesake application from the Experimental Physics and Industrial Control System (EPICS) toolkit [5].

* Work supported by Oak Ridge National Laboratory for UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy.

General UI Behavior

The CSS Probe layout adjusts to the window size. When restarted, the previous window location is restored. The PV name field offers a drop-down list of recently entered names. Eclipse supports localization and Probe includes German, Chinese, and English translations, depending on the locale of the operating system.

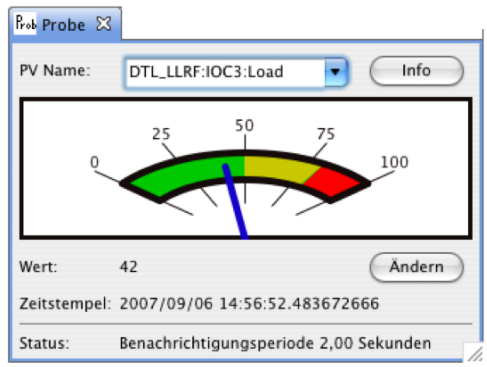

Figure 1: Snapshot of CSS Probe in German locale.

Preference Pages

CSS tools present their settings via the Eclipse preference UI. Probe itself currently needs no preferences, but it uses a communication plugin to obtain PV samples. If the EPICS plugin is loaded, Probe utilizes the EPICS network protocol, unaware of the required subnet addresses and other settings. The EPICS plugin itself adds the necessary preference pages. The user can consequently inspect and configure the network protocols that Probe uses via a convenient graphical UI.

In addition to the preference page UI, system administrators can define site-specific defaults for CSS and Eclipse settings on various levels.

Online Help

CSS plugins for tools and support libraries contribute to the Eclipse online help system. This is especially useful for libraries: when libraries are added, the user automatically sees the corresponding online help and preference pages without changes to the tools that use them.

Version Info

Previous UI applications had inconsistent support for version information. In Eclipse, everything is packaged in one or more plugins, each of which has a version number. There is a common UI, accessible from the 'Help' menu, which lists all plugins with version detail and optional legal information.

EPICS PV TREE

This application displays a hierarchical view of EPICS record input links with current values and alarm states. While limited to a specific control system, it nicely demonstrates the integration with other CSS tools.

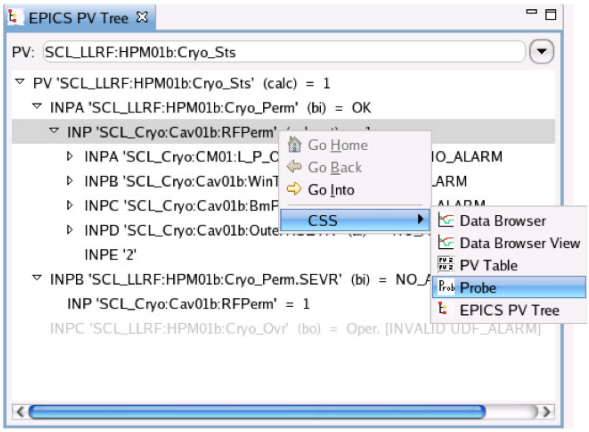

Figure 2: PV Tree with context menu.

CSS Context Menus

When opening the context menu on a PV or FEC name in a CSS application, it includes a list of other CSS tools that handle this type of item. In the example of Fig. 2, clicking the right mouse button on a PV in the tree opened a context menu with tree navigation options as well as a "CSS" submenu that lists other CSS utilities for PVs. When selecting "Probe", a new Probe instance starts to display the current value of the selected PV.

This offers a previously hard to achieve integration between control system applications. Users no longer need to be aware of all the other PV-related tools, know the specifics of how to start them and how to transfer a PV name into them, because context menus automate this.

Moreover, users can select multiple PVs at the same time, and use the context menu to send them to a new instance of the Data Browser described later on. Probe, on the other hand, would appear disabled in the context menu because it was not designed to handle more than one PV.

As a technical detail, this is implemented via Eclipse Object Contributions. When a new PV-related tool is added to CSS, there is no need to recompile or reconfigure existing tools. The new tool automatically appears in the respective context menus. Eclipse provides this powerful Object Contribution mechanism, and CSS defines data types to attach context menus to PVs, FECs, archive data sources and sample sequences.

NAMESPACE BROWSER, ALARM TREE, LOG VIEWER

The Namespace Browser interactively locates PV names on a database server. The Alarm Tree and Log Viewer are part of a system for handling control system alarm messages, which is somewhat specific to the setup at DESY (Hamburg, Germany). Space limitations prevent a more detailed description in this context.

The important point, however, is that one can develop CSS plugins which are specific to a certain site or CS, and still benefit from full integration with other CSS tools. After obtaining a PV name from the Namespace browser, the context menu leads to other CSS tools. At an EPICS site, this includes the PV Tree. At other sites, it could include a program to display PV configuration detail from a local relational database.

Site-specific tools no longer need to be standalone, oddball applications without preferences or online help. If your application handles PVs, it can easily send those to other CSS tools. More importantly, other CSS tools can now send PVs to your tool.

DATA BROWSER

The Data Browser is a generic CSS tool that combines Strip Tool and Archive Viewer functionality. It can display live samples as well as archived data in a plot, or export the data to files. Based on plugins for archive data sources, it can currently interface to the Channel Archiver, the DESY AAPI server, and the EPICS Archive Record. Each PV may have multiple data sources: for example, the Data Browser can merge samples from an archive record for recent history with those from a mid-term and long-term archive.

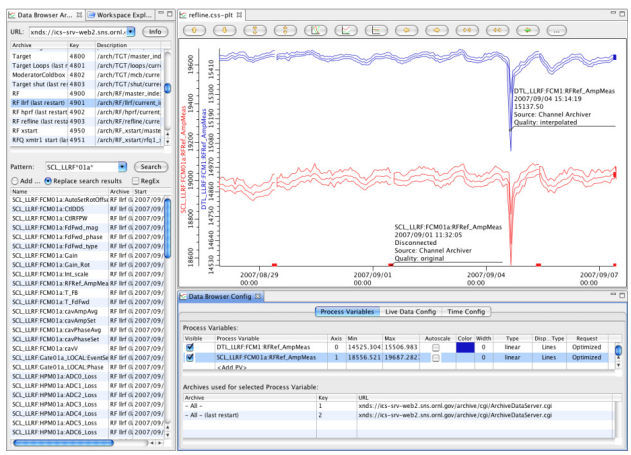

Figure 3: Data Browser Example.

The Data Browser was designed with usability in mind. Compared to existing EPICS tools, changes to axis assignments, colors, or data ranges require fewer mouse clicks or keystrokes. When a PV name is received via drag-and-drop onto the plot, the PV is added with its own new vertical axis; when dropped onto an existing axis, it is added to that axis.

The plot can show Markers for selected samples, which indicate the data source: live, archive, ... If supported by the data server, the sample Quality (original, interpolated) is used to display a reduced sample count with minimum, maximum and average values while viewing a broad time range, automatically switching to the original samples when zooming in close enough.

During zoom or pan operations, the graph redraws quickly with the available data, while background tasks request new samples and trigger a redraw on the arrival of data. The standard Eclipse "Progress" view can optionally be used to monitor or abort ongoing background tasks.

When opening multiple instances of the Data Browser, they share the same view panels to configure the plot, explore individual samples, or export data. For example, the "Data Browser Config View" that is visible in the lower-right section of Fig. 3 always reflects the configuration of the most recently selected Data Browser instance.

Such shared views are common in modern programs. Text editors, for example, can open multiple documents that share the same formatting palette. Older control system applications, however, were usually written without a framework to support these ideas, thus wasting screen space with per-instance views, which often even come in the form of modal and fixed-size dialogs.

SYNOPTIC DISPLAY

The Synoptic Display is a graphical editor for operator interface screens. It was designed to support multiple control systems, and offer widgets where every property can be dynamic. A label widget can display a static text like "Pressure", but it can also be configured with a text property that reflects the current value of a PV, for example one that contains a pressure reading. Similarly, the position, size, color, or font of a widget can update in response to PV changes, allowing for very powerful display options.

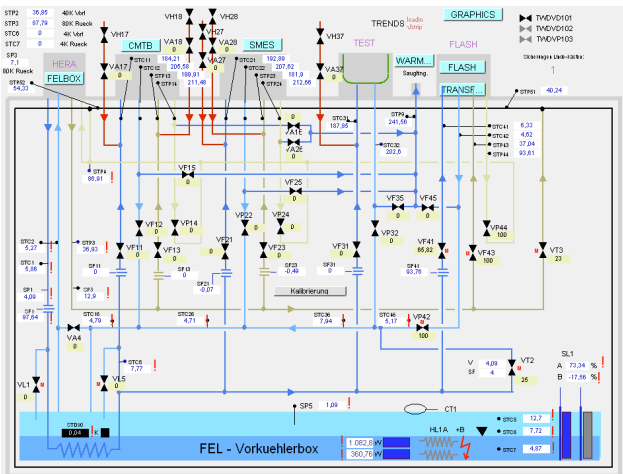

Figure 4: Synoptic Display Example.

INTEGRATION

Eclipse offers numerous ways for sites to package, brand, and deploy CSS as well as other plugins as what is then called a Product.

Packaging and Deployment

A minimal CSS product can be provided that allows users to download the desired tool plugins. You can also prepackage desired components for your site as one product. Updates can be provided via manual or automated download.

Startup Choices, "Splash" and "Welcome"

Custom product startup code can provide an initial login dialog for authentication, a workspace selector, or fulfill other site-specific requirements.

Product configuration also includes a "Splash" screen to display, for example, site-specific legal information, and an initial "Welcome" help screen to guide new users through their first steps.

ISSUES

There is no free lunch. The high usability of CSS comes at the expense of added work for the application developer, and the person who handles the integration and deployment. Collaboration amongst different sites, targeting different control systems and allowing for data exchange, is much more time consuming than writing a standalone application with a limited feature set. The CSS code base continually evolves as we learn about better solutions for data exchange, the use of the Eclipse Workspace, and the quirks of the control systems we attempt to interface.

Instead of designing everything from scratch, CSS greatly benefits from the Eclipse ecosystem, but this alone entails a steep learning curve. Fortunately, a worldwide open-source community supports Eclipse. Numerous books and newsgroups can help when problems are encountered.

CONCLUSION

CSS adds control system APIs and guidelines to Eclipse. The results are modern, portable control system tools, which are not built on a site-specific library, but on a proven open-source framework.

There are already several general-purpose CSS applications, and you can add site-specific applications, which are then fully integrated: common look, behavior and data exchange. CSS can be packaged and deployed as needed.

Acknowledgements

Kenneth Evans (ANL) wrote the first "Probe" for Eclipse. Xiaosong Geng (ORNL) added Chinese translations, and worked on the Data Browser formula package. Jan Hatje (DESY) provided Fig. 4 and DESY application detail. See http://css.desy.de for more.

REFERENCES

[1] M. Clausen and G. Tkacik, "EPICS Office," ICALEPCS'05, Geneva, Oct. 2005.
[2] Eclipse open source community home page, http://www.eclipse.org.
[3] Java technology home page, http://java.sun.com.
[4] J. Hatje, "Control System Studio", ICALEPCS'07, Knoxville, Oct. 2007.
[5] EPICS home page, http://www.aps.anl.gov/epics.