NEW USER INTERFACE CAPABILITIES FOR CONTROL SYSTEMS*

K.U.Kasemir, ORNL, Oak Ridge, TN 37830, U.S.A.

Abstract

Latest technologies promise new control system User Interface (UI) features and greater interoperability of applications. New developments using Java and Eclipse aim to unify diverse control systems and make communication between applications seamless. Web based user interfaces can improve portability and remote access. Modern programming tools improve efficiency, support testing and facilitate shared code. This paper will discuss new developments aimed at improving control system interfaces and their development environment.

LEGACY USER INTERFACES

Control System (CS) UIs were typically limited to one operating system. Tools were often a disjoint mixture of standalone applications. A top-level UI launcher can attempt to provide common access to the tools, but there is little data exchange between the running instances of for example the main operator interface panel, the strip-charting tool, or the alarm display [1]. Each tool has a different optical design, which is typically static and cannot be customized. If tools provide online help, it is found in different places.

MODERN USER EXPERIENCE

In contrast, prevalent desktop automation tools like word processors or email clients have a consistent design. They all include online help; their features are highly configurable. They interoperate to allow sending a text document as email, or to open a received document in the word processor [2]. More of these applications now become web-based, accessible from every operating system with a web browser [3,4]. Some online tools even try to offer automatic links to related information, where a stock price ticker might for example point to news that explain the variations in the price of a stock [5].

CONTROL SYSTEM UI TRENDS

It is impossible to list all developments related to control system UIs worldwide. Some are based on commercial packages [6], others start to use web technology [7].

Java

A common denominator for developments that are shared by more than one installation seems to be the choice of Java technology [8]. One application example is a powerful control panel editor for multiple CS network protocols [9]. Another is the accelerator application library (XAL) and toolbox used by several sites for the rapid development of numerous physics applications [10].

Because of Java, the development tools as well as the resulting applications can be free and portable. Most development tools support full source-level debugging, sophisticated source code navigation, and automated unit testing, which help to develop robust software.

Unfortunately, end users can often tell Java applications from those *native* to an Operating System (OS). Launched by scripts to configure the CLASSPATH and invoke the Java virtual machine, the task bar or process list will show "java" instead of the actual application name. In addition, a mere collection of Java applications is still as disjoint as the legacy applications written in other languages.

Eclipse

The Eclipse Rich Client Platform (RCP) [11] is a framework for building Java applications that appear native to the user and offer a modern, professional look. RCP supports multiple document views, online help in various forms, user-settable preferences, and persistence of settings across application restarts.

RCP is based on a dynamic plug-in model and an elaborate extension point mechanism. Plug-ins are self-describing and auto-initializing bundles of code. Plug-ins can be developed by different people at possibly distant sites, yet hook into the same menu bar, contribute to a common online help system, and thereby present themselves to the end user in a uniform product [12].

CONTROL SYSTEM STUDIO

The Control System Studio (CSS) initiative uses Eclipse RCP to develop a modern, portable and consistent set of CS applications [13]. CSS defines data types for CS-specific items like Process Variables (PVs). It declares extension points for providing live or historic PV data, and includes implementations for EPICS [14] or other control systems.

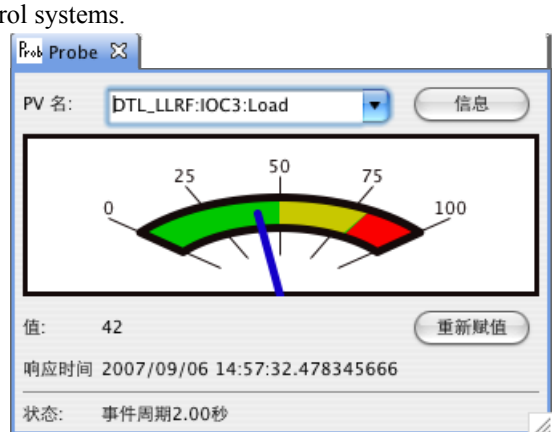

Figure 1: "Probe" with Chinese localization.

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

CSS APPLICATION PLUG-INS

The following list of CSS plug-ins is not complete but meant to show which types of plug-ins are possible.

Simple CSS Tools

The CSS "Probe" plug-in allows users to display the current value of a PV (Fig. 1). The "EPICS PV Tree" displays EPICS record links. These are indeed simple tools, but compared to the legacy EPICS "probe" tool this version is fully resizable, keeps a per-user history of previously inspected PV names, and also offers localization.

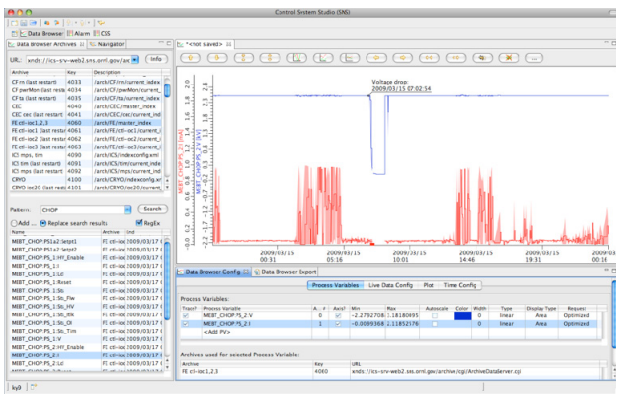

Figure 2: "Data Browser" plot with archive browser and configuration panel.

Data Browser

The Data Browser (Fig. 2) is an interactive plotting or strip-charting tool for historic as well as live PV samples. Users can zoom in and out, pan back and forth, add annotations, customize the appearance of the plot in several ways or export the data. It supports multiple archive data sources at the same time.

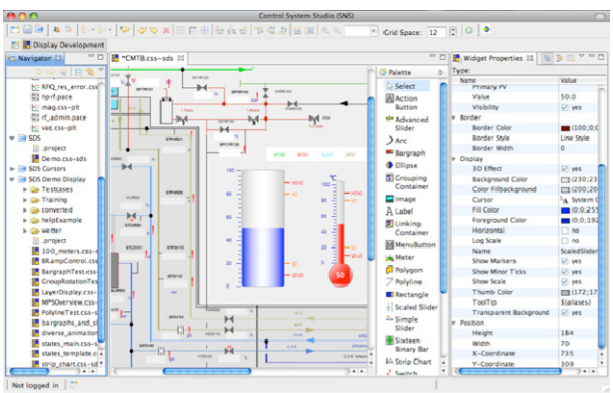

Figure 3: "Synoptic Display" editor.

Synoptic Display

The Synoptic Display (Fig. 3) is an operator interface editor and runtime engine. It offers the usual graphical base elements as well as control system widgets like meters, sliders or buttons. It has extensive editing features.

At runtime, every aspect of a widget can be *dynamic* so that the position, size, color or font of a widget changes to reflect the current value of a PV. It therefore bridges the gap between easy to use yet limited control system UI editors, and tools which offer more flexibility but require programming skills for even the simplest operator display.

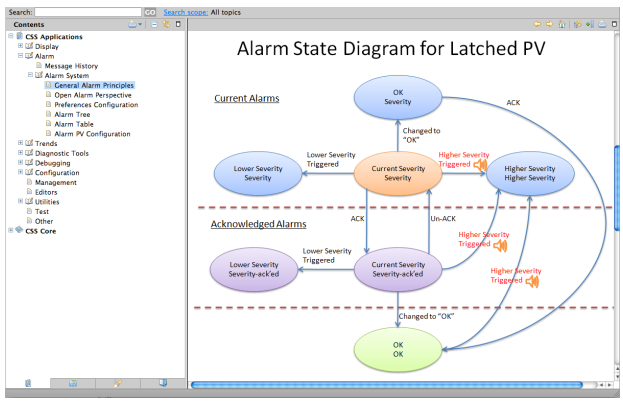

Figure 4: SNS Alarm System State diagram in online help.

Spallation Neutron Source (SNS) Alarm System

The SNS alarm system (Fig. 4) uses CSS plug-ins to display the currently active alarms in either a tabular list or a hierarchical tree view, allowing operators to acknowledge alarms, invoke related operator displays to handle the alarm or access guidance messages. It is also used to configure the alarm system online, or to inspect the history of a certain alarm.

ECLIPSE FEATURES

Like their legacy counterparts, each of the tools highlighted so far would be useful in an accelerator control room as individual applications. They can become more useful, however, when they interact within a unifying context.

Workbench

The Eclipse Workbench presents all CSS tools in a common desktop environment with a shared menu and tool bar. Users can open, close and position the available panels, for example the configuration section of the Data Browser and an instance of Probe, to their liking, and this state is then preserved through restarts of CSS. Users can define more than one layout and save each as a named *Perspective*. The Workbench includes a file browser that will for example start the Data Browser when opening one of its configuration files.

It is important to understand that simply adding a plug-in to CSS will cause its menu entries and tool bar icons to appear in the Workbench, just as its online help and preferences options described below will be integrated into the CSS product. There is no need to manually "rebuild" the CSS product when plug-ins are added or removed.

Localization

Eclipse supports localization during the software development cycle by aiding in the externalization of UI texts to files, and at runtime by loading the correct localization files based on the operating system's locate settings. This way, most CSS tools now support English and German localization, some even Chinese (Fig. 1).

Integrated Help and Preference System

All plug-ins can contribute their help (Fig. 4) or preference settings (Fig. 5) to the Eclipse help resp. preference UI, allowing users to search the online help for information on any plug-in, or to configure any of the installed plug-ins in a common way.

In addition to a help system similar to web pages Eclipse offers a *Cheat Sheet* system where users can select interactive step-by-step instructions, for example for displaying archived data in a plot.

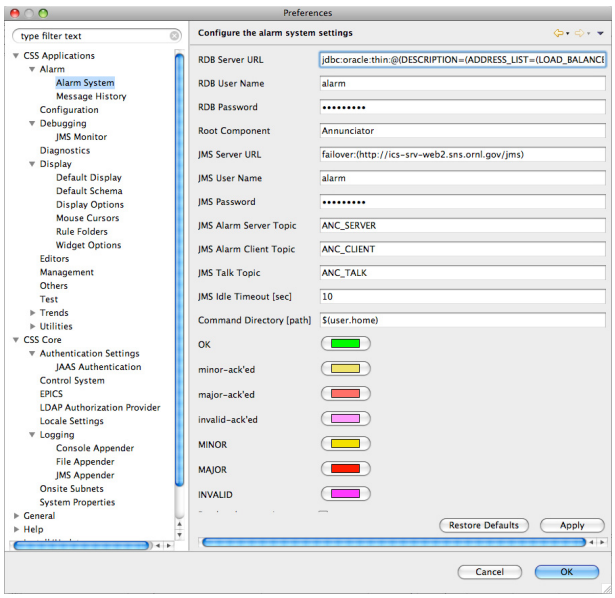

Figure 5: Preference panel opened to edit Alarm System settings.

CSS FEATURES

CSS enhances Eclipse with extension points and default implementations for control system connectivity.

Live and Historic Data Support

Tools like Probe or the Data Browser are not hard-coded to the interfaces or network protocols of a specific control system. Instead they utilize extension points to access live and historic data access. Implementations are available for live data from EPICS, and historic data from various versions of the Channel Archiver.

PV Exchange

CSS defines data types for control system related objects including PVs. This allows CSS tools to implement type-specific Drag-and-Drop data exchange, where the tools can react in different ways to for example a received PV name or Front-End controller name.

An Eclipse mechanism called Object Contribution allows PV tools like Probe or the Data Browser to request addition to context menus. At the same time, tools that display PV related information notify the Eclipse Workbench about currently selected PV names.

In combination this allows users to right-click on most PV related display items, and send the PV names to any of the other PV-related CSS tools. Figure 6 shows how the context menu of an active alarm in the Alarm Table leads to the Data Browser, which would then open up and display the recent history of the PV. If not already running, Eclipse will start the invoked tool and send the selected PV names to it. This CSS approach is much more convenient than legacy tools which required users to manually start the "target" application, then copy and paste PV names between tools.

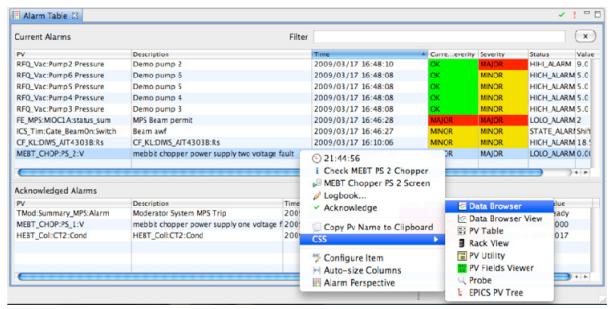

Figure 6: Context Menu activated on a PV in the SNS Alarm Table, allowing user to send the PV to for example the Data Browser.

Electronic Logbook Access

CSS provides an API for making text and image entries into an electronic logbook (E-Log), so that users can conveniently log information about current alarms from the SNS alarm UI, or submit commented snapshots of Data Browser plots.

Since electronic logbook systems differ for each accelerator, a CSS extension point allows site-specific plug-ins to provide the logbook implementation, which is currently available for the Oracle-based SNS ELog [15].

Authentication, Authorization

For applications like an alarm handling system where actions should be performed by known, i.e. authenticated users, and configuration changes must be limited to certain individuals, CSS offers role-based authentication and authorization. The implementation is again based on extension points, with existing plugins for Kerberos or LDAP-based Authentication and Authorization.

Site-Specific Packaging

The addition or removal of plugins from a CSS installation is fundamentally as easy as adding or removing a Java Archive 'jar' file. CSS generates error

messages when a required extension is missing, for example when no plugin for live PV data access is found, or it will simply suppress related features like E-Log submission when no implementation is available.

The sheer number of available plug-ins and their interdependencies can, however, be confusing, so each site should choose a suitable deployment strategy. Some may provide a bare-bones CSS product and allow end-users to download collections of related plug-ins, bundled as Eclipse *Features*, from an online site which can later also provide version updates. Others may prefer to provide a site-specific CSS product that already contains all relevant plug-ins, with preconfigured settings for the local accelerator [16].

The CSS administrator at a site can adjust the application startup code such that it prompts for login information, a workspace directory or other site-specific needs. The addition of a "splash" screen and a "welcome" page to guide users during their first steps then results in a product that is customized for local needs, but at the same time offers professional appearance and functionality.

WEB APPLICATIONS

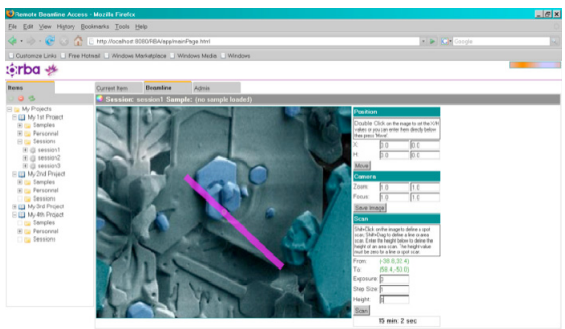

Figure 7: Selecting a scan region in a web-based remote beam-line tool [17].

Following the trend towards web technology, there are first CS applications that offer zero-install remote access from a web browser, exemplified in Fig. 7. While very convenient for the user, they still face a few problems.

Web browsers differ, especially regarding features that are important for highly interactive web sites, a problem which is beginning to be addressed by compatibility libraries [18].

When CS data needs to be redirected through web protocols, the response time suffers. Add-ons that allow the browser to directly communicate with the control system solve this problem [19], but are no longer zero-install, face browser incompatibility, and reopen security issues which the move to web protocols was supposed to solve.

Finally, web applications consist of code running on the web server, often in Java or a scripting language, code running in the browser, usually a combination of Java Script, style sheets and HTML. This distributed mix of technologies is harder to debug and maintain than a single-source Java application. The Eclipse Rich Ajax Platform (RAP) promises an RCP-like Java development environment, but resulting in a product can be deployed as a web application [20, 21].

CONCLUSION

Many ongoing CS UI developments are based on Java to create portable tools. By also adopting Eclipse technology like CSS, a disjoint collection of individual tools can turn into a homogeneous product.

ACKNOWLEDGEMENTS

CSS is very much a combined effort, see [13].

REFERENCES

[1] "MEDM", "StripTool", "ALH" and other network client tools for EPICS, http://aps.anl.gov/epics/extensions
[2] Microsoft Office, http://office.microsoft.com
[3] Google Docs, http://docs.google.com
[4] Google Mail, http://mail.google.com
[5] Google Finance, http://www.google.com/finance
[6] Xihui Chen et al, "Detector Control System of BESIII", ICALEPCS07, Knoxville, TN.
[7] E. Matias, "Remote Access to the Canadian Light Source", PCaPAC08, Ljubljana, Slovenia.
[8] Java Technology, http://java.sun.com
[9] Elke Sombrowski et al, "JDDD: A Java DOOCS Data Display for the XFEL", ICALEPCS07, Knoxville, TN.
[10] Tomas Pelaia II, "XAL Application Framework and Bricks GUI Builder", ICALEPCS07, Knoxville, TN.
[11] Eclipse, http://www.eclipse.org
[12] Eric Clayberg, Dan Rubel, "Eclipse Plug-ins", Addison-Wesley, 2008
[13] Control System Studio, http://css.desy.de
[14] Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics
[15] Thomas Pelaia II, "SNS Electronic Logbook", EPICS Collaboration Meeting, Santa Fe, NM, 2004
[16] CSS for the Spallation Neutron Source, http://ics-web.sns.ornl.gov/css
[17] Presentation slides for [7]
[18] Google Web Toolkit, http://code.google.com/webtoolkit
[19] Web CA, http://webca.cosylab.com
[20] Ralf Sternberg, Rüdiger Herrmann: "Single Sourcing RCP and RAP" - Desktop and web clients from a single code base: EclipseCon 2009, Santa Clara, CA.
[21] Eclipse Rich Ajax Platform, http://www.eclipse.org/rap