CONTROL SYSTEM STUDIO (CSS)

Jan Hatje, M. Clausen, Ch. Gerke, M. Moeller, H. Rickens, DESY, Hamburg, Germany

Abstract

Most applications for the control system EPICS are developed for UNIX and X-Windows. They are independent from each other, have a different look and feel, and it is difficult to exchange data. To solve these problems, the Control System Studio (CSS) is under development. CSS is a common platform for new control system applications and provides developers with management infrastructure and a centralised connection to external data sources like JDBC-databases, JMS-, LDAP-servers, etc. CSS defines interfaces to avoid dependencies on special implementations. This design makes sure that an application can easily be integrated or exchanged. Another important feature is the accessibility of data through all applications via CSS-data types defined in CSS. The Data Access Layer (DAL) assures the transparent access to any control system protocol. Thus CSS is not only a platform for EPICS but for any control system protocol that implements the DAL. The intention to modularise CSS and run it on any operating system led to the decision to use the Eclipse RCP based on the OSGi technology. Technically CSS is a set of essential core-plugins and application plugins selected by the user.

MOTIVATION

The current operator interfaces are all individual applications that are designed primarily for reliability and performance. They are mainly running only on UNIX because they are developed with X-Windows and are written for the control system EPICS. State of the art software features like data exchange by objects, common network interfaces, etc. are missing.

The Control System Studio (CSS) is designed to be operating system independent with a common look and feel. CSS provides services and API's to make the development of new applications easy. It is decoupled from specific control system protocols, implementations, etc. [1]

ECLIPSE RCP

CSS does not create a new framework but uses the Rich Client Platform (RCP) Eclipse. The Eclipse RCP is written in Java and runs on every operating system for which a Java Virtual Machine is available. Eclipse is based on plugin technology and therefore easy extensible. It provides many generic features for applications like menus, preferences, help system, etc.

The decision for Eclipse and not for Netbeans or another RCP was made because of the largest community of developers, and the implementation of the OSGi framework. The OSGi alliance is an open standard organisation that has specified a Java based service platform defining layers, services and API's.

The Eclipse Runtime is the basis of each Eclipse RCP installation (see fig. 2). The runtime manages and loads all available plugins that are providing the functionality of the application.

Extension Points

To add new functionality to the system, Eclipse uses the extension point mechanism. A plugin defines an extension point and other plugins can implement the extension point to offer additional services (see fig. 1). An extension point consists of the rules for the extension in a XML document. In general there is also an interface that has to be implemented.

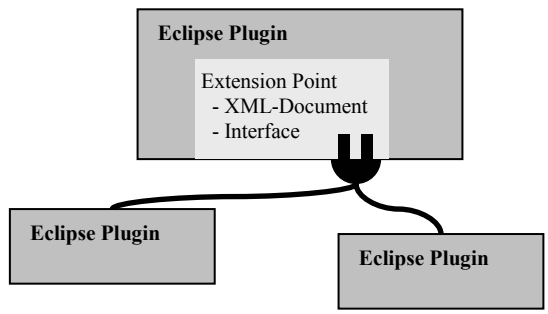

Figure 1: Eclipse extension point

The advantage of extension points is that new plugins with an implementation of the extension can be added without changing the existing system.

CSS DESIGN

CSS consists of two parts: the CSS core plugins (headless and UI) with common features like control system data types, management of CSS instances, etc. and the CSS application plugins [2].

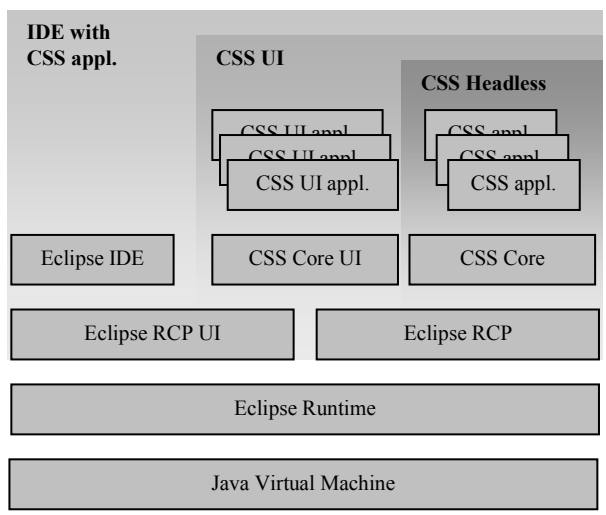

Figure 2: Architecture of Eclipse IDE and CSS

Technically CSS plugins are ordinary Eclipse plugins. If all CSS core plugins are present on which a CSS application has a dependency it is possible to use them also in an Eclipse IDE (see fig. 2).

Another advantage of using the Eclipse RCP compared to stand alone Java application is the quick startup. All CSS application plugins run in the same Java Virtual Machine (JVM) of the Eclipse runtime. Common services like database connections are handled by the CSS core plugins and have to be initialized only once. Stand alone applications have to use their own JVM and sharing services is not possible.

Headless Mode

For some applications a User Interface (UI) is not necessary but it would be useful to use features of CSS core like the logging service or management.

All Eclipse- and CSS plugins are separated if they are related to UI or not. A plugin without dependencies to other UI-plugins can be executed in the headless mode of Eclipse (see fig. 2).

ECLIPSE FEATURES

Eclipse provides many features that CSS uses by the extension point mechanism. For the CSS menu, help system and preferences CSS defines a structure of categories like CSS/Display or CSS/Diagnostics. Each CSS plugin that implements the extension points will be automatically added in the CSS menu. The preference pages are available and the HTML help pages will appear in the help system.

The localization is managed by simple text files in which all Strings that appear on GUI elements are stored. To use a new language in CSS, a file with the translation has to be added in the plugin directory.

Update Site

The Update Site is a very convenient way for the user to select only the applications that he wants to install. Normally the options to configure an installation of common software are very limited.

CSS loaded from the CSS homepage consists only of the core plugins without any applications. The CSS applications can be initially installed or updated from the CSS Update Site. Eclipse checks automatically if there are unresolved dependencies in the selection of plugins.

Object Contribution, Drag and Drop

CSS defines data types for control systems like ‘Process Variable’ or ‘Front End Controller’. To exchange objects of these data types between plugins, Eclipse provides Drag and Drop and object contribution.

To use this mechanism, the CSS plugin has to implement an extension point.

Fig. 3 displays an example for object contribution. Each row of the table is represented by an object of the type ‘Process Variable’. With a right click on a row a menu pops up with all plugins that can handle ‘Process Variables’. Eclipse also checks if an array of objects is selected and filters the plugins in this case.

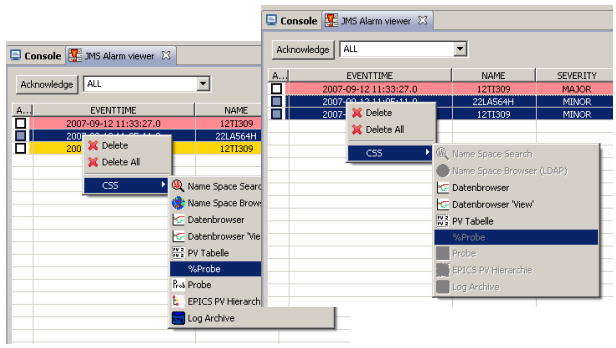

Figure 3: Example for object contribution

CSS FEATURES

On top of Eclipse CSS adds features and services for CSS applications.

CSS core contains some common libraries like JDBC, JMS, etc. to prevent that each plugin comes along with its own libraries. Thus, there are no different versions in the installation and the basic settings can be managed via one preference page.

Preferences

The CSS settings for e. g. the control system or database connections differ for institutes and even for the installations in the same institute. To make the adjustment easy, there is only one file with all preferences in the root directory of CSS. The preferences can be changed manually and will be updated on the next startup.

Another way is the export function for preferences. The preferences can be set in one CSS installation. Using the export function they can be written to a file and distributed to other CSS instances.

Logging service

CSS provides a convenient and customizable logging mechanism, which is based on LOG4J. It can be used to get information about the runtime status of the application or errors that occur. The log settings can be changed during runtime via the preference pages.

The available methods correspond to the LOG4J log level hierarchy:

DEBUG, INFO, WARN, ERROR, FATAL

Three different destinations for the log messages are now available: the console, file system and JMS server. More destinations can be added by implementing the extension point of the CSS logging.

Authorization, Authentication

For security reasons it is important to set restrictions that not all users can execute all commands. As the authorisation and authentication can be handled in many different ways, extension points and an API are implemented for this purpose.

For authentication an implementation for the Kerberos server is currently available. The implementation for

authorisation uses an LDAP directory to store the access rights for the user groups.

Management of CSS instances

Especially for headless CSS applications or if there are many CSS instances, it is convenient to manage them remotely.

The CSS management is built on top of the Extensible Messaging and Presence Protocol (XMPP). Each plugin can use an extension point to add new remote actions to the management system.

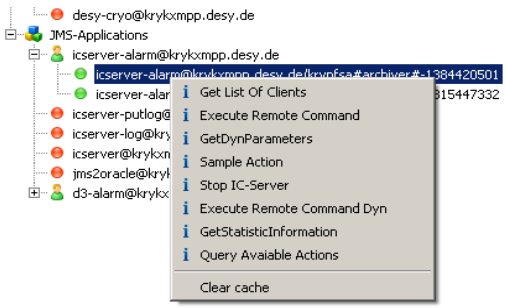

Figure 4: CSS instances with remote actions

In the snapshot of fig. 4 is a list of registered CSS instances. A green flag indicates that the instance is online. The context menu displays all available actions that the instance provides. It is also possible to select more than one instance and execute remote commands for all of them e. g. to force an update or deploy new preferences.

DATA ACCESS LAYER (DAL)

The Data Access Layer (DAL) [3] is a consistent set of interfaces to access the dynamic data of control systems. The API decouples the data access from the implementation for a certain control system (see fig. 5).

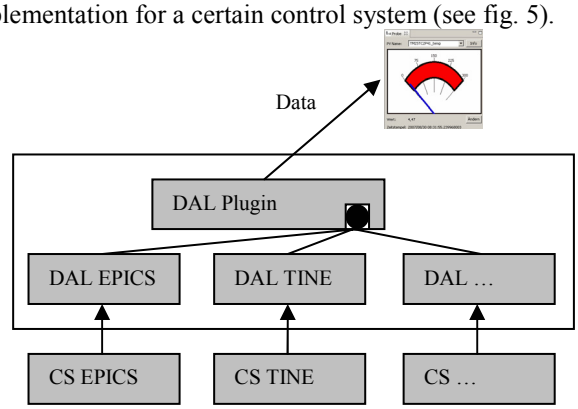

Figure 5: Architecture of the Data Access Layer

The API is integrated in a plugin with an extension point. Therefore, a plugin with a new implementation for a control system can be installed without recompiling the CSS core.

To decide which implementation is adequate, the name of a Process Variable has to start with a prefix like <control system name>://<process variable name>. A Factory in CSS core parses the prefix and returns the appropriate DAL implementation. If no prefix is provided, the default control system defined in the preferences will be returned.

Currently DAL implementations are available for the control systems EPICS and TINE.

ARCHIVE ACCESS LAYER (AAL)

The Archive Access Layer is an API for accessing archive data sources. The architecture is similar to the one of the DAL.

Currently are implementations for the channel archiver over XML-RPC and via the AAPI server (DESY) available. Other archive server sources are the archive record and TINE archive server.

CSS APPLICATIONS

Many applications are now available for CSS. The most general applications like Probe, Data Browser or PV Table are developed at ORNL by Kay Kasemir [2]. An EPICS Namespacebrowser and Alarm System come from DESY but these applications need additional software like LDAP server or SQL database.

The Synoptic Display Studio (SDS) is developed on the basis of a scientific contract with the University of Hamburg: It will replace dm2k in the future. All widgets are now available in SDS and the final tests by the operators are beginning this month. The ADL-converter converts the ADL-files into the XML format of SDS. This way it is possible to reuse the former displays.

The architecture of SDS is based on plugins and extension points. Therefore, it is very easy to add new widgets to SDS by implementing just two extension points.

OUTLOOK

The development of version 1.0 of CSS core is now available. Only few parts in the DAL and in the CSS management need to be changed or improved.

For the next versions the granularity of the user rights management will be enhanced. A “save and restore functionality” is planned [4] to make often occurring tasks easier to handle.

Besides the technical aspects it should be as easy as possible for developers to create CSS plugins. Because of the complex Eclipse API it is important to offer wizards and a good documentation for the startup. CSS is a collaborative development and each new developer will have to play his role to improve the software.

REFERENCES

[1] M. Clausen and G. Tkacik, “EPICS Office,” ICALEPCS’05, Geneva, Oct. 2005.

[2] K. Kasemir, "Control System Studio Applications", ICALEPCS’07, Knoxville, Oct. 2007.

[3] Igor Kriznar, Data Acces Layer Plug Implementation, http://css.desy.de

[4] Debbie Rogind, “SLAC’s Save/Restore Application Development in the Eclipse RCP Architecture”, ICALEPCS’07, Knoxville, Oct. 2007