"JDDD": A JAVA DOOCS DATA DISPLAY FOR THE XFEL

E.Sombrowski, A. Petrosyan, K. Rehlich, P. Tege, DESY, Hamburg, Germany

Abstract

The X-ray Free-Electron Laser (XFEL) [1] is a new accelerator currently under construction at DESY. It will be a powerful X-ray source for many scientific disciplines ranging from physics, chemistry and biology to material sciences, geophysics and medical diagnostics. The commissioning is planned in 2014 and the preparation of the control system has been started. The XFEL makes high demands on the control system and its user interface. For this reason "jddd", a new Java Data Display program for the Distributed Object Oriented Control System (DOOCS) [2], has been developed. jddd is a graphical editor for designing and running control panels. The editors functionality is similar to standard IDEs like NetBeans or Eclipse. Complex control panels can easily be created without programming. jddd offers all components needed for control panel design. The Components are reusable Java Beans like labels, buttons, plots and complex dynamic components as Switches. The jddd panel structure is stored in an xml format. jddd will be a replacement of the DOOCS data display (ddd) [3] program. For compatibility reasons the old ddd storage format can be converted to the new jddd xml format.

MOTIVATION

A pilot facility for the future XFEL is the Free electron LASer Hamburg (FLASH) [4], which is in operation since 1999. It is a user facility providing laser-like radiation in the VUV and soft X-ray range to various user experiments. FLASH is operated with DOOCS and here ddd is used as a graphical editor and synoptic display program. With the ddd editor graphical displays can easily be designed without any progamming. This concept has proved to be a good choice. More than 1300 control panels have been designed by FLASH operators and experts until today.

From the experience gained by using ddd in the past 10 years many requests for a new graphical user interface emerged, like:

- Platform independency
- A modern, more standard-like graphical editor with a Component Inspector, Undo & Redo functionality, etc.
- Improved components (e.g. plots with mathematical functionality for simple data analysis)
- New components (e.g. a new Switch component)
- Layers
- A web interface
- The possibility to reuse jddd panels in other high level applications.

IMPLEMENTATION

Because of the platform independency it was decided to use Java with standard Swing components. For all reusable components like labels, buttons, etc. the Java Beans technology was used. The existing JDOOCS API [5] was reworked to match the requirements for the connection to the DOOCS control system.

For writing the editor, there are several existing frameworks which can be used as a base. An obvious way would be to use Eclipse or Netbeans and to write additional plugins. Another option would be the Netbeans visual library, which can also be used as an editor API. But after several tests and discussions the decision was made that jddd has to be independent of any external libraries and needs a completely new editor. This guarantees the highest flexibility for writing special components which need extraordinary editor features.

THE JDDD EDITOR

The editor was designed similar to other existing standard graphical editors. It constists of four subwindows (see Fig. 1), which will be described in the following sections.

Editor Window

The central point of view is the Editor Window. It contains a design-time view of a control panel in an editor tab. Selected components can be arranged, aligned, flipped, rotated or grouped. A grid can also be used for alignment.

Component Inspector

On the left hand side is the Component Inspector, which displays a tree hierarchy of all components contained in the currently openend tab. It offers a clearly arranged view on the used components. This view is important, because complex hierarchical elements like nested "If" components or transparent buttons are difficult to see in the Editor Window.

Component Palette

On the upper right is the Component Palette. It contains a list of all available components that can be used for control panel design. There are five different component types:

- Pane components which help structuring the control panels (i.e. tabbed panes).
- Static components like labels, icons and other graphical components.
- Dynamic components which are used for the control and visualization of control system values (i.e. buttons, values, checkboxes, sliders...).
- Logic components whose appearance depends on a user defined control system value (i.e. the "If" and the "Switch" component).
- Plot components which display one or multiple data channels.

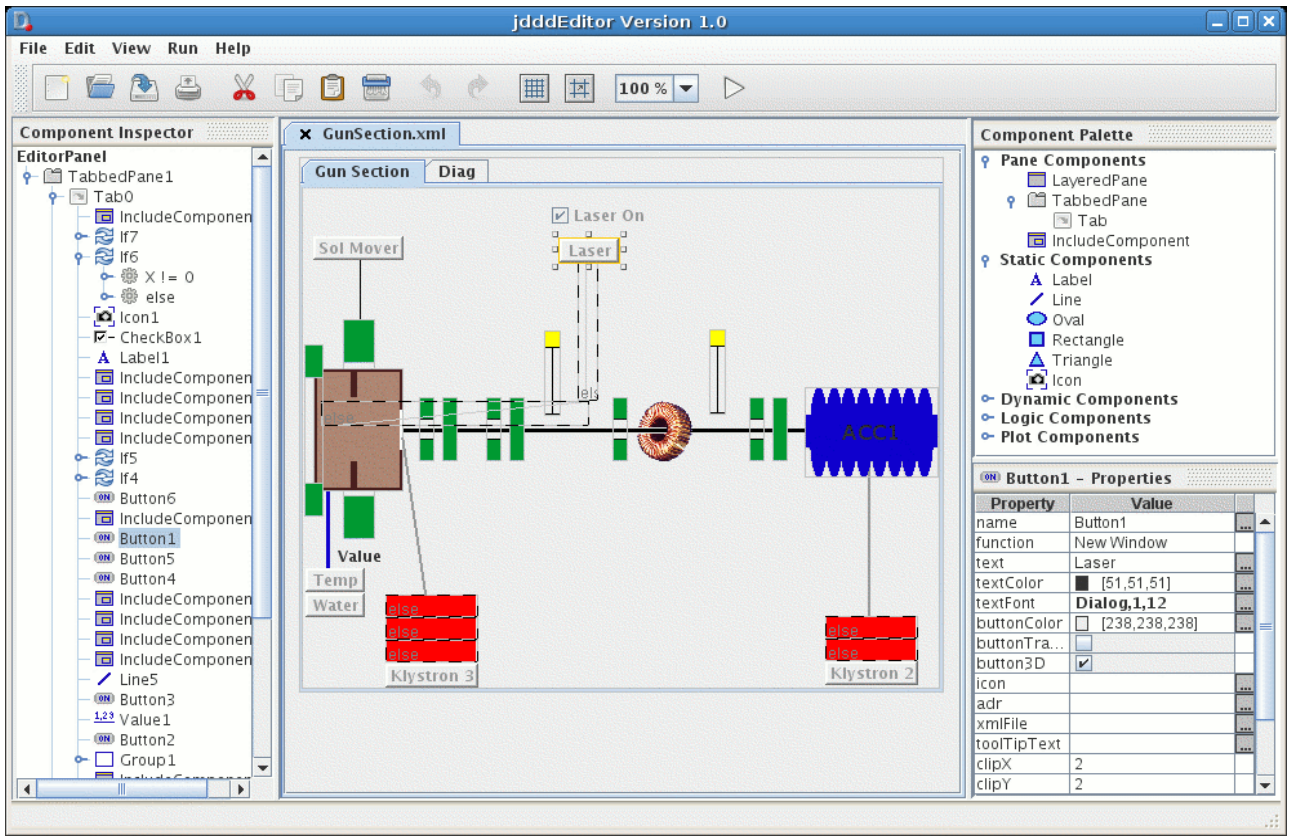

Figure 1: jddd editor screenshot.

Component Properties

On the lower right is the Component Properties table. It displays the editable settings for the currently selected component(s). The property values are typed directly in the value field. Alternatively the buttons on the right hand side open special dialog boxes for different property types. Special editors are a text editor, a bounds editor, a color chooser, a font chooser, a DOOCS address chooser and a file open dialog.

THE RUNTIME MODE

The jddd program has two different execution modes. The first one is the jddd editor mode for drawing and testing panels. The created panels are saved in a xml file format. The second one is a run-time mode, where these xml files are parsed and executed.

In run-time mode the data displayed on the panels update with a rate which is set for each component in the editor. Further information like archived data are accessed by mouse click from the status displays.

The displays can be organized in a hierarchical manner. Then buttons are defined to open new panels which provide more detailed information on certain subsystems.

A drag & drop functionality enables the operator to drop additional data into a plot and e.g. to compare spectra at runtime. The plot component also offers simple data analysis functionality like fitting or fourier analysis of online spectra (see Fig. 2).

For documentation the displays can be printed directly or else a screenshot can be saved in png file format.

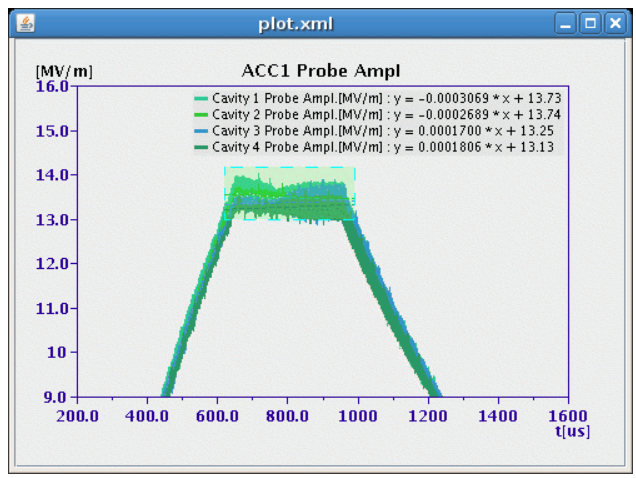

Figure 2: The plot component with a linear fit.

STARTING JDDD

For starting jddd the Java Web Start technology is used. The jddd jar archives are stored on a central web server and can be accessed from all around the world at: http://jddd.desy.de

Java Web Start ensures the most current version of the application will be deployed, as well as the correct version of the Java Runtime Environment.

SPECIAL EDITOR FEATURES

With the assistance of the jddd editor synoptical displays are designed in a short time without any programming effort or knowledge.

All displays may be used as generic library components and may be added to other jddd displays. For example if there is a panel displaying a steerer, multiple instances of this steerers can be included in a panel displaying the beam pipe (see Fig. 3). Then each steerer gets its individual device address.

Library components help to create control panels in an efficient way, because similar components which differ only in the device address have to be created only once and can be reused afterwards. In addition changing the original component changes all its clones. This saves a lot of work, because instead of editing every single component, only the “template” has to be modified.

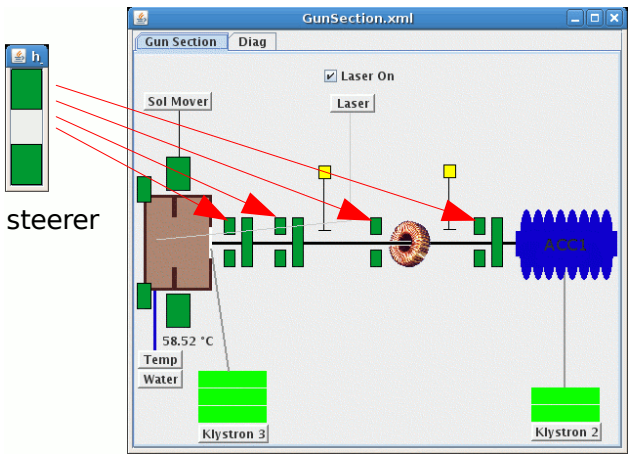

Figure 3: Example for a steerer library component.

For some high level displays the jddd editor comes - despite of its logic components - to its limit. The development of such high level applications is simplified with different export possibilities of jddd displays (see Fig. 4):

The first one is to save a jddd display as a Java JFrame or JPanel. This way complex displays are easiliy designed with the jddd editor and complex functionality can be added to the Java source code. The only disadvantage of this method is that panels cannot be revised with the jddd editor after external modification of the Java code.

The second way is therefore more flexible: jddd displays may be reused as Java Beans in other applications. Only a few lines of codes are needed to add a jddd display to an existing Java application. The properties of all components in this jddd display can easily be accessed from the application. This way the jddd xml files can be modified any time without affecting the applications source code.

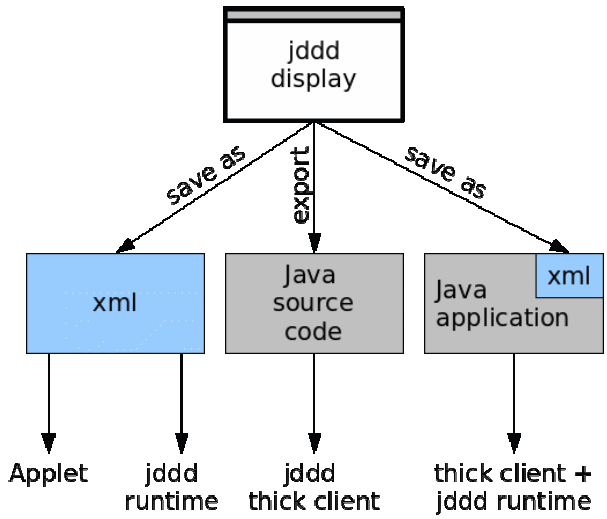

Figure 4: jddd save and export possibilities.

CONCLUSION

Java with Swing components has proved to be a good choice for graphical control panel design. The “selfmade” jddd editor offers the required flexibility and has a good performance even with huge displays, which contain more then 1000 components. It is easy and intuitive to use without reading a manual. Up to now jddd has already 25 different components including complex dynamic and logical components. The functionality of these components exceeds the functionality of the old ddd components and enables the user to create more complex control system panels. With jddd in a first step the old ddd displays will be converted then in a second step a new generation of control displays will be designed.

The next milestone in jddd development will be a central place for data storage. This central place is planned to be a subversioning sytem. It has the advantage that older versions of a panel are not simply replaced by a newer verison and the design history of the displays doesn't get lost.

REFERENCES

[1] http://xfel.desy.de
[2] http://doocs.desy.de
[3] K.Rehlich, “An Object-Oriented Data Display for the TESLA Test Facility”, ICALEPCS’97, Beijing, June 1997.
[4] http://flash.desy.de
[5] K. Rehlich and V. Kocharyan, “JDOOCS – a Java Library for DOOCS”, PCaPAC’02, Frascati, October 2002.