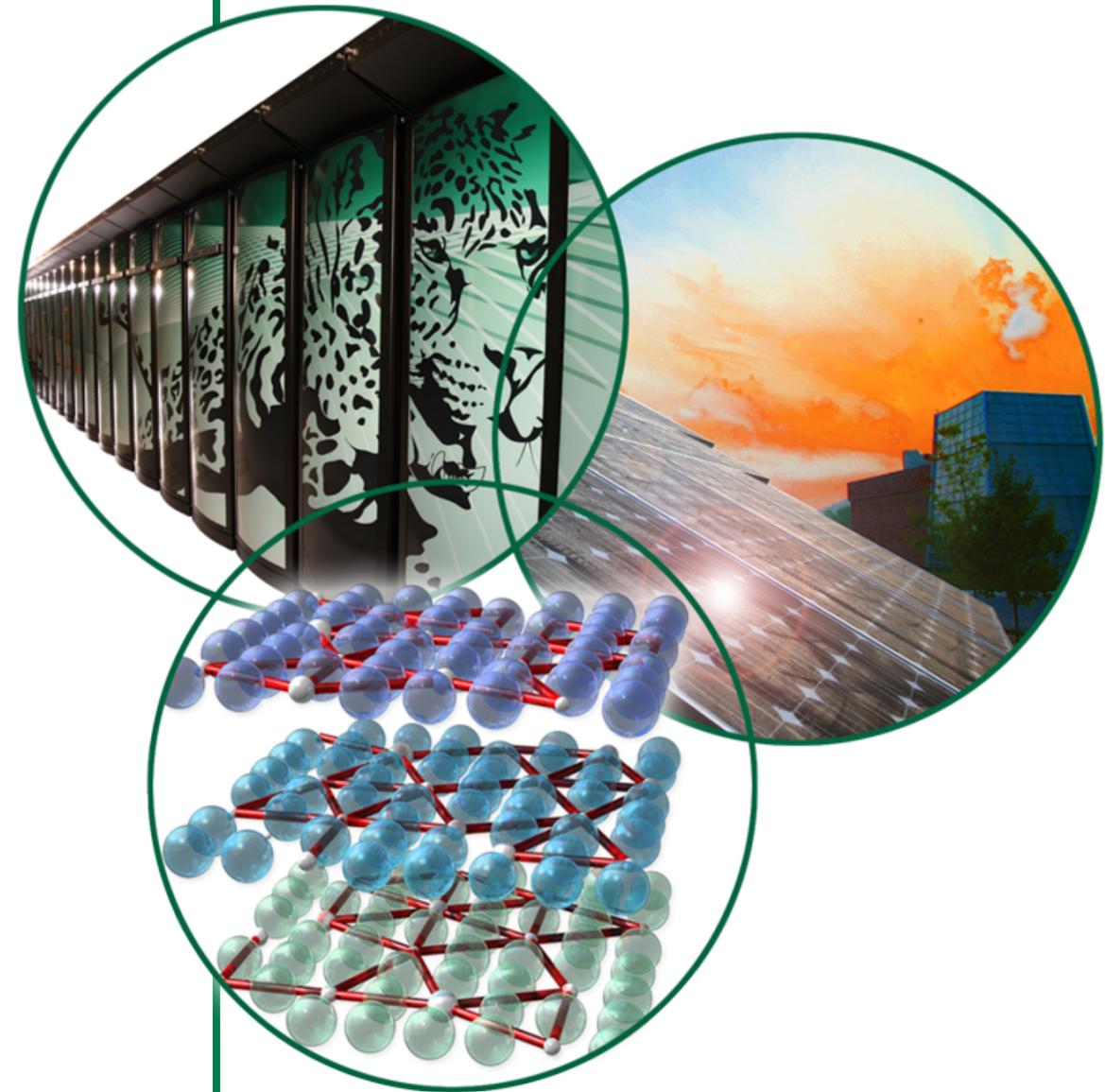


Writing Widgets for BOY & A brief introduction to XYGraph Library

Xihui Chen

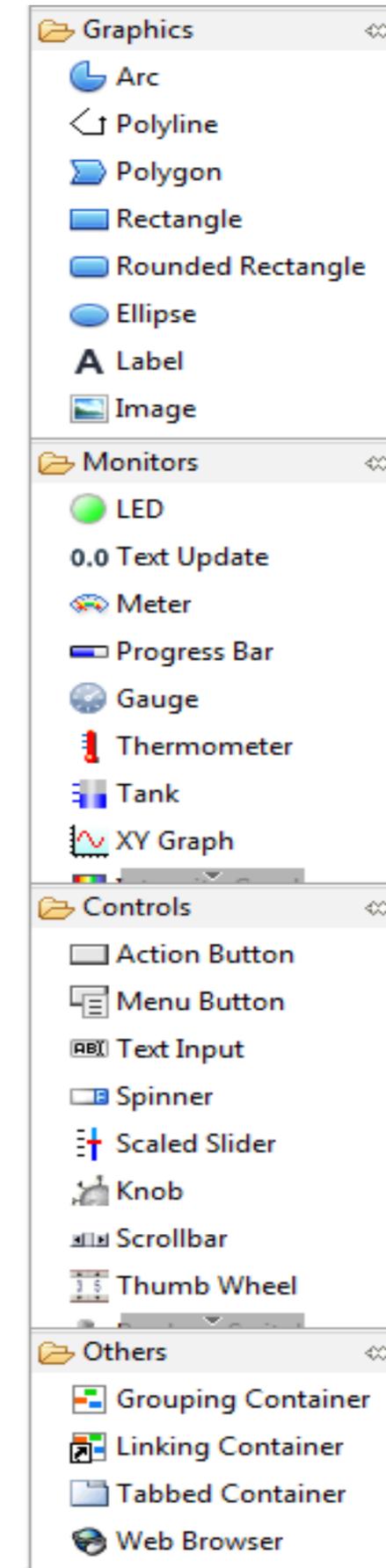
chenx1@ornl.gov

Fall 2010 EPICS Collaboration Meeting

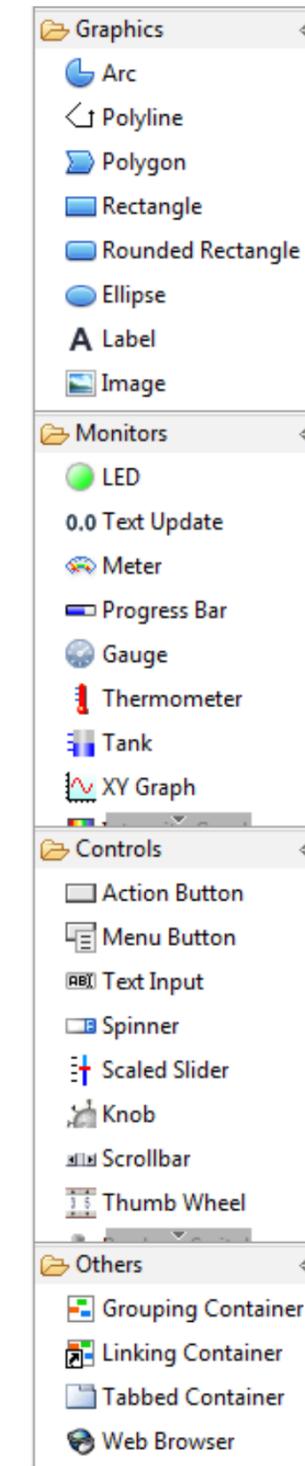
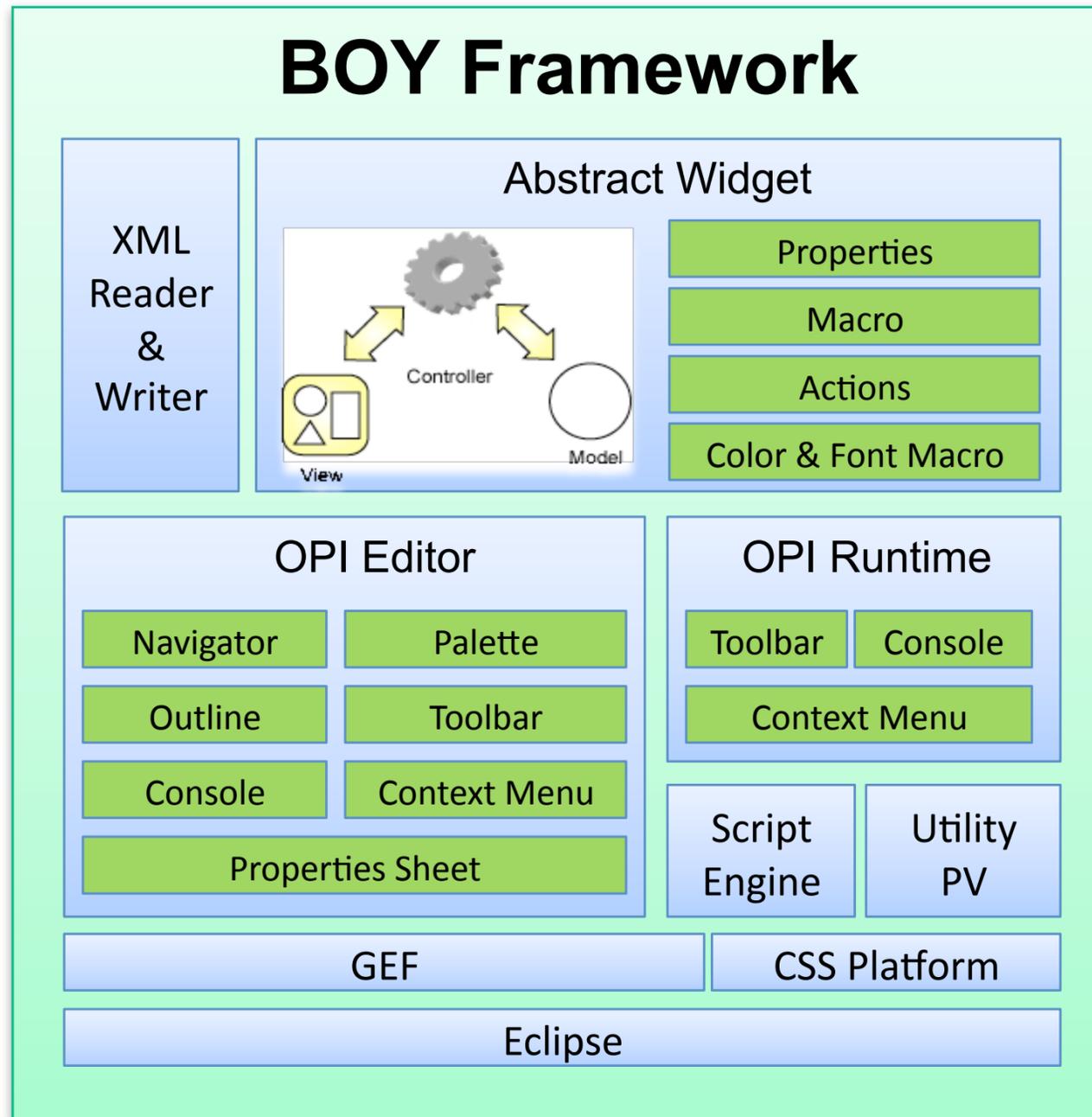


What is a BOY widget?

- The brick to construct an OPI
- A graphical and function unit with a set of properties
- No limitation on its function
 - Decoration
 - Display PV value
 - Set PV value
 - Container
 - Anything for your needs



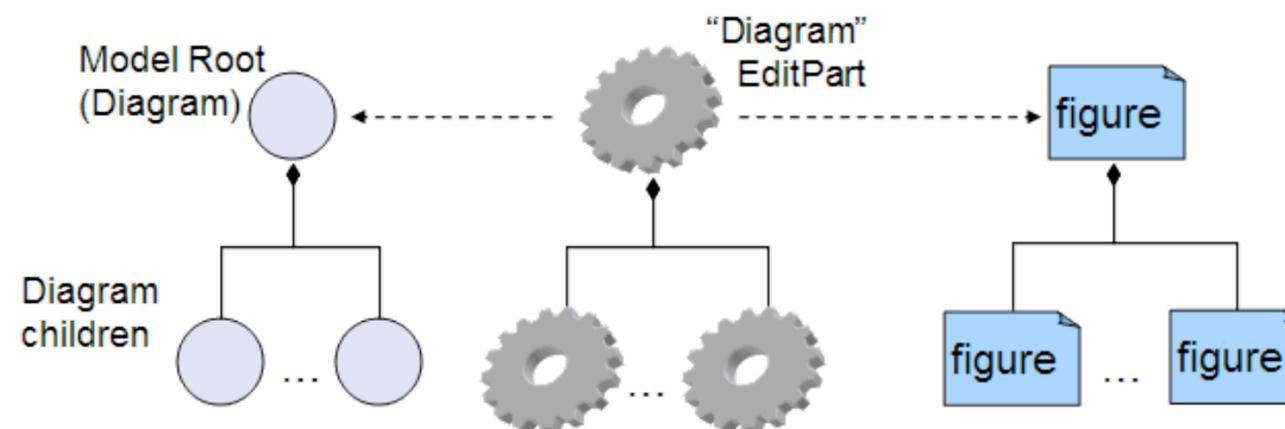
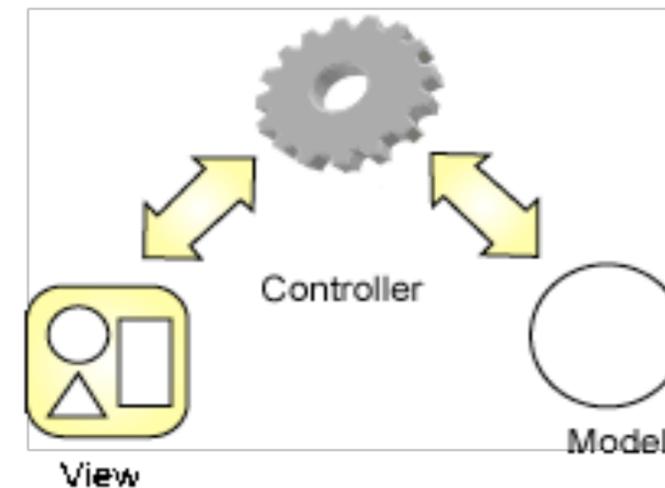
BOY Architecture



Yes, you can plug your own widgets to BOY!

MVC Pattern

- All BOY widgets should follow Model-View-Controller Pattern
 - Separate responsibilities
 - Decouple complexity
 - Increase flexibility
 - Code reuse



Widget Model

- **Definition**
 - **Defines the set of properties**
 - **Stores the according values**
 - **All information of the widget should be stored in its model**

Widget Model

- **Implementation**

- **All widgets models must subclass AbstractWidgetModel**

- Provided the basic properties for all widgets, such as Width, Height, X, Y, Name, Foreground Color, Background Color, Rules, Scripts, Actions...

- **All PV widgets should subclass AbstractPVWidgetModel**

- Provided the basic properties for all PV widgets, such as PV Name, PV Value, border/foreground/background color alarm sensitive

- **All Container Widgets must subclass AbstractContainerModel**

- Provided the functions to manage children
 - Macros Property

Widget Model Example

```
public class SimpleBarGraphModel extends AbstractPVWidgetModel{

    /** Lower limit of the widget. */
    public static final String PROP_MIN = "max"; //$NON-NLS-1$

    /** Higher limit of the widget. */
    public static final String PROP_MAX = "min"; //$NON-NLS-1$

    public final String ID = "org.csstudio.opibuilder.widgetExample.SimpleBarGraph"; //$NON-NLS-1$

    /**
     * Initialize the properties when the widget is first created.
     */
    public SimpleBarGraphModel() {
        setForegroundColor(new RGB(255, 0, 0));
        setBackgroundColor(new RGB(0,0,255));
        setSize(50, 100);
    }

    @Override
    protected void configureProperties() {
        addProperty(new DoubleProperty(PROP_MIN, "Min", WidgetPropertyCategory.Behavior, 0));
        addProperty(new DoubleProperty(PROP_MAX, "Max", WidgetPropertyCategory.Behavior, 100));
    }

    @Override
    public String getTypeID() {
        return ID;
    }

    ...
}
```

Widget Properties

- **Currently supported property types:**

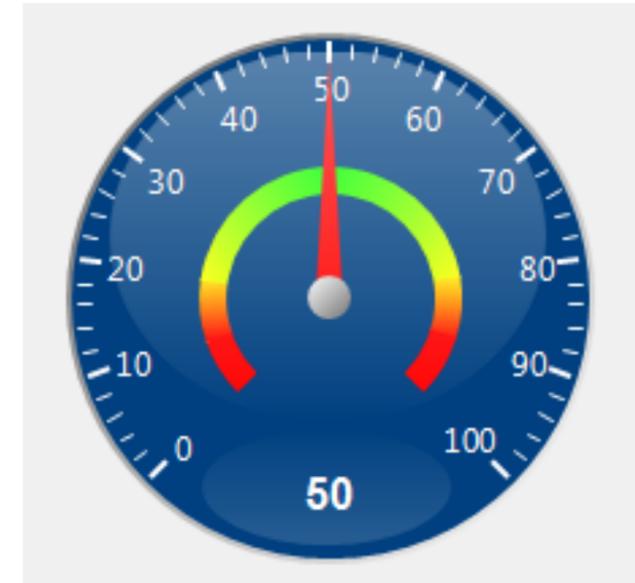
Property Type	Example Properties
Boolean Property	Enabled, Visible
Integer Property	Height, Width, X, Y
Double Property	Meter.Level HIHI, Meter.Maximum
Combo Property	Border Style
String Property	Name, PV Name, Text
Color Property	Background Color, Foreground Color
Font Property	Font
File Path Property	Image.Image File, Linking Container.OPI File
PointList Property	Polyline.Points, Polygon.Points
Macros Property	Macros
ColorMap Property	IntensityGraph.Color Map
String List Property	Items
Rules Property	Rules
Actions Property	Actions
Scripts Property	Scripts

- **Create your own property type by extending *org.csstudio.opibuilder.properties.AbstractWidgetProperty***

Widget Figure (View)

- **Definition**

- The graphical representation of the widget
- Response to mouse or keyboard event
- No dependency on model and controller.
- It should be autonomous, which means it should have its own behavior independent from model and controller



- **Implementation**

- Need Draw2D knowledge
 - <http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.draw2d.doc.isv/guide/guide.html>
- All widgets figure must be an instance of `org.eclipse.draw2d.IFigure`
- In most cases, just subclass `org.eclipse.draw2d.Figure` or other existing figures will make it easier.

Widget Figure Example

```
public class SimpleBarGraphFigure extends Figure {

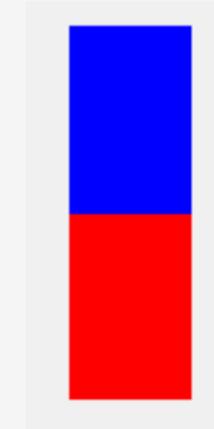
    private double min = 0;
    private double max = 100;
    private double value = 50;

    @Override
    protected void paintClientArea(Graphics graphics) {
        super.paintClientArea(graphics);
        //fill background rectangle
        graphics.setBackground(getBackgroundColor());
        graphics.fillRect(getClientArea());

        //fill foreground rectangle which show the value's position
        graphics.setBackground(getForegroundColor());
        //coerce drawing value in range
        double coercedValue = value;
        if(value < min)
            coercedValue = min;
        else if (value > max)
            coercedValue = max;
        int valueLength = (int) ((coercedValue-min)*getClientArea().height/(max-min));
        graphics.fillRect(getClientArea().x,
            getClientArea().y + getClientArea().height -valueLength,
            getClientArea().width, valueLength);
    }

    public void setValue(double value) {
        this.value = value;
        repaint();
    }

    public double getValue() {
        return value;
    }
    ...
}
```



Widget Editpart (Controller)

- **Definition**
 - The link between model and view
 - Responsible for the behavior of the widget when model properties value changed
 - Responsible for initialization and deactivation
- **Implementation**
 - All widgets editparts must subclass `AbstractWidgetEditPart`
 - All PV widgets should subclass `AbstractPVWidgetEditPart`
 - All Container Widgets must subclass `AbstractContainerEditpart`

Widget Editpart Example

```
public class SimpleBarGraphEditpart extends AbstractPVWidgetEditPart {

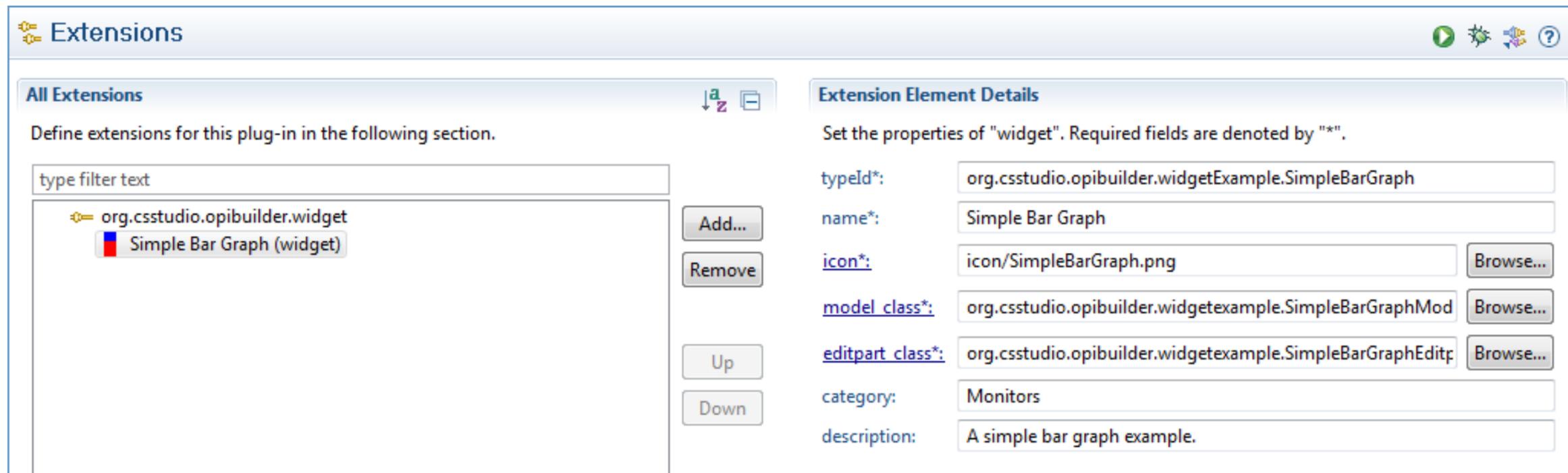
    /**
     * Create and initialize figure.
     */
    @Override
    protected IFigure doCreateFigure() {
        SimpleBarGraphFigure figure = new SimpleBarGraphFigure();
        figure.setMin(getWidgetModel().getMin());
        figure.setMax(getWidgetModel().getMax());
        return figure;
    }

    @Override
    protected void registerPropertyChangeHandlers() {
        // The handler when PV value changed.
        IWidgetPropertyChangeHandler valueHandler = new IWidgetPropertyChangeHandler() {
            public boolean handleChange(final Object oldValue,
                final Object newValue,
                final IFigure figure) {
                if(newValue == null)
                    return false;
                ((SimpleBarGraphFigure) figure).setValue(ValueUtil.getDouble((IValue)newValue));
                return false;
            }
        };
        setPropertyChangeHandler(AbstractPVWidgetModel.PROP_PVVALUE, valueHandler);

        //The handler when max property value changed.
        IWidgetPropertyChangeHandler maxHandler = new IWidgetPropertyChangeHandler() {
            public boolean handleChange(Object oldValue, Object newValue, IFigure figure) {
                ((SimpleBarGraphFigure) figure).setMax((Double)newValue);
                return false;
            }
        };
        setPropertyChangeHandler(SimpleBarGraphModel.PROP_MAX, maxHandler);
        ...
    }
    ...
}
```

Register the widget with BOY

- Use the standard Eclipse extension point mechanism
- Extension point: *org.csstudio.opibuilder.widget*



The screenshot shows the Eclipse 'Extensions' dialog. On the left, under 'All Extensions', a list contains the extension point `org.csstudio.opibuilder.widget` and its associated widget `Simple Bar Graph (widget)`. On the right, the 'Extension Element Details' panel shows the configuration for the widget:

- `typeId*`: `org.csstudio.opibuilder.widgetExample.SimpleBarGraph`
- `name*`: `Simple Bar Graph`
- `icon*`: `icon/SimpleBarGraph.png` (with a 'Browse...' button)
- `model class*`: `org.csstudio.opibuilder.widgetexample.SimpleBarGraphMod` (with a 'Browse...' button)
- `editpart class*`: `org.csstudio.opibuilder.widgetexample.SimpleBarGraphEditp` (with a 'Browse...' button)
- `category`: `Monitors`
- `description`: `A simple bar graph example.`

Integrate the widgets with CSS

- **CSS Developer**
 - Include the widgets plugin into your build
- **CSS User**
 - Export it to a deployable plugin JAR file and copy it to your CSS dropin folder
- **Same as adding a general plugin to an Eclipse RCP**

Connect to PV

- **AbstractPVWidgetEditpart**
 - Handled the connection to PV
 - Handled the disconnection border
 - Handled the alarm sensitive border/foreground/background color
 - Mapped the PV value change event to property change event of Prop_PVValue
- **Set PV value**
 - **AbstractPVWidgetEditpart.setPVValue(PV_PropID, Value)**

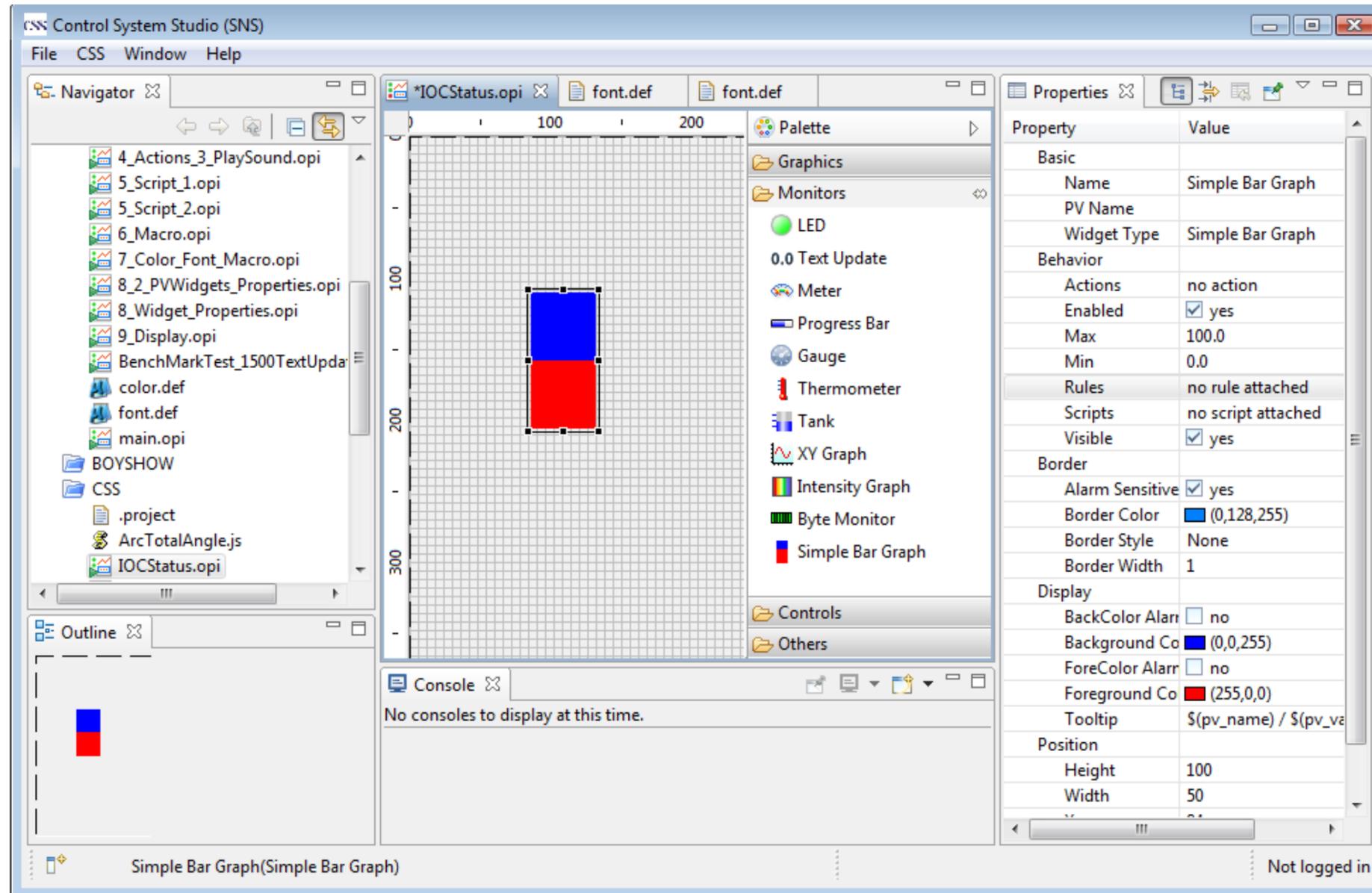
Be aware of resources usage

- **Color, Font, Image and Cursor need to be disposed explicitly**
- **Use `org.csstudio.platform.ui.util.CustomMediaFactory` if you want the resources to be disposed by system automatically.**

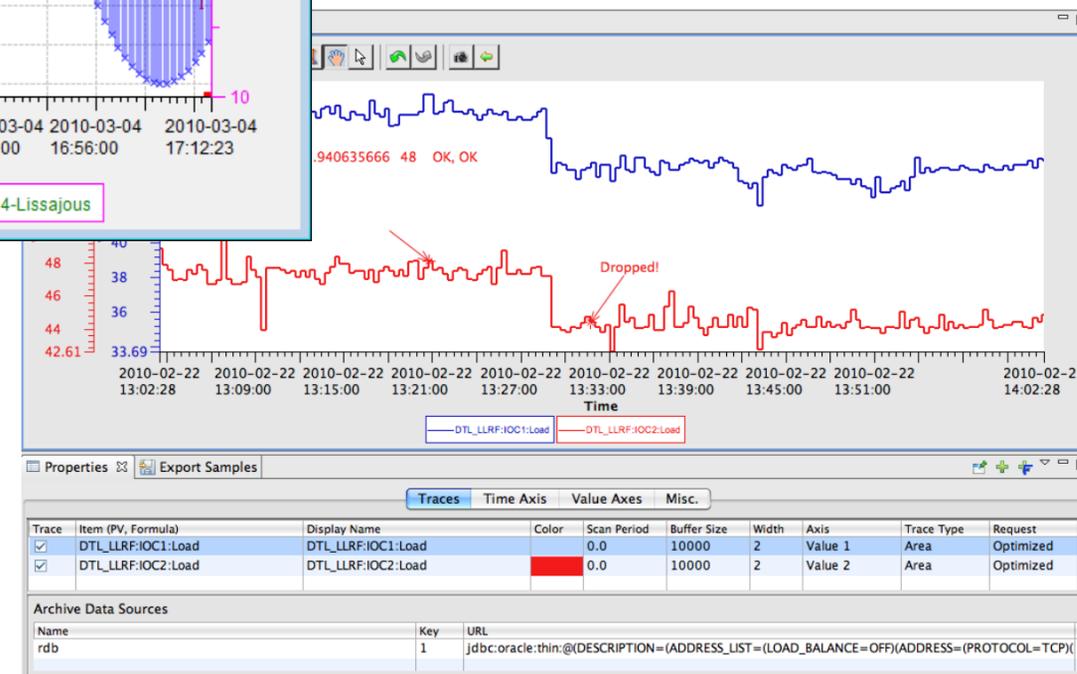
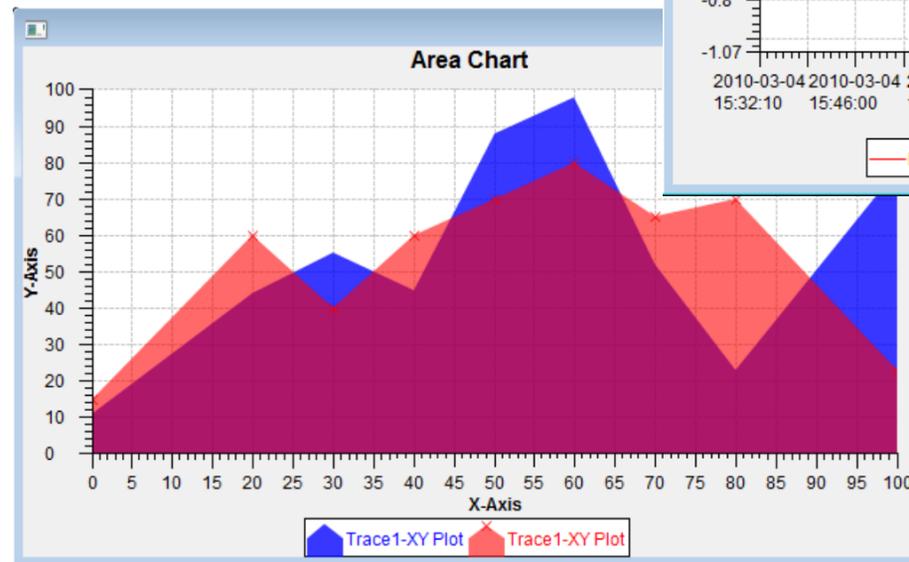
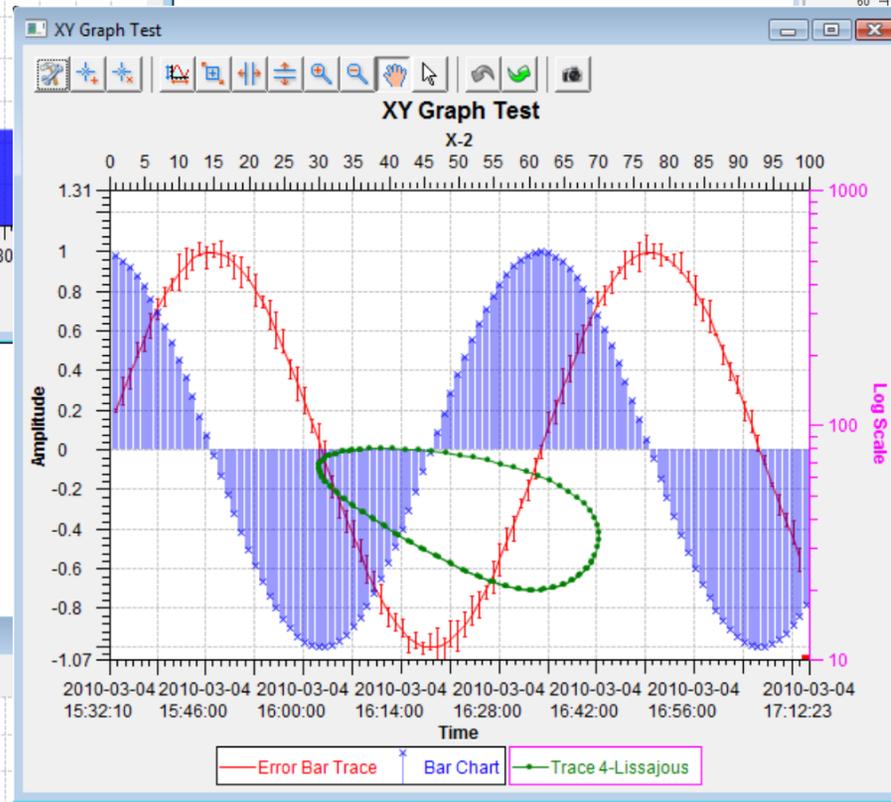
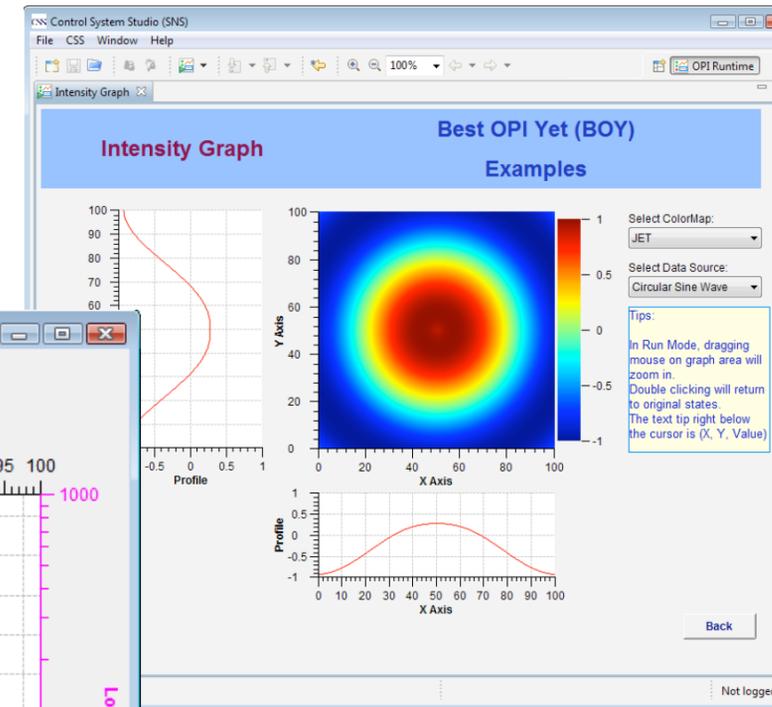
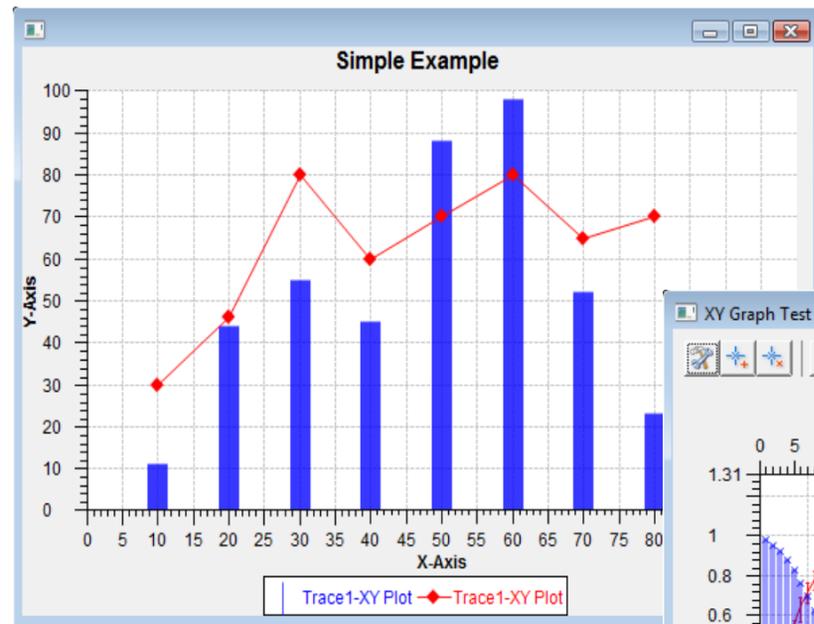
Available Resources

- **org.csstudio.swt.xygraph**
 - Linear Scale
 - SWT XYGraph
- **org.csstudio.swt.widgets**
 - Round Scale
 - Existing Widgets

Simple Bar Graph Widget Demo



SWT XYGraph Library

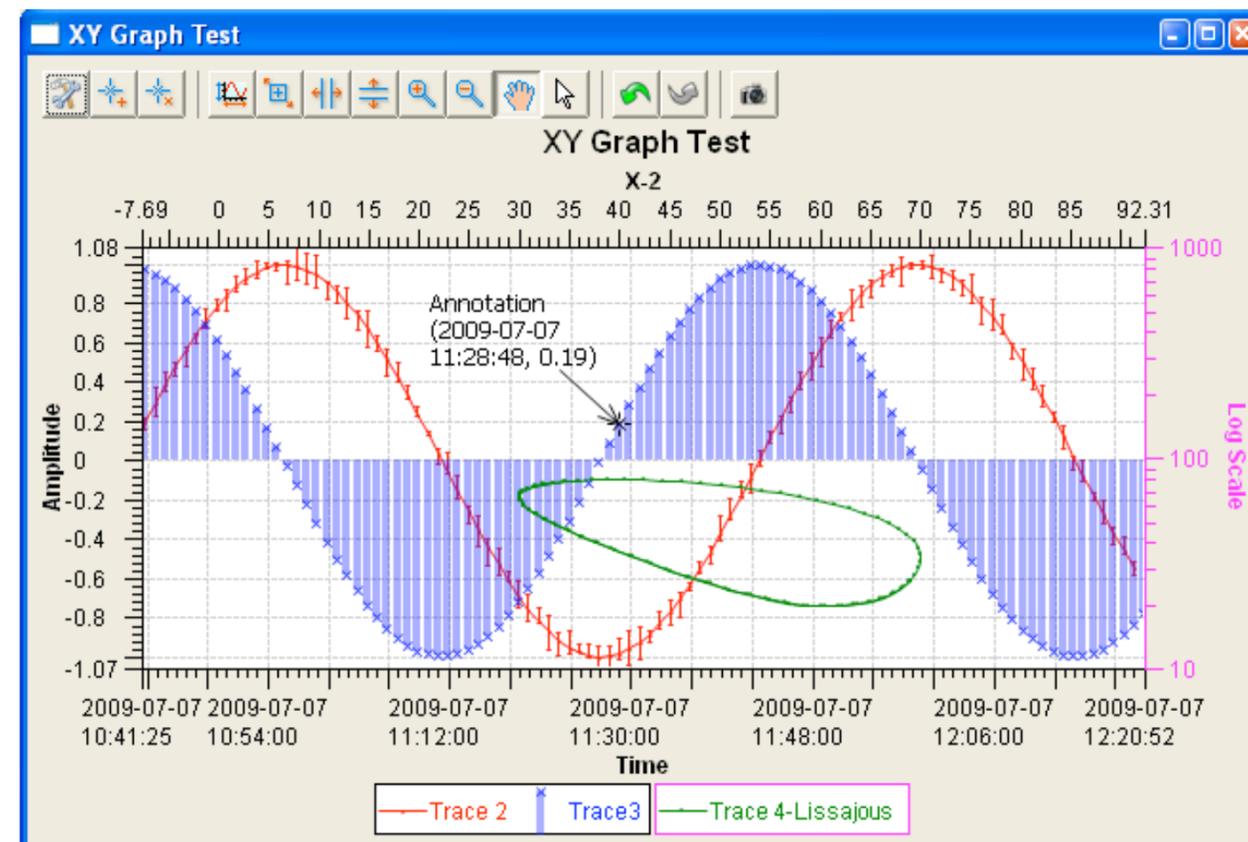


Introduction

- **Based on Draw2D, so it can be used in any SWT or GEF based applications**
- **Good for scientific and engineering 1D or 2D data plotting**
- **Used in BOY and Data Browser**

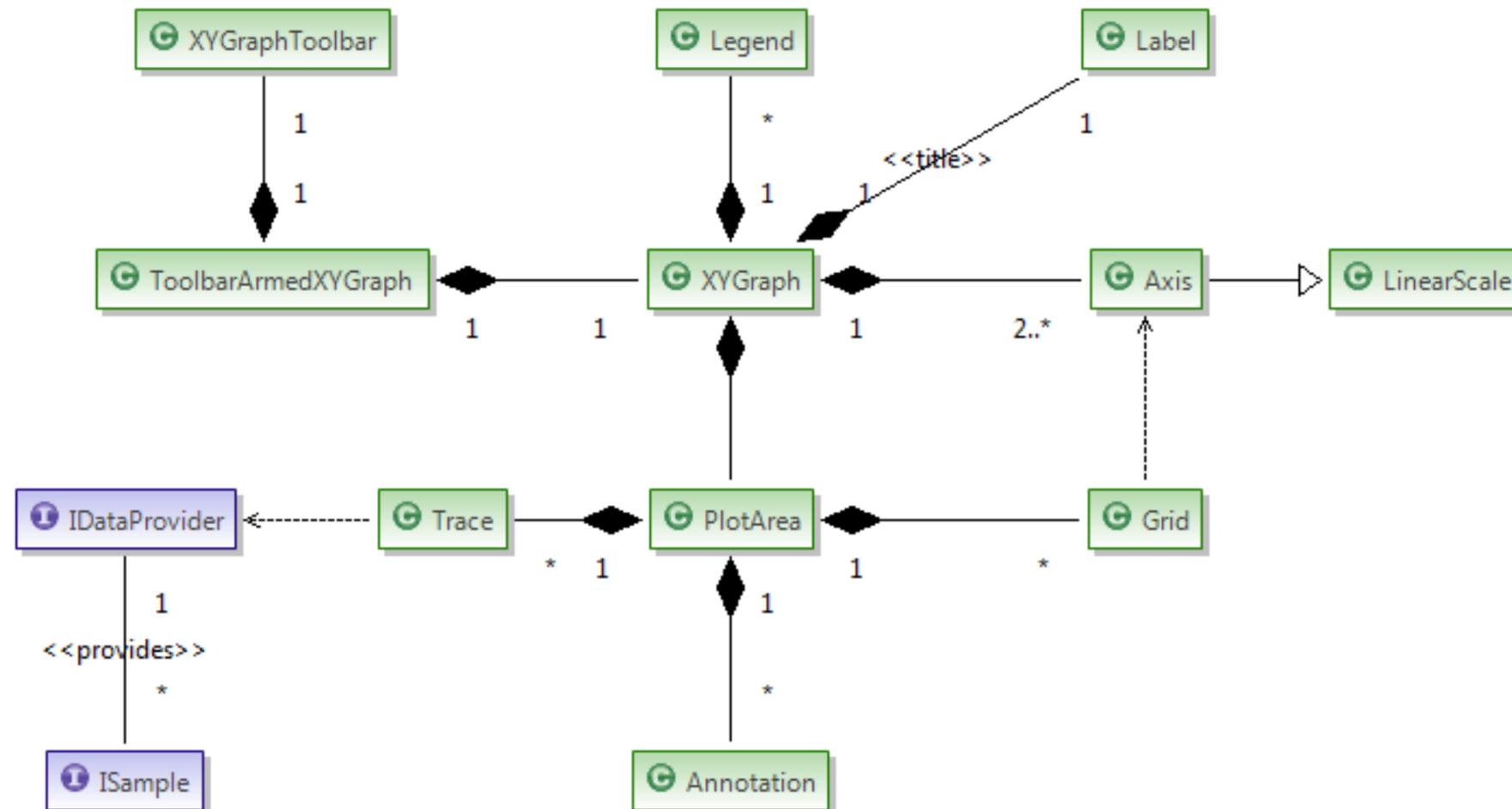
SWT XYGraph Library

- Support line chart, scatter chart, bar chart, step chart, area chart and more...
- Automatically zoom, Rubberband Zoom, Horizontal Zoom, Vertical Zoom
- Panning on both graph area and axes
- Autoscale
- Annotation support
- Multiple axes support
- Log scale, date time format support
- Grouping legends by axes
- Runtime Configuration for everything
- Customized data provider support
- Taking snapshot
- Undo/Redo



Architecture

- **Component Structure**
 - Easy to have arbitrary number of Axes or Traces.
 - Easy to configure the visibility and properties for each component
- **IDataProvider**
 - Allow Customized Data Provider with customized data storage structure or data source



Comparison with JFreeChart

- **JFreeChart**

- **Cannot be used in GEF application**
- **Low performance for real time data plotting**
 - **You have to copy all your data to its predefined data structure for every plotting, which will waste lots of CPU and memory.**
- **Good at static graph because of its comprehensive chart types.**

- **SWT XYGraph**

- **Only support numeric value plots on 2D axes.**
- **Does not support Pie Chart, Gantt Charts, Polar chart...**
- **Can be used in GEF, SWT applications**
- **Armed with comprehensive interactive tools, such as zoom, panning, undo/redo**
- **Good at real time data plots**

A simple XYGraph Example

```
public class SampleView extends ViewPart {

    public SampleView() {
    }

    @Override
    public void createPartControl(Composite parent) {

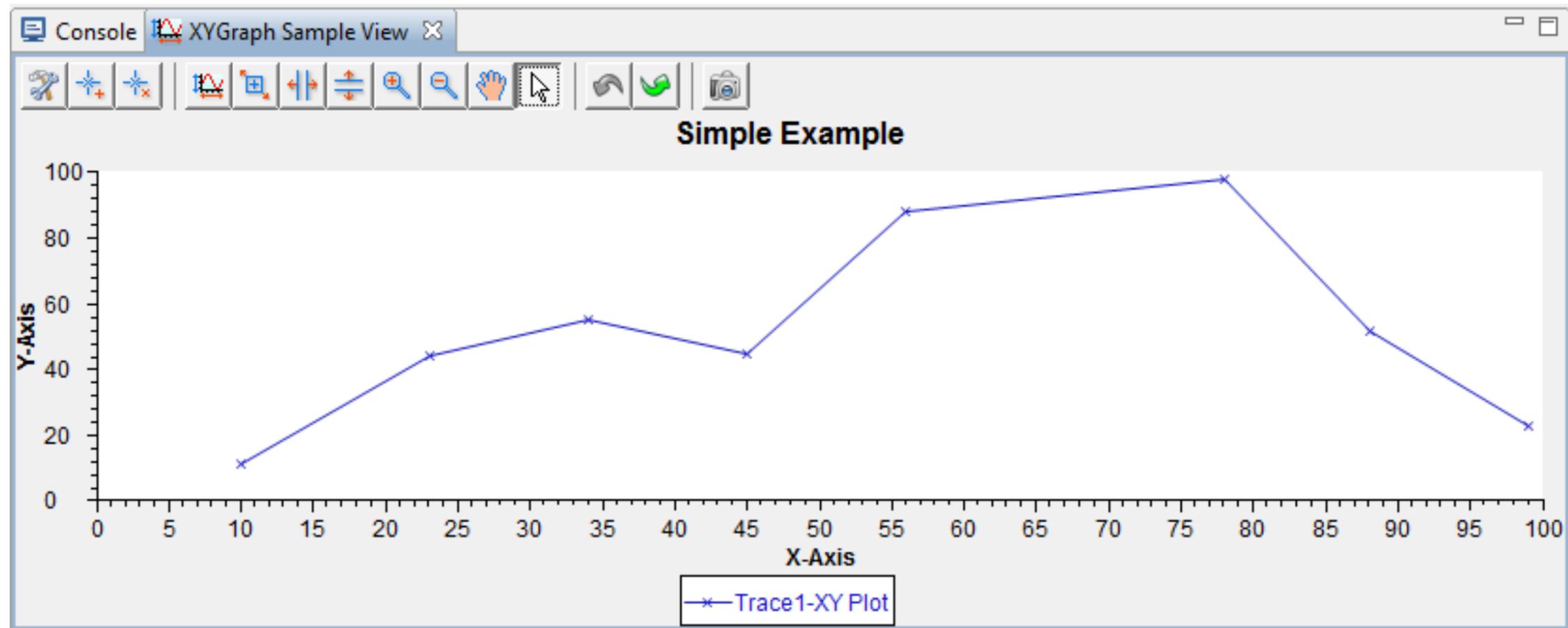
        //use LightweightSystem to create the bridge between SWT and draw2D
        final LightweightSystem lws = new LightweightSystem(new Canvas(parent, SWT.NONE));

        //create a new XY Graph.
        ToolBarArmedXYGraph toolbarArmedXYGraph = new ToolBarArmedXYGraph();
        //set it as the content of LightweightSystem
        lws.setContents(toolbarArmedXYGraph);

        XYGraph xyGraph = toolbarArmedXYGraph.getXYGraph();
        xyGraph.setTitle("Simple Example");
        //create a trace data provider, which will provide the data to the trace.
        CircularBufferDataProvider traceDataProvider = new CircularBufferDataProvider(false);
        traceDataProvider.setBufferSize(100);
        traceDataProvider.setCurrentXdataArray(new double[]{10, 23, 34, 45, 56, 78, 88, 99});
        traceDataProvider.setCurrentYdataArray(new double[]{11, 44, 55, 45, 88, 98, 52, 23});

        //create the trace
        Trace trace = new Trace("Trace1-XY Plot",
        xyGraph.primaryXAxis, xyGraph.primaryYAxis, traceDataProvider);
        //set trace property
        trace.setPointStyle(PointStyle.XCROSS);
        //add the trace to xyGraph
        xyGraph.addTrace(trace);
    }
}
```

That's it!



Summary

- **BOY Widgets**
 - All BOY widgets must follow the MVC pattern
 - Each widget should have at least three classes: Model, Editpart and Figure
 - Using Eclipse extension point to register widgets with BOY
- **XYGraph**
 - It is good at drawing real time numeric data compare to JFreeChart

Thank you!

- See BOY online help **Creating Customized widget** for more details
- XYGraph homepage:
 - <http://code.google.com/p/swt-xy-graph/>