

# Drag and Drop Display and Builder.

Timofei B. Bolshakov,  
Andrey D. Petrov  
FermiLab

# Motivation

- Operations of complicated colliders (Tevatron, the LHC, and the ILC ) require **sophisticated** control systems.
  - Security
  - Data pool management
  - Alarms
  - Logging
- The people who must build, operate, and maintain these accelerators
  - Operators
  - Engineers
  - Accelerator physicistsrequire **rapid development** of control displays and application programs.

# Motivation

- For rapid development, the **system expert** (operator, engineer, or physicist) should be the one to **develop** the displays or applications
- These advanced control systems can seem **overwhelming** to non controls experts.
  - This is why Lab View is so popular
  - However Lab View offers little of the benefits of an advanced control system.

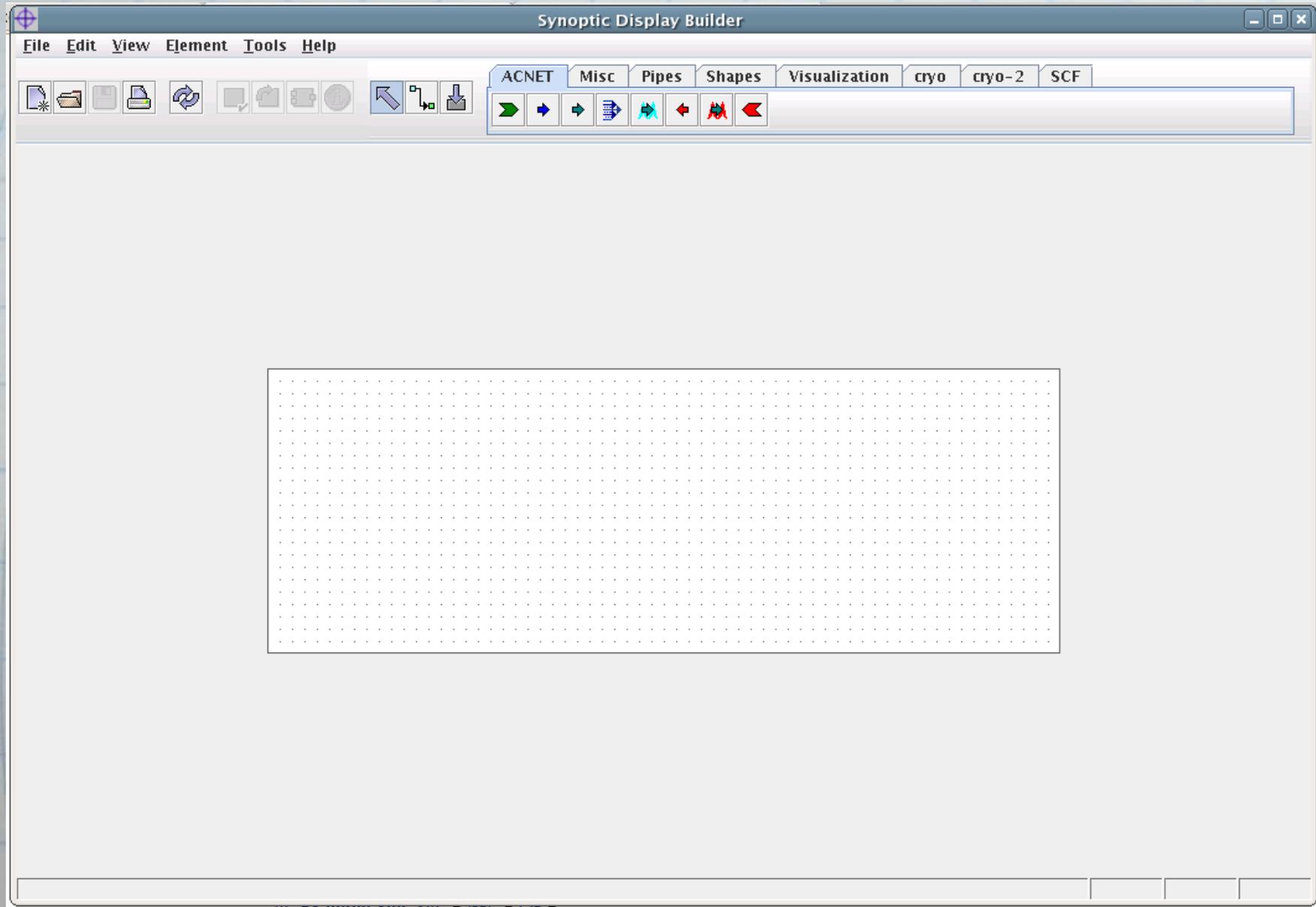
# Solution

- Drag and Drop is an environment that gives non-control system experts the ability to quickly build controls displays which operate within a context of the control system.
- Drag and Drop:
  - is easy to use
  - sophisticated enough to handle complex displays
  - uses web browsers and/or Java Web Start
  - is easily extendible
  - is a mature application
    - First developed in 2001
    - Fermilab Cryogenics department are heavy users

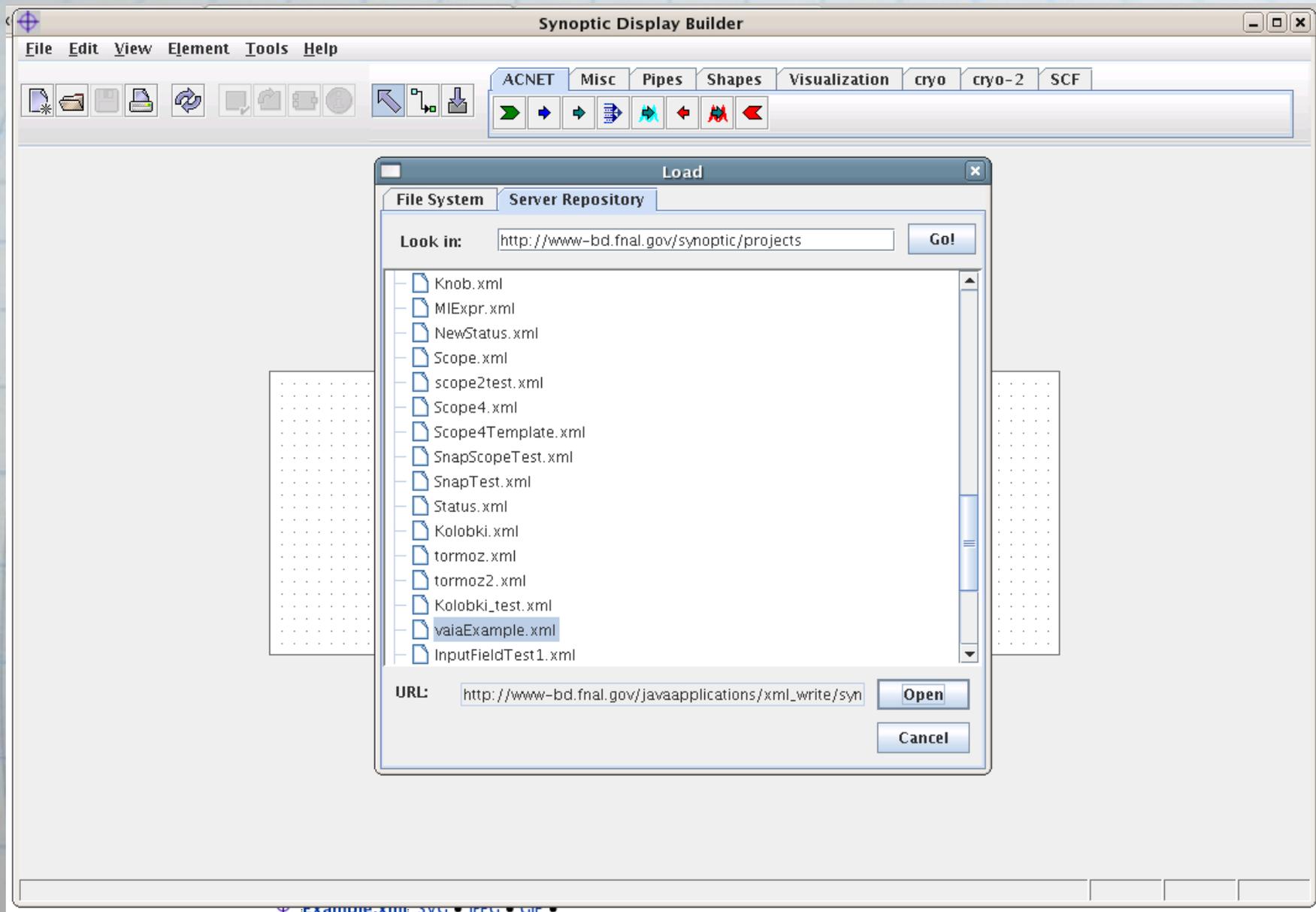
# Drag and Drop Display and Builder

- Drag and Drop a consists of two parts:
  - Display
  - Builder
- The Drag and Drop Display can be run from a web browser (readings only) so it can viewed anywhere in the world.
  - Files are stored on a web server (well organized and secure)
  - Displays are extremely quick because it uses Scalable Vector Graphics (SVG) so that the screen does not constantly have to be re-drawn.
- The Builder has a simple graphical user interface that offers a rich set of graphical components
  - Display can be built and deployed in a matter of minutes
  - The builder is easily extendible
  - can be run on any machine because it is based on Java

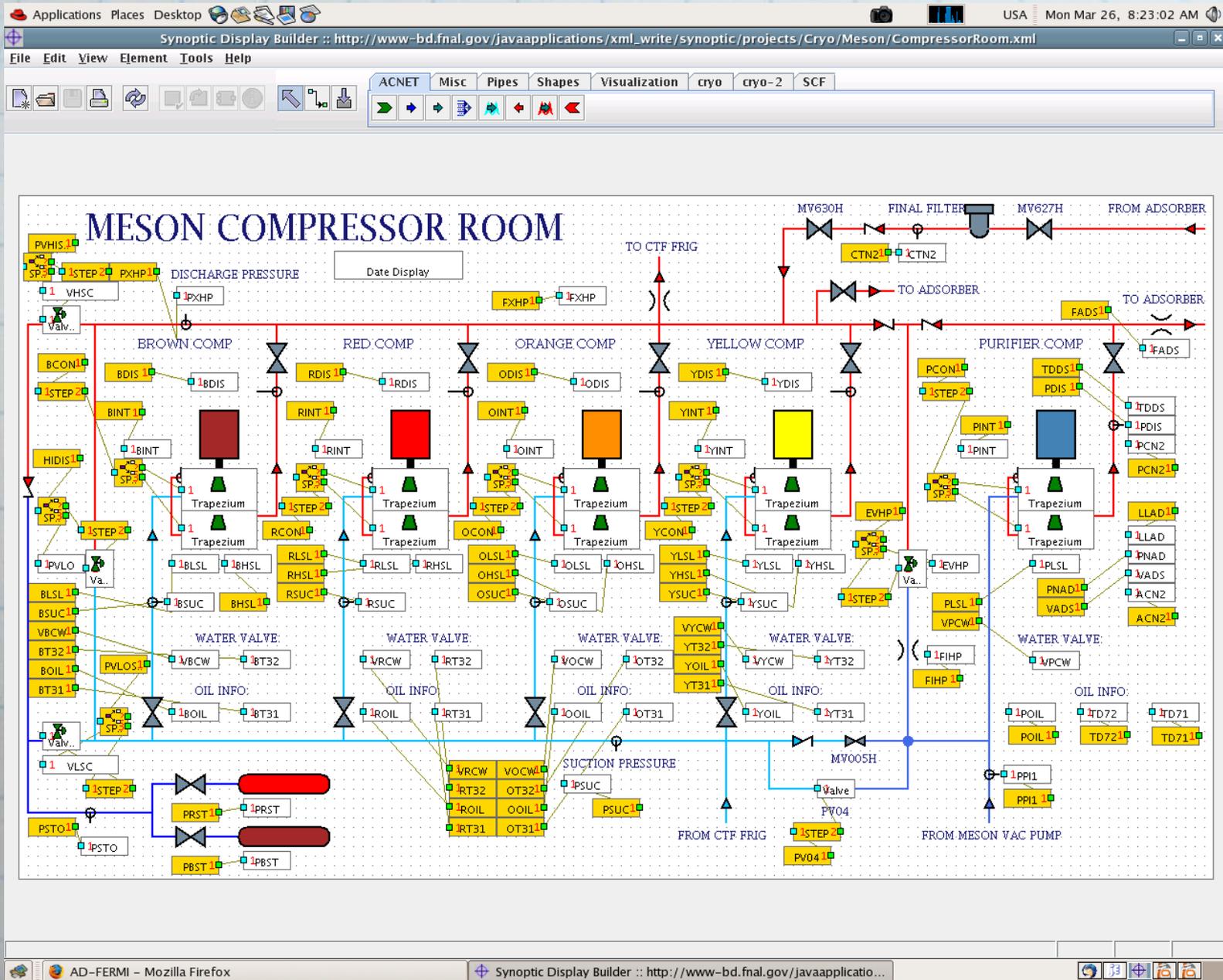
# Demo – Empty Builder



# Demo – Builder, Open Project

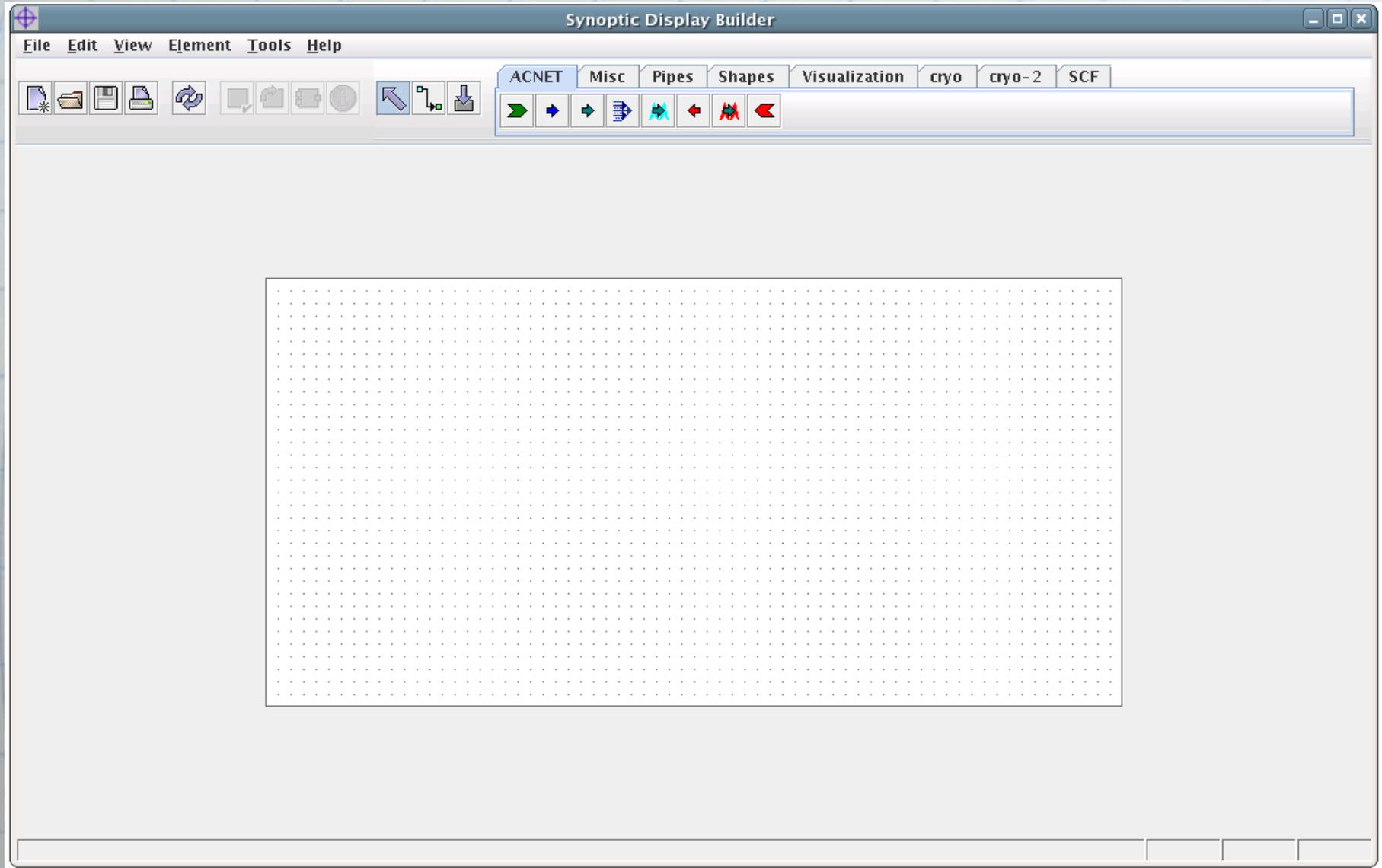


# Demo - Builder, Meson Compressor Room





# Demo – Klystron Step 1, Empty Builder



# Demo – Klystron, Step 2, Project Properties

The screenshot shows the Synoptic Display Builder application window. The main workspace contains two dotted rectangular areas. A dialog box titled "Property of GenericProject" is open, displaying a list of properties and their values. The "Background Color" property is selected, and a color palette is shown below it. The status bar at the bottom indicates "Selection: GenericProject" and "To edit value: press F2 or double click." The coordinates "X: 310 Y: 70" are also visible.

Property	Value
Minimum Width	20
Minimum Height	10
Name	
X	0
Y	0
Width	680
Height	340
minInputs	0
maxInputs	-1
minOutputs	0
maxOutputs	-1
Background Color	

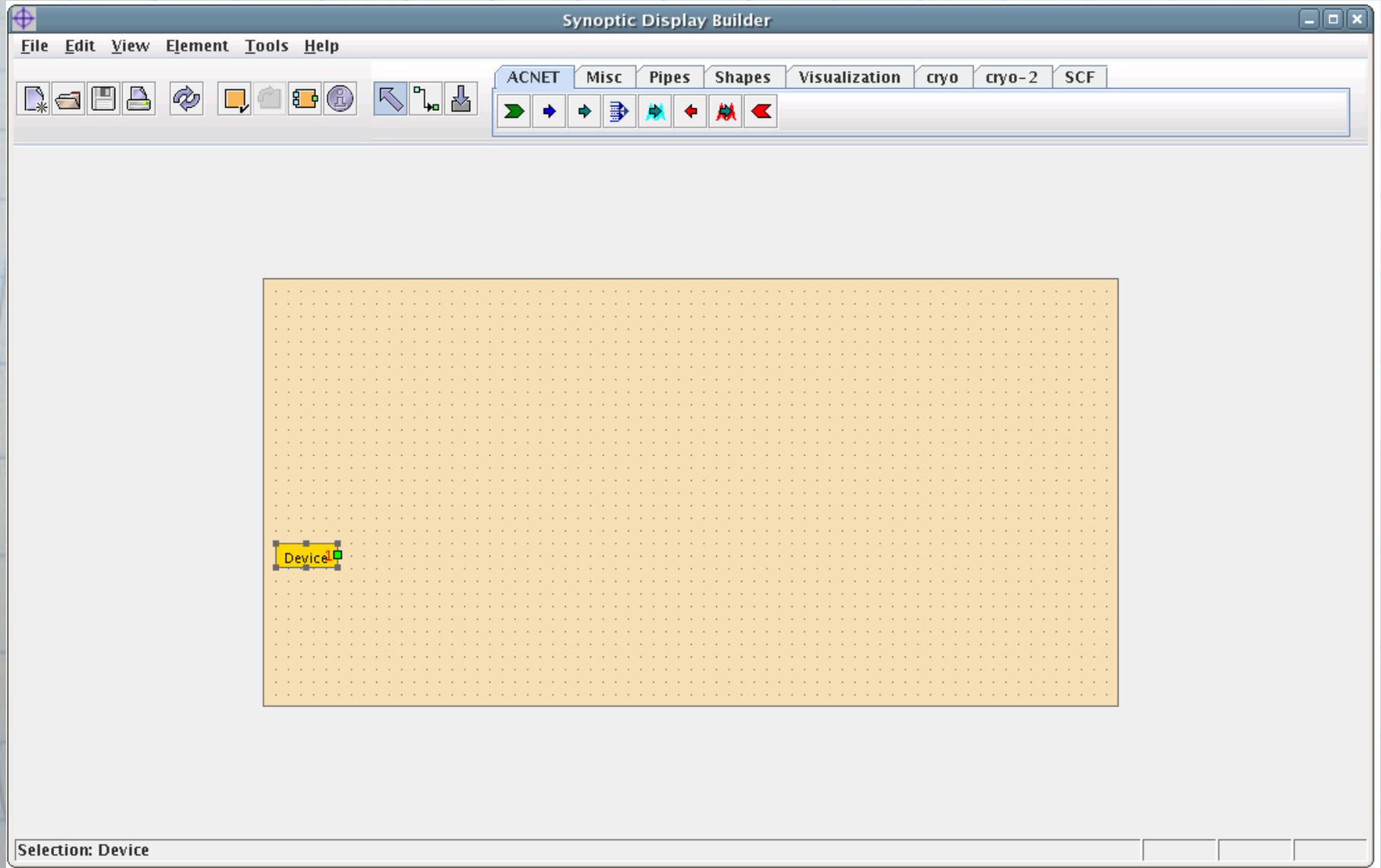
Selection: GenericProject

To edit value: press F2 or double click.

X: 310 Y: 70

OK Cancel

# Demo – Klystron, Step 3, Adding a Device



# Demo – Klystron, Step 4, Setting Up Device

The screenshot shows the Synoptic Display Builder interface. A 'Property of Device' dialog box is open, displaying the following properties:

Property	Value
Min. Width	20
Min. Height	10
Name	Device
X	10
Y	210
Width	50
Height	20
Min. Inputs	0
Max. Inputs	0
Min. Outputs	1
Max. Outputs	1
Background	gold
Device Name	L:C6GRAD,Reading
Event	periodic, 1Hz
Source	Accelerator
Reflect changes only	true

The background of the software shows a grid with a yellow dotted area. A small label 'Device' is visible on the grid. The status bar at the bottom indicates 'Selection: Device' and 'X: 40 Y: 220'. The dialog box has 'Ok' and 'Cancel' buttons at the bottom.

# Demo – Klystron, Step 5, Adding Indicator

**Synoptic Display Builder**

File Edit View Element Tools Help

Property of Arrow Device

Property	Value
Min. Width	80
Min. Height	80
Name	Arrow Device
X	20
Y	10
Width	130
Height	120
Min. Inputs	1
Max. Inputs	1
Min. Outputs	0
Max. Outputs	1
Background	sandybrown
Value Format	##0.##
Date Format	hh:mm:ss
Minimum	0.0
Maximum	1.6
Alarm Minimum	0.0
Alarm Maximum	2.0
Step	0.2
Scale Format	##0.#
Min and Max Format	##0.#
Scale Color	black
Arrow Color	black
Fill Color	blue
Text Color	black
Alarm Color	red

To edit value: press F2 or double click.

Ok Cancel

Selection: Arrow Device X: 110 Y: 40

# Demo – Klystron, Step 6, Adding Graphics

The screenshot shows the Synoptic Display Builder software interface. The main window contains a diagram on a grid background. The diagram includes an orange square labeled "Arrow Device" with a clock icon inside, a blue rectangle labeled "Numeric Display", and a red rectangle. Two yellow boxes labeled "Device" are positioned near the orange square and the red rectangle. A vertical line with arrows at both ends connects the top of the red rectangle to the bottom of the orange square. A "Property of Shape" dialog box is open on the right, showing a color palette for the selected shape. The dialog box has a table with "Property" and "Value" columns. The "Fill Color" property is selected, and the color palette is displayed. The "Stroke Color" and "Stroke Width" properties are also visible but not selected. The status bar at the bottom of the main window shows "Selection: Shape" and "X: 210 Y: 110".

Synoptic Display Builder

File Edit View Element Tools Help

ACNET Misc Pipes Shape

Property of Shape

Property	Value
Fill Color	[Color Palette]
Stroke Color	[Color Palette]
Stroke Width	[Color Palette]

Selection: Shape

To edit value: press F2 or double click. X: 210 Y: 110

OK Cancel

# Demo – Klystron, Step 7, Adding Graphics

The screenshot shows the Synoptic Display Builder software interface. The main window contains a diagram on a grid background. The diagram includes several components: an orange square labeled 'Arrow Device' with a yellow circle inside; a yellow vertical rectangle; a red horizontal rectangle at the bottom; a blue horizontal rectangle labeled 'Numeric Display' connected to the red rectangle; and a yellow circle on the right. An orange line connects the top of the yellow vertical rectangle to the yellow circle on the right. A 'Property of Shape' dialog box is open over the diagram, showing the 'Stroke Color' property set to 'black' and a color palette.

**Synoptic Display Builder**

File Edit View Element Tools Help

ACNET Misc Pipes Shapes Visualization cryo cryo-2 SCF

Property of Shape

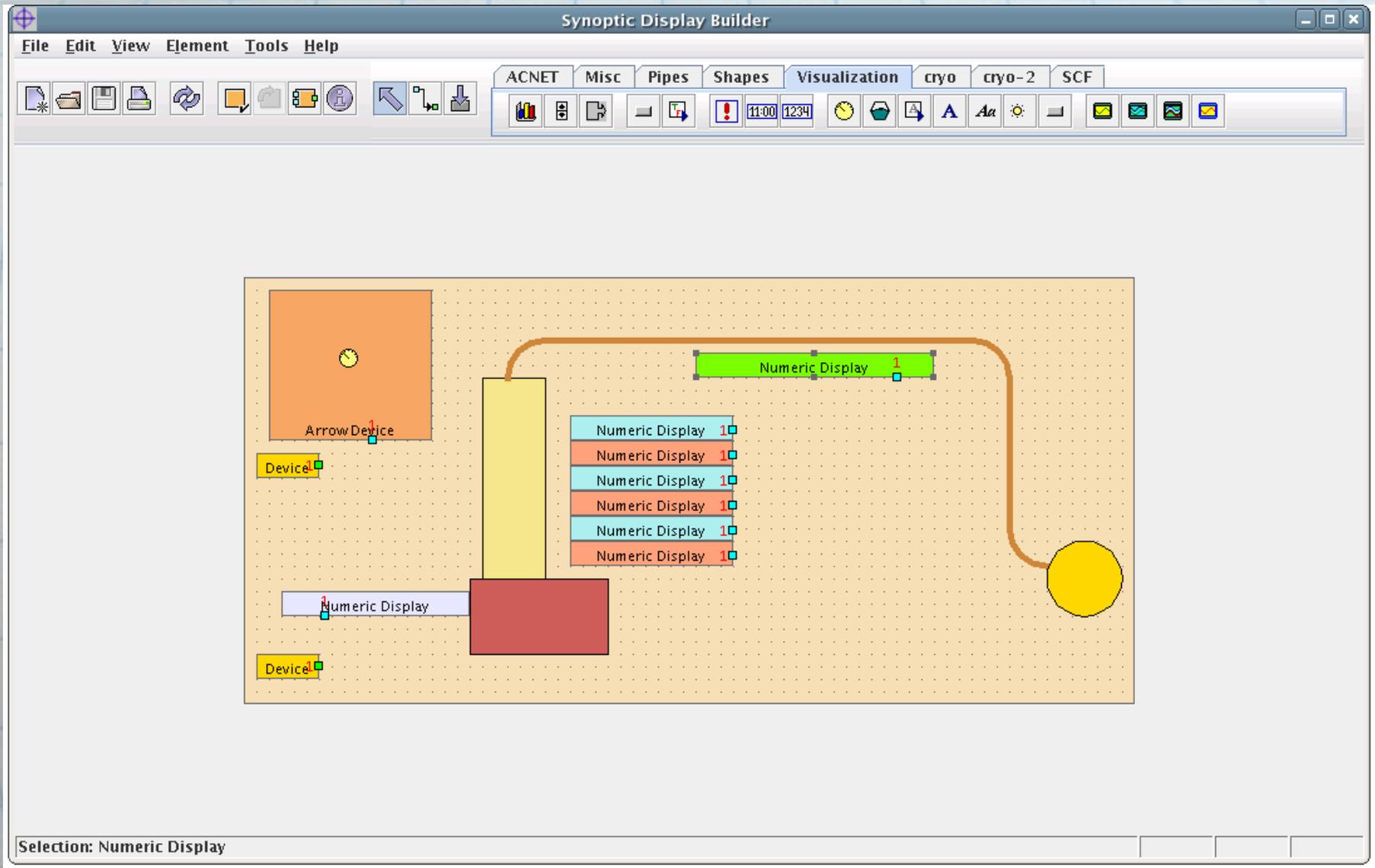
Property	Value
Stroke Color	black
Stroke Width	

To edit value: press

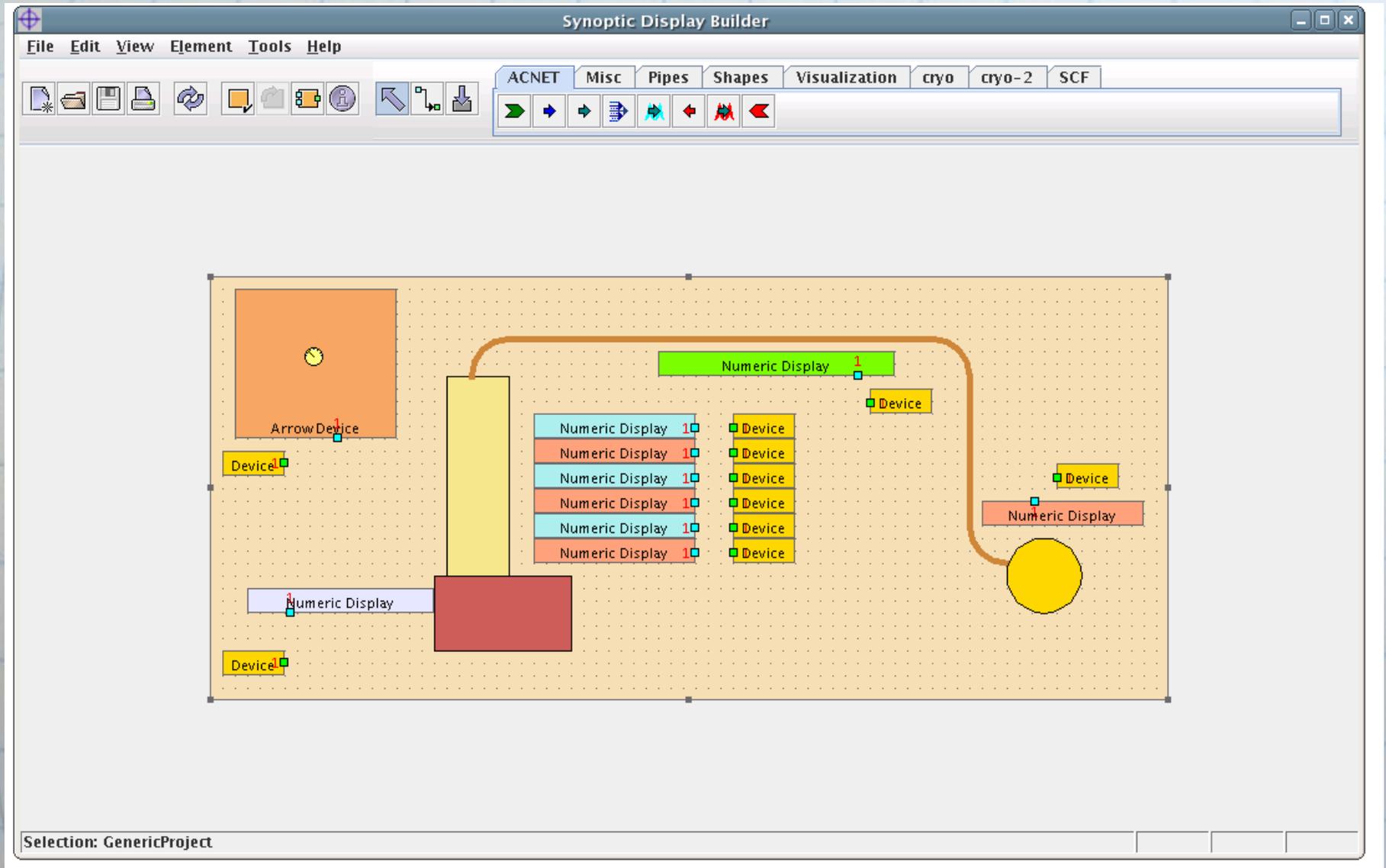
OK

Selection: Shape X: 630 Y: 220

# Demo - Klystron, Step 8, More Components



# Demo – Klystron, Step 9, More Devices



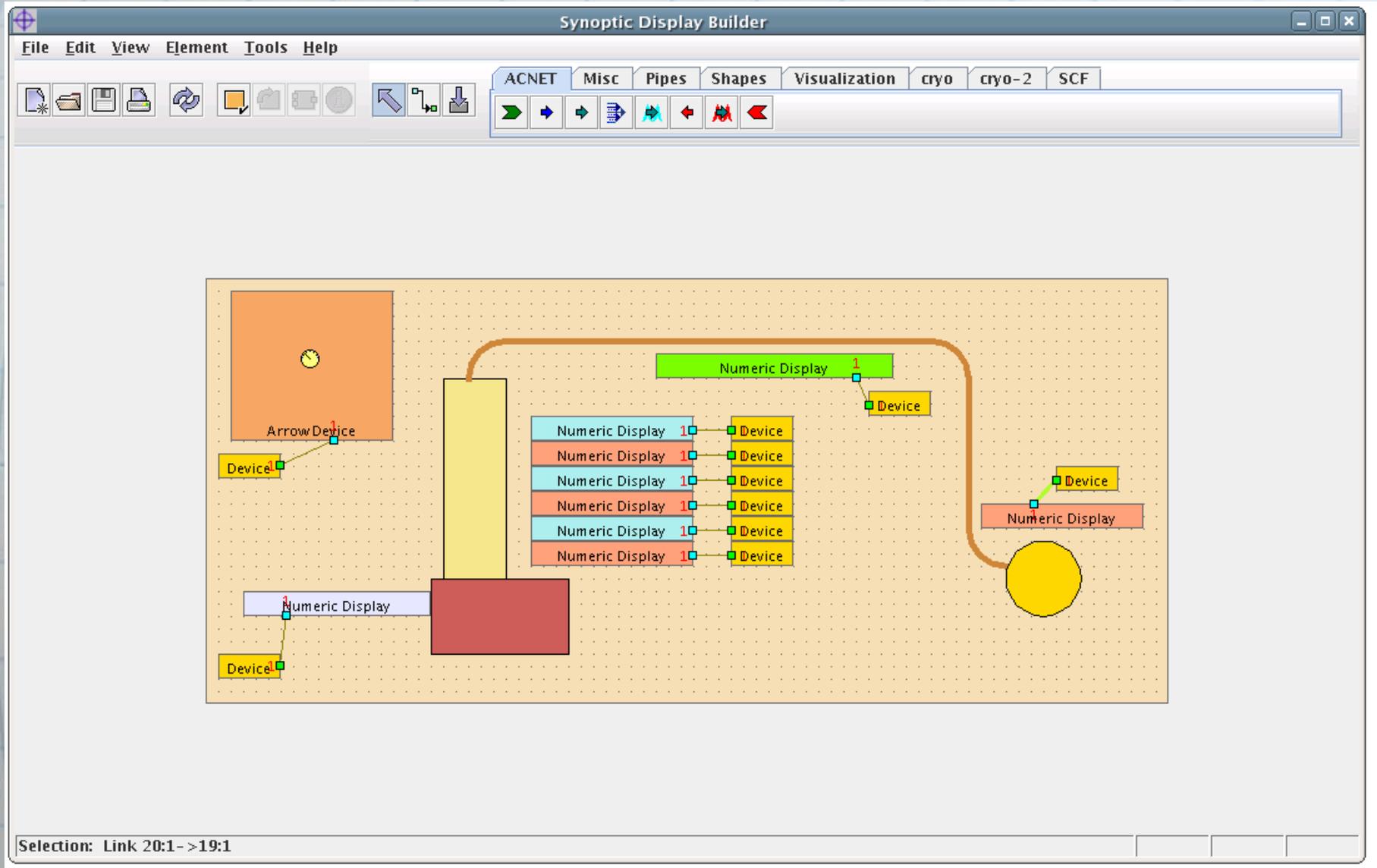
# Demo – Klystron, Step 10, Setting Up Device

The screenshot shows the Synoptic Display Builder interface. The main workspace contains a diagram with several components: an orange square labeled 'ArrowDevice', a yellow vertical bar, a red square, a blue 'Numeric Display' box, a green 'Numeric Display 1' box, and a vertical stack of six 'Numeric Display' boxes (alternating cyan and orange). A yellow 'Device' box is connected to the green 'Numeric Display 1' box. The 'Property of Device' dialog is open, showing the following table:

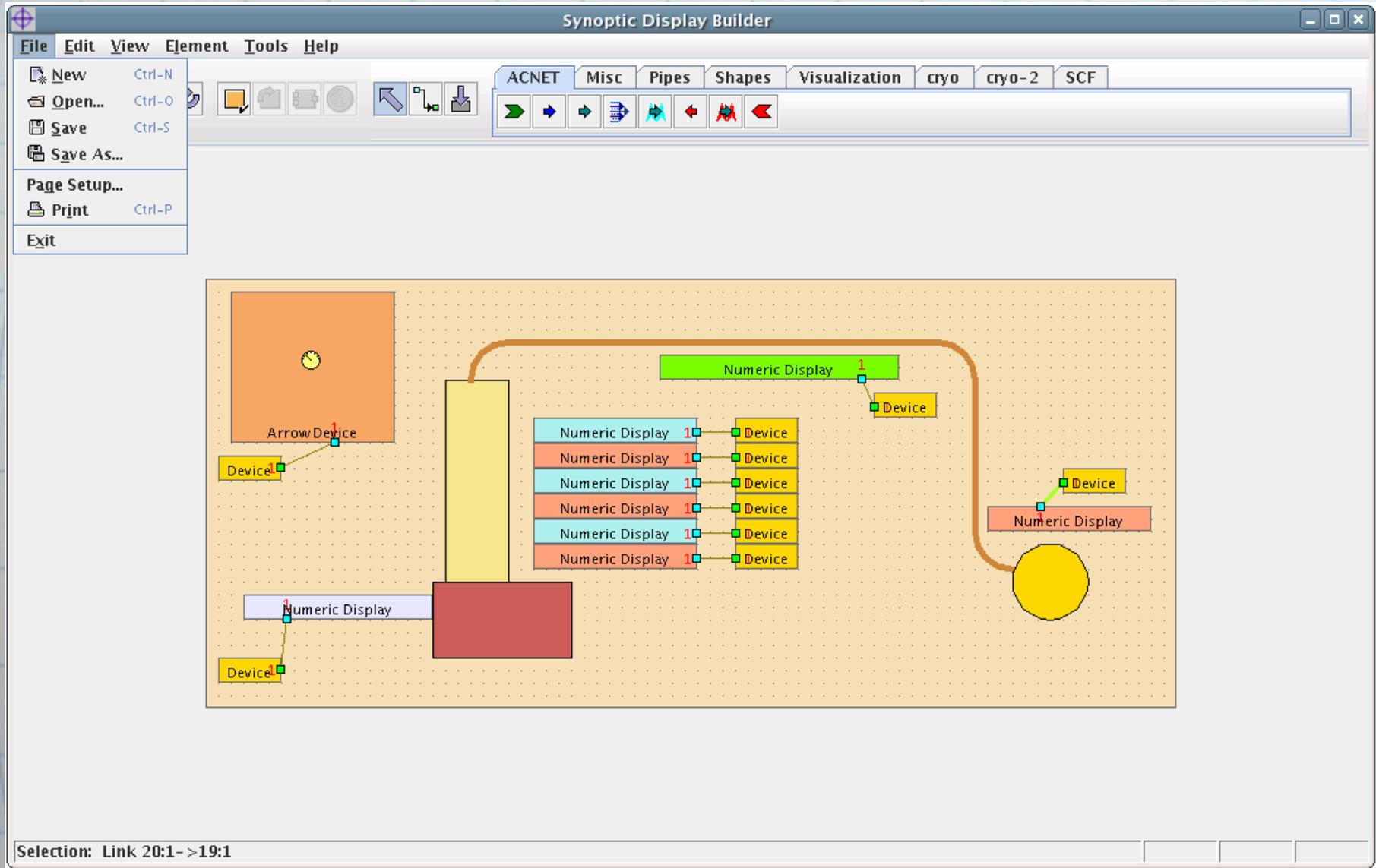
Property	Value
Min. Width	20
Min. Height	10
Name	Device
X	420
Y	110
Width	50
Height	20
Min. Inputs	0
Max. Inputs	0
Min. Outputs	1
Max. Outputs	1
Background	gold
Device Name	L:K6SOL1_Reading
Event	periodic, 1Hz
Source	Accelerator
Reflect chages only	true

At the bottom of the dialog, it says "To edit value: press F2 or double click." and has "OK" and "Cancel" buttons. The status bar at the bottom of the software window shows "Selection: Device" and coordinates "X: 440 Y: 110".

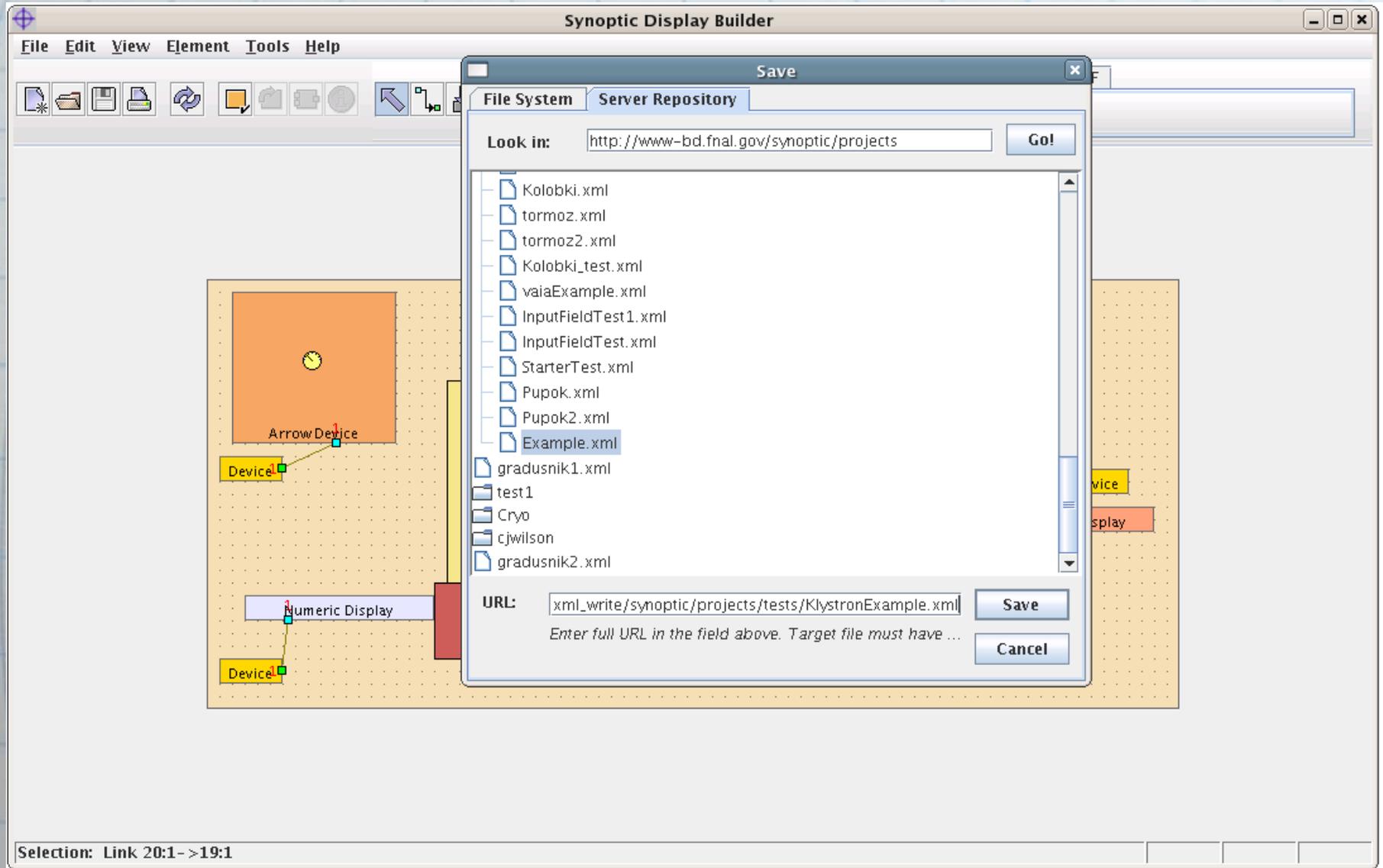
# Demo – Klystron, Step 11, Connecting



# Demo – Klystron, Step 12, Saving Project



# Demo – Klystron, Step 13, Saving Project 2



# Demo – Klystron, Step 14, Projects Repository

The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://www-bd.fnal.gov/synoptic/tree.html`. The browser's address bar also shows a search engine set to Google. The page content displays a file tree structure for the AD-FERMI project. The tree is expanded to show the 'tests' directory, which contains a list of XML files. The file 'KlystronExample.xml' is highlighted with a red circle. The browser's status bar at the bottom shows 'Done' and the system tray includes a clock showing 0.322s and an Adblock extension.

Applications Places Desktop

AD-FERMI – Mozilla Firefox

File Edit View History Bookmarks Tools Help

`http://www-bd.fnal.gov/synoptic/tree.html` Google

Red Hat, Inc. Red Hat Network Support Shop Products Training Current gov/fnal/controls/ap...

Gmail – Входящие (8) AD-FERMI AD-FERMI

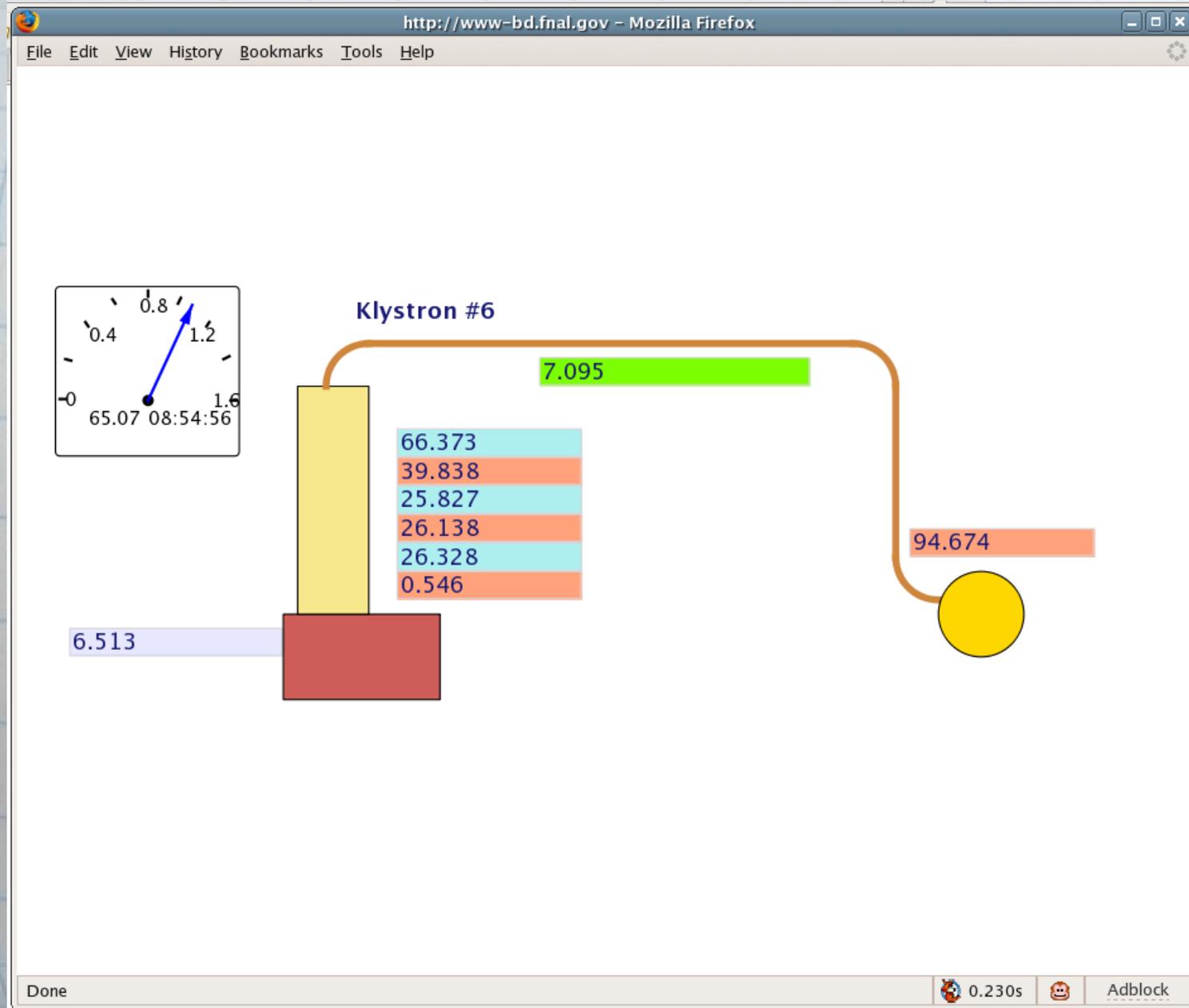
- SlowPlots
- test
- tests
  - BooleanDiscriminator.xml SVG • JPEG • GIF •
  - Button.xml SVG • JPEG • GIF •
  - CompDeviceTest.xml SVG • JPEG • GIF •
  - delay.xml SVG • JPEG • GIF •
  - FB.xml SVG • JPEG • GIF •
  - FSMPanelTest.xml SVG • JPEG • GIF •
  - FSMTest.xml SVG • JPEG • GIF •
  - FTPHistTest.xml SVG • JPEG • GIF •
  - FTPTest.xml SVG • JPEG • GIF •
  - HistogrammTest.xml SVG • JPEG • GIF •
  - JustATest.xml SVG • JPEG • GIF •
  - Knob.xml SVG • JPEG • GIF •
  - MIExpr.xml SVG • JPEG • GIF •
  - NewStatus.xml SVG • JPEG • GIF •
  - Scope.xml SVG • JPEG • GIF •
  - scope2test.xml SVG • JPEG • GIF •
  - Scope4.xml SVG • JPEG • GIF •
  - Scope4Template.xml SVG • JPEG • GIF •
  - SnapScopeTest.xml SVG • JPEG • GIF •
  - SnapTest.xml SVG • JPEG • GIF •
  - Status.xml SVG • JPEG • GIF •
  - Kolobki.xml SVG • JPEG • GIF •
  - tormoz.xml SVG • JPEG • GIF •
  - tormoz2.xml SVG • JPEG • GIF •
  - Kolobki\_test.xml SVG • JPEG • GIF •
  - vaiaExample.xml SVG • JPEG • GIF •
  - InputFieldTest1.xml SVG • JPEG • GIF •
  - InputFieldTest.xml SVG • JPEG • GIF •
  - StarterTest.xml SVG • JPEG • GIF •
  - Pupok.xml SVG • JPEG • GIF •
  - Pupok2.xml SVG • JPEG • GIF •
  - Example.xml SVG • JPEG • GIF •
  - KlystronExample.xml SVG • JPEG • GIF •**
- test1
- Cryo
- cjwilson
  - composite.xml SVG • JPEG • GIF •
  - D11.xml SVG • JPEG • GIF •
  - gradusnik.xml SVG • JPEG • GIF •
  - gradusnik\_1.xml SVG • JPEG • GIF •
  - gradusnik\_2.xml SVG • JPEG • GIF •
  - internal.xml SVG • JPEG • GIF •

Done

0.322s Adblock

AD-FERMI – Mozilla Firefox [Synoptic Display Builder :: `http://www-bd.fnal.gov/javaappli...`]

# Demo – Klystron, Step 15, Display



# Architecture: Components-based.

- Software component is a system element offering a predefined service and able to communicate with other components.
- They do not share state and communicate by exchanging messages carrying data.
- Criteria:
  - Multiple-use
  - Non-context-specific
  - Composable with other components
  - Encapsulated i.e., non-investigable through its interfaces
  - A unit of independent deployment and versioning

# Architecture: Components-based.

A simpler definition can be: *A component is an object written to a specification.*

```
import org.w3c.dom.Element;  
import java.util.concurrent.BlockingQueue;  
public interface RuntimeComponent extends Runnable{  
    public void init(Element root);  
    public void start();  
    public void stop();  
    public void setSink (  
        int i,  
        BlockingQueue <TimedData>q  
    );  
    public void setSource (  
        int i,  
        BlockingQueue <TimedData>q  
    );  
}
```

# Philosophy: Components libraries.

Each component has only one function (e.g., simple and specialized).

Either **visualization** or data **acquisition**.

TimedData travels via **connecting pipes**.

Similar components are grouped into libraries:

Controls system interface (ACNET) : reading and writing data from hardware.

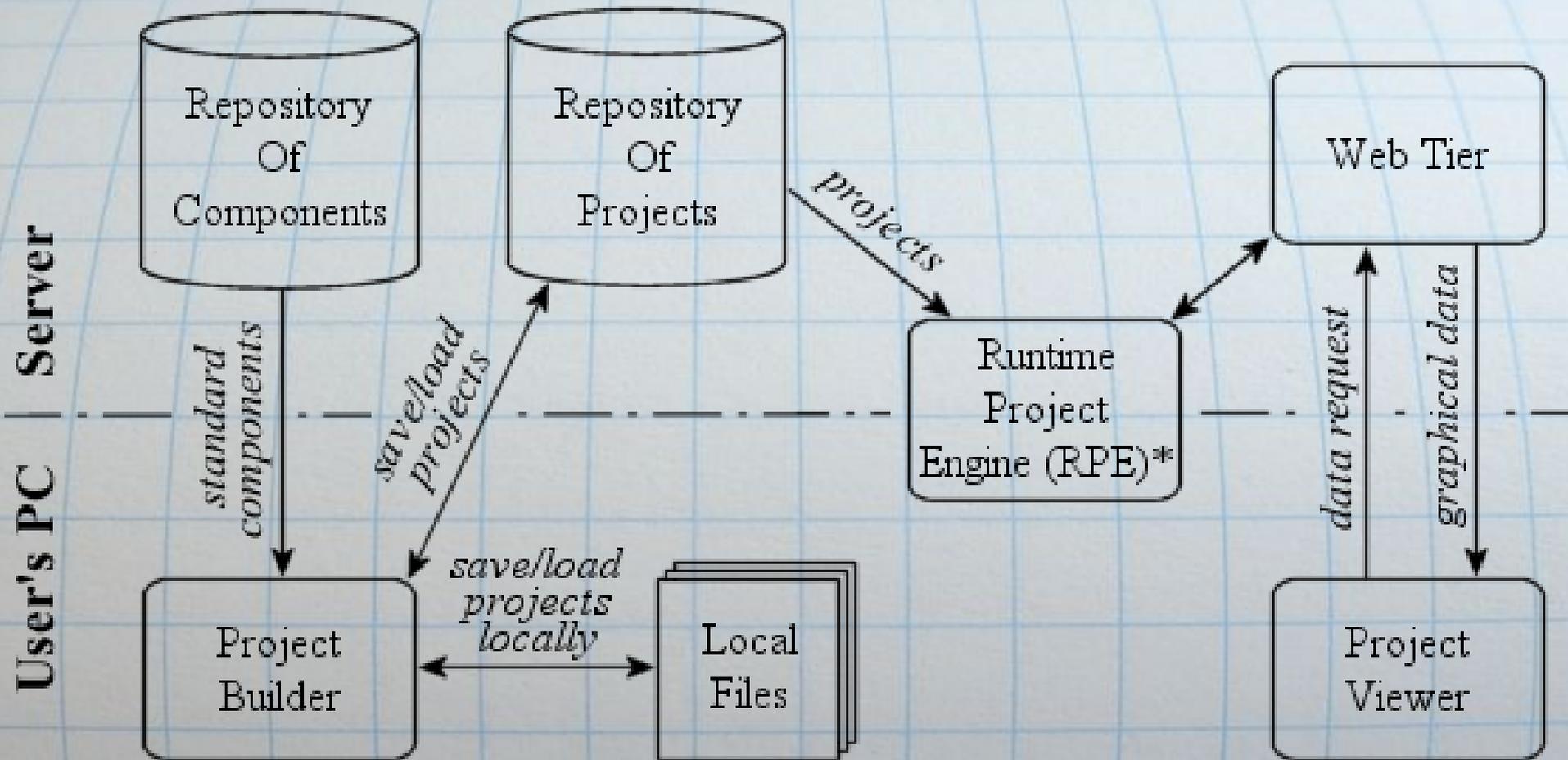
Visualization – formatted number, barrels with liquid level, plots, histograms, oscilloscopes, etc.

Data entry – input field and slider.

Static components – arcs, bars, specialized cryogenic symbols etc.

Processing pipes – components that process data and send it to visualization components.

# Architecture



\* RPE may be started either on server side or locally

# Architecture

## Project Builder

is a special-purpose graphical editor that allows users to define logical flows of information from data sources to data consumers through data handlers and pipes.

## Repositories of Components and Projects

## Runtime Project Engine

downloads project XML files from the repository and starts project as Java application.

## Web Tier and Project Viewer

Upon the first user request, Web Tier sends the full SVG image to the client. On all subsequent requests, Web Tier sends just a difference between current image and the previous one.

# Why redesign ?

Any Java application written in 2001 should be refactored to accommodate new language features and new standard libraries.

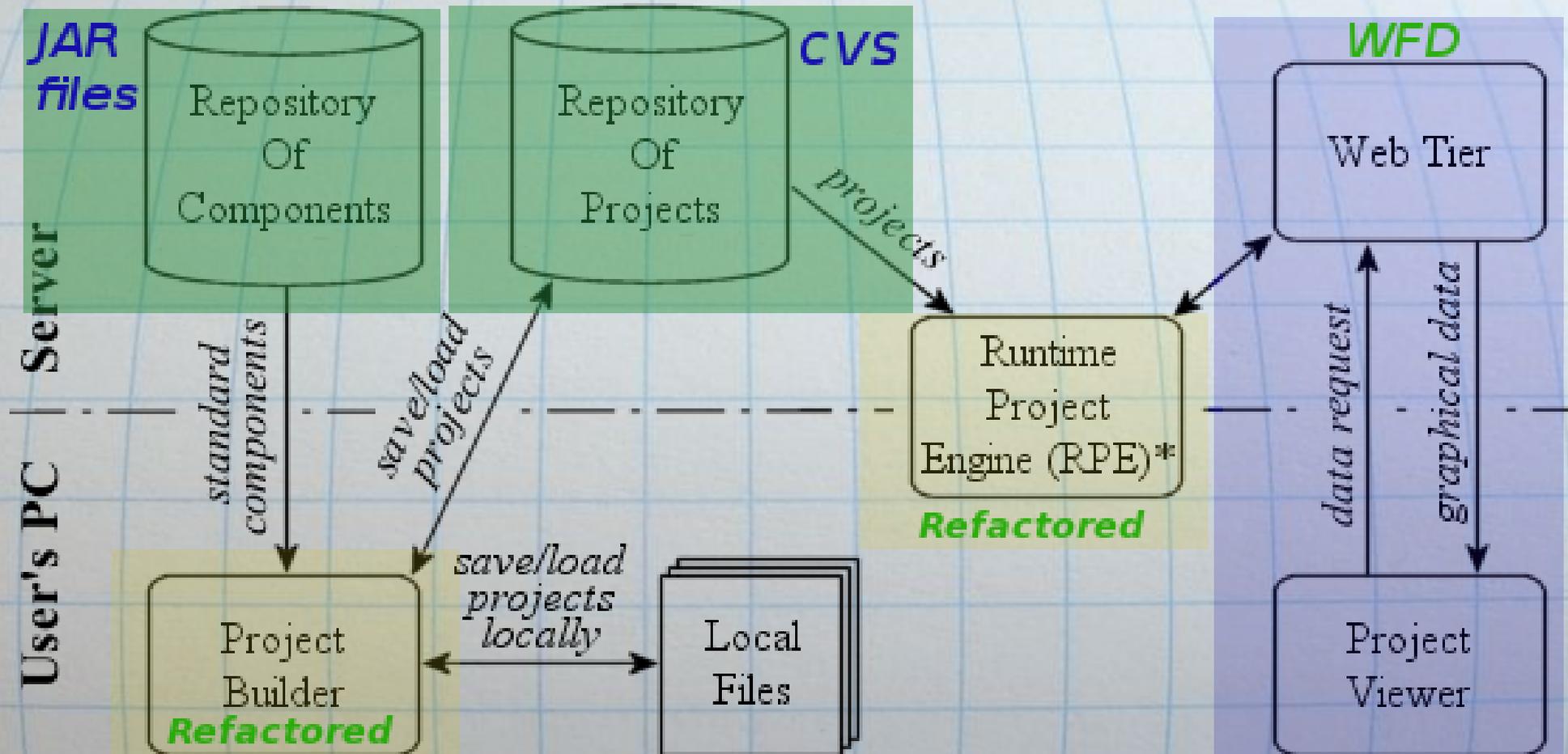
**LAFS** (LHC At Fermilab Software) group was formed in Fermilab in Autumn 2006.

**LAFS** Goal - share experience & software with CERN and learn from new CERN control system.

Drag and Drop web tier was separated into independent project and implemented.

**Requirements for new version of Drag and Drop Display and Builder are discussed right now.**

# Changes in Architecture



\* RPE may be started either on server side or locally

# WFD - Web Fixed Displays.

The web-tier was refactored and called Web Fixed Display (WFD).

CERN developers reviewed and modified Requirements and controlled quality of the implementation.

New features:

- WFD shows ANY Java Swing- or AWT-based application on the web and works with multiple frames.

- WFD allows every application to have a separate classpath.

- WFD has a live index page.

- Application is described in a property file.



# WFD – How does it work.

WFD has 2 major parts – server side Java web application and client side AJAX Javascript.

Server side ApplicationManager (AM) starts preconfigured applications and produces Scalable Vector Graphics (SVG) images of these applications by periodical rendering them on special SVGGraphics2D.

SVG is a convenient graphical format – it is an XML W3C standard. Text in SVG:

```
<text class="cls33" id="849" x="2" y="14">
```

```
    ACN2: 0.7 PPM
```

```
</text>
```

```
<path class="cls11" id="863"
```

```
    d="M879 114 L854 94 L854 114 L879 94 zM879 114 "/>
```

# WFD – How does it work.

Client side AJAX script downloads SVG image of application once.

After that it starts to request the differences from the server. Difference comes in following format:

```
<changes id="849" content="ACN2: 0.6 PPM"/>
```

```
<changes id="863" attr="d"
```

```
content="M879 114 L854 94 L854 114 L879 94 L880 102 z"/>
```

And AJAX script changes SVG elements on client side using DOM API.

As you may see it is very economical way of updating live graphical web page! With just hundreds bytes per second you got rich updating picture!

# Acknowledgments

To David McGinnes  
for managing LAFS workgroup and  
encouraging DnD redesign.

To Jakub Wozniak  
for working from CERN side on WFD.

Thank you for your attention.