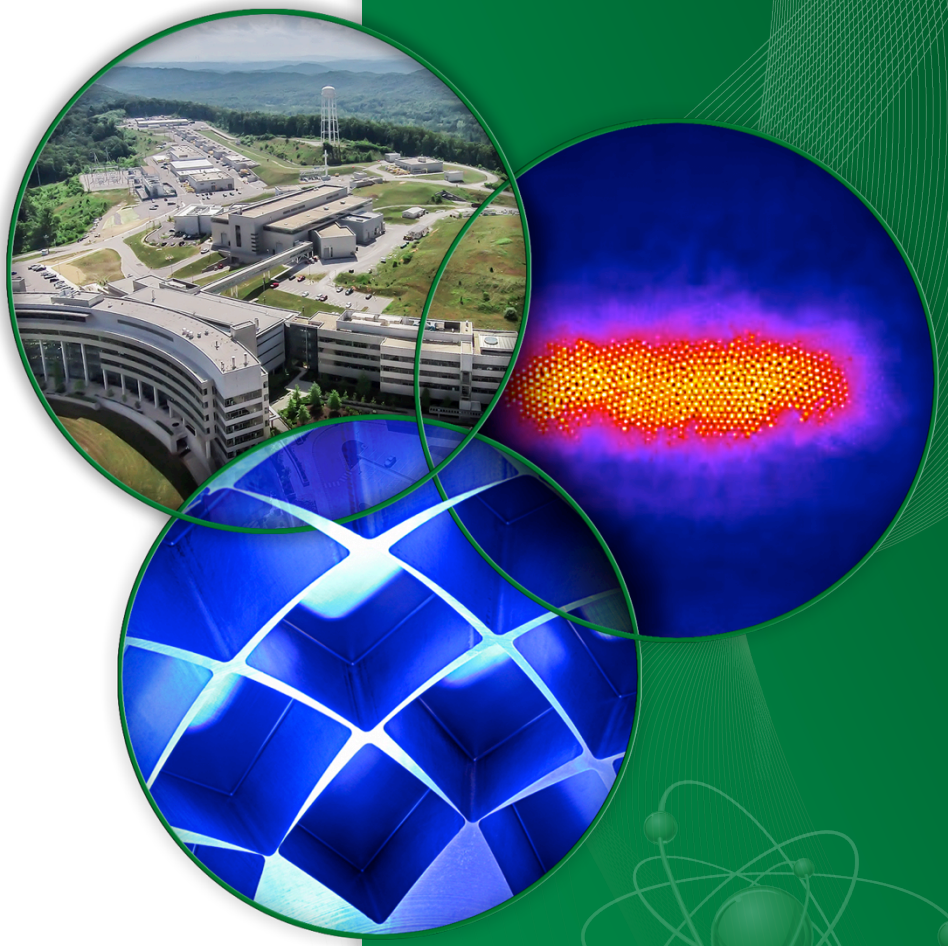
display.builder Update Project



Oct. 2015

Kay Kasemir

ORNL is managed by UT-Battelle
for the US Department of Energy

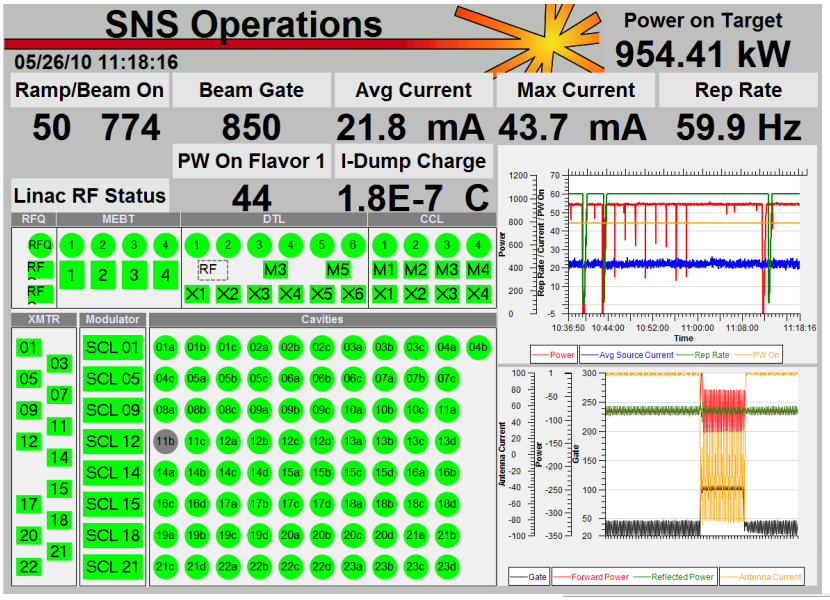
OAK RIDGE National Laboratory | SPALLATION NEUTRON SOURCE

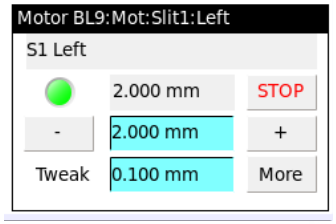
Why?

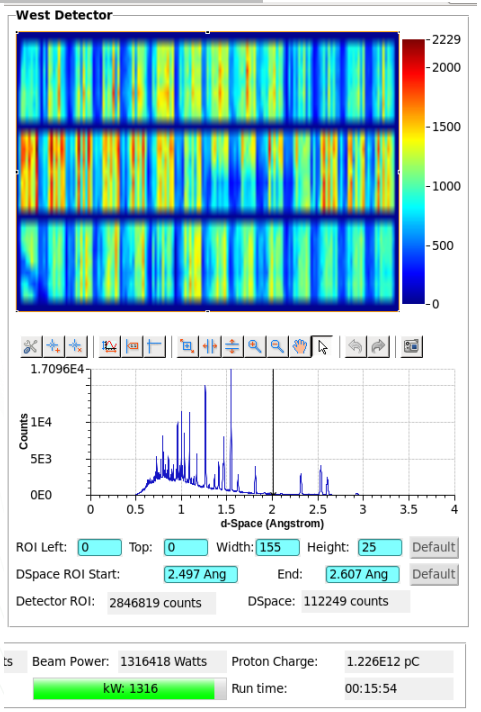
BOY is <u>the</u> control system user interface on SNS beam lines

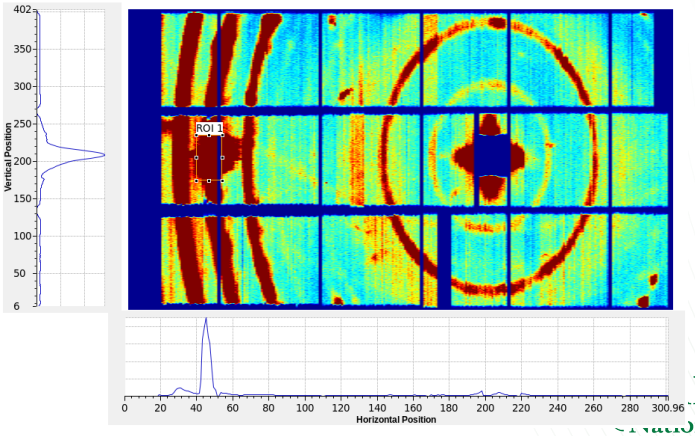
.. also used at NSLS-II, KEK, FRIB, ITER,

DLS invested in good EDM translation





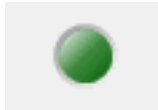


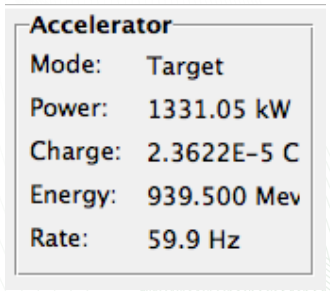


What's Good

- Editor
  - Various Alignment options
  - Edit properties of multiple widgets

- Widgets describe meaning
  - LED with PV and alarm-sensitive border, *not* Circle with border and background color dynamically linked to PVs
  - Group, *not* Rectangle that happens to be around other widgets





Scripts

- Are evil and should be avoided

  .. but still better than separate python/Qt, tcl/tk, perl/ wxWindows, .. tools

Model's XML

- .. is dump of widget property map.
- "Save As" without change results in different file; impossible to track real changes.

→Save elements in consistent order

Model Loading

- Is on UI thread, freezing application.
- Keeps warning “..was created with newer version..” because of compile time, not actual version changes

→Model load/save in background threads.

→Let widgets handle version changes.

Data Flow

- .. moves early on to UI thread,
- .. including scripts!

→Rewrite.

In background thread:
PV updates → optional scripts → model changes

On UI thread:
UI displays model changes

Why don't we just change it?

- Model tied to SWT/Draw2d widgets.
- Runtime depends on G*Editor*F.

→Rewrite.

Model: Hierarchy of Widgets w/ Properties.
Representation: Maps model to SWT, JavaFX.
Runtime: Connect to PVs

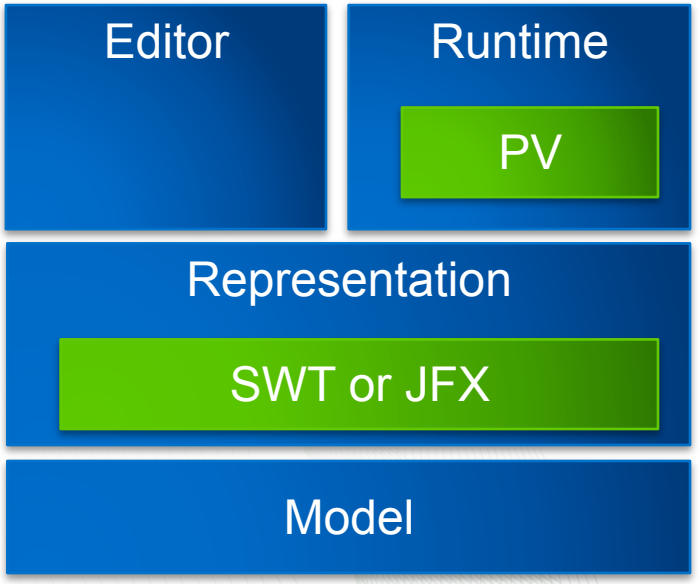
Could have different web repr&run

display.builder Idea

To end user, very similar to BOY.

Internally, independent…

1. Model
   - Widgets and properties (no SWT/Draw2D/RCP)
2. Representation
   - SWT, JavaFX: Replaceable
3. Runtime
   - Handles PVs, Scripts
4. Editor



Model

- Widgets
  - Rectangle, Label, TextUpdate, LED, ProgressBar, Image, XYPlot

    ... for now, implemented only essential properties
  - Group (contains other widgets)
  - Embedded Display
  - ActionButton
- Properties
  - Change notification
  - Well defined category & order
  - Colors, Fonts
  - Structures, Arrays (XYPlot “axis”, “traces”)
- Persistence
  - Loads existing *.opi files
  - Saves in defined order
- Macros
  - For text as well as ‘x’, ‘visible’, ..

TODO

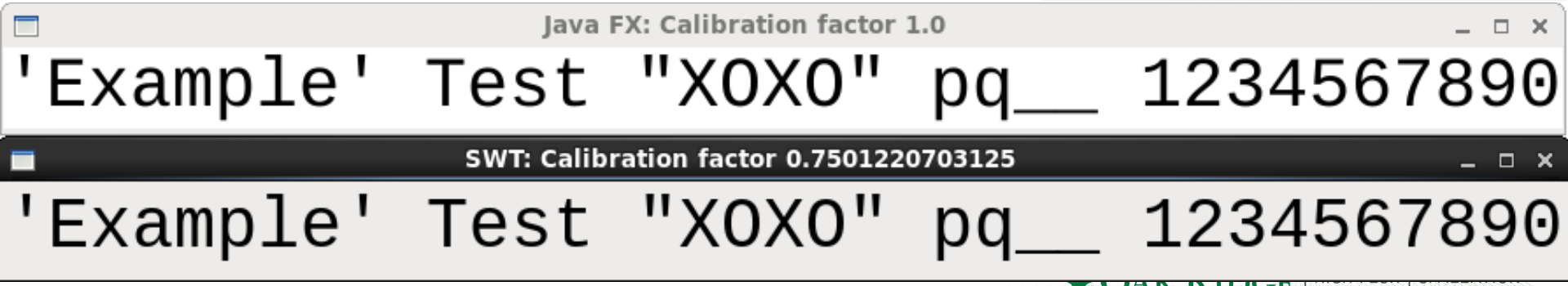
- Many more properties and widgets
- Defaults based on ‘class’ property

Representation

- SWT
  - Incomplete, just to show it's possible
- Java FX
  - For all widgets
- Fixed the font size problem!
  - BOY didn't distinguish between "point size" and pixel

TODO

- Many more widgets
- Many properties: Alarm sensitive, ..
- Web Representation



Runtime

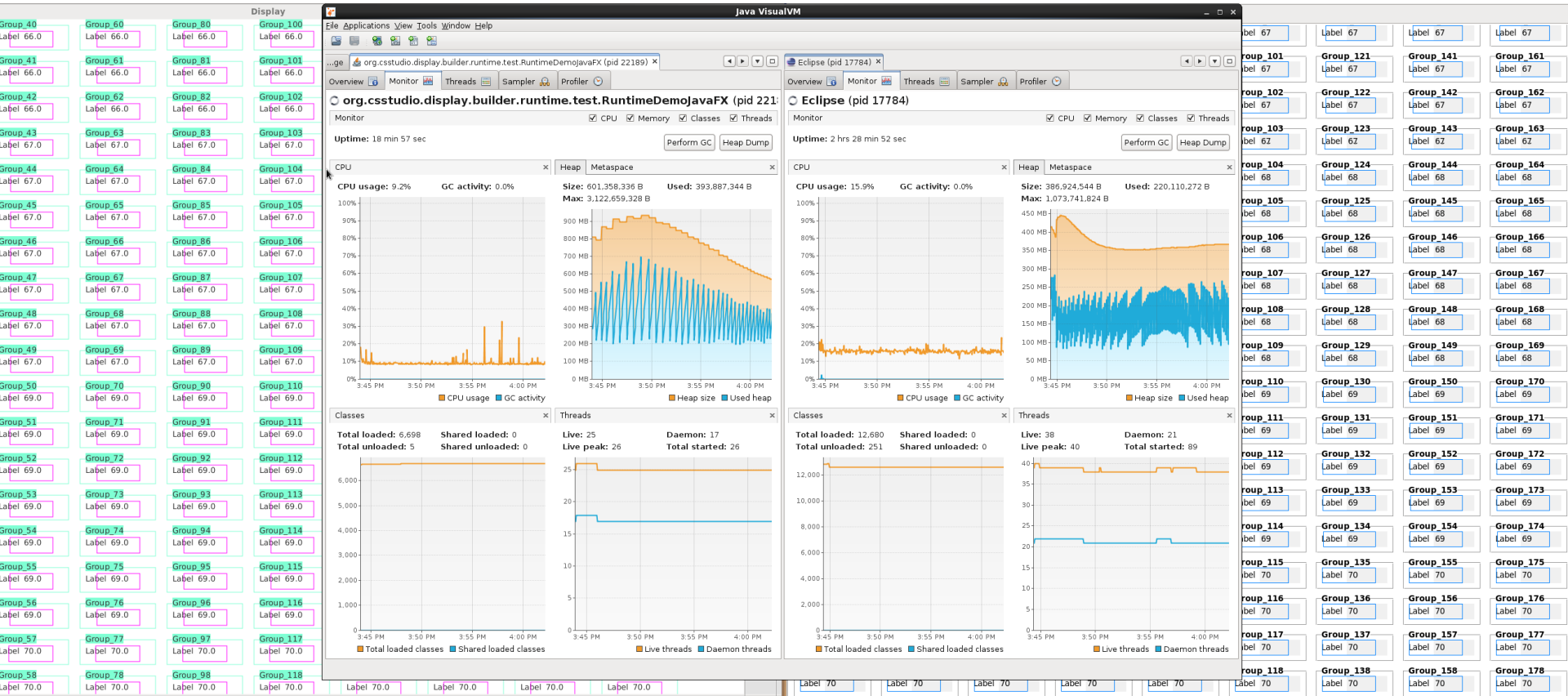
- Common Widget Runtime
  - Loads model in background
  - Creates representation on UI thread
  - Handles PVs and scripts in background
  - One Jython interpreter & thread per window
  - Throttles UI updates
- EmbeddedDisplay Runtime
  - Keeps hosted display private
- ActionButton Runtime
  - Replaces current or opens new window
  - Writes PVs

TODO

- Navigate back/forward

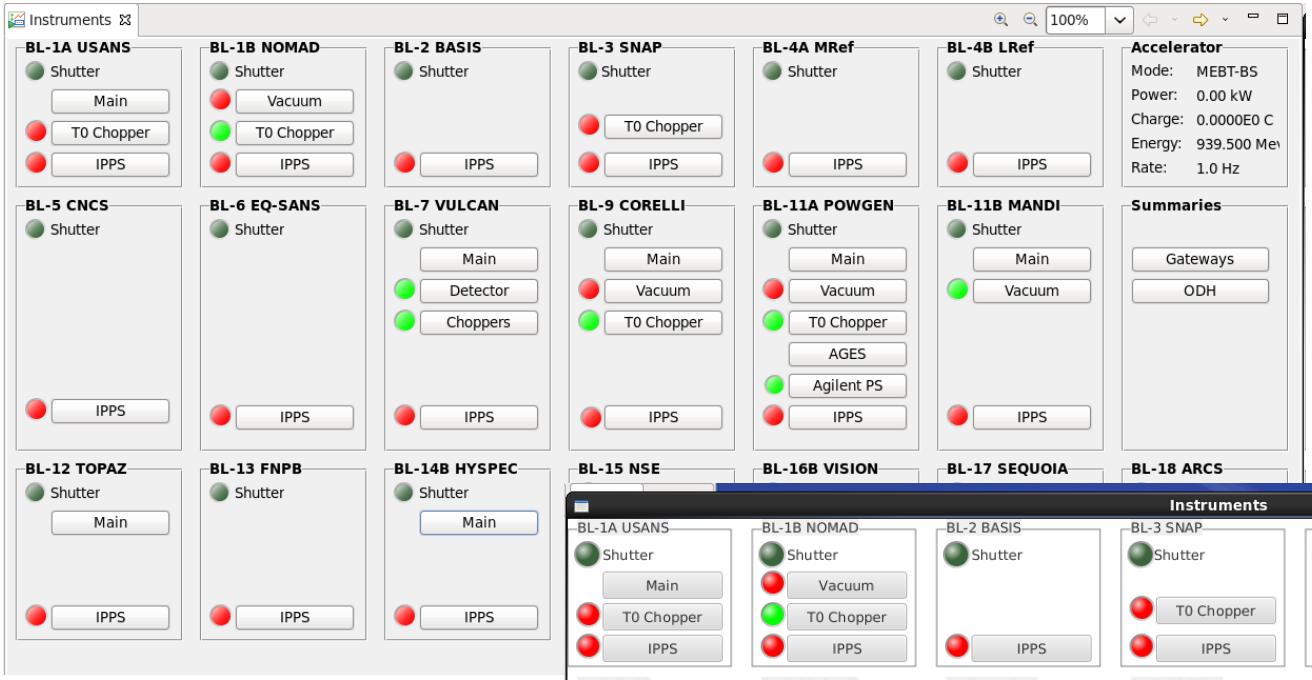
CS-Studio/SWT vs. display.builder/JavaFX

- 200 { Group, Label, TextUpdate (10Hz), Rectangle (10 Hz Script) }
- Scripts, PVs, Model updates off the UI thread
- Windows: Same CPU Load.
  Linux: Lower CPU Load for JavaFX ☺☺



Compare BOY and display.builder

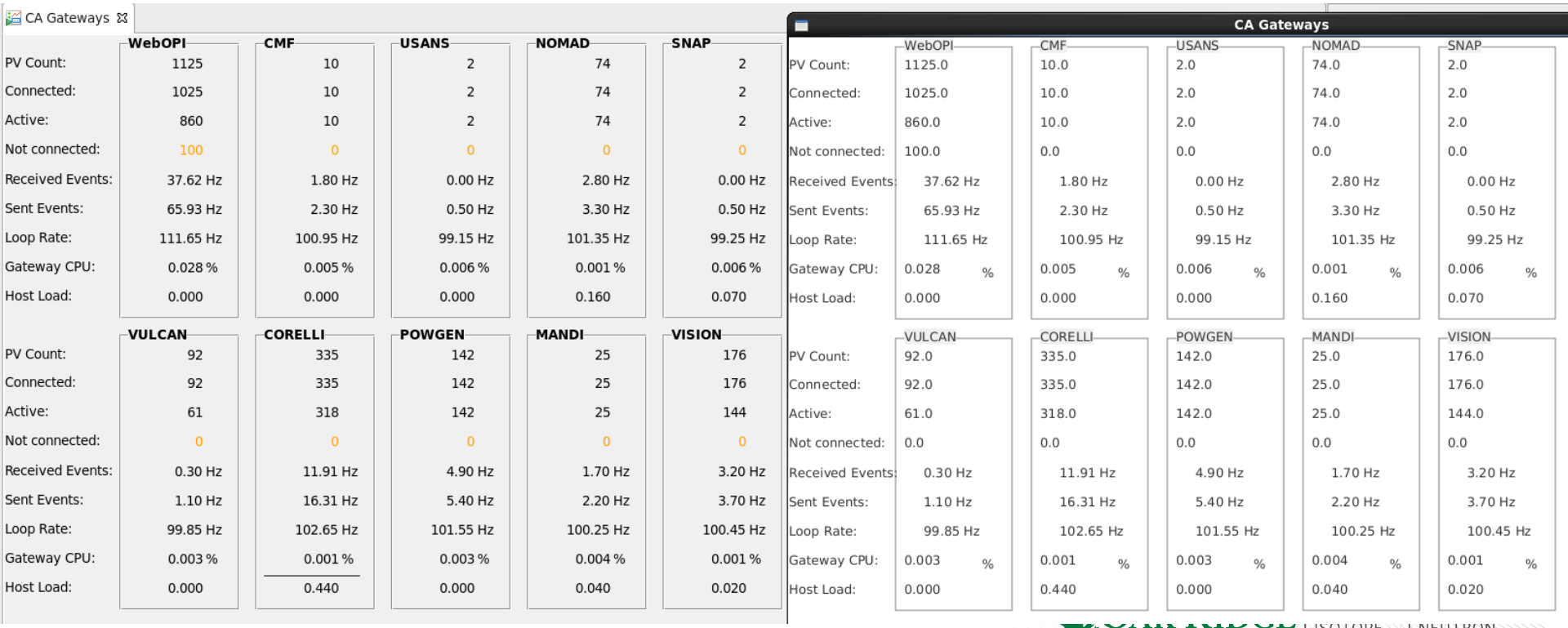
SNS Instruments Overview: very similar!



Compare BOY and display.builder

- Loads via http://
- Groups update macros of contained widgets

TODO: ‘alarm sensitive’ color/border/..



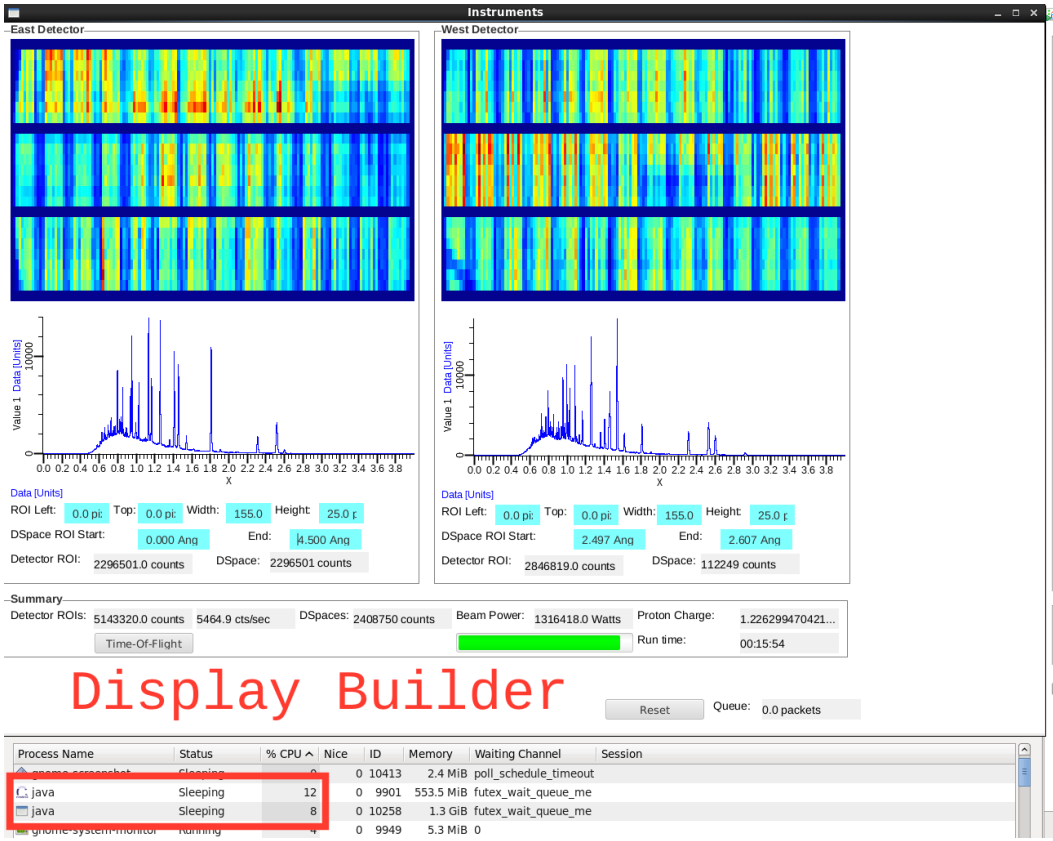
Compare BOY and display.builder

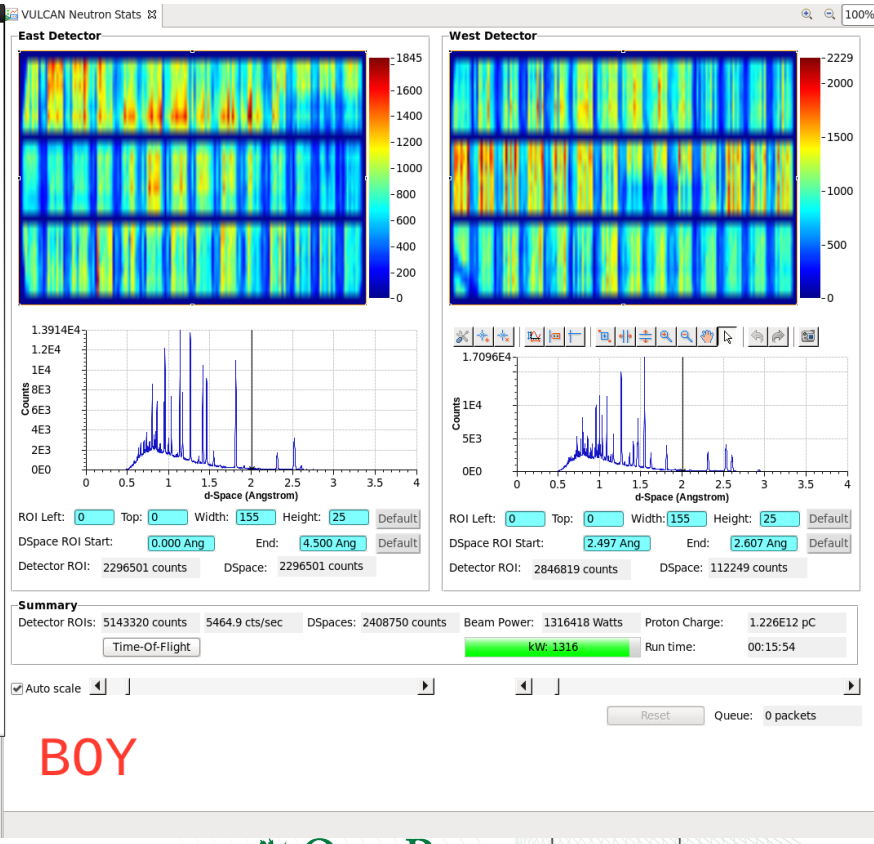
Nested groups, embedded displays.

Resolves file locations ‘relative’ to parent, passes macros

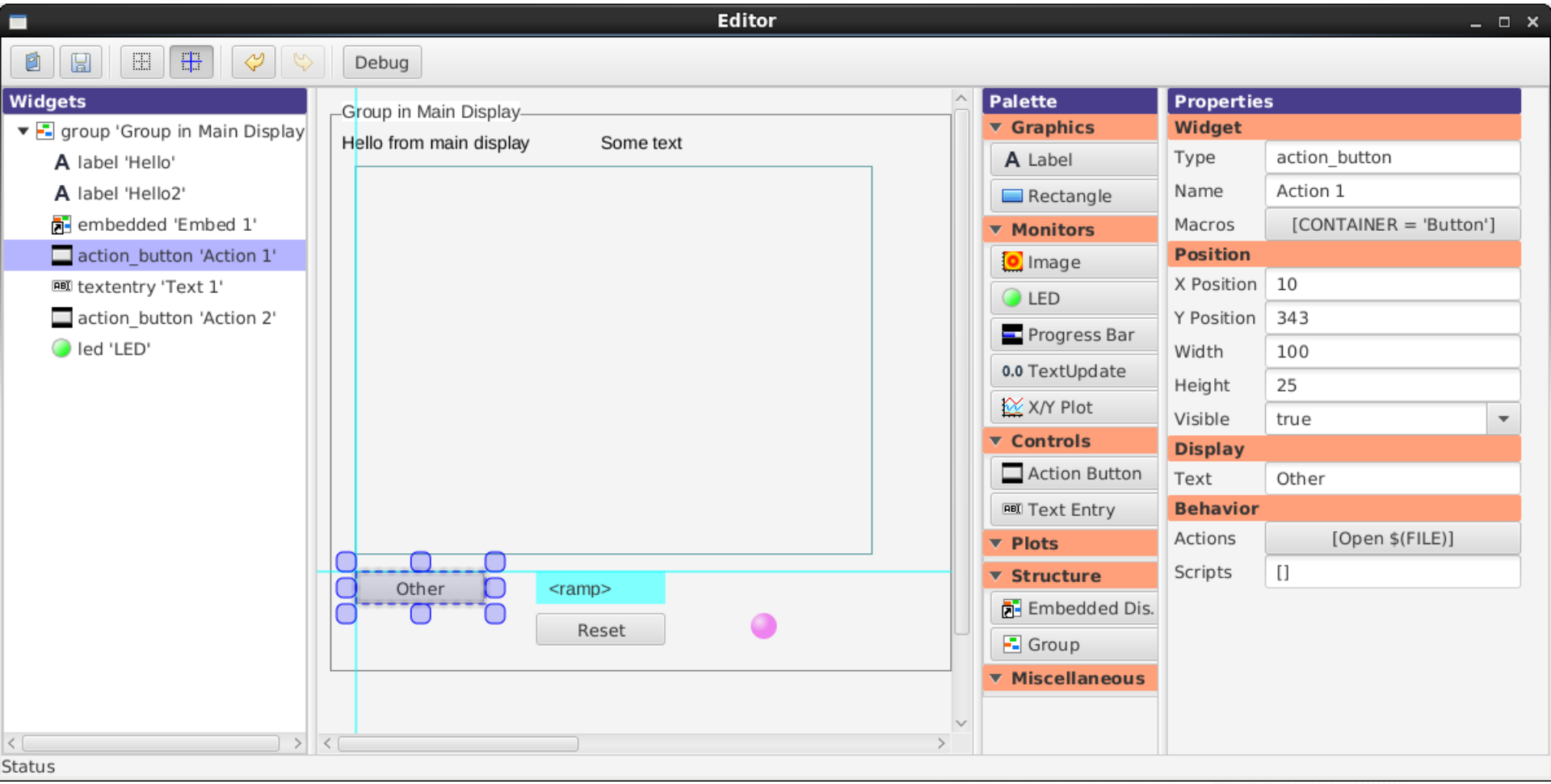
Plotting

Basics of Image and XYPlot





Editor Exploration



- Grid, Guidelines
- Common Properties

Editor Exploration

- Palette of available Widgets
- Show current model: Display and Widget-Tree Outline
- Editable Properties
  - In defined order
- Tracker to move/resize
  - Snap-to-Grid
  - Snap-to-Geometry (parallelized)
- Toolbar
- Selection:
  Rubberband, multi-widget,
  in and out of ‘Group’

TODO

- Complex properties
  - Scripts
  - XYPlot “Axis”,
    “Trace” structure
- Rulers
- Copy/Paste

Summary

Reads existing files, looks the same

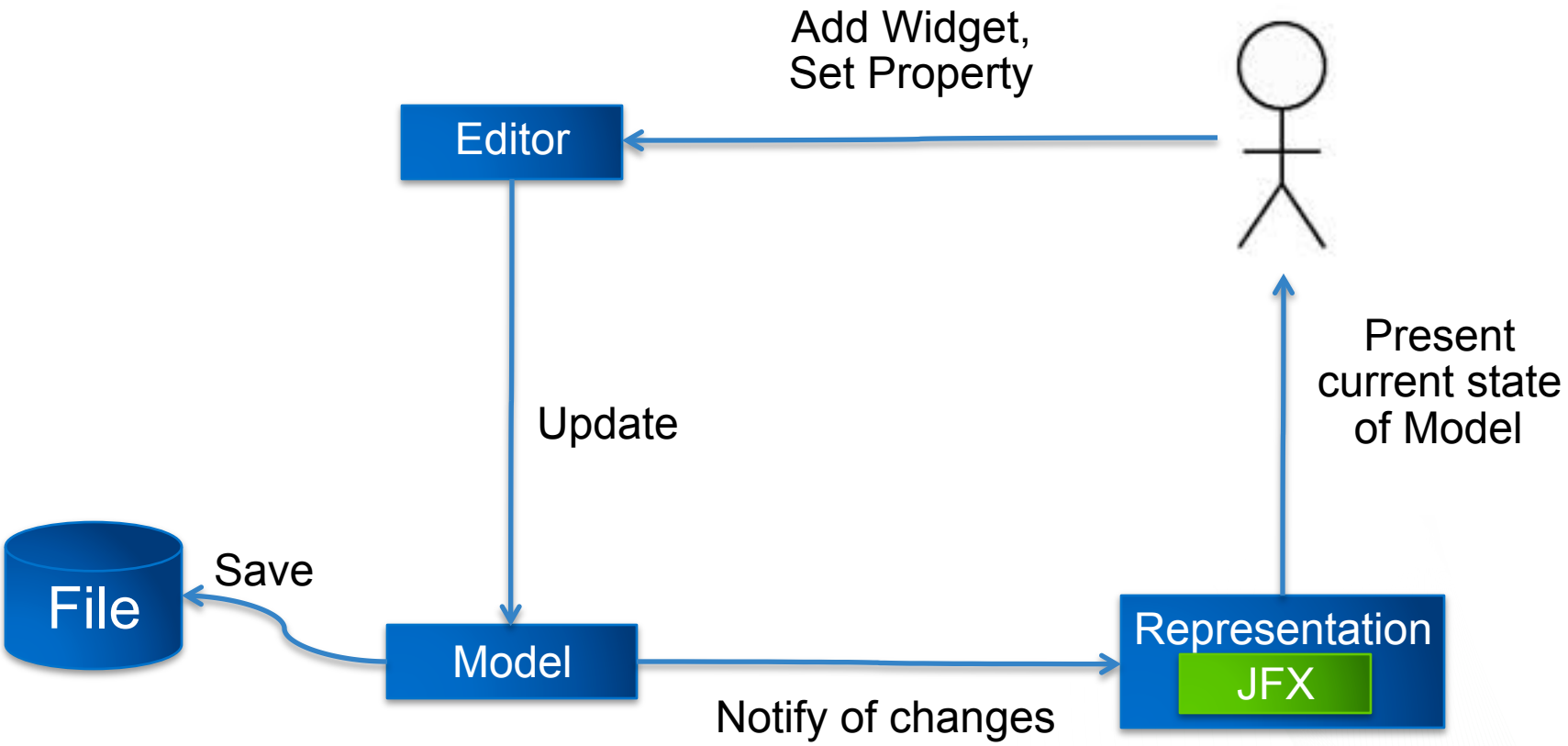
- Separate Model, Representation, Runtime
- Background threads whenever possible (less freezing)
- Typically faster

To Do

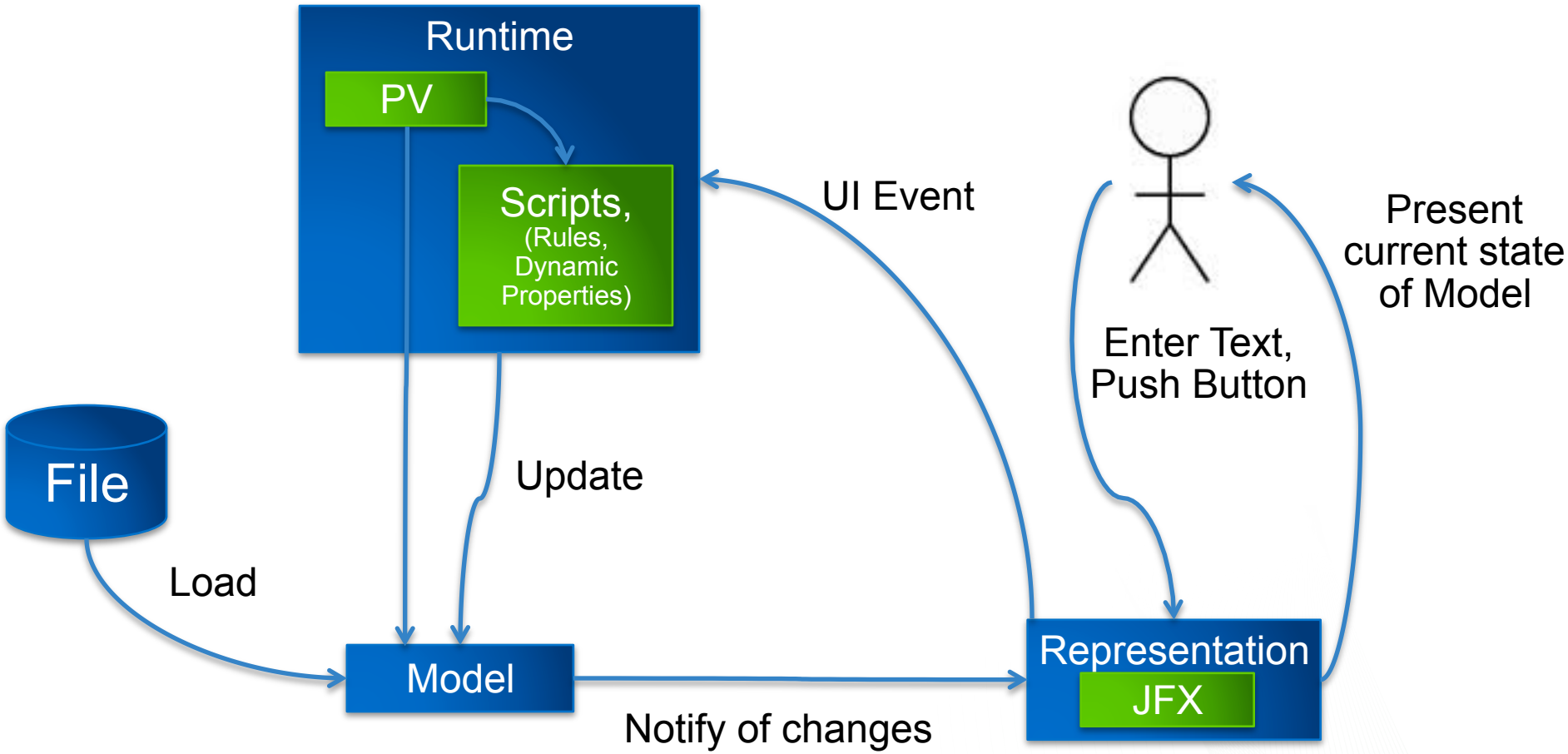
- Many widgets, gazillion properties
- Editor smoothness
- Eclipse RCP integration (FXCanvas, context menus)

Boy, only Better

Design Time Data Flow



Runtime Data Flow



JavaFX Quirks ☹

- Java API vs. Style sheets
  - LED uses API for basic color, css for ‘highlight’
    - css limited to single –fx-effect, while API can chain multiple.
  - ProgressBar
    - Default css is nice, but blue.
    - API for setting color would get flat, solid rectangle.
- ToolTip only for ‘Control’, not ‘Shape’
- Mouse clicks unaware of Shift/Control/Alt/.. State
  - Additional onMouseDown(), which needs to ‘arm()’ button
- Custom-drawn Widgets
  - ‘Canvas’ draws on UI thread
  - WritableImage only offers set(x, y, color)
  - Using AWT to draw image in background, then display in Canvas

Linking Container

- Model merges linked content into one large model, then uses “instanceof AbstractContainer” to hack around the result.

→Rewrite.
Container has *internal* model of linked content.

‘Connectors’ to widgets within container can’t be supported because content may change.

Editor

- GEF?
  - “Will not be developed further”
  - Requires Draw2D representation for every widget
- GEF4?
  - Provisional API
  - GEF4 MVC = Demo of Bezier curve editor?
  - No ‘Palette’, no Context Menu