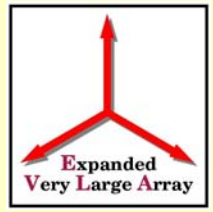
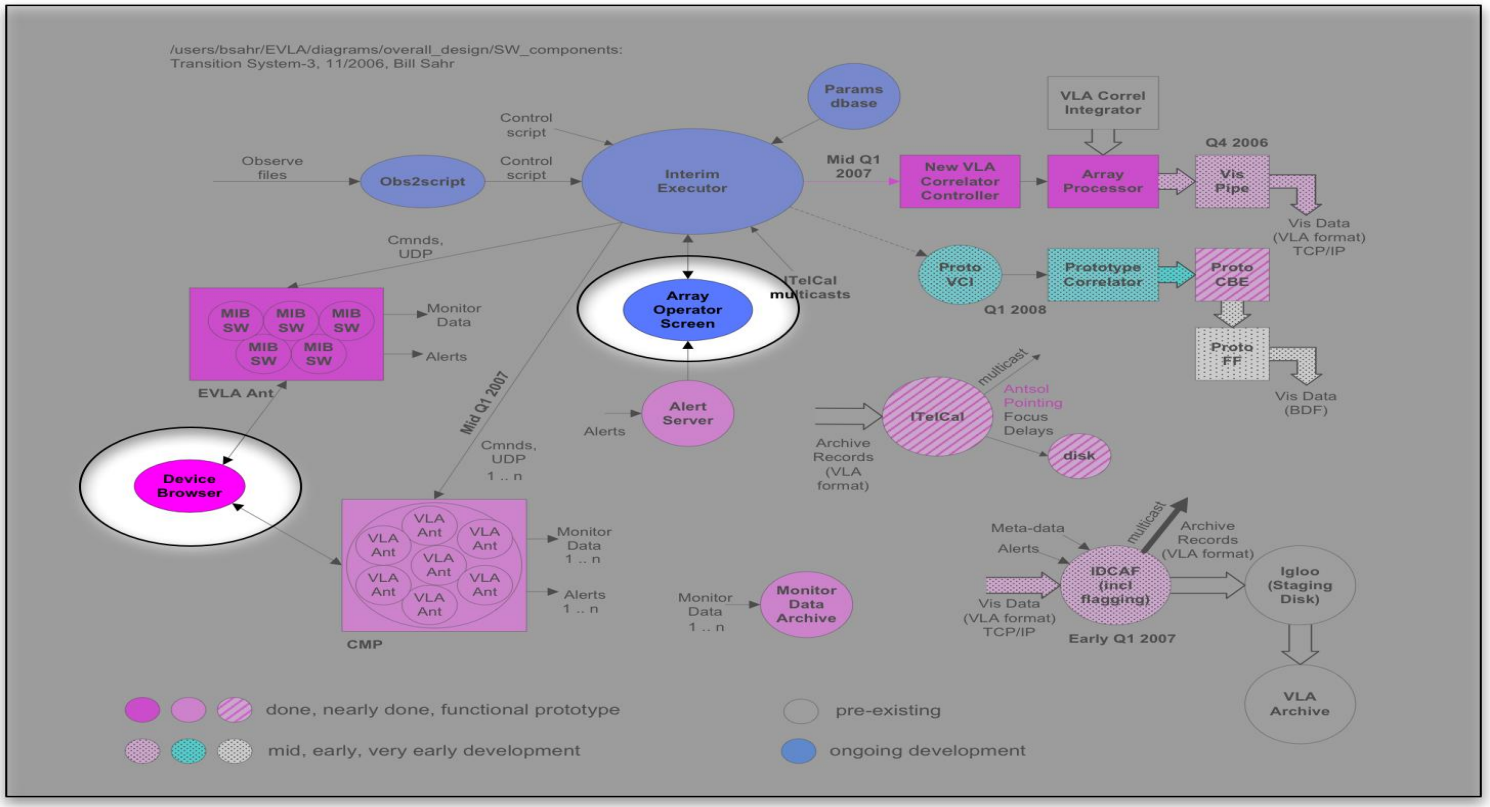


User Interfaces

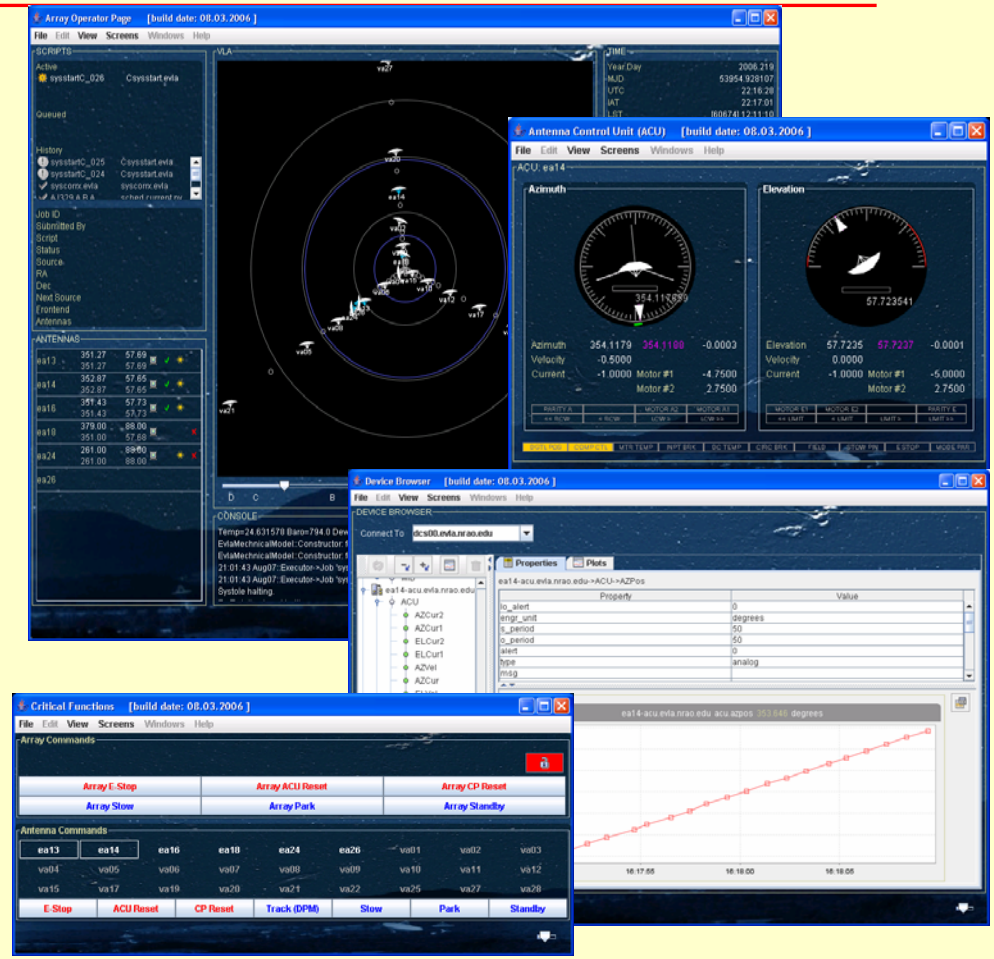
Rich Moeser

Place in the System



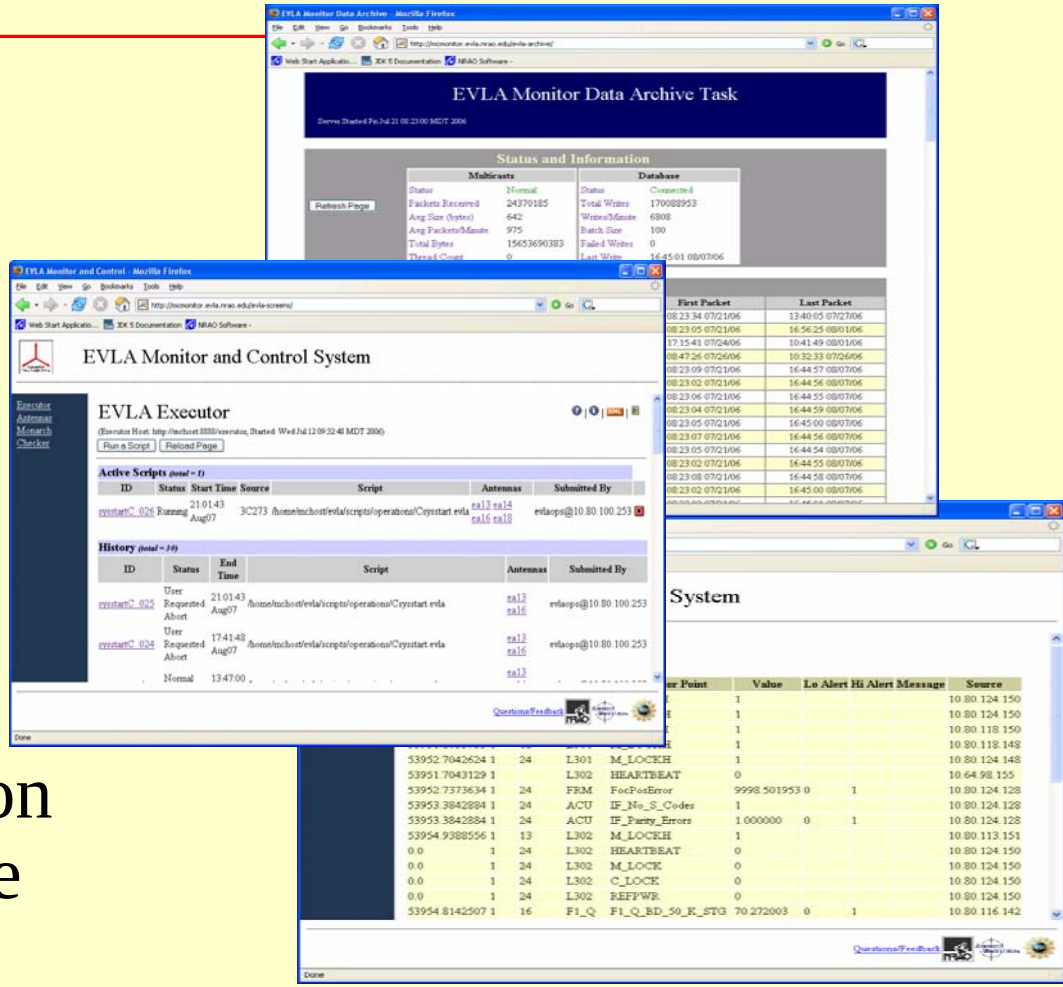
EVLA User Interfaces

- Two types of user interfaces for the EVLA M&C system
  - Java application
  - Web applications
- Java Application
  - EVLA Tool Suite
    - Array Operator Screen
    - Module Screens
    - Device Browser
    - Critical Functions Screen



EVLA User Interfaces

- Web Applications
  - Mostly functional prototypes
  - Executor interface
  - Monitor data archive interface
  - Alerts interface
  - More to come…
- This talk will focus on the EVLA Tool Suite



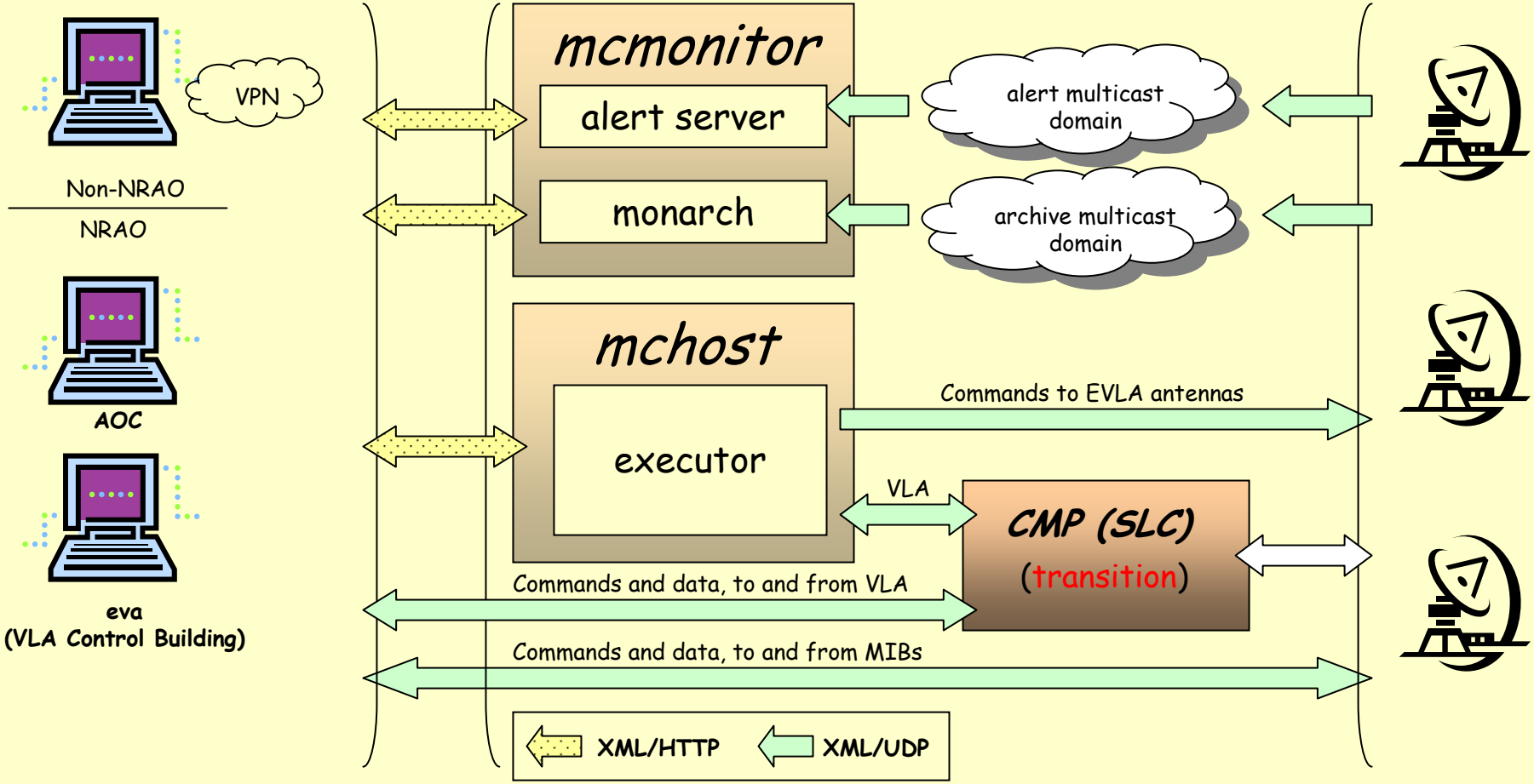
EVLA User Interface History

Date(s)	Description
2001-2002	Requirements docs, prototyping
2003	Work on communications infrastructure, examined XML-based GUI frameworks, prototyping, etc. Early work on Device Browser (Thinlet-based)
2004	**Device Browser** field tested and put into operation. MIB to LabView software interface developed. (Using LabView's DataSocket Server) Web-based **Alerts interface**
2005	• Web-based **Executor interface** (released in May) • **EVLA Tool Suite** (Operator Screen, Device Browser, ACU screen, etc) Released in September
Oct 2005-Dec 2006	**EVLA Tool Suite** - 13 releases. (Oct05, Nov05, Dec05, Feb06 (2), Mar06, Jun06, Jul06 (2), Aug06, Sep06, Oct06, Nov06)
Current Status	50 source files, ~24000 lines of Java code, 9 jar files (2 in-house, jfreechart, swinglabs, 5 JAXB)

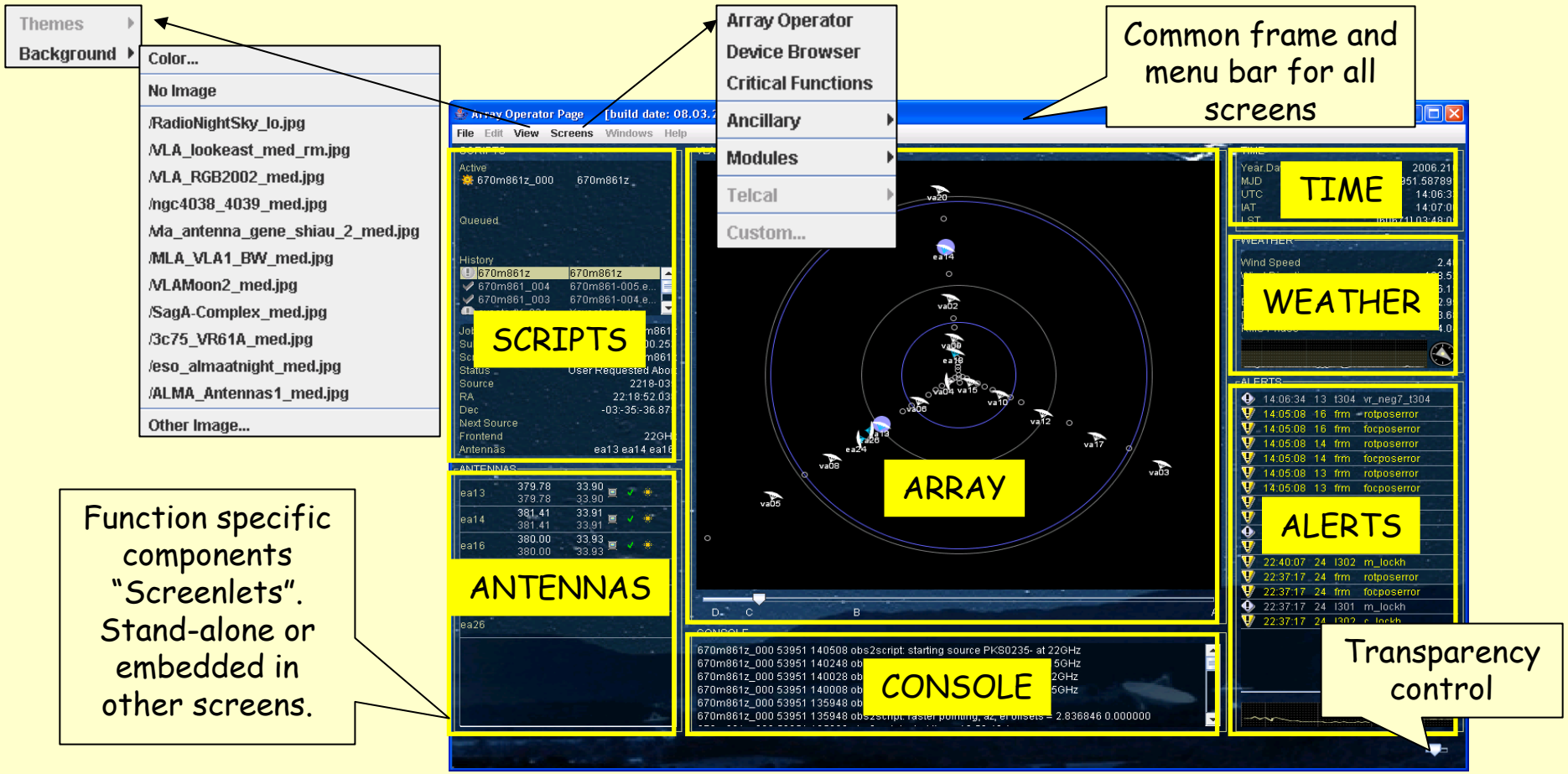
Technologies

- Java (version 1.5)
  - Swing (JTable, JPanel, JFrame, etc, etc, etc)
  - Graphics2D (custom components)
  - Java Web Start
  - JAXB (Java Architecture for XML Binding)
  - SwingLabs (http://swinglabs.org)
    - So far using only a few SwingLabs components.
    - Many of these components eventually end up in swing.
- XML/HTTP (REST)
- XML/UDP
- JFreeChart (version 1.0.1, http://www.jfree.org/jfreechart)
- Abandoned technologies
  - Thinlet (replaced by Swing)
    - Difficult to create custom components
    - Could not use newer features of Java
    - Concerned about future support
  - Castor (replaced by JAXB)
    - Concerned about future support, otherwise worked as well as JAXB

EVLA System Overview



Array Operator Screen Layout

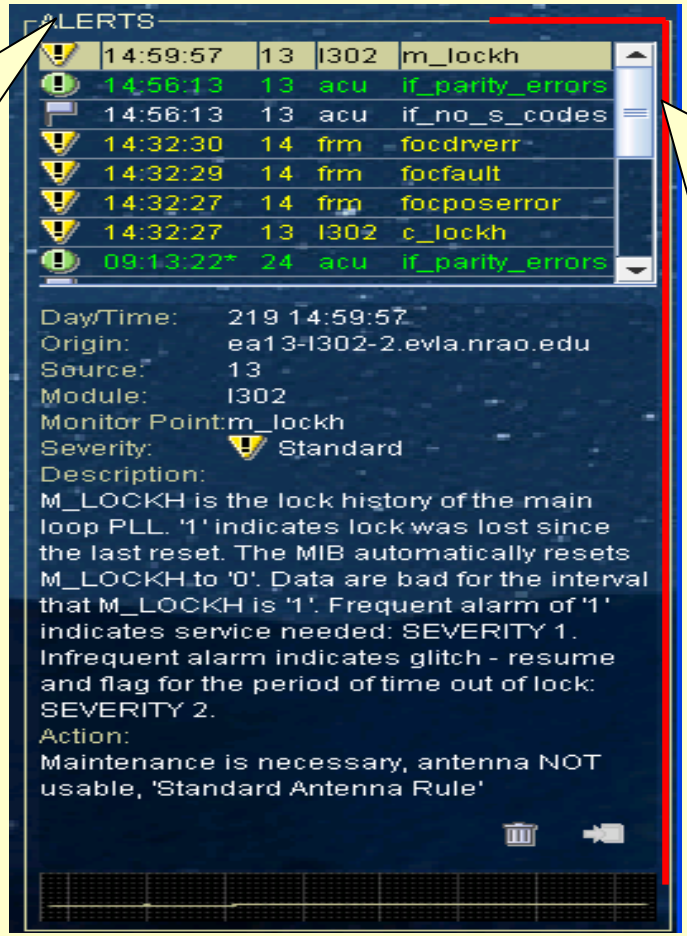


Elements of a "Screenlet"

The **title** of the screen. If you put the mouse pointer over the title and **right-click** a popup-menu will appear.

Clicking anywhere on a screen will (should) not have an adverse effect. For example, it should not send a command to the array.

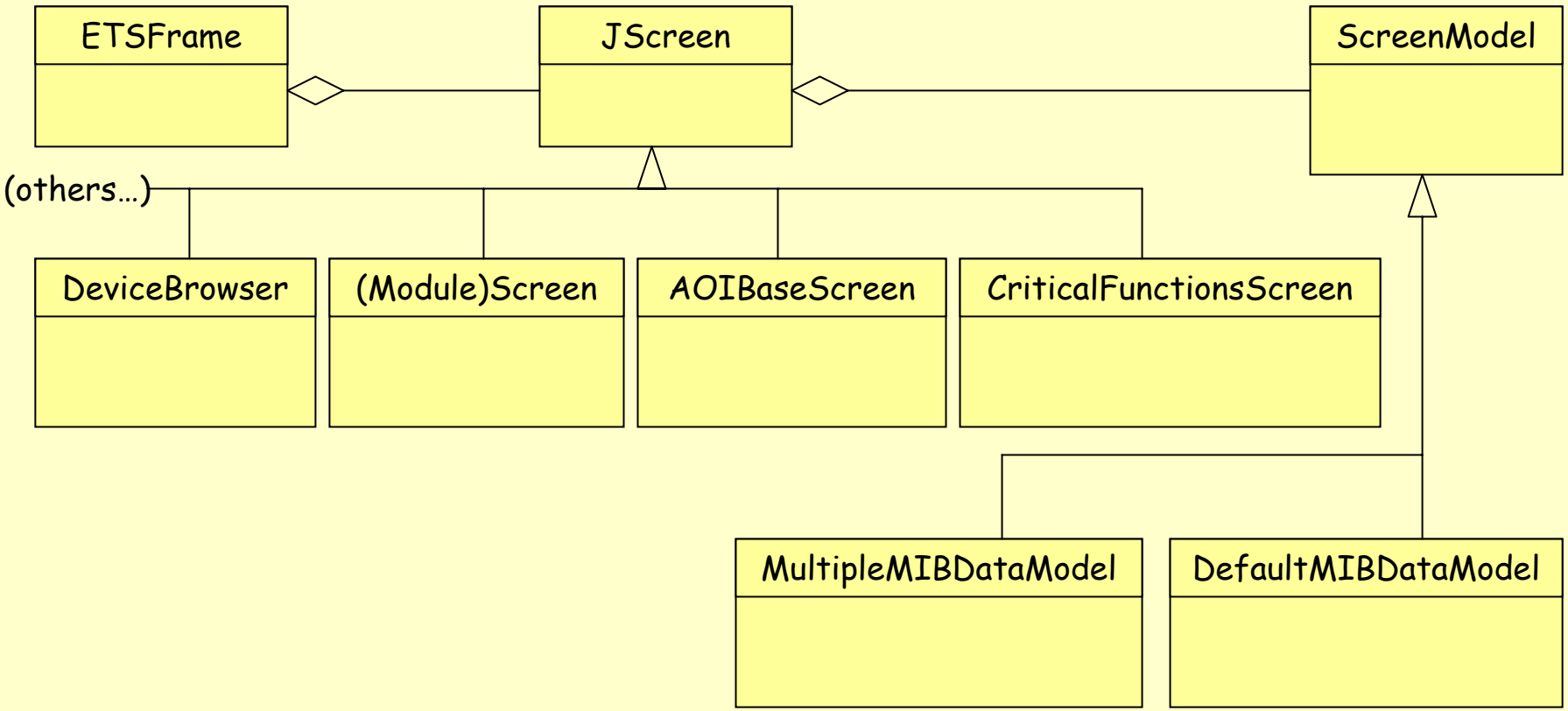
Modifiable **update rate**. Right-click to invoke popup menu and select *Options* and then *Update Rate*



A red **border** indicates a possible problem. Typically it means a communication error has occurred or the data received was invalid. Don't worry if it "flickers" occasionally, only if it's solid red. (If you move the mouse cursor over the title, text describing the error will be displayed.)

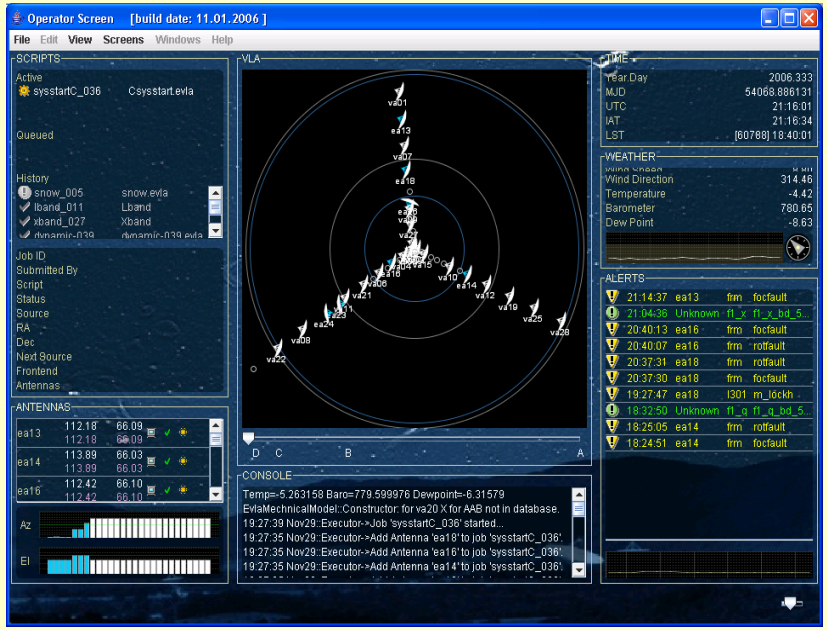
For the most part, screenlets are **self-contained**. Meaning they can be launched in their own window.

Software: Class Diagram



Software: Frame class - ETSFrame

- ETSFrame
  - Common Frame
    - All screen and/or tools have the same frame
  - Common Status Bar
    - Set alpha transparency of screens
    - Common message area
  - Common Menu
    - Exit/Close windows
    - Select screens/tools
    - Select preferences (colors, backgrounds, etc.)



Software: Panel class - JScreen

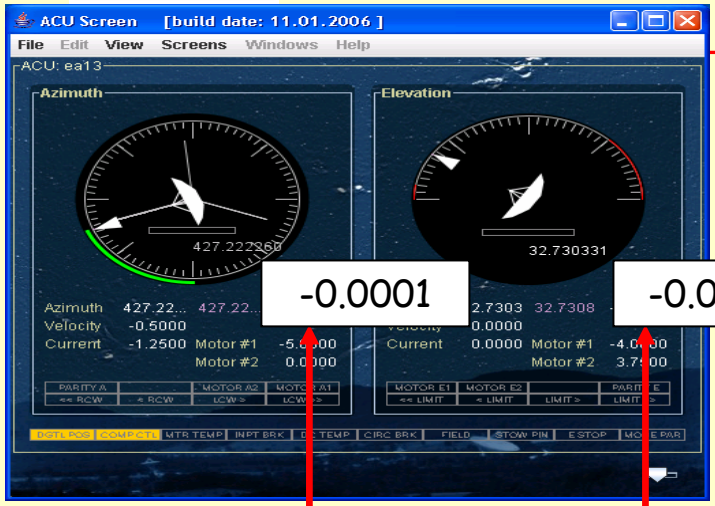
- JScreen
  - Common panel
  - Common look and feel
    - Border provides communication status
    - Alpha transparency
  - Provides a common framework
    - For data collection and display
    - For the sending of commands

Software:
Data model class - DefaultMIBDataModel

- DefaultMIBDataModel
  - Handles communication to MIBs and makes data available to the screens
  - Requires hostname, command and update rate
    - Ex: “ea14-frm.evla.nrao.edu”, “get frm.*”, 5000
  - Polls the MIB for data at the requested update rate
  - The XML from the MIB is unmarshalled into a JAXB object and the data is stored into a table accessible to the screen
  - The parameters in the table are of the form <devId>.<mpId>
    - For example: *frm.anemom1* or *frm.anemom2*
  - When the update has completed the model fires a notification message to the screen.

- MultipleMIBDataModel (a collection of DefaultMIBDataModels)

Software: How screens are updated



-0.0001

-0.0006

JScreen

Update screen components with current value

Read JAXB object from table and get the value

Tell the screen to update itself

DefaultMIBDataModel

Send UDP request "get acu.*"

Unmarshall XML into JAXB object and load table

acu.azposerror	JAXB obj
acu.elposerror	JAXB obj

Notify screen that new data has arrived

UDP request

UDP response XML

ea13-acu MIB

Software: UDP Communications

- UDP is used to communicate with MIBs and the CMP.
- All MIBs and the CMP have a common UDP interface called the *service port*.
- Simple command structure using *gets* and *sets*:
  - `get *.*`
  - `get acu.*.* frm.*`
  - `set acu.azposcmd=30`
  - `Set acu.elposcmd=45`

```
//Example:
//sending a command to a MIB
import edu.nrao.evla.commons.net.*;
...
EVLAClient mib =
    new EVLAClient("ea14-acu",
    EVLAClient.UDP);
mib.connect();
mib.sendOnly("set acu.azposcmd=250");
...
//keep active until done, then close
mib.close();
```

Software
UDP Communications

```
//Example:
//receiving data from a MIB
import edu.nrao.evla.commons.net.*;
...
EVLAClient mib =
    new EVLAClient("ea14-acu",
   EVLAClient.UDP);
mib.connect();
String xml = mib.send("get *.*");
...
//convert to DOM or JAXB object
...
mib.close();
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<EVLAMessage location='Antenna 14' timestamp='53851.7036242'
     timestamp='53851.7036242' >
<device name='ACU'>
  <monitor name='AZCur2' type='analog' value='0.000000'
     alert='0' />
  <monitor name='AZCur1' type='analog' value='0.000000'
     alert='0' />
  <monitor name='ELCur2' type='analog' value='0.000000'
     alert='0' />
  <monitor name='ELCur1' type='analog' value='0.000000'
     alert='0' />
  <monitor name='AZVel' type='analog' value='0.000000'
     alert='0' />
  <monitor name='AZCur' type='analog' value='0.000000'
     alert='0' />
  <monitor name='ELVel' type='analog' value='-1.500000'
     alert='0' />
  <monitor name='ELCur' type='analog' value='-1.500000'
     alert='0' />

...

  <monitor name='ElMtrStat' type='analog' value='0x0'
     alert='0' />
  <monitor name='AzMtrStat' type='analog' value='0x0'
     alert='0' />
  <control name='AzPosCmd' type='analog' value='379.000000'
     value='379.000000' />
  <control name='MtrCtrlCmd' type='analog'
     value='32768.000000' />
  <control name='ElPosCmd' type='analog' value='88.000000'
     />
  <control name='LimitTime' type='analog'
     value='53851.498363' />
</device>
...
</EVLAMessage>
```

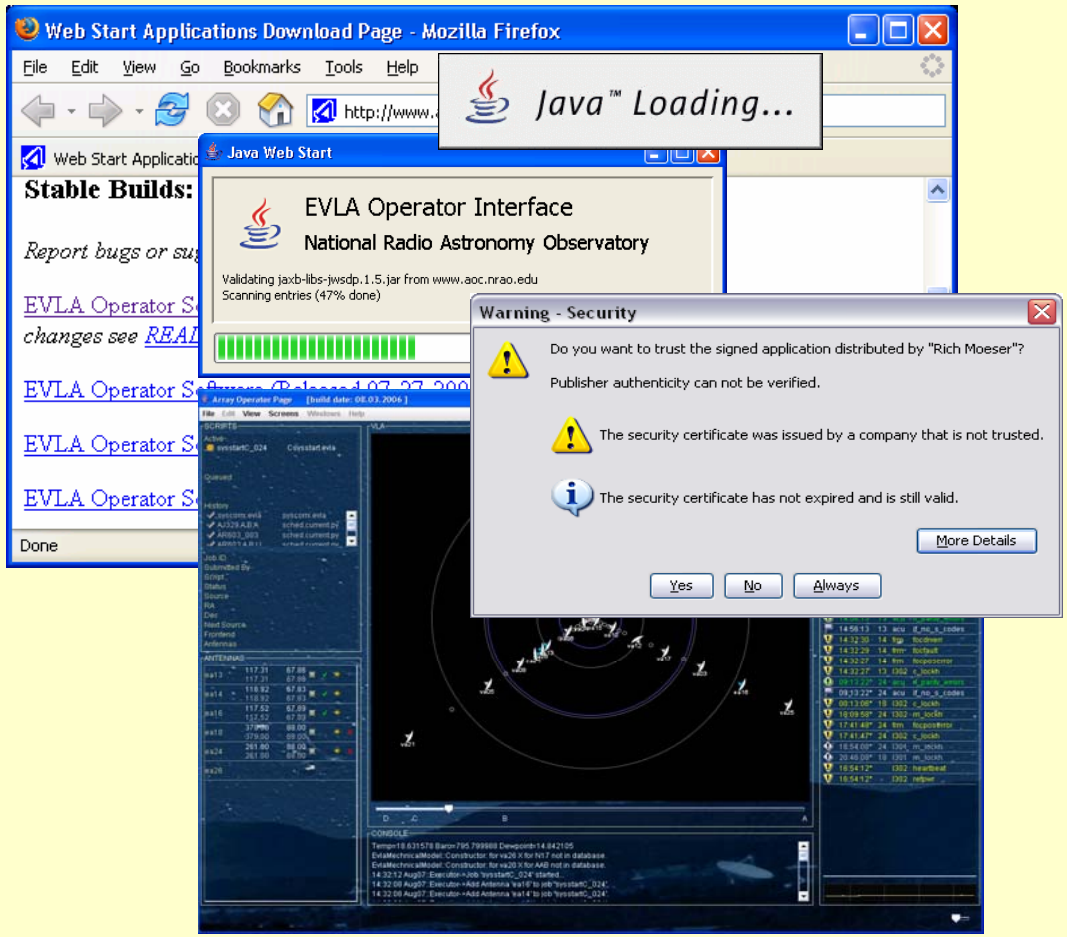
Software
HTTP Communications

```
//Example:
//requesting information from
//the alert server

import edu.nrao.evla.commons.net.*;
...
String xml =
    EVLAClient.GET("http://mcmonitor.evla
    .nrao.edu/evla-
    checker?operation=alertsinfo");
...
//convert to DOM or JAXB object
```

```
//Example:
//requesting information from
//the executor

import edu.nrao.evla.commons.net.*;
...
String xml =
    EVLAClient.GET("http://mchost.evla.
    nrao.edu:8888/executor?operation=Ex
    ecutorInfo");
...
//convert to DOM or JAXB object
```

Deployment using Java Web Start

- Web Start is a mechanism for deploying Java applications over the Web.
- Point a browser to http://www.aoc.nrao.edu/asg-internal/jnlp
- Select the link labeled “EVLA Operator Software” and the latest version of the application
- The browser associates the “.jnlp” file with the javaws application (jnlp = java network launching protocol)
- You will see the “Java Loading…” message
- If the software has changed the download dialog will appear
- “Trust this guy?” dialog
- Application launches



Future Work

- Create screens for all EVLA modules. Currently there are only 7 EVLA modules that have screens (acu, frm, f317, f320, l301, m302, and m303).
- Create Web apps with similar functionality – but, much less dynamic - to support remote access to the EVLA system. The only means of remote access right now is to have VPN software installed on the remote client. The first will be a web-based version of the Device Browser.
- Define and implement screens needed before Modcomps can be retired. Most of the AOI screens will be reproduced during this effort.
- And, much, much more…

Demo…

- Web Apps
  - http://mcmonitor.evla.nrao.edu/evla-screens
- EVLA Tool Suite
  - www.aoc.nrao.edu/asg-internal/jnlp