The Active Notebook

1. Details of subsystems in this component
    1.1. Server Side
        1.1.1. Zope Server
        1.1.2. Apache Server
        1.1.3. Zope content
            1.1.3.1. SeisRes Pipeline Control
            1.1.3.2. SeisRes Submission & Results
                1.1.3.2.1. SeisRes Control
                1.1.3.2.2. SeisRes Projects
            1.1.3.3. Product Folders
        1.1.4. CGI-BIN Server Side Scripts
    1.2. Client Side
        1.2.1. TCL Plugin
        1.2.2. Broadway Plugin
        1.2.3. TCL Scripts
        1.2.4. MetaGui
        1.2.5. BLT Graphs
        1.2.6. TIX Widgets
        1.2.7. SDV Visualizer
2. List of source file and directory structures composing this component.
3. Environment variables used by this component.
4. Detailed definitions of file formats used by this component and examples, e.g., permission file.
    4.1. Template files
    4.2. Workflow graph example
5. Definition and comments about classes and data structures used by this component.
6. Class diagram (in PDF).
7. Detailed documentation for the source code created automatically from source.comments and header files.
8. Active Notebook Appendix A: Example of VTK in the Notebook Browser
9. MetaGui and Workflow Graph documentation.

1. Details of subsystems in this component

1.1. Server Side

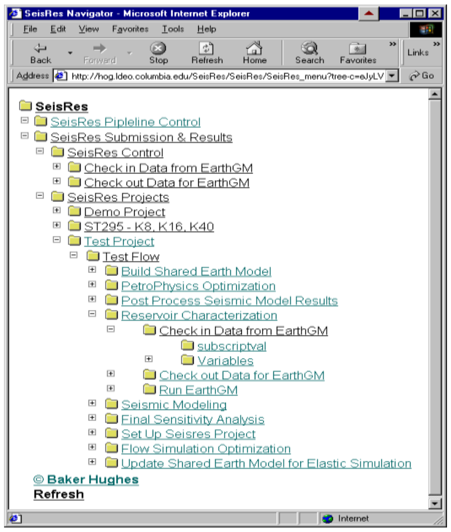
1.1.1. Zope Server

Zope is called a web application server. It is open source. It uses an object oriented model for web content. It is based in python and uses a model of publishing objects on the web. It has an object database where web content is stored. It uses a multitasking server based on Medusa. For details on Zope see: **http://www.zope.org**.

1.1.2. Apache Server

Apache is a popular open source web server. We run Zope under Apache by using an Apache rewrite rule to map urls to Zope. This is done so that there can also be cgi-scripts which Zope does not do. For more on Apache see **http://www.apache.org**.

1.1.3. Zope content


SeisRes Navigator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Links
Address
Go
Internet

http://hog.ldeo.columbia.edu/SeisRes/SeisRes/SeisRes_menu?tree-c=eJyLV

```
SeisRes
⊟ SeisRes Pipleline Control
⊟ SeisRes Submission & Results
    ⊟ SeisRes Control
        ⊞ Check in Data from EarthGM
        ⊞ Check out Data for EarthGM
    ⊟ SeisRes Projects
        ⊞ Demo Project
        ⊞ ST295 - K8, K16, K40
        ⊟ Test Project
            ⊟ Test Flow
                ⊞ Build Shared Earth Model
                ⊞ PetroPhysics Optimization
                ⊞ Post Process Seismic Model Results
                ⊟ Reservoir Characterization
                    ⊟ Check in Data from EarthGM
                            subscriptval
                        ⊞ Variables
                    ⊞ Check out Data for EarthGM
                    ⊞ Run EarthGM
                ⊞ Seismic Modeling
                ⊞ Final Sensitivity Analysis
                ⊞ Set Up Seisres Project
                ⊞ Flow Simulation Optimization
                ⊞ Update Shared Earth Model for Elastic Simulation
  © Baker Hughes
  Refresh
```

Figure 16. *Drill down of SeisRes content*

Note on our system at Lamont there is another parent folder, SeisRes, that contains this. This is so that other folders can be added as trials without them being in the main SeisRes folder sequence.

The main folder contains the portal's initial Index page.

1.1.3.1. SeisRes Pipeline Control

This folder contains some index documents for Admin, Util, and Control functions. These indices refer to html documents in the SeisRes Control folder in SeisRes

Submission & Results (see below). This is for historical reasons and is likely to go away.

1.1.3.2. SeisRes Submission & Results

This folder contains the main content. In this folder, are the dtml methods for setting up the metagui interfaces for the TCL plugin, for processing submissions, and storing parameter/values as properties, etc. DTML methods defining the look and feel for SeisRes Index pages, etc, are defined here. These are inherited by *acquisition* by the child folders of this folder.

1.1.3.2.1. SeisRes Control

This folder contains the various html documents for general control, export/import, and utilities. The contents of this folder are not versioned by different projects.

1.1.3.2.2. SeisRes Projects

This folder contains all the Projects. Within each of the projects are individual workflows. A workflow contains all the workflow step folders. Generally a workflow step folder will have an index document and documents for setting up the execution, monitoring, and results documents. The synopsis document is used to display the step status in the workflow box for that step. The workflow box can handle HTML text and images. (It is a simple html browser.) Submission documents that involve setting up a GUI interface in the tcl plugin will have their own subfolder with the variable folders with the metagui (metadata) properties on them. The index page for a workflow contains the tcl graph code gui layout for the workflow graph. Both the project folder and the workflow folder have methods for making new projects and workflows. The following table shows a drill down of a project, Test Project, one workflow within it, Test Flow, the workflow steps, and a drill down of the Reservoir Characterization step.

1. Drill down of Project, Workflow, Workflow Steps

- Demo Project
  - ST295 - K8, K16, K40
  - Test Project
    - Test Flow
      - Build Shared Earth Model
      - PetroPhysics Optimization
      - Post Process Seismic Model Results
      - Reservoir Characterization
        - Check in Data from EarthGM
        - Check out Data for EarthGM
        - Run EarthGM
      - Seismic Modeling

```
    Final Sensitivity Analysis
    Set Up Seisres Project
    Flow Simulation Optimization
    Update Shared Earth Model for Elastic Simulation
```

1.1.3.3. Product Folders

Zope has a notion of *Products*, which are web content that can be instantiated by making a *clone* of a product. A clone can be given particular attributes to tailor it to a specific use. This is how new SeisRes Projects and Workflows are generated. The table below shows a typical set of products. (A basic set comes with a Zope install and more can be downloaded from **www.zope.org**.) The contents of the workflow product contain two dtml documents for tailoring the clone -- one for displaying a form and another for processing the html post action. There is the workflow contents folder that will be cloned in the process of instantiating a new workflow. Finally. there is a Zope factory method for making clones. More details on making Zope products is available at **www.zope.org**, in the *Zope Content Manager's Guide*.

```
The contents of the Products Folder looks like:
Product Management at /SeisRes / Control_Panel / Products

  Squishdot (Installed product Squishdot (Squishdot-0-3-2))
  TinyTable (Installed product TinyTable (TinyTable-0-8-2))
  ZCatalog (Installed product ZCatalog (Catalog-1-0-0))
  ZDBase (Installed product ZDBase)
  ZDConfera (Installed product ZDConfera)
  ZGadflyDA (Installed product ZGadflyDA)
  ZSQLMethods (Installed product ZSQLMethods)
  seisres_project (SeisRes WorkFlow)
  seisres_project_folder (New Folder for SeisRes Project)
```

```
The contents of The SeisRes Worklow Product looks like:

Product at /SeisRes / Control_Panel / Products / seisres_project

   Designer (Form for adding a new SeisRes Workflow)
   SeisRes_WorkFlow (Generic Workflow (1/22/00))
   build_it (Build New Workflow)
   siesres_workflow (SiesRes Workflow Template)
```

2. Zope SeisRes products.

1.1.4. CGI-BIN Server Side Scripts

The table below summarizes the scripts used on the server side:

Zope.cgi	Script to invoke Zope with Apache rewrite rule
procmon.tcl	Script to monitor the results of batch submissions
test-monitor	These two scripts are used to monitor SesRes events
test-monitor.tcl	
xplugin.pl	use to run EarthGM

3. Server side scripts.

1.2. Client Side

1.2.1. TCL Plugin

The TCL plugin is used to build GUIs within the Notebook pages. Tcl scripts for the plugin also interface to the SeisRes C++ code that has been wrapped with swig. One can use either the original TCL plugin or the one from NASA.
See **http://www.demailly.com/tcl/plugin/** for the tclplugin
See **http://heasarc.gsfc.nasa.gov/Tools/maki/plugin/** for the LHEATcl Plugin.

1.2.2. Broadway Plugin

See **http://www.broadwayinfo.com/** for the Broadway plugin

1.2.3. TCL Scripts

1.2.4. MetaGui

See **http://www.stratasys.com/software/metagui/**

1.2.5. BLT Graphs

See: **http://www.tcltk.com/blt/**

1.2.6. TIX Widgets

See: **http://www.neosoft.com/tcl/ftparchive/sorted/packages-7.6/devel/Tix4.1.0.006.README** and **http://www.go.dlr.de/fresh/unix/src/contrib/Tix4.1.0.007.tar.gz** to download

1.2.7. SDV Visualizer

See SDV document.

2. List of source file and directory structures composing this component.

Various gui tcl scripts:
$SEISRES_HOME/gui/srgui/
$SEISRES_HOME/pio/examples/
$SEISRES_HOME/srfc/examples/
$SEISRES_HOME/srio/examples/
$SEISRES_HOME/optimizer/tcl/
$SEISRES_HOME/cgc/tcl/

Workflow gui code:
$SEISRES_HOME/gui/workflow/

MetaGui code and enhancements:
$SEISRES_HOME/gui/metagui/

Apache:
$APACHE_HOME/cgi-bin/
Zope.cgi
procmon.tcl
test-monitor
test-monitor.tcl
xplugin.pl

Zope:
$ZOPE_HOME/
start_zope
$ZOPE_HOME/Extensions/
makeScript.py
$ZOPE_HOME/lib/python/Products/
Squishdot/
TinyTable/

ZDConfera/

3. Environment variables used by this component.

Set the following environment variables pertaining to the notebook:

VARIABLE	VALUE	OPTIONS
TIXHOME	/usr/local/lib/tix4.1	Typically. If you use TIX in plugin.
TCL_PLUGIN_WISH	1	Needs to be set to 1 instead of default 0
SEISRESWISH_CLIENT_FLAG	1	Governs how the pioChoosers are run. Can have the following values: 0 - Run inside the current process. 1 - Launch, using exec. (The default value.) 2 - Run in the separate "seisreswish-client".
SEISRESWISH_SERVER_PORT	9876	If SEISRESWISH_CLIENT_FLAG is set to 2
SEISRESWISH_CLIENT_HOST	Ip of host	If SEISRESWISH_CLIENT_FLAG is set to 2

Table 12. *Shell environment variables for the notebook.*

4. Detailed definitions of file formats used by this component and examples, e.g., permission file.

4.1. Template files

Script template files contain %%name%%-delimited placeholder variables that get substituted for with the input values from a submission form. The idea is that it is easy to turn an existing script an any language into a generic script that can be instantiated with form values. An example template is blow:

```
#!/bin/ksh
# @ input = %%input%%
# @ output = %%output%%
# @ error =  %%error%%
# @ notify_user = %%notify%%
# @ class = %%class%%
# @ notification = %%notification%%
# @ checkpoint = %%checkpoint%%
# @ restart = %%restart%%
# @ requirements = (Arch == "R6000") && (OpSys == "AIX42") && (Adapter ==
"hps_user") &&  (Pool == 01)
# @ min_processors =%%min_processors%%
# @ max_processors =%%max_processors%%
# @ job_type = %%job_type%%
# @ queue

export MP_PROCS=24
export MP_RMPOOL401
export MP_EUIDEVICE=css0
export MP_EUILIB=us
export MP_STDOUTMODE=ordered
export MP_INFOLEVEL=3
export MP_LABELIO=yes
export MP_STDOUTMODE=ordered

/usr/bin/poe %%program%% %%parameter_file%%
# send notice
mail %%notify%% < %%output%%
mail %%notify%% < %%error%%
```

Table 11. *Example Script Template file for SP2 job submission*

4.2. Workflow graph example

```
<embed type="application/x-tcl"
script='

catch {policy trusted}
set urlHead
http://hog.ldeo.columbia.edu/SeisRes/SeisRes/SeisResSub/projects/Demos/demo_workflow_
st295
set baseurl http://hog.ldeo.columbia.edu/SeisRes/

#set baseurl http://hog/SeisRes/
option add *background AntiqueWhite1
option add *font        -Adobe-Helvetica-Medium-R-Normal--*-120-*
option add *bold_font   -Adobe-Helvetica-Bold-R-Normal--*-120-*
option add *menu_font   -Adobe-Helvetica-Bold-R-Normal--*-120-*
option add *italic_font -Adobe-Helvetica-Bold-O-Normal--*-120-*
option add *fixed_font  -*-courier-medium-r-*-*-14-*-*-*-*-*-*-*
option add *border1     1
#source $env(SEISRES_HOME)/gui/workflow/demowork.tcl

catch {policy trusted}
catch {cd I:/seisres/gui/workflow}
catch {cd $env(SEISRES_HOME)/gui/workflow}
. configure -bg  #FFA07A

set wffile $env(SEISRES_HOME)/gui/workflow/workflow.tcl                    ;# home
document
source $wffile


set nodeHorzGap 50
#
#
#
initWorkLoadCanv

if { ![info exists urlHead] } {
   set urlHead "http://hog.ldeo.columbia.edu/SeisRes/SeisRes/SeisResSub/ST295/Demo-10-
22-99"
}

proc testLoad { } {
   global urlHead baseurl
   # testInit
   set nodeUrl0 "${urlHead}/4D_RAI/synopsis.html"
   set nodeUrl1 "${urlHead}/ResChar/synopsis.html"
   set nodeUrl2 "${urlHead}/EarthModel/synopsis.html"
   set nodeUrl3 "${urlHead}/SimOpt/synopsis.html"
   set nodeUrl4 "${urlHead}/PetroPhysicsOpt/synopsis.html"
   set nodeUrl5 "${urlHead}/UpdateEarthModel/synopsis.html"
   set nodeUrl6 "${urlHead}/SeisMod/synopsis.html"
   #set nodeUrl7 "${urlHead}/PostProcessSeis/synopsis.html"
   set nodeUrl7 "${urlHead}/Differencing/synopsis.html"

   set paramUrl0 "${urlHead}/4D_RAI/4drai/4drai.html"
   set paramUrl1 "${urlHead}/ResChar/"
   set paramUrl2 "${urlHead}/EarthModel/"
   set paramUrl3 "${urlHead}/SimOpt/"
   set paramUrl4 "${urlHead}/PetroPhysicsOpt/"
   set paramUrl5 "${urlHead}/UpdateEarthModel/"
   set paramUrl6 "${urlHead}/SeisMod/"
   #set paramUrl7 "${urlHead}/PostProcessSeis/"
   set paramUrl7 "${urlHead}/Differencing/"

   prep_addNode "TEXT" "T0" "4D RAI" [list [list "brown" "I01" ] ] [list [list "green"
"O01" ] ] $nodeUrl0 $paramUrl0

   prep_addNode "TEXT" "T1" "EarthGm" [list [list "green" "I11" ] [list "brown" "I12" ]
[list "brown" "I13" ] ] [list [list "green" "O11" ] ] $nodeUrl1 $paramUrl1

   prep_addNode "TEXT" "T2" "Build Earth Model" [list [list "green" "I21" ] ]  [list  [list
"green" "O21" ] ] $nodeUrl2 $paramUrl2

   prep_addNode "TEXT" "T3" "Fluid Flow" [list [list "green" "I31" ] [list "brown"  "I32" ]]
[list [list  "green" "O31" ] [list "brown" "O32" ] [list "brown" "O33" ] ] $nodeUrl3
$paramUrl3

   prep_addNode "TEXT" "T4" "Petro Physics" [list [list "green" "I41" ] [list "brown" "I42"
] ]  [list [list "green" "O41" ] [list "brown" "O42" ] [list "brown" "O43" ] ] $nodeUrl4
$paramUrl4

   prep_addNode "TEXT" "T5" "Reassemble Earth Model"  [list [list "green" "I51" ] ]  [list
[list "green" "O51" ] ] $nodeUrl5 $paramUrl5

   prep_addNode "TEXT" "T6" "Omega FDM Model & Migration"  [list [list "green" "I61" ]
]  [list [list "green" "O61" ] ] $nodeUrl6 $paramUrl6

   prep_addNode "TEXT" "T7" "Seismic Differences" [list [list "green" "I71" ] ]  [list  [list
"brown" "O71" ] ] $nodeUrl7 $paramUrl7

   #prep_addNode "TEXT" "T8" "Sensitivity Analysis"  [list [list "green" "I81" ] ] ""
$nodeUrl8 $paramUrl8

   prep_addArc "A0" "T0" "O01" "T1" "I11"
   prep_addArc "A1" "T1" "O11" "T2" "I21"
   prep_addArc "A2" "T2" "O21" "T3" "I31"
   prep_addArc "A3" "T3" "O31" "T4" "I41"
   prep_addArc "A4" "T4" "O41" "T5" "I51"
   prep_addArc "A5" "T5" "O51" "T6" "I61"
   prep_addArc "A6" "T6" "O61" "T7" "I71"

   #Feedbacks

   prep_addArc "A8" "T3" "O32" "T3" "I32"
```

```
  prep_addArc "A9" "T4" "O42" "T4" "I42"
  #prep_addArc "A10" "T6" "O62" "T6" "I62"

  prep_addArc "A11" "T3" "O33" "T1" "I12"
  prep_addArc "A12" "T4" "O43" "T1" "I13"

  prep_addArc "A7" "T7" "O71" "T0" "I01"

  prep_complete
}
testLoad
'
width="900" height="340"  >
```

Table 15. *Workflow graph example.*

5. Definition and comments about classes and data structures used by this component.

Not relevant to the Notebook.

6. Class diagram (in PDF).

Not relevant to the Notebook.

7. Detailed documentation for the source code created automatically from source comments and header files.

Not relevant for the notebook.

8. Active Notebook Appendix A: Example of VTK in the Notebook Browser

The user specifies chair Visualization. The parameters of this technique are specified in a form. The "Build Pipeline" causes a post to the server, which then writes the following web content back:

```
<HTML>
<HEAD>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
  <META NAME="Author" CONTENT="Albert Boulanger">
  <META NAME="GENERATOR" CONTENT="Mozilla/4.04 [en] (WinNT; I)
[Netscape]">
  <TITLE>Untitled</TITLE>
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#FDEDAC" LINK="#0000EE"
VLINK="#551A8B" ALINK="#FF0000">
```

```
<H1>
Step 1: Visualize the Seismic via Chair Visualization</H1>

<HR SIZE=4 NOSHADE WIDTH="100%">

<h2>Step Parameters</h2>

<br><br>
<INPUT TYPE=Submit  ID="Buildit1" NAME="Buildit1" value="Build Pipeline">
</form>
<HR SIZE=2 NOSHADE WIDTH="100%">
<CENTER>

<embed type="application/x-tcl"
script='
catch {policy trusted}
catch {load vtktcl}

vtkImageReader reader
vtkImageGradient gradient
vtkImageViewer viewer
vtkImageConstantPad pad
vtkImageHSVToRGB hsv
vtkImageShiftScale ss

#reader-SetDataByteOrderToLittleEndian
#reader DebugOn
reader SetHeaderSize 65
reader SetDataExtent 0 100 0 100 0 175
reader SetFileDimensionality 3
reader SetDataScalarTypeToUnsignedChar
#reader SetFileName "\\Hog\aboulang\ox\4d-vrml\seg98\data\timb88.fld"
reader SetFileName "H:/st295/timb88.fld"
reader UpdateWholeExtent

vtkImagePermute mute
mute SetInput [reader GetOutput]
mute SetFilteredAxes 0 1 2

pad SetInput [mute GetOutput]
pad SetOutputNumberOfScalarComponents 3
pad SetConstant 255.0
```

```
hsv SetInput [pad GetOutput]

# ************************************************

# Create outline
vtkChairDisplay chair
chair SetInput [hsv GetOutput]
chair SetXNotchSize 40
chair SetYNotchSize 60
chair SetZNotchSize 20

vtkPolyDataMapper chairMapper
   chairMapper SetInput [chair GetOutput]

vtkActor chairActor
   chairActor SetMapper chairMapper

vtkTexture atext
atext SetInput [chair GetTextureOutput]
atext InterpolateOn

chairActor SetTexture atext
[chairActor GetProperty] SetAmbient 0.2


# create render window
vtkRenderWindow renWin
set ren [vtkTkRenderWidget .ren  -height 500 -width 500 -rw renWin]
BindTkRenderWidget $ren


set nx [scale .nx -from 0 -to 100 -res 1 -orient horizontal \
          -label "Notch X"]

set ny [scale .ny -from 0 -to 100 -res 1 -orient horizontal \
          -label "Notch Y"]

set nz [scale .nz -from 0 -to 175 -res 1 -orient horizontal \
          -label "Notch Z"]

set rframe [frame .rframe]

grid $rframe -sticky news
grid $ren - -sticky news
grid $nx $ny $nz -sticky news -padx 10 -ipady 5
pack propagate $rframe no


set renWin1 [$ren GetRenderWindow]


# Create renderer stuff
#
vtkRenderer ren1
ren1 SetAmbient 1 1 1
$renWin1 AddRenderer ren1


ren1 AddActor chairActor
ren1 SetBackground 0.1 0.2 0.4
$renWin1 Render


#iren SetUserMethod {wm deiconify .vtkInteract}
#iren Initialize


proc setXn {chair win amnt} {
   $chair SetXNotchSize $amnt
   $win Render
}

proc setYn {chair win amnt} {
   $chair SetYNotchSize $amnt
   $win Render
}

proc setZn {chair win amnt} {
   $chair SetZNotchSize $amnt
   $win Render
}


$nx   config -command "setXn chair $renWin1"
$ny   config -command "setYn chair $renWin1"
$nz   config -command "setZn chair $renWin1"

###END OF PLUGIN
```

```
'
width="90%" height="600">
</center>

</BODY>
</HTML>
```

Table 16. Example tclet script for vtk chair display.

This document contains a form and a tclet script, which is a simple VTK pipeline executed on the client. The browser widow is shown below:

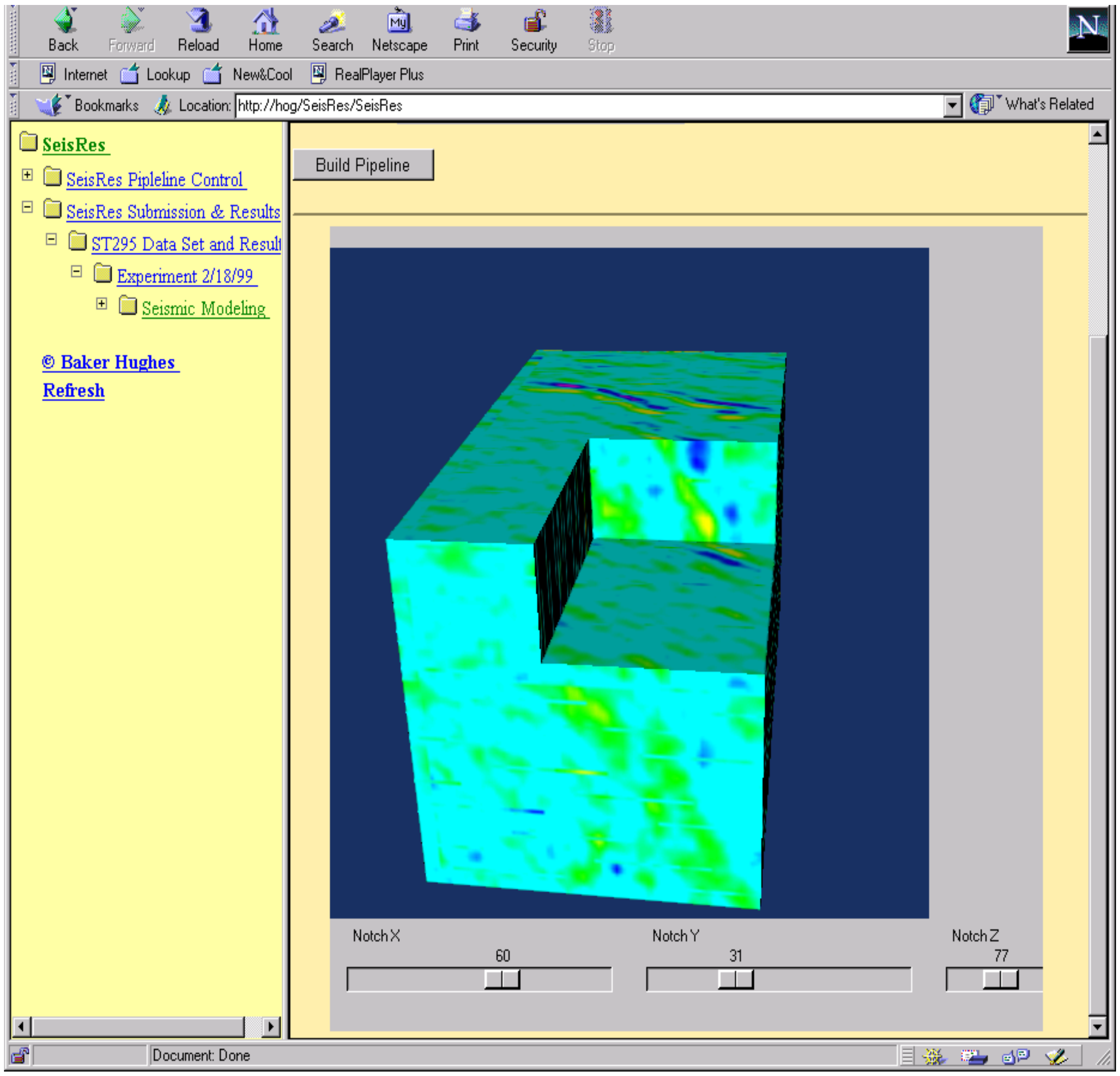


Figure 17. *VTK visualization pipline in Netscape Browser.*

9. MetaGui and Workflow Graph documentation.

See the documentation at **http://www.stratasys.com/software/metagui/** . Our modifications to that package are outlined below

```
*******************************************************
defvar testVar1 {
        -type       string
        -label      "Test1"
        -default    "fraz2"
        -balloon    "Test String"
    };
defvar testVar2 {
        -type       dirname
        -label      "Test2"
        -default    "fraz2"
        -balloon    "Test Dirname"
    };
defvar testVar3 {
        -type      pio
        -label     "PIO Value"
        -default   "PIO_B"
        -balloon   "Specify (Single) PIO Value."
    };
defvar testVar4 {
    -type      pioMultiple
    -label     "MultiPIO Value"
    -default   "PIO_B PIO_C"
    -balloon   "Specify (Multiple) PIO Value."
    };

*******************************************************
defvar testVar1 {
        -type       string
        -label      "Test1"
        -default    "fraz2"
        -balloon    "Test String"
    };
defvar testVar2 {
        -type       filename
        -label      "Open File Test"
        -default  "/usr/people/nichael/metagui-nlc/clwtest.text"
        -balloon    "Open Filename"
```

```
   };
defvar testVar3 {
      -type      filename
      -label     "Save File Test"
      -default "/usr/people/nichael/metagui-nlc/clwtest.text"
      -balloon    "Save Filename"
      -filemode   "save"
   };
defvar testVar4 {
      -type      point2
      -label     "Two Floats"
      -default { 10.0 20.0 3.0 }
      -balloon    "point2 (floats)"
   };
defvar testVar5 {
      -type      point2
      -label     "Two Ints"
      -default { 10 20 }
      -balloon    "point2 (integers)"
      -intonly    "true"
   };
defvar testVar6 {
      -type      point3
      -label     "Three Floats"
      -default { 1.0 2.0 3.0 }
      -balloon    "point3 (floats)"
   };
defvar testVar7 {
      -type      point3
      -label     "Three Ints"
      -default { 1 2 3 }
      -balloon    "point3 (integers)"
      -intonly    "true"
   };


*****************************************************
1] The PioChoosers now accept a keyword -pioKind to specify
the selection type.

# Example for specifying a single well_zone
defvar testVar_well {
   -type    pio
   -label   "PIO: well_zone"
   -default ""
   -pioKind "well_zone"
   -balloon "Specify well_zone Value."
   };

# Example for specifying a multiple well_zones
defvar testVar_well_multi {
   -type      pioMultiple
   -label    "PIO: Multi well_zone"
   -default  ""
   -pioKind  "well_zone"
   -balloon  "Specify multi well_multi Values."
   };
```

2] Note that the old "hierarchical" keyword is gone.

3] Below is a list of the currently legal values for -pioKind

```
EE
EarthGM
fluidSim
impedence
project
seismic
specs
well
well_bore
well_core
well_header
well_remark
well_production
well_sidewallcore
well_curve
well_perf
well_pick
well_table
well_velocity
well_zone
```

4] Finally well_perf and well_velocity have bugs (which I'll need to talk to liqing about).

5] There's also a browsing mode, (i.e. go anywhere in the repository, but

```
don't return a value) but I don't think you're interested in that right
now.

N

*****************************************************
defvar testVar_well {
   -type      pio
   -label "PIO: Well"
   -default "PIO_B"
   -pioKind "well"
   -balloon    "Specify well Value."
   };
defvar testVar_well_save {
   -type      pio
   -label   "PIO: Well (Save)"
   -default "PIO_B"
   -pioKind "well"
   -balloon  "Specify well Value."
   -filemode save
   };

defvar testVar_well_multi {
   -type      pioMultiple
   -label "PIO: Well_Multi"
   -default "PIO_B"
   -pioKind "well"
   -balloon    "Specify well_multi Value."
   };
defvar testVar_well_curve {
   -type      pio
   -label "PIO: Well_Curve"
   -default "PIO_B"
   -pioKind "well_curve"
   -balloon    "Specify well_curve Value."
   };
defvar testVar_well_zone {
   -type      pio
   -label "PIO: Well_Zone"
   -default "PIO_B"
   -pioKind "well_zone"
   -balloon    "Specify well_zone PIO Value."
   };
```

```
defvar testVar_seismic {
   -type      pio
   -label "PIO: Seismic"
   -default "PIO_B"
   -pioKind "seismic"
   -balloon    "Specify seismic PIO Value."
   };

defvar testVar_EE {
   -type      pio
   -label "PIO: EE"
   -default "PIO_B"
   -pioKind "browser"
   -balloon    "Specify EE PIO Value."
   };

*****************************************************
Using the multiple-file-chooser:

1] You need to souce the following file:


  source $env(METAGUIHOME)multiFileChooser.tcl



2] To call the chooser inside the metagui:

defvar testVar2 {
   -type      filenameMultiple
   -label "NLCMultipleFilename"
   -initDir "/usr/nichael/testoid/frazzle/"
   -default { foo bar bax }
   };
*****************************************************

1] In the pio code, to launch a pop-up browser, call:

  PioBrowser


2] To embed a pioBrowser, call:
  EmbedPioBrowser .tlwindow
```

Where .tlwindow is the id for the toplevel flag into which the browser is to be embedded.

****

Here's an example of the parameterizable-choosing stuff:

```
defvar testVar_well_param {
   -type      pioParamMultiple
   -label "PIO: Well_Param"
   -default { {fraz { parm1 f4 attribute f2 } } {baz { parm1 b4 attribute
b2 } } }
   -pioKind "well"
   -pioParam "attribute"
   -balloon    "Specify well_param Value."
   };
```

1] Note that and item in the list of values the form:

```
  { <pathname> { <key1> <val1> <key2> <val2> .... } }
```

2] Setting the pioParam above will specify which of the keys to edit.

3] Also, note that because this is being used in a list --which can be morphed into an array-- and since this has to be passed via command-line args, there can't be any spaces in either a key or in a value.

Specifically, the user-input for a new value first has an leading/ending whitespace trimmed off. Any whitespae in the middle of the string is replaced with underscores.

****

The Files+Parameters chooser is done and checked in.

This is basically just an extensin of the filenameMultiple type. An example of its use in the metagui is:

```
 defvar testVar3 {
    -type      filenameMultiple
    -label "ParameterizedFilenames"
    -initDir "~/guest/"
    -default { { ~/fraz { parm1 fp attribute fa } } { ~/baz { parm1 bp
attribute ba } } }
    -pioParam "attribute"
    };
```

This also has another parameter:

```
 -acceptNoParam
```

When this is "1" it will allow files to be specified without a parameter
value being speicified.

The default is "0"; i.e. each file must have the parameter specified.

****
At the bottom of the file:
  gui/metagui/pioRepos.tcl

is a proc:
  pioRepos_makeNewReposDir

This is handed the name of the "directory" to be made.

The two lines worth of stuff that I tried to do (and which I think aren't working in my build) is commented out.

An example of the use of this in metagui:

```
defvar testVar21 {
   -type  pioReposDir
   -pioKind "well"
   -label "pioReposDir"
   -dirmode "make"
   -hideFiles 1
   -default "well"
}
```

 -type:

```
"pioReposDir"


 -pioKind:
If this is provided, the directory works on that subdirectory.

If it is not provided, the entire repository can be browsed.


 -dirmode:
If the value is "make", this behaves as "repository/sub-directory
maker".

If this is not provided, the default behavior is to behave as
"sub-directory chooser".


 -hideFiles:
If the value is 1, then none of the actual files are displayed
in the chooser along with the "sub--directories".

If it is not provided, the default behavior is to display the files, but
they are not selectable (although, the files are describable and
visualizable).

(Also, if the files are displayed, their names are surround
with bracets: e.g.  "[mywell]" )


******************************************************

  - tpioDirect
Use the tix-list gadget to choose a directory.
  - tpioDirectMultiple
Ditto, multiple directories.

  - tpioRepos
Use the tix-list gadget to choose a repository.
  - tpioReposMultiple
Ditto, multiple repositories.

  - tpioFile
Use Liqing's tix-based dialog to (remotely) select a file.
  - tpioFileMultiple
```

```
Ditto, multiple files.

  - tpioObj
Use Liqing's tix-based dialog to (remotely) select an object.
(Also uses the parameters -typePattern.)
  - tpioObjMultiple
Ditto, multiple objects.


*****************************************************************

Stuff for the seireswish server:



The script to launch the seireswish server is in the file:

    gui/metagui/seisreswishServer.tcl


Environment variables:

   SEISRESWISH_SERVER_PORT
(Used by both the client and server.)
Port that the server uses (defaults to 9876).


   SEISRESWISH_CLIENT_HOST
(Used by the client.)
The host where the Server lives (defaults to "localhost").


   SEISRESWISH_CLIENT_FLAG
(Used by the client.)
Governs how the pioChoosers are ran.
Can have the following values:
   0 - Run inside the current process.
   1 - Launch, using exec.  (The default value.)
   2 - Run in the separate "seisreswish-client".


**********************************************************
```

To pass the port and host to the Tix-like remote file-selection gadget:

In the file gui/metagui/pioChooser.tcl

The proc tixFileChooser_1 is called with the port and the host,
but they need to be passed to the proc pio:createPioFileSelectDialog
when it is ready to accept them.

****

multiFilenameChooser_1 is the actual internal "guts" function that actually
does the real calling.

multiFilenameChooser is a function that will dispatch on how to call the
function (i.e. either "locally", or using the "seiesresWish server").

The second function should be defined in pioLaunch.tcl (Perhaps this is a
tcl_index problem? Maybe the index needs to be rebuilt for the metagui
directory.)

****

```
defvar testVar_string_multi {
   -type      pioStringMultiple
   -label "PIO: String_Multi"
   -default { "The third String" "This string" }
   -itemsList { "The first string" "The second String" "The third String" "This string" "That
string" "The very last string"}
   -balloon    "Specify multiple Strings."
   };
```

Done. This now takes a -lister arg:

```
  -lister someProcName
```

where someProcName returns a list of strings.

NOTE: In order to maintain compatibility with the other uses of -lister,
this proc takes one argument, the name of the defvar on which it is called.

****

1] I just checked some code that should handle the embedded browser with
the Seisreswish Server.

2] There is a new script to support this:

```
    gui/metagui/browseinwin2.tcl
```

```
*******************************************************
```

Table 17. *Our extensions to the metagui package.*

Here is a description on how to add new types to the metagui.

```
*******************************************************
```

Help on adding new pio types.......

```
*******************************************************
```

Add new types:
In the following I'll walk through the examples of adding two new
metagui-types: tpioDirect and tpioDirectMultiple.

The second type is "multi-valued"

```
**********
```

1] In file frameDisplay.tcl, proc updateEntry:
If this is a "multiple-valued" parameter, add a clause to the
if-cluster at near the bottom. (This loads the display-gadget.
The single-valued case is already taken care of.)

```
   } elseif { $type == "tpioDirectMultiple" } {
          # NLC--
          foreach item [varDisplayValue $varName] {
             $entryWidget insert end $item
          }
```

```
**********
```

2] In file frameDisplay.tcl, add a proc near the bottom.
This will have a name of the form: "<typename>Command"

This is what happens when the "button gets pressed".

For the "single-valued" type, this will look something like the following: (where "launch_PioDirectListChooser" is my proc that pops up the chooser, etc.)

```
proc    tpioDirectCommand { varName entryWidget } {
   global                $varName;
   upvar #0 $varName   varValue;


   set res [launch_PioDirectListChooser]

   if { $res != "__CANCEL__" } {
        set $varName $res;
   }

   $entryWidget configure -fg black;
   $entryWidget delete 0 end;
   $entryWidget insert 0 $varValue;
   $entryWidget xview end;
}
```

For the "multi-valued" type, this will look something like the following:

```
proc    tpioDirectMultipleCommand { varName entryWidget } {
   global                $varName;
   global mfcDefaultCancelFlag
   upvar #0 $varName   varValue;

   set typePattern "";
   set typePattern [vardata $varName "-typePattern"];


   set res [multiFilenameChooser $varValue "" "" "" "" 0 4 $typePattern]

   if { $res != $mfcDefaultCancelFlag } {
        set $varName $res;
   }

   $entryWidget configure -fg black;
   $entryWidget delete 0 end;

   foreach item $varValue {
        $entryWidget insert end $item
   };
}

**********
```

3] In file metaframe.tcl, in the proc displayvar, you need to add a
clause to the "if-cluster".

This controls how the value gets displayed on the metagui-display.

For the single-valued, this looks like the following:

```
        # NLC--
        #
        #       tpioDirect variables.
        #
   } elseif { $vartype == "tpioDirect" } {
        global  dirimage;
        if { $dirimage == "" } {
           global       gifLoc;
           set dirimage [image create bitmap -file
$gifLoc/pickdir.xbm];
        }

        frame $root.$varName;
        entry $root.$varName.entry;
        if { $entryWidth != "" } {
           $root.$varName.entry configure -width $entryWidth;
        } else {
           $root.$varName.entry configure -width 40;
        }

        $root.$varName.entry insert 0 [varDisplayValue $varName];
        $root.$varName.entry xview end;
        entryBindings $root.$varName.entry $varName;
        appendTrace $varName "updateEntry $root.$varName.entry";
        button $root.$varName.pick -image $dirimage -takefocus 0 -command
\
           "tpioDirectCommand $varName $root.$varName.entry";
        pack $root.$varName.entry -side left -expand 1 -fill x;
        pack $root.$varName.pick -side right -fill y;
        eval grid $root.$varName -row $row -column 2 -sticky ew $pad;
        set     widgetList "";
        lappend widgetList $root.$varName.entry;
        lappend widgetList $root.$varName.pick;
```

For the multi-valued, this looks like the following:

```
        # NLC--
        #
        #       tpioDirectMultiple variables.
        #
   } elseif { $vartype == "tpioDirectMultiple" } {
        global  dirimage;
        if { $dirimage == "" } {
           global       gifLoc;
           set dirimage [image create bitmap -file
$gifLoc/pickdir.xbm];
        }


        Scrolled_Listbox $root.$varName \
           $root.$varName.entry 5 \
           $root.$varName.pick "" $dirimage

        $root.$varName.pick configure  \
           -command "tpioDirectMultipleCommand $varName
$root.$varName.entry";

        foreach item [varDisplayValue $varName] {
           $root.$varName.entry insert end $item
        };

        entryBindings $root.$varName.entry $varName;
        appendTrace $varName "updateEntry $root.$varName.entry";

        eval grid $root.$varName -row $row -column 2 -sticky ew $pad;


        set     widgetList "";
        lappend widgetList $root.$varName.entry;
        lappend widgetList $root.$varName.pick;
```

****
4] In file metavar.tcl, in the proc "defvar":

- Search for the following:

```
   errormsg "Variable $name: Has no default value.";
```

In the conditional for the if-statement just before this, you
probably want to add a line like the following:

```
    $type != "tpioDirect" && \
```

(This will ensure that variables of this type do not have to have
default values.)

- Search for the following:

```
   errormsg "Variable $name: Illegal or missing variable type.";
```

Add a clause to the preceeding if-cluster, like the following:

```
    } elseif { $type == "tpioDirect" } {
```

(This does nothing; it just ensure the metagui will recognize this
type.)

****
5] In file "varValidate.tcl", add a proc near the bottom:

```
proc tpioDirectValidate { name value newValue {chkReadonly true}} {
   return [genericValidate $name $value $newValue $chkReadonly]
}
```

(I'm not completely sure when or how this gets used. But the metagui
seems to bitch if it's not there.)

Table 18. *How to add extensions to the metagui package*

Here is some documentation on the workflow graphing package

3] To alter the html code in a node:

```
  nodeRenderFromText-external nodeTag htmlText
```

****

1]   nodeRenderFromText_external { nodeTag htmlText }

Sets the node in a "text-like" display mode, where htmlText is a string to render in an HTML-like way.

2]   nodeResetUrl_external { nodeTag nodeUrl }

Reset the URL in the node.

3]   setNodeState_external { nodeTag state }

Set the display-state of the node.

State can be: "normal", "test" or the name of a color.

4]   pollNodeUrls { {updateInterval "" } }

Causes the URLS of the nodes to be re-loaded ever <updateInterval> milliseconds. Default is every 15secs.

Helper function:

5]   nodeTagToArrTag { nodeTag }

Converts nodeTag to nodeArrTag

***Table 19**. Some information on the workflow graphing package.*