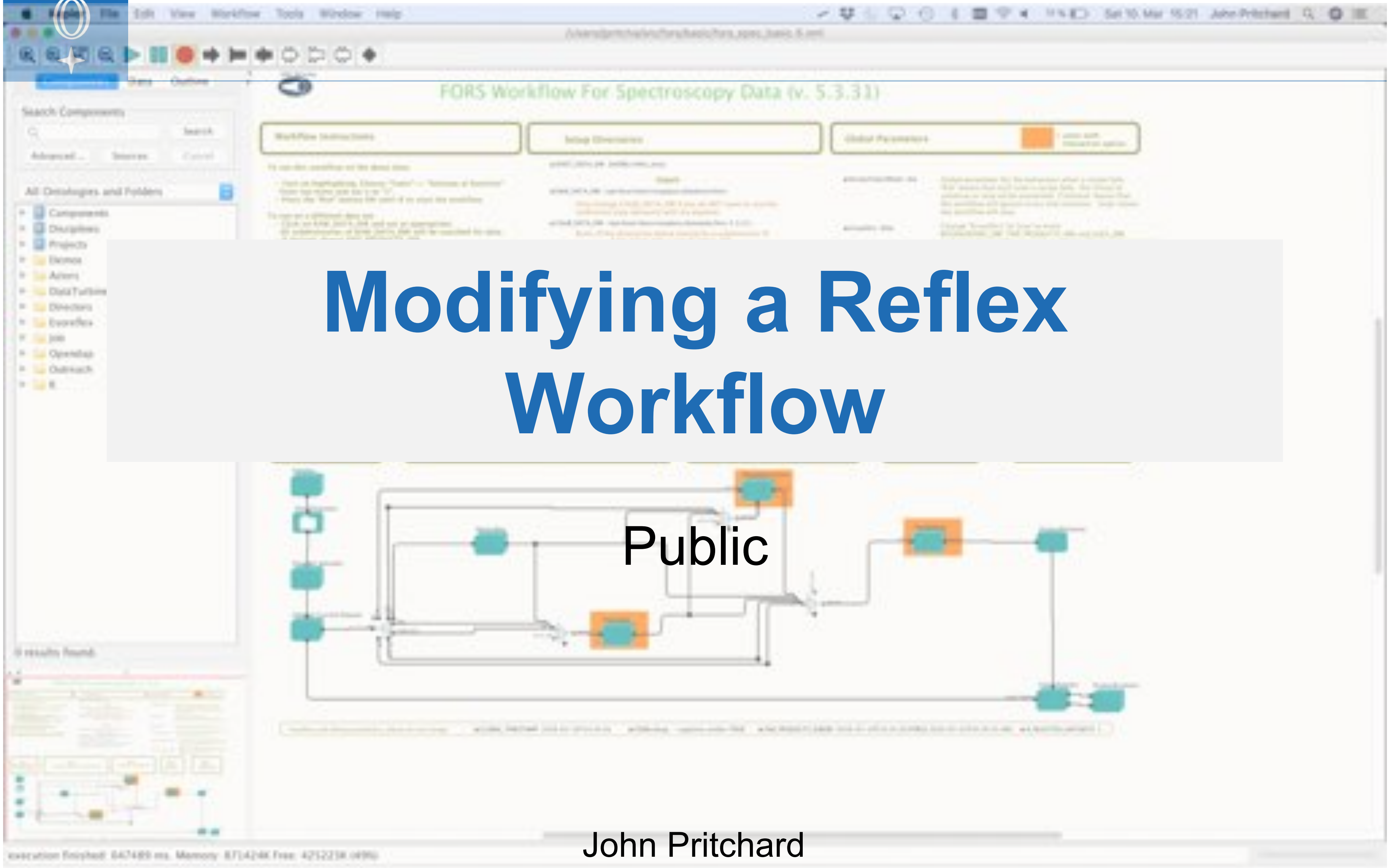


Modifying a Reflex Workflow

Public

John Pritchard

ESO-Reflex and Kepler

- **EsoReflex** is the *ESO Recipe Flexible Execution Workbench*, an environment to run ESO VLT pipelines which employs a workflow engine (***Kepler***) to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers.

Glossary

- Actors:
  - (mostly) Green folder like things…
  - ‘things’ that act on ‘stuff’, in an EsoReflex workflow, the important actors are the **recipe executor**s and the graphical display actors.
- Connections:
  - The lines joining things up
  - connect ‘things’ allowing the ‘stuff’ coming out of one ‘thing’ to ‘flow’ to another ‘thing’
- Relations:
  - The black diamonds here and there…
  - allow ‘stuff’ to be sent to/from multiple ‘things’

Notes

- The following is being done on a Mac, with fors-5.3.31 and giraf-2.16.2 workflows, installed via MacPorts
- If working on RPM based installation replace:

/opt/local/ → /usr/

- If working on a install-script based installation replace:

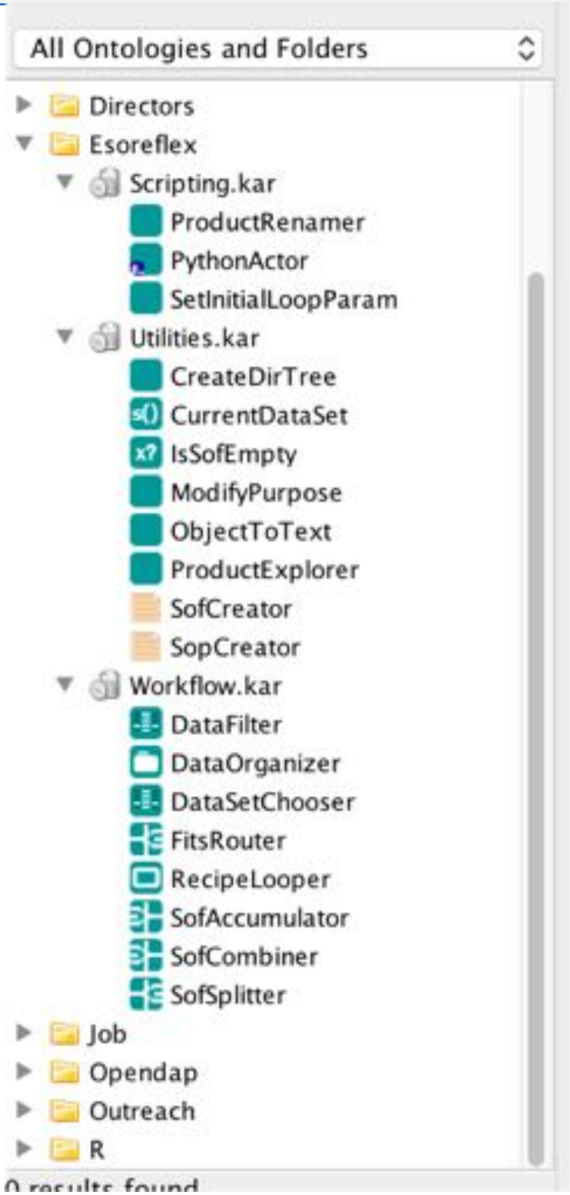
/opt/local/ → <dir_spec>/install/

SOFs – The ‘stuff’ that flows…

- SOFs: Set Of Files
  - original concept from esorex, a Filename+Category, e.g. bias.fits MASTER_BIAS
  - Extended for esoReflex by adding “purposes”, checksums, class…
- SOFs are the main objects that flow around an esoReflex workflow
  - they are passed from the output port of one actor to the input port one one or more other actors “via” the connectors.
  - The actors act on the input SOF(s), processing the files and/or modifying the SOF that they then output

“Kepler” & ESO Actors

- A number of the standard Kepler actors are useful, e.g. to set variables, or to replicate objects…
- But, most of the actors used in the esoReflex workflows are ESO specific and are found in the “Esoreflex” folder
- To add an actor to a work flow, simply drag and drop from the side bar onto the workflow…



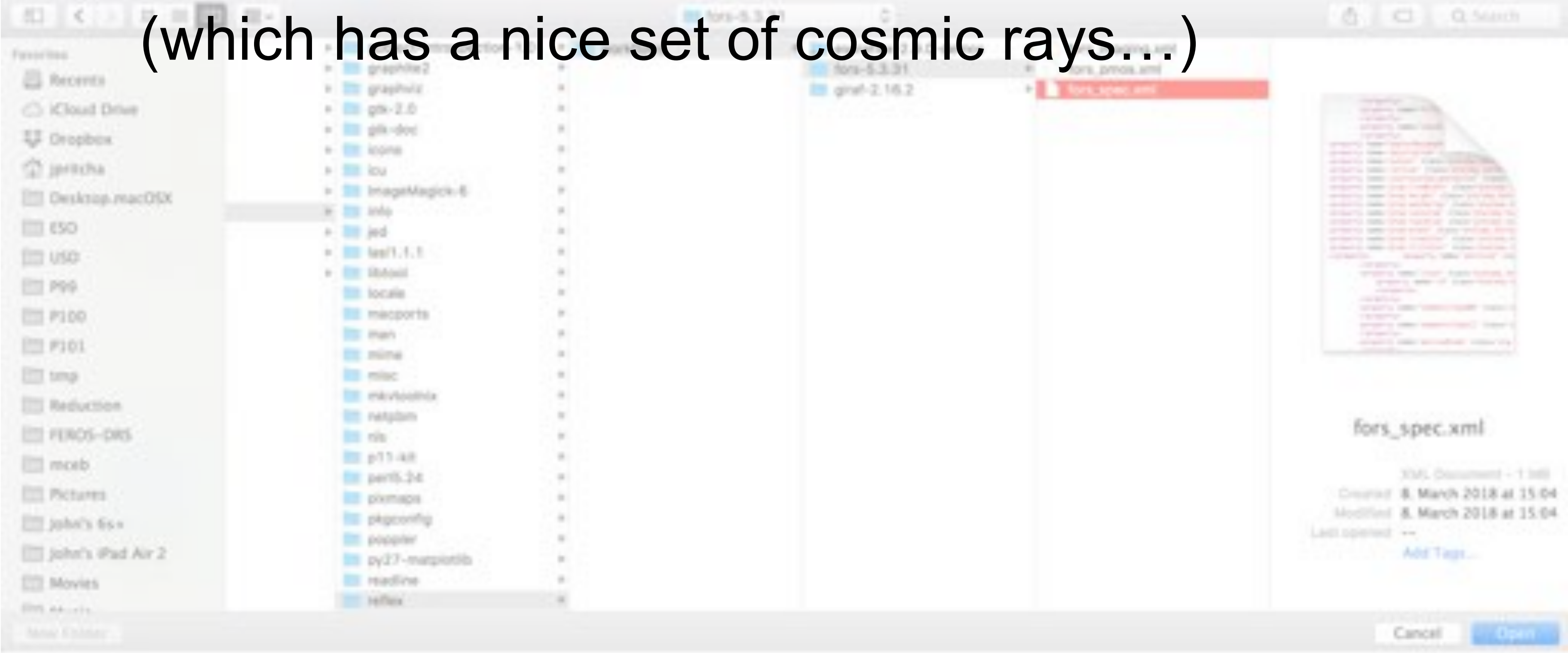
All Ontologies and Folders

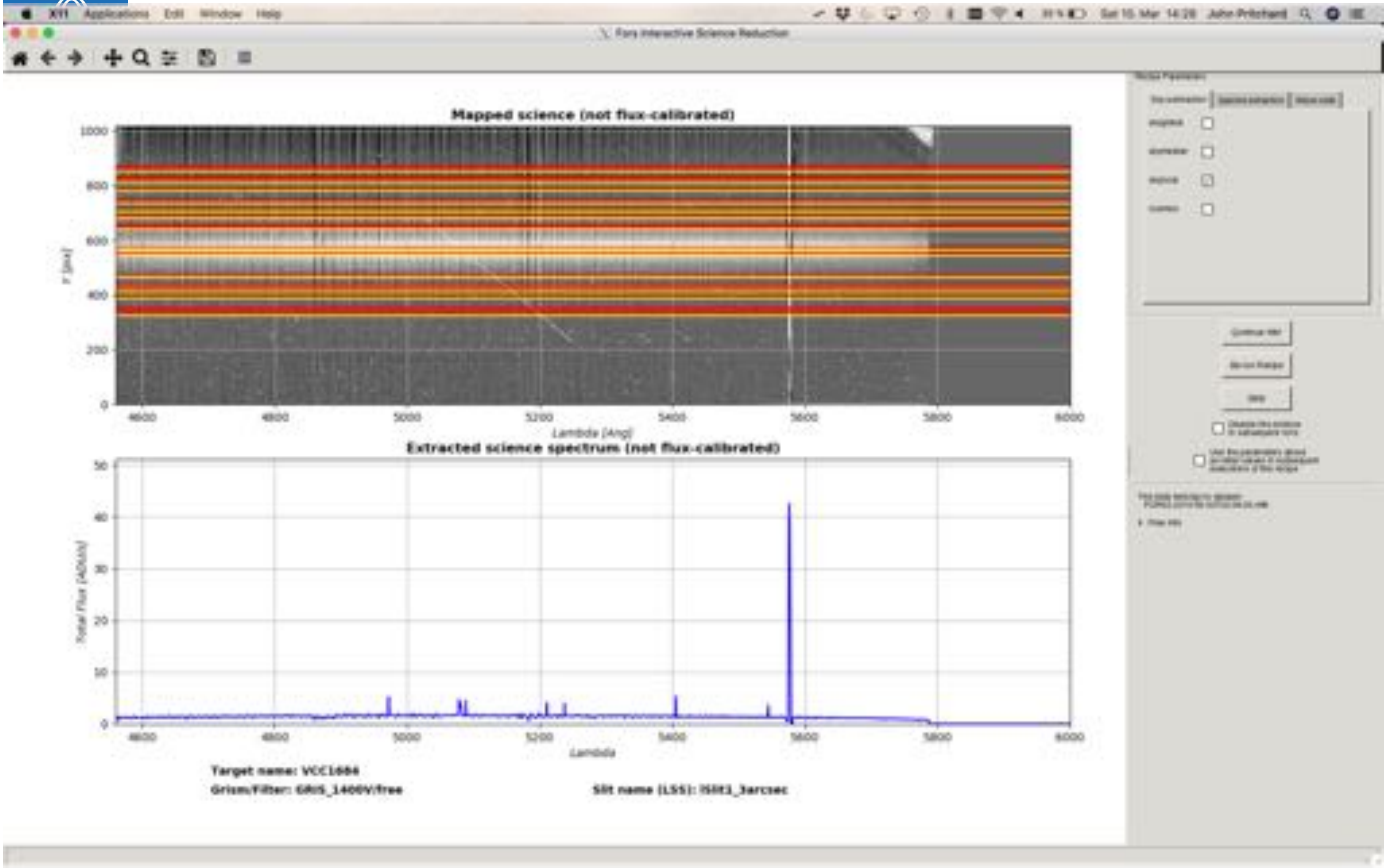
- Directors
- Esoreflex
  - Scripting.kar
    - ProductRenamer
    - PythonActor
    - SetInitialLoopParam
  - Utilities.kar
    - CreateDirTree
    - CurrentDataSet
    - IsSofEmpty
    - ModifyPurpose
    - ObjectToText
    - ProductExplorer
    - SofCreator
    - SopCreator
  - Workflow.kar
    - DataFilter
    - DataOrganizer
    - DataSetChooser
    - FitsRouter
    - RecipeLooper
    - SofAccumulator
    - SofCombiner
    - SofSplitter
- Job
- Opendap
- Outreach
- R

0 results found

Step 0: open the WKF and export as…

- Open the standard fors_spec workflow and then export as XML to a new directory.
- Then run the recipe selecting just the 4th dataset (which has a nice set of cosmic rays...)



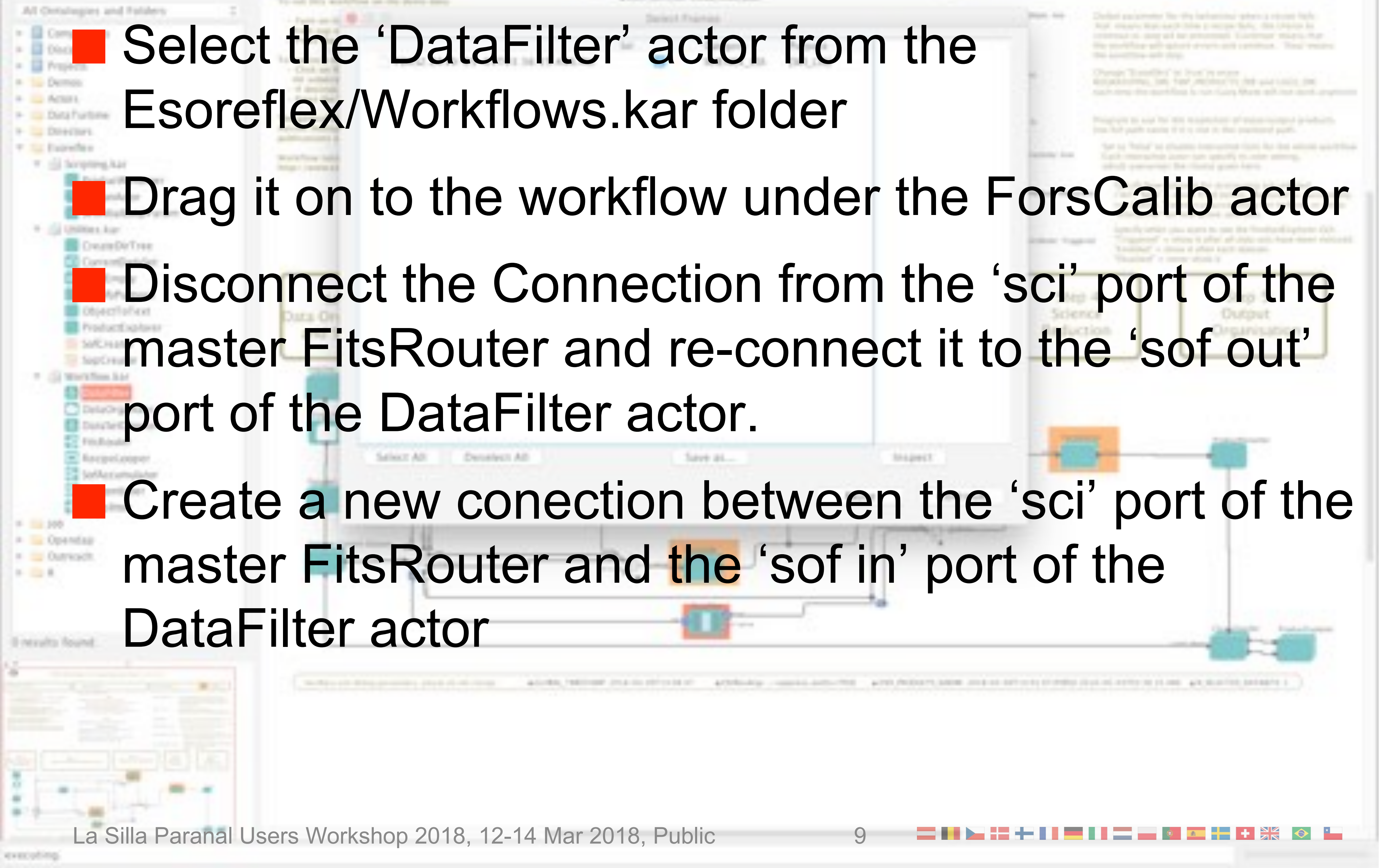


Target name: VCC1684

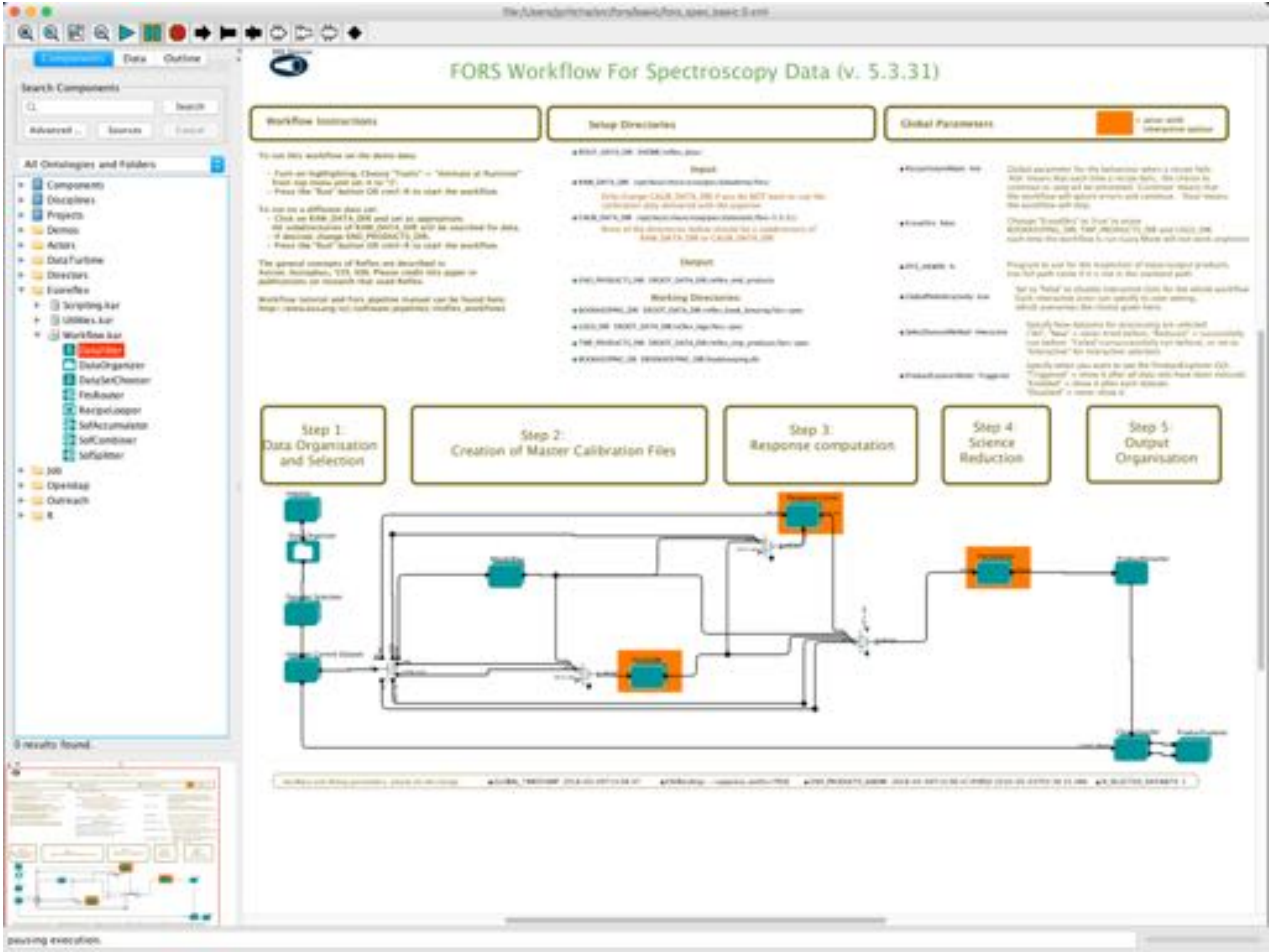
Grism/Filter: GRIS_1400V/free

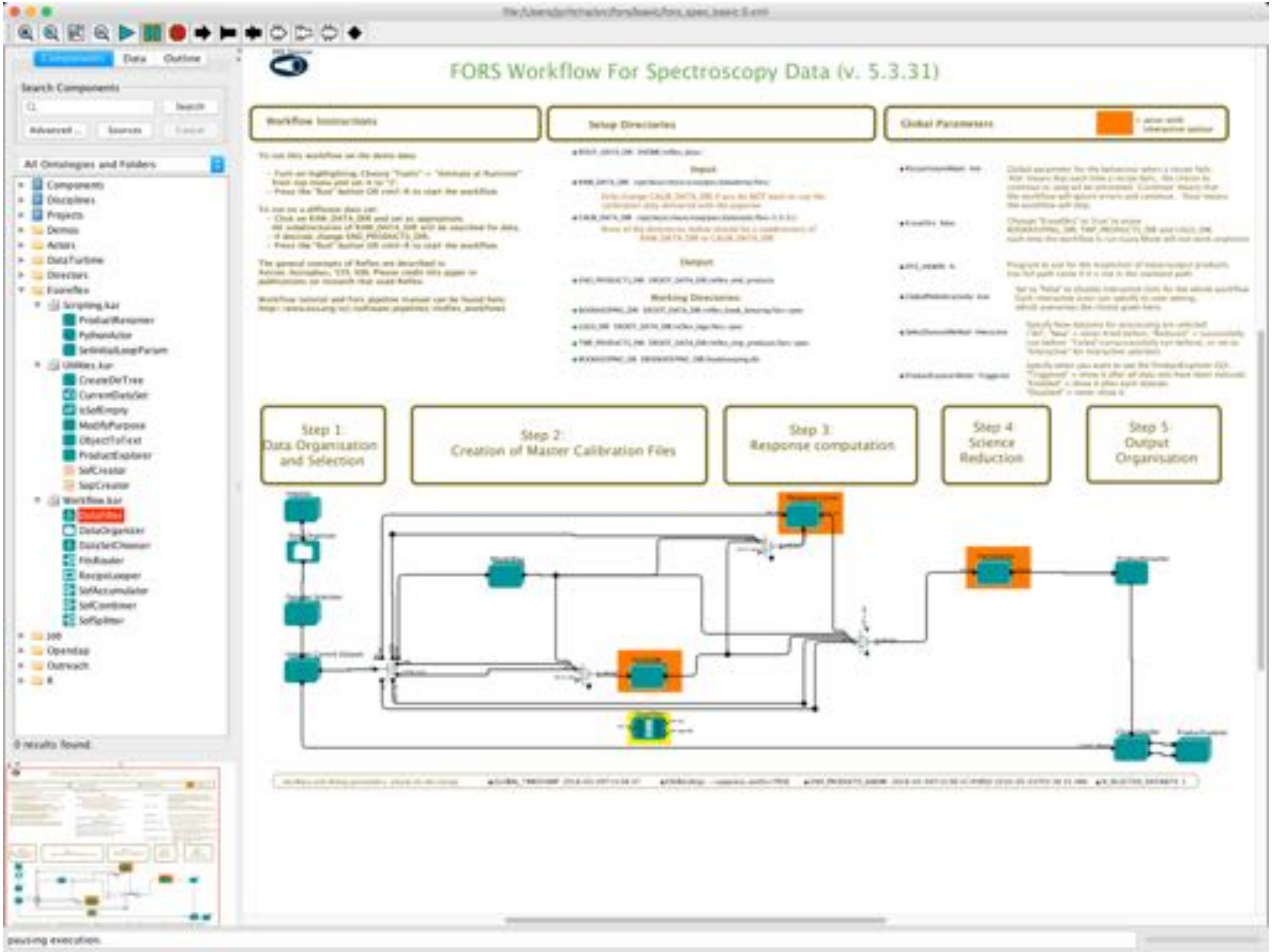
Slit name (LSS): lSlit1_1arcsec

Step 1: add a ‘simple’ actor

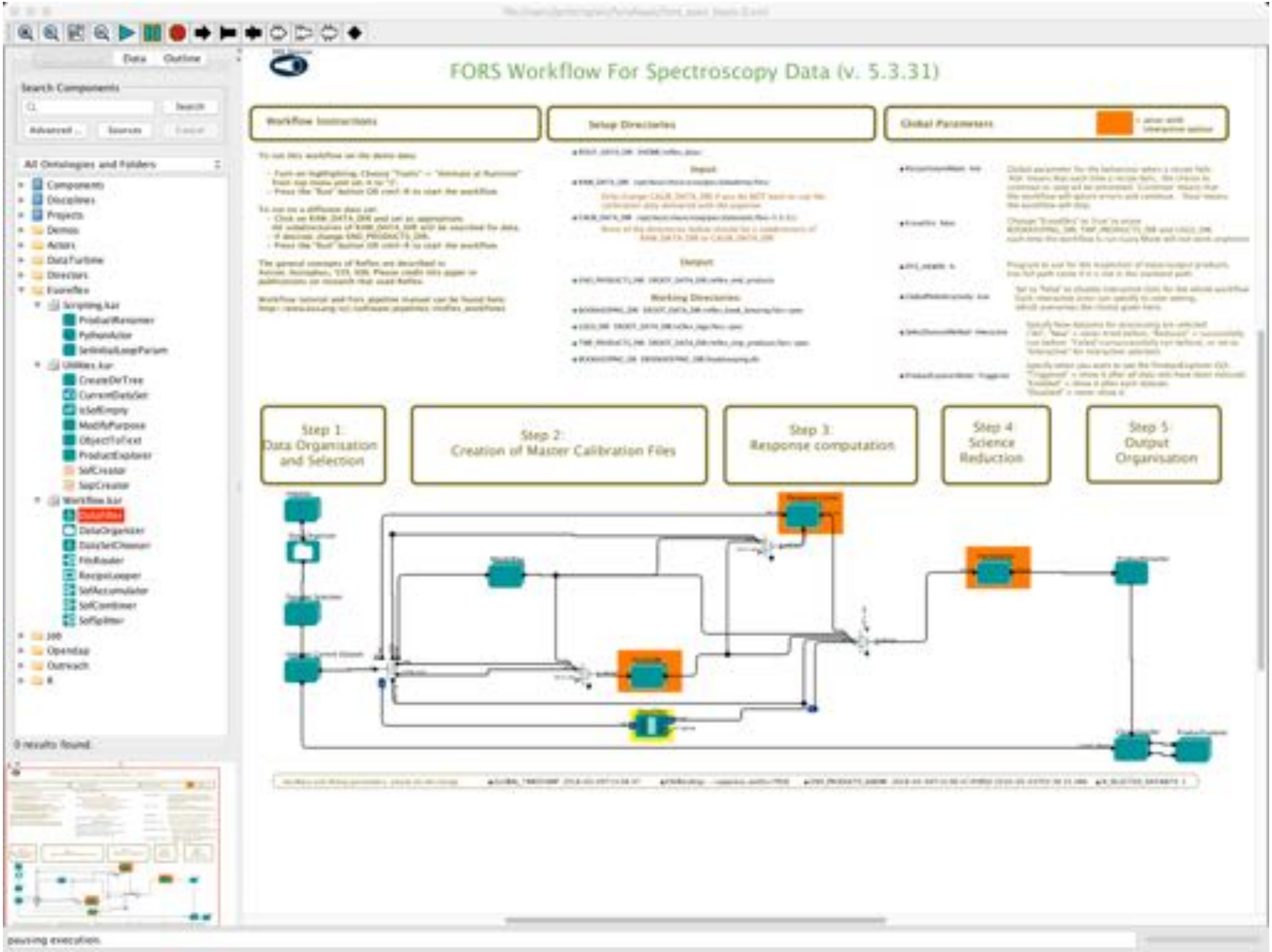


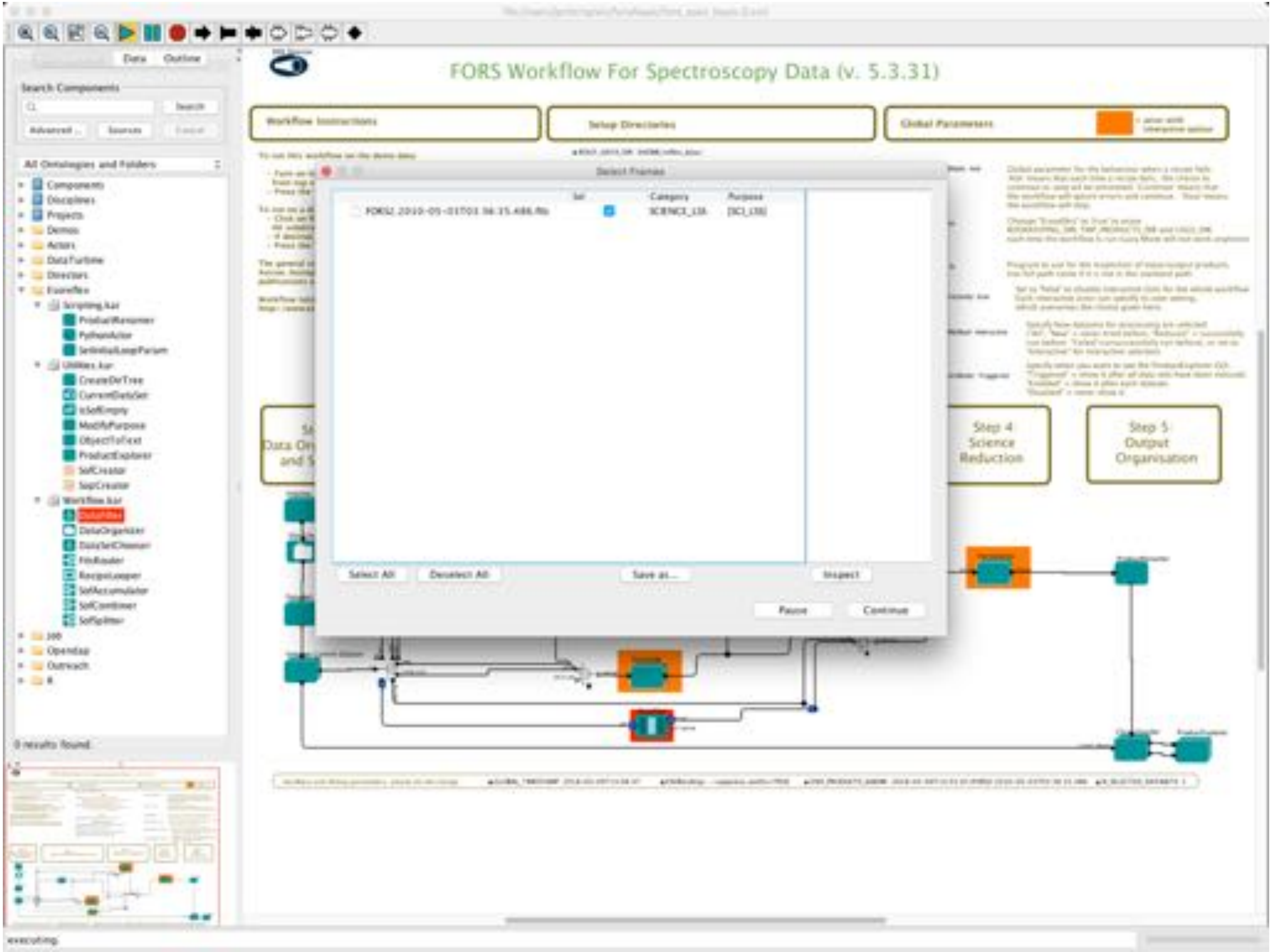
- Select the ‘DataFilter’ actor from the Esoreflex/Workflows.kar folder
- Drag it on to the workflow under the ForsCalib actor
- Disconnect the Connection from the ‘sci’ port of the master FitsRouter and re-connect it to the ‘sof out’ port of the DataFilter actor.
- Create a new conection between the ‘sci’ port of the master FitsRouter and the ‘sof in’ port of the DataFilter actor


Components
Data
Outline
Search Components
Search
Advanced
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Esoreflex
Scripting.kar
Utilities.kar
Workflow.kar
DataFilter
DataOrganizer
DataSetChooser
FitsRouter
RecipeLooper
SofAccumulator
SofCombiner
SofSplitter
JOB
Outreach
R
0 results found.
FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Input
Output
Working Directories
Global Parameters
Step 1:
Data Organisation and Selection
Step 2:
Creation of Master Calibration Files
Step 3:
Response computation
Step 4:
Science Reduction
Step 5:
Output Organisation
pausing execution



FORS Workflow For Spectroscopy Data (v. 5.3.31)


Data
Outline
Search Components
Search
Advanced ...
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Esoreflex
Scripting.kar
ProductRenamer
PythonActor
SetInitialLoopParam
Utilities.kar
CreateDirTree
CurrentDataSet
IsSofEmpty
ModifyPurpose
ObjectToText
ProductExplorer
SofCreator
SopCreator
Workflow.kar
DataFilter
DataOrganizer
DataSetChooser
FitsRouter
RecipeLooper
SofAccumulator
SofCombiner
SofSplitter
Job
Opendap
Outreach
R
0 results found
FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Global Parameters
Input
Output
Working Directories
Step 1:
Data Organisation and Selection
Step 2:
Creation of Master Calibration Files
Step 3:
Response computation
Step 4
Science Reduction
Step 5
Output Organisation
pausing execution

FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Global Parameters
Select Frames
Sel
Category
Purpose
FORS2.2010-05-01T03:36:15.486.fits
SCIENCE_LSS
[SCI_LSS]
Select All
Deselect All
Save as...
Inspect
Pause
Continue
Step 4
Science
Reduction
Step 5
Output
Organisation
Search Components
Search
Advanced
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Esoreflex
Scripting.kar
ProductRenamer
PythonActor
SetInitialLoopParam
Utilities.kar
CreateDirTree
CurrentDataSet
IsSofEmpty
ModifyPurpose
ObjectToText
ProductExplorer
SofCreator
SopCreator
Workflow.kar
DataFilter
DataOrganizer
DataSetChooser
FitsRouter
RecipeLooper
SofAccumulator
SofCombiner
SofSplitter
Opendap
Outreach
R
0 results found.
Data
Outline
executing

The end…

- And that’s “How to modify a workflow…”

…Not the end…

- But seriously…

Step 2.1: Add a basic Python Actor

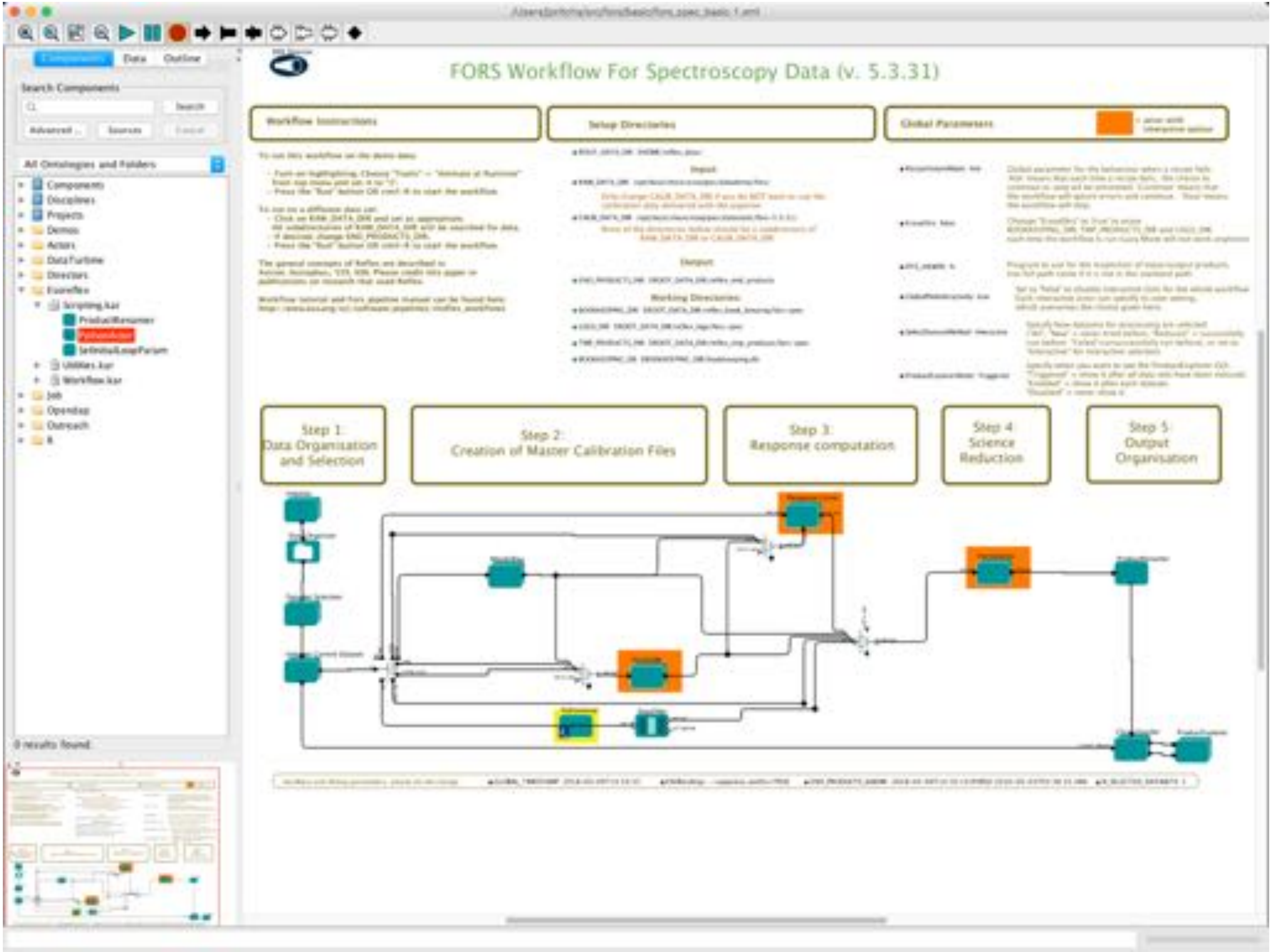
- Select the ‘PythonActor’ actor from the Esoreflex/Scripting.kar folder
- Drag it on to the workflow beside and to the left of the DataFilter actor
- Note it (currently) has no ports…
- In a terminal do:

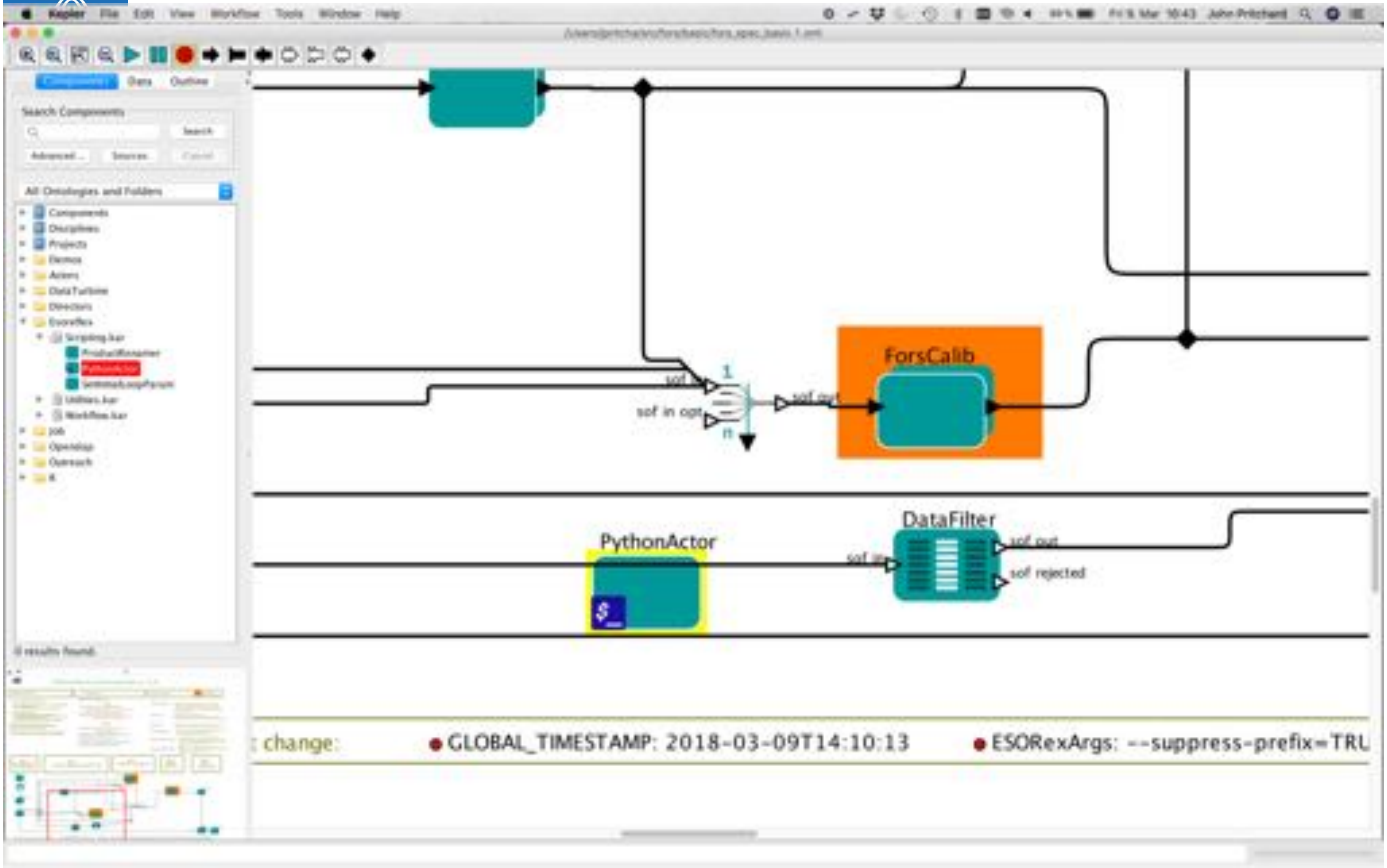
```
cp /opt/local/share/esopipes/giraf-2.16.2/reflex/giraf_in_out.py fors_in_out.py
```

- Edit it (double click, or right click and select Edit) and set ”Python script” to fors_in_out.py
- Note how the two ports appear...

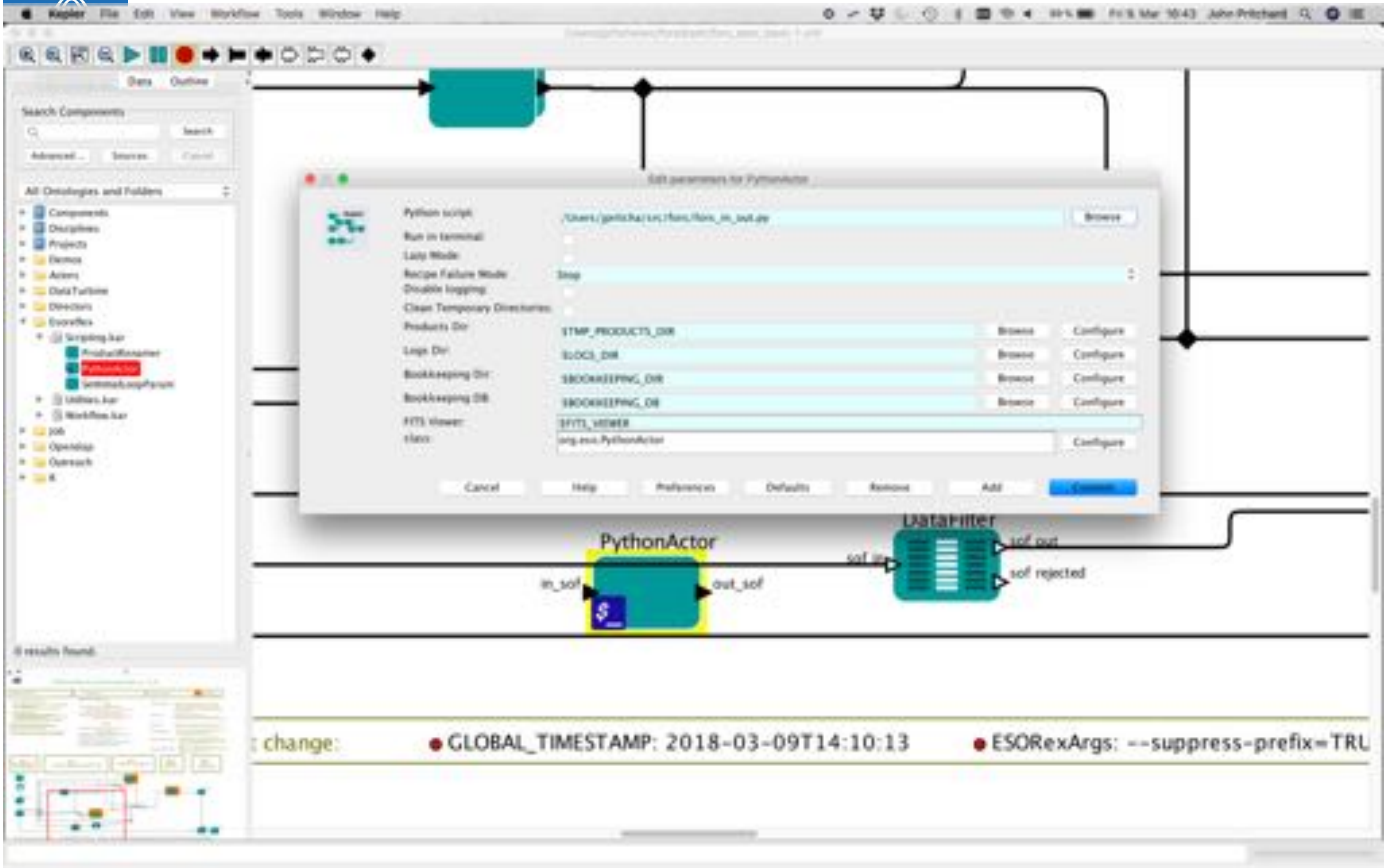
Step 2.1: Add a basic Python Actor

- Disconnect the Connection from the ‘sof in’ port of the DataFilter actor and re-connect it to the ‘sof out’ port of the PythonActor actor.
- Create a new conection between the ‘sof out’ port of the PythonActor actor and the ‘sof in’ port of the DataFilter actor
- Right button Click on the PythonActor actor and select ‘Customize name’.
- Set the name to InOut.
- The InOut actor doesn’t do anything, except write the in and out SOFs to the bookkeeping directory.


Components
Data
Outline
Search Components
Search
Advanced ...
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Esoreflex
Scripting.kar
ProductRenamer
PythonActor
SetInitialLoopParam
Utilities.kar
Workflow.kar
Job
Opendap
Outreach
R
0 results found.
FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Global Parameters
Input
Output
Working Directories
Step 1:
Data Organisation and Selection
Step 2:
Creation of Master Calibration Files
Step 3:
Response computation
Step 4:
Science Reduction
Step 5:
Output Organisation



change: ● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13 ● ESORexArgs: --suppress-prefix=TRU



Edit parameters for PythonActor

Recipe Failure Mode: Stop

Products Dir: $TMP_PRODUCTS_DIR

Logs Dir: $LOGS_DIR

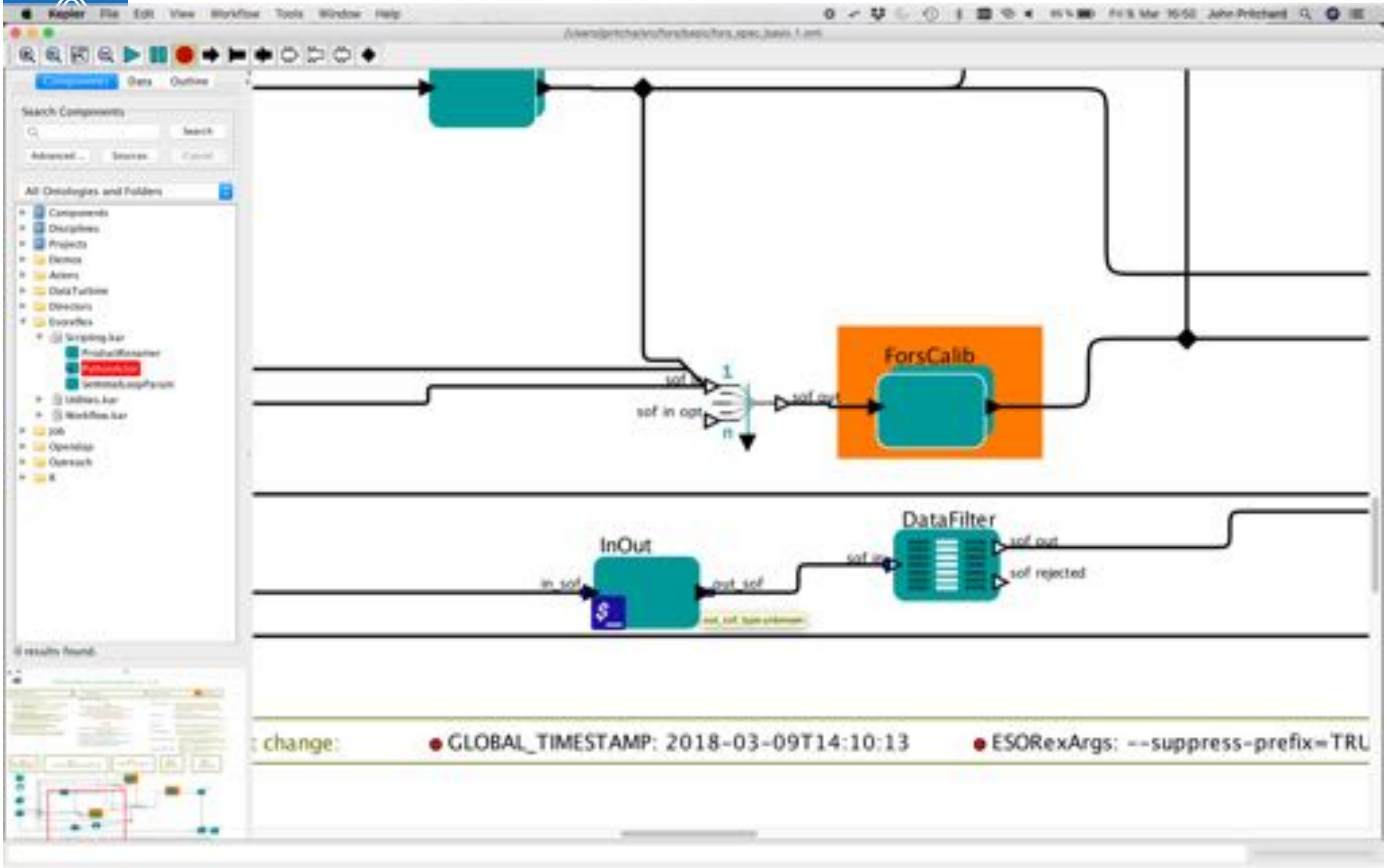
Bookkeeping Dir: $BOOKKEEPING_DIR

Bookkeeping DB: $BOOKKEEPING_DB

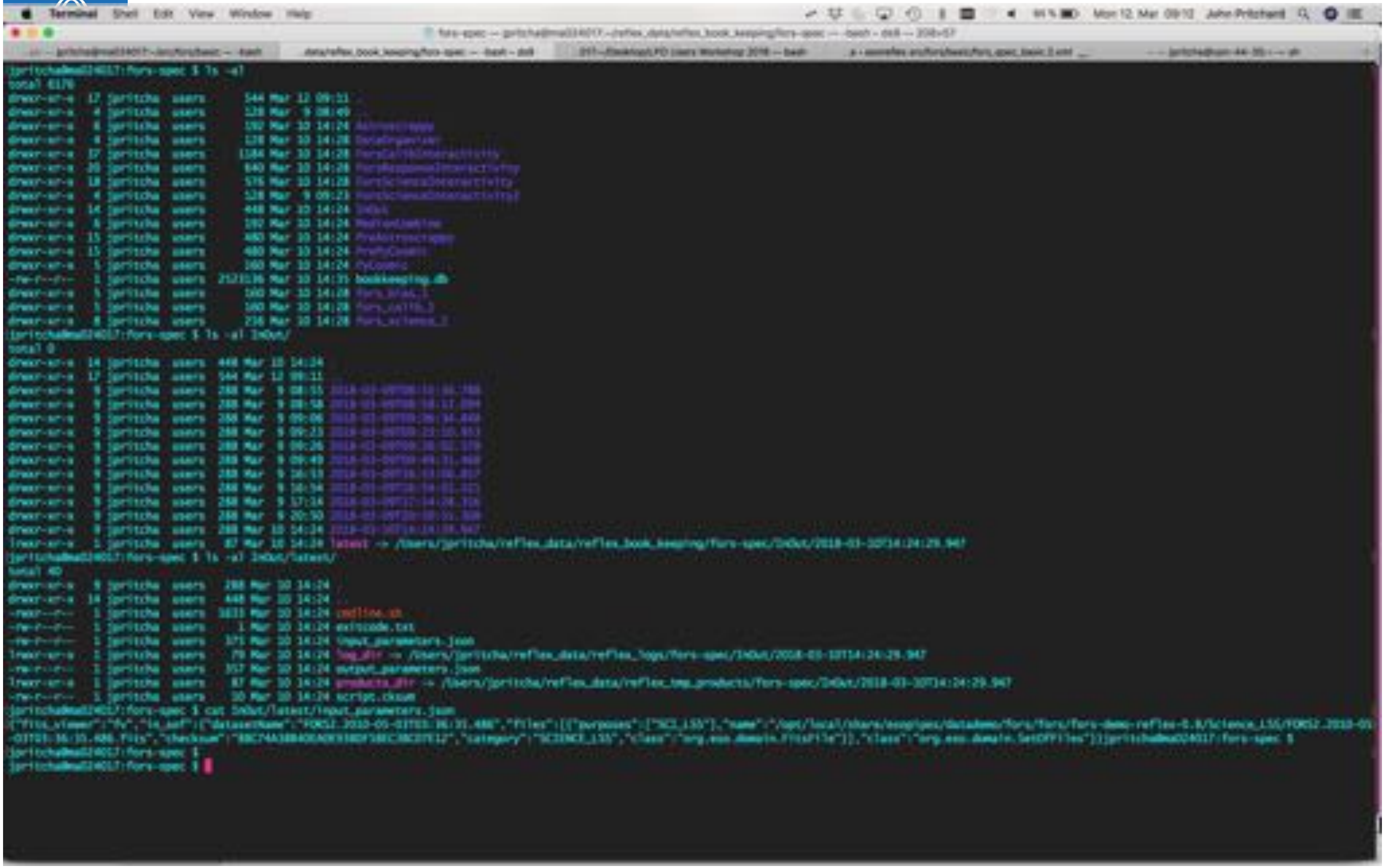
FITS Viewer: $FITS_VIEWER

class: org.eso.PythonActor

change: ● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13 ● ESORexArgs: --suppress-prefix=TRU



change: ● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13 ● ESORexArgs: --suppress-prefix=TRU

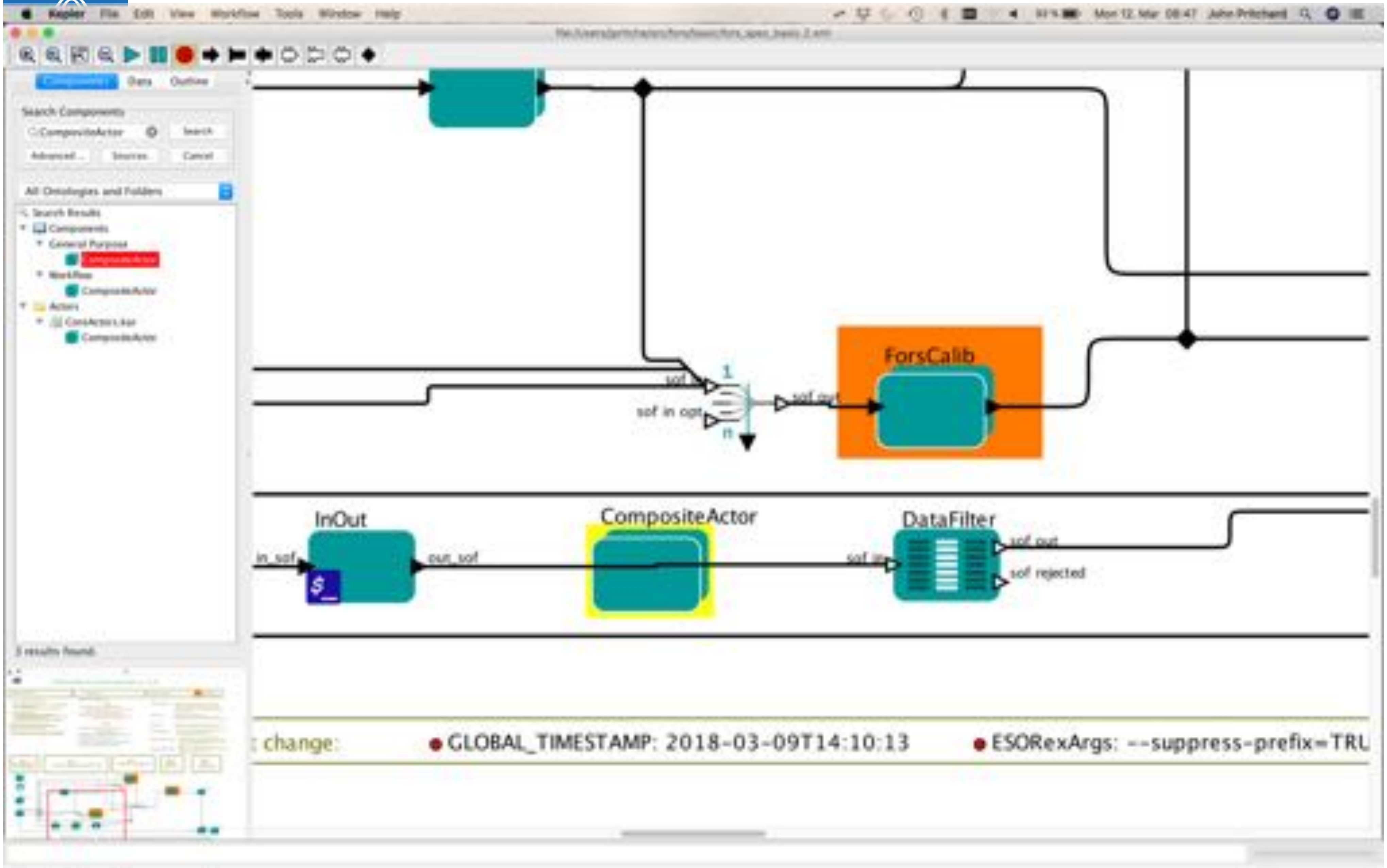


Step 2.2: Add a Recipe Executor

- The Esoreflex Recipe Executor is a special case.
- There is no template actor to drag and drop.
- Instead… as per the Reflex Workflow Development Guide (https://www.eso.org/sci/software/reflex):

The RecipeExecuter should be instantiated using the Tools -> Instantiate Component option. The component should be called org.eso.RecipeExecuter. This will show a list of available recipes as seen by the esorex command which is in the current path. If your recipe is not shown there, check your esorex and pipeline installation.

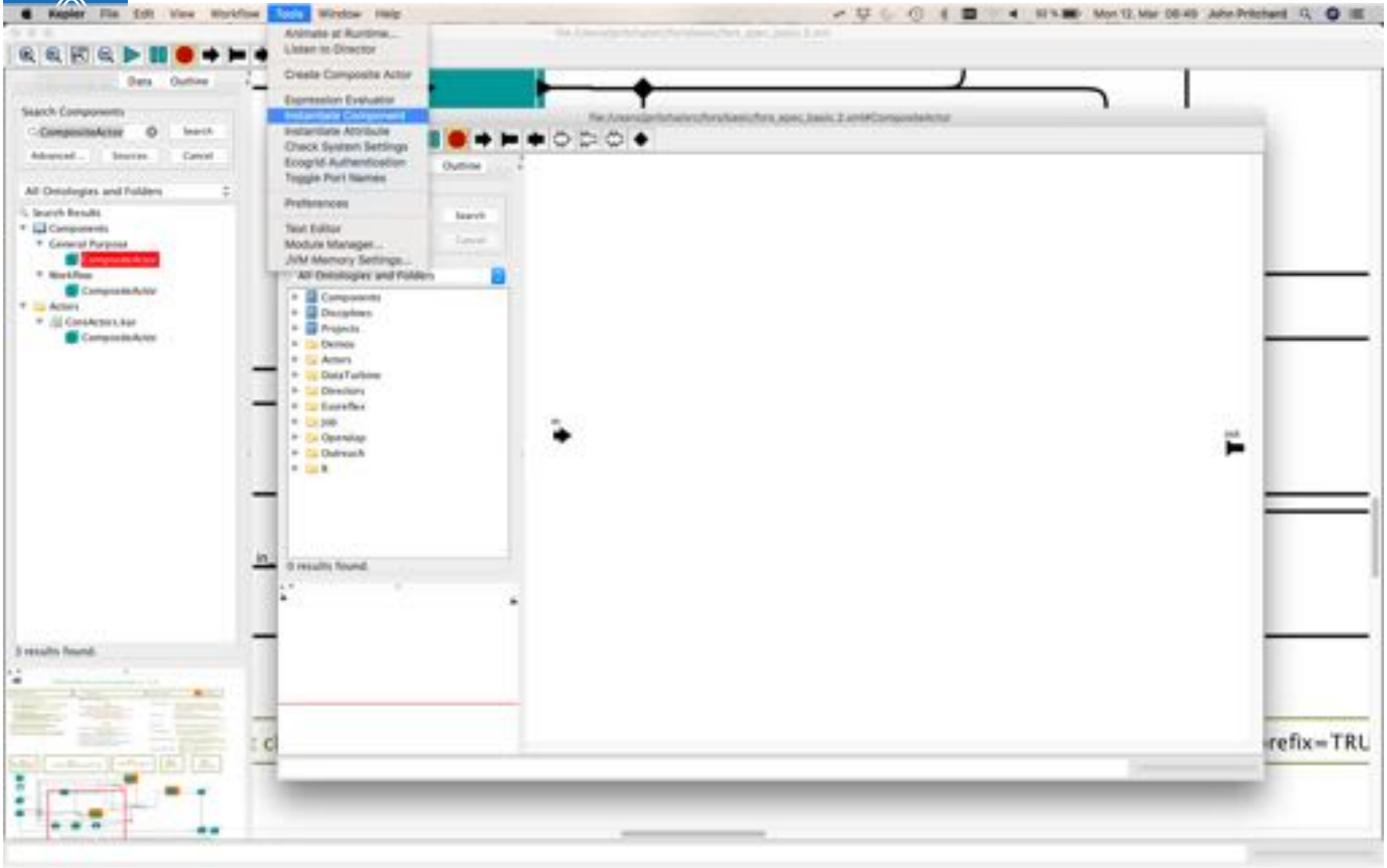
- Add inside a CompositeActor, together with a sofSplitter and a sofAccumulator.

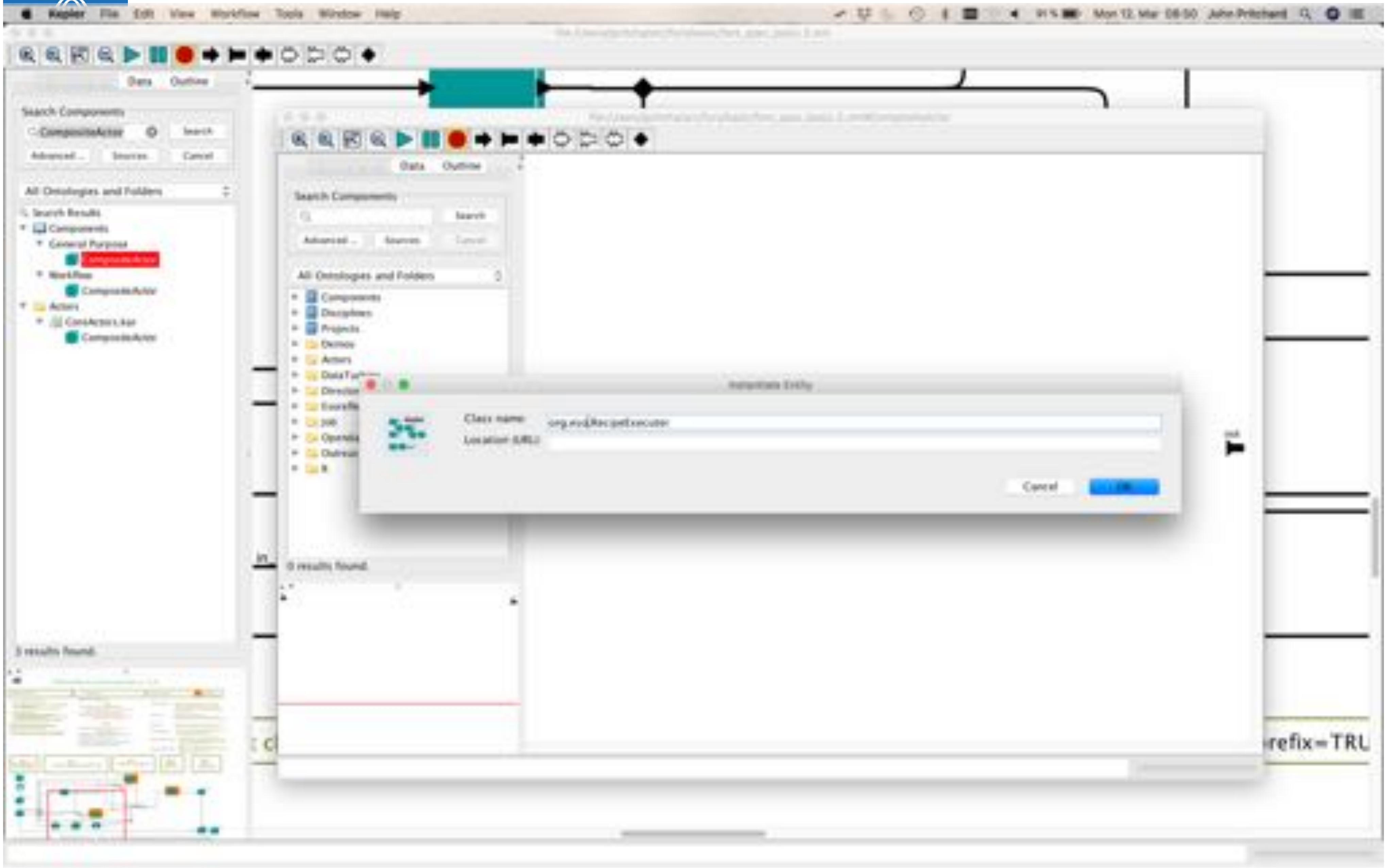


change:

● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13

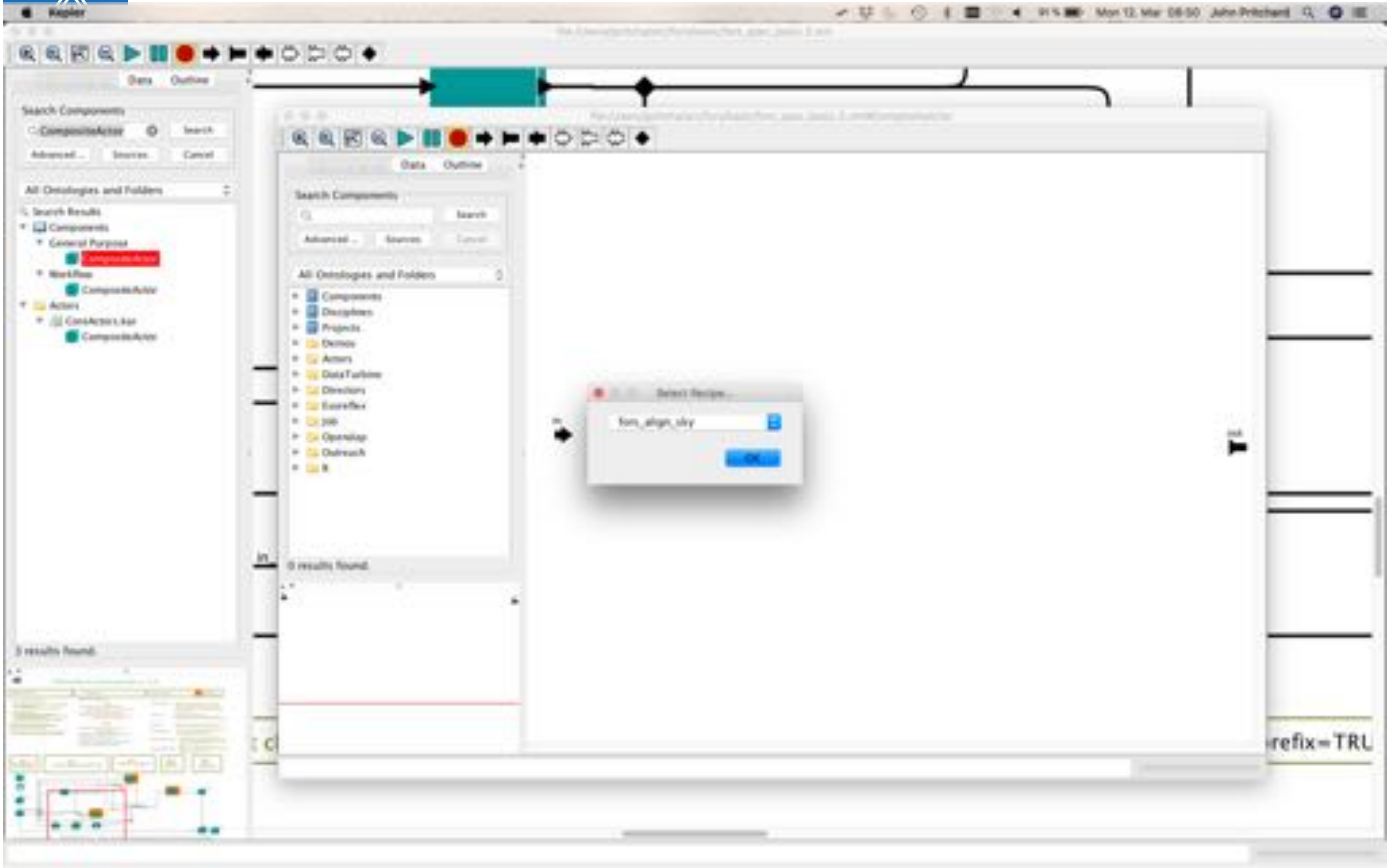
● ESORexArgs: --suppress-prefix=TRU


Kepler
File
Edit
View
Workflow
Tools
Window
Help
Animate at Runtime...
Listen to Director
Create Composite Actor
Expression Evaluator
Instantiate Component
Instantiate Attribute
Check System Settings
Ecogrid Authentication
Toggle Port Names
Preferences
Text Editor
Module Manager...
JVM Memory Settings...
Data
Outline
Search Components
CompositeActor
Search
Advanced...
Sources...
Cancel
All Ontologies and Folders
Search Results
Components
General Purpose
CompositeActor
Workflow
CompositeActor
Actors
CoreActors.kar
CompositeActor
3 results found.
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Esoreflex
job
Openalap
Outreach
R
0 results found.
in
refix=TRU



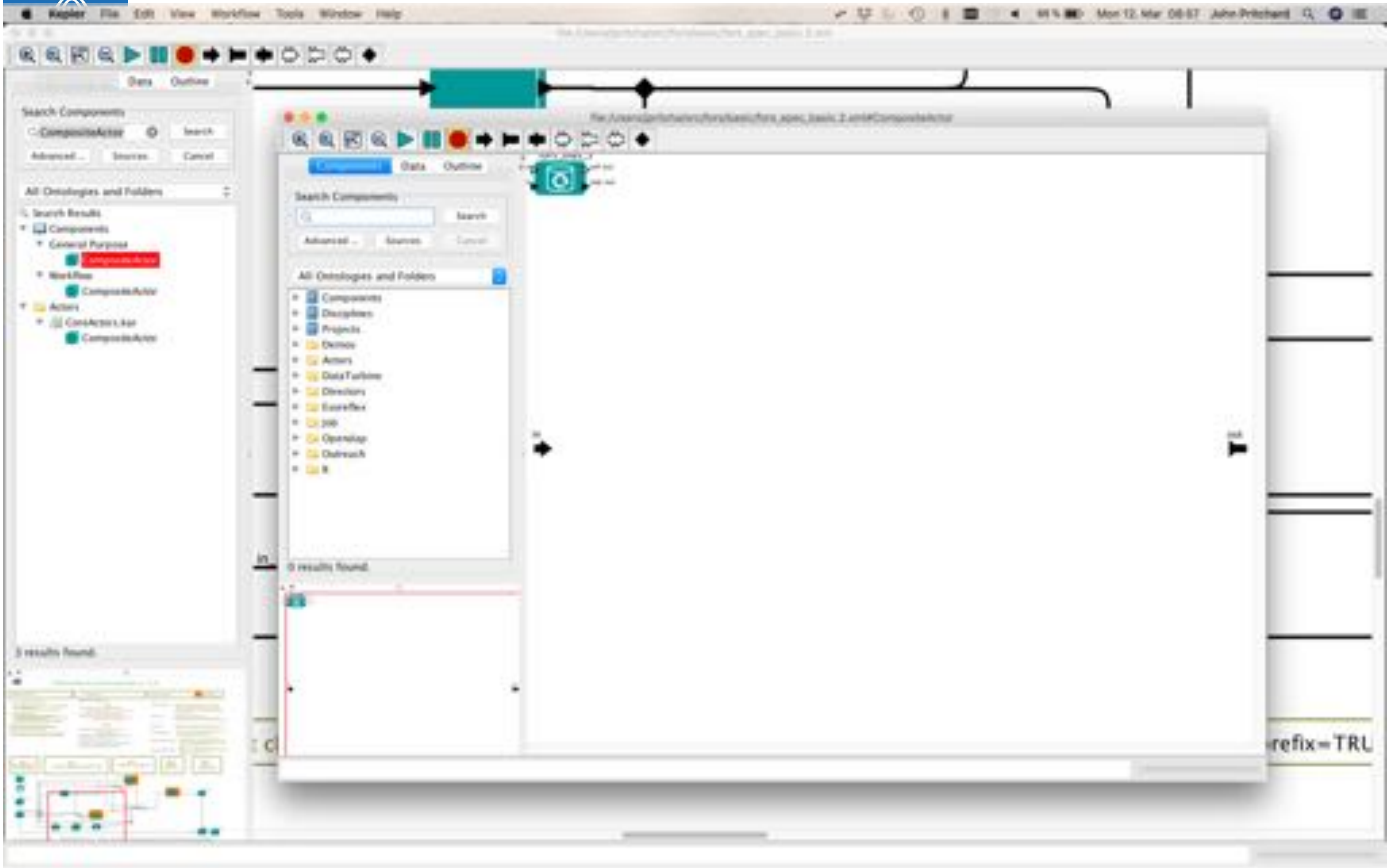
Class name: org.eso.RecipeExecuter

Location (URL):



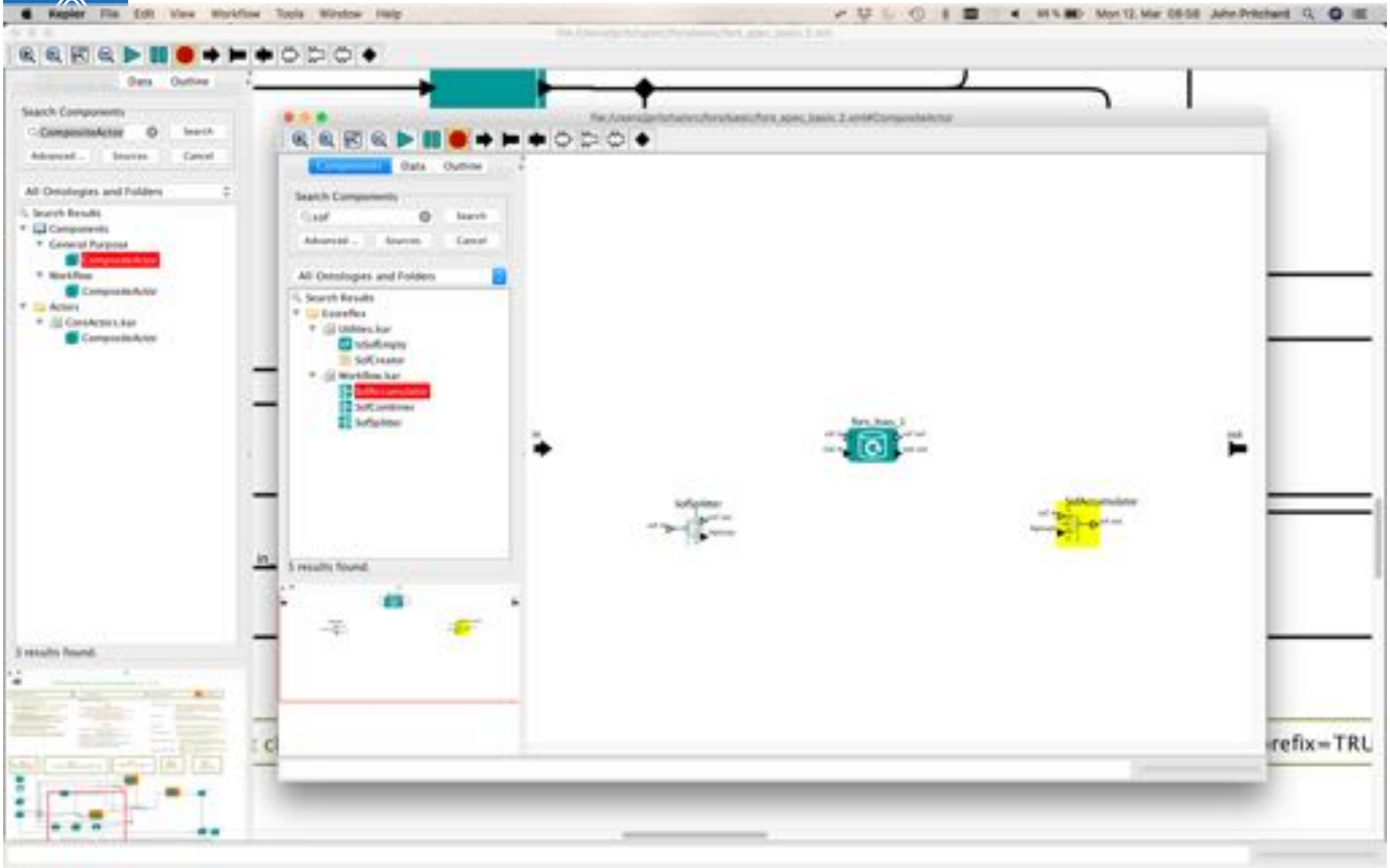
Select Recipe...

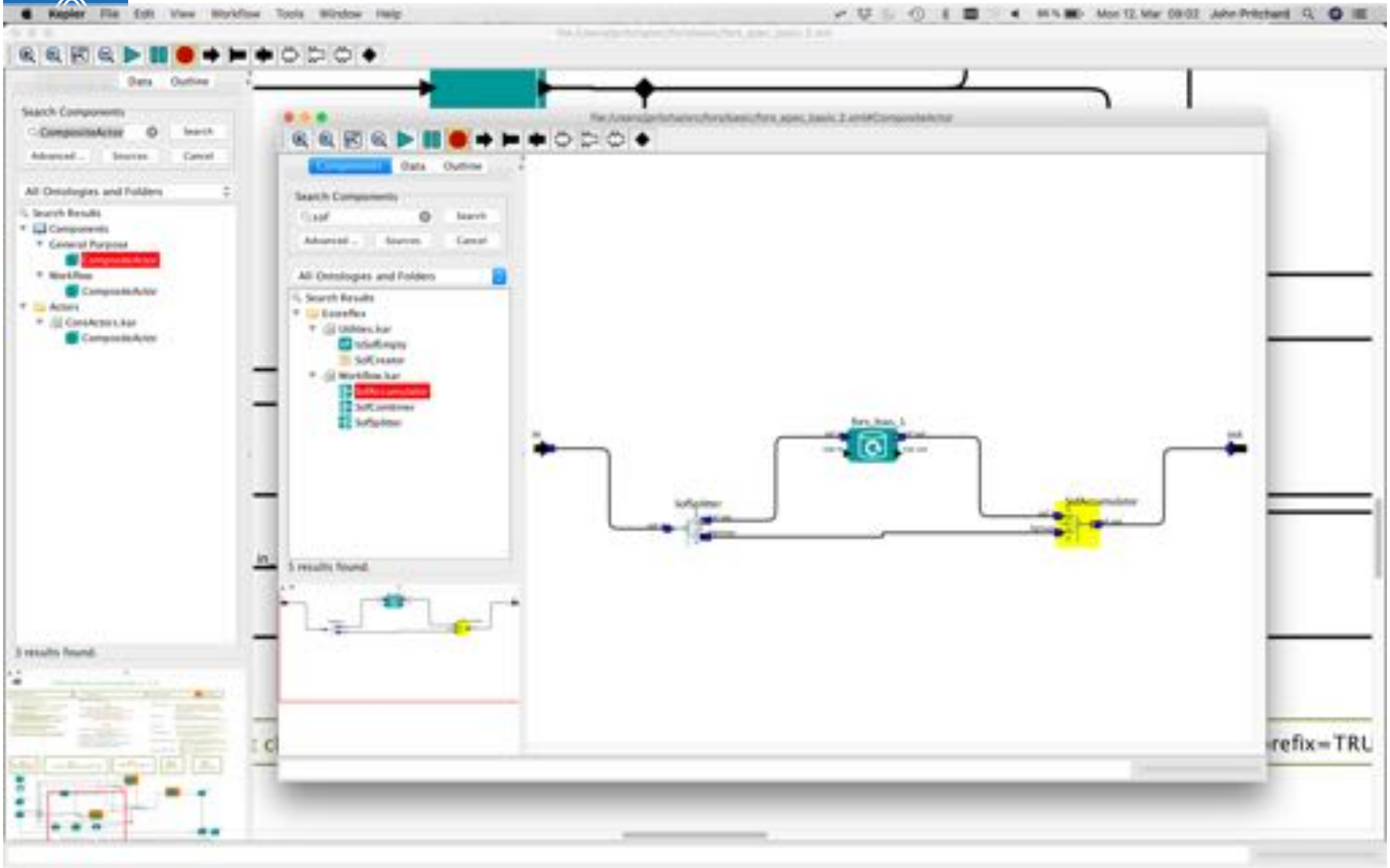
fors_align_sky

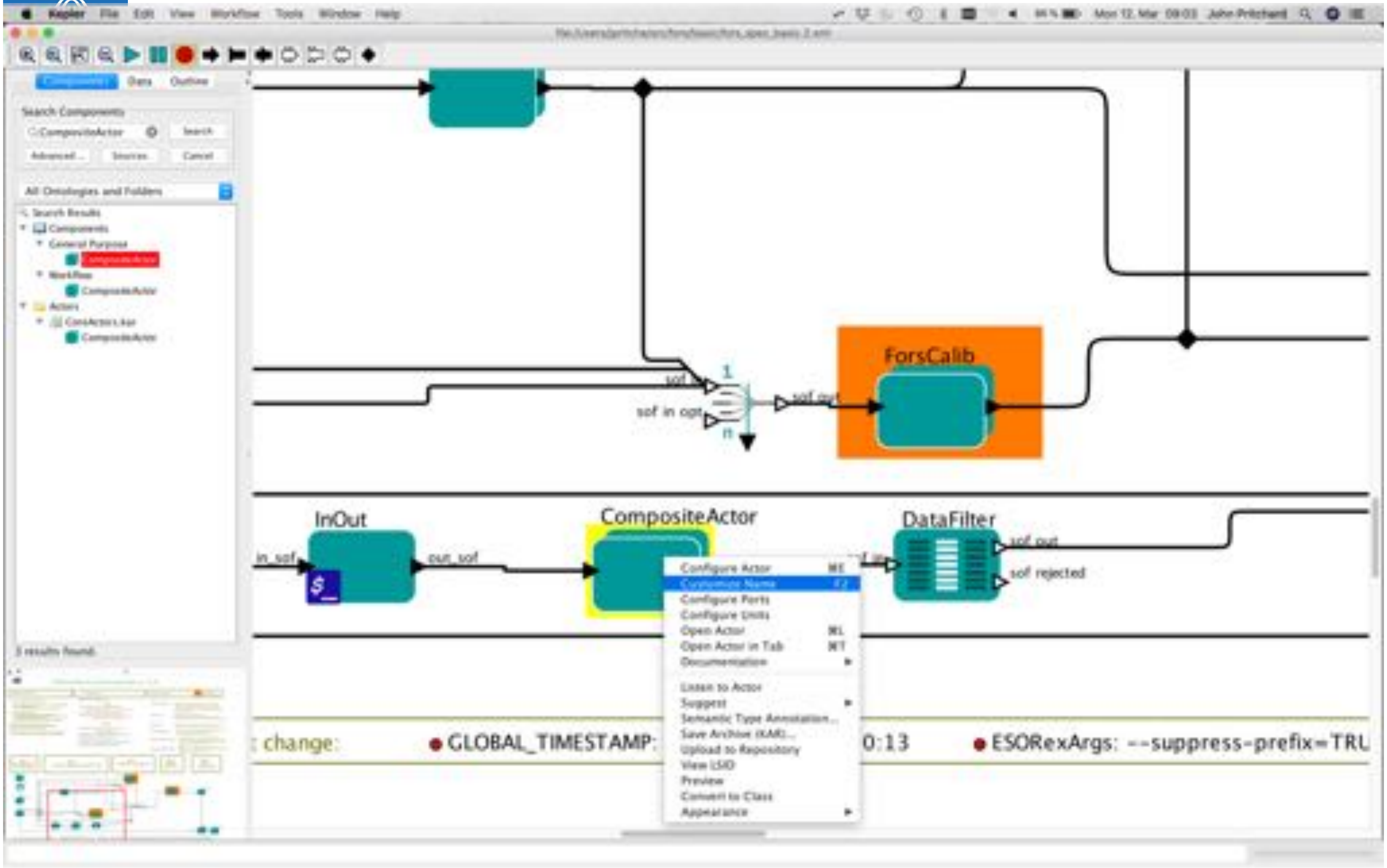


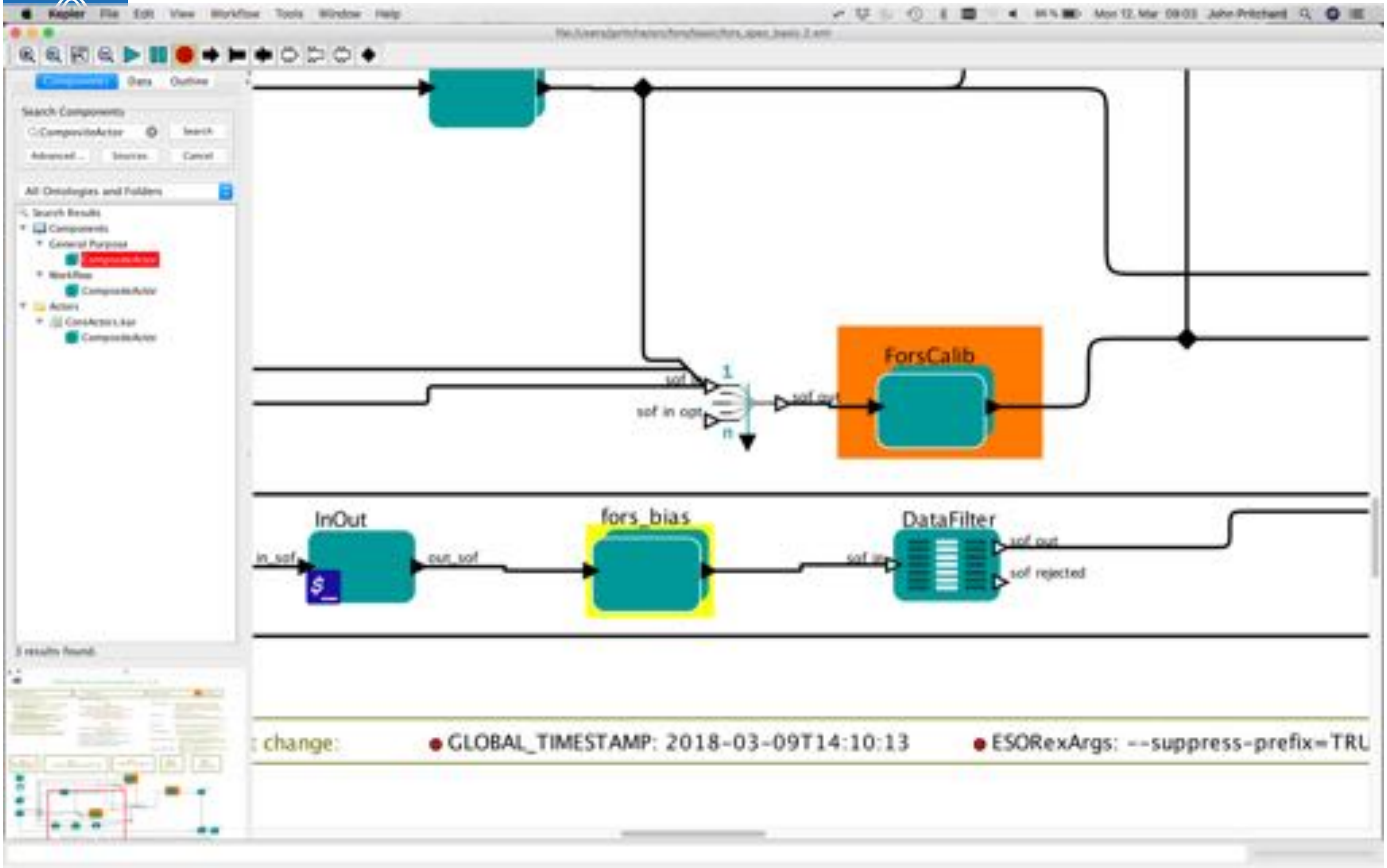
3 results found.

0 results found.


Kepler
File
Edit
View
Workflow
Tools
Window
Help
Data
Outline
Search Components
CompositeActor
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Search Results
Components
General Purpose
Workflow
CompositeActor
Actors
CompositeActor
3 results found
Components
Data
Outline
Search Components
sof
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Search Results
Esoreflex
Utilities.kar
IsSofEmpty
SofCreator
Workflow.kar
SofAccumulator
SofCombiner
SofSplitter
3 results found
fors_bias_1
SofSplitter
SofAccumulator
in
out
refix=TRU


Kepler
File
Edit
View
Workflow
Tools
Window
Help
Data
Outline
Search Components
CompositeActor
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Search Results
Components
General Purpose
CompositeActor
Workflow
CompositeActor
Actors
CompositeActor
3 results found
Components
Data
Outline
Search Components
sof
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Search Results
EsoReflex
Utilities.kar
IsSofEmpty
SofCreator
Workflow.kar
SofAccumulator
SofCombiner
SofSplitter
3 results found
fors_bias_1
SofSplitter
SofAccumulator
in
out
in
refix=TRU

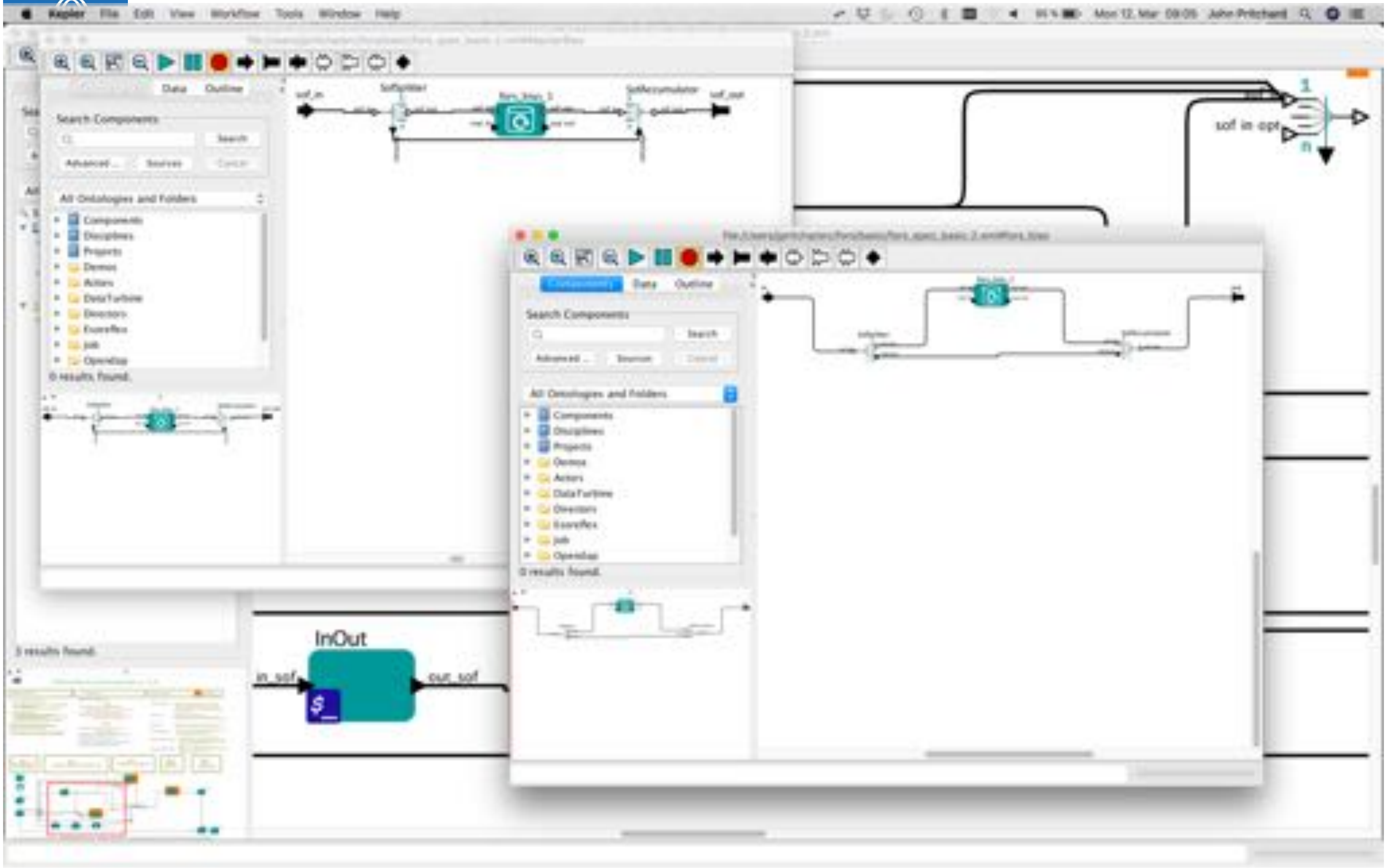

Kepler File Edit View Workflow Tools Window Help
Components Data Outline
Search Components
CompositeActor
Search
Advanced... Sources Cancel
All Ontologies and Folders
Search Results
Components
General Purpose
CompositeActor
Workflow
CompositeActor
Actors
CoreActors.kar
CompositeActor
3 results found
ForsCalib
sof in
sof in opt
sof out
1
n
InOut
in_sof
out_sof
$_
CompositeActor
DataFilter
sof in
sof out
sof rejected
Configure Actor
Customize Name
Configure Ports
Configure Units
Open Actor
Open Actor in Tab
Documentation
Listen to Actor
Suggest
Semantic Type Annotation...
Save Archive (KAR)...
Upload to Repository
View LSID
Preview
Convert to Class
Appearance
change:
GLOBAL_TIMESTAMP:
0:13
ESORexArgs: --suppress-prefix=TRU



change:

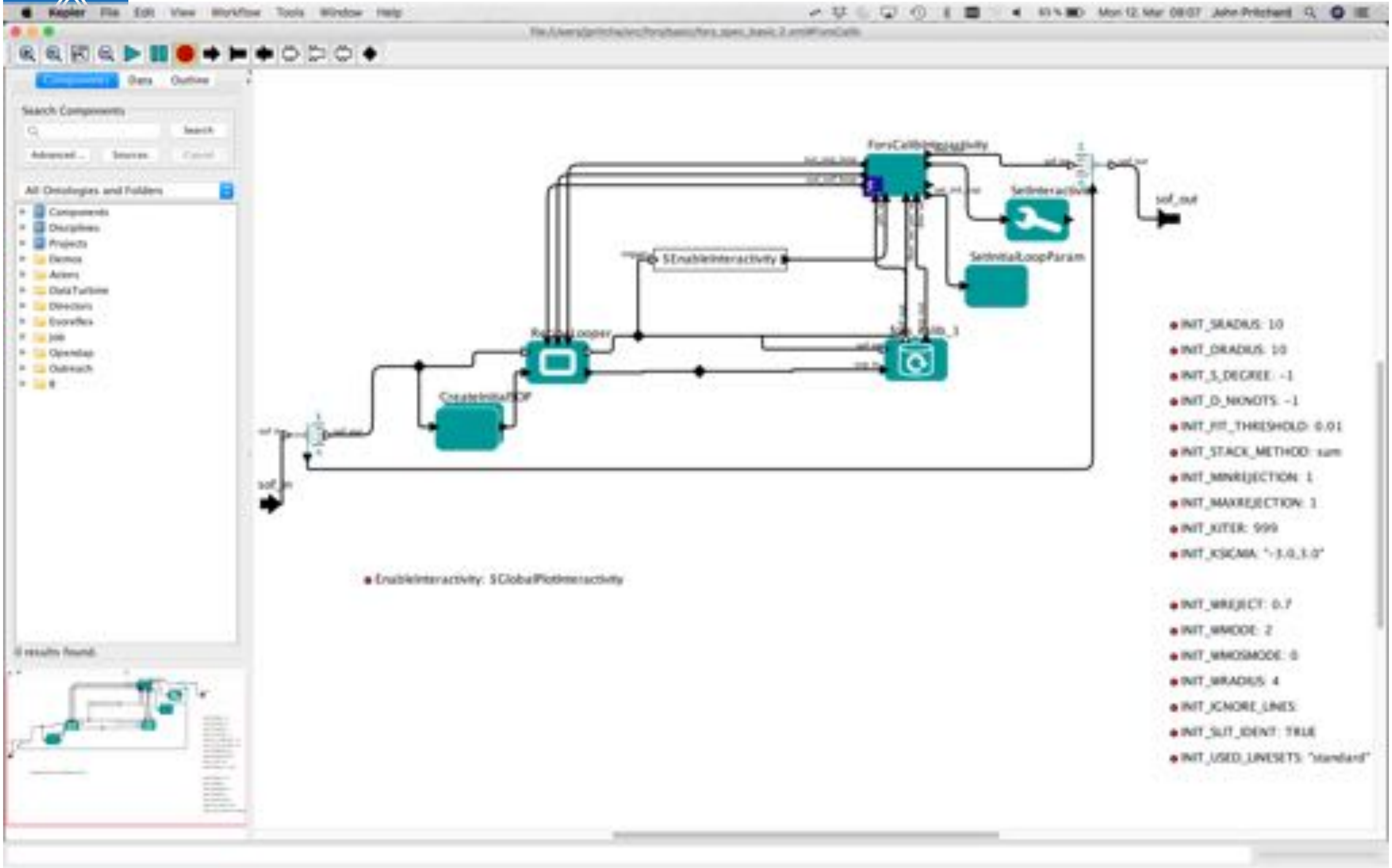
● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13

● ESORexArgs: --suppress-prefix=TRU



0 results found.

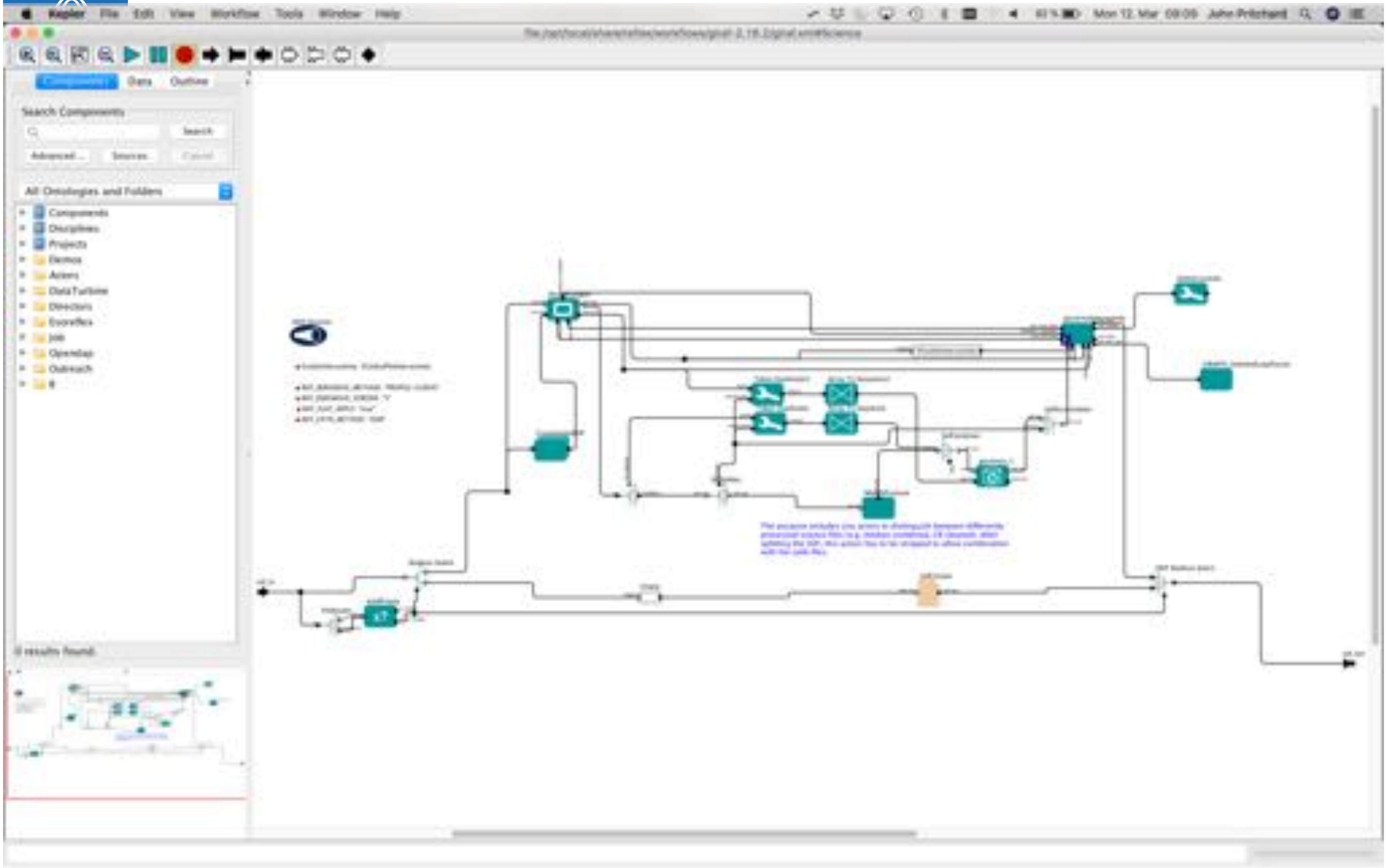
3 results found.



- INIT_SRADIUS: 10
- INIT_DRADIUS: 10
- INIT_S_DEGREE: -1
- INIT_D_NKNOTS: -1
- INIT_FIT_THRESHOLD: 0.01
- INIT_STACK_METHOD: sum
- INIT_MINREJECTION: 1
- INIT_MAXREJECTION: 1
- INIT_KITER: 999
- INIT_KSIGMA: "-3.0,3.0"

- INIT_WREJECT: 0.7
- INIT_WMODE: 2
- INIT_WMOSMODE: 0
- INIT_WRADIUS: 4
- INIT_IGNORE_LINES:
- INIT_SLIT_IDENT: TRUE
- INIT_USED_LINESETS: "standard"

- EnableInteractivity: $GlobalPlotInteractivity

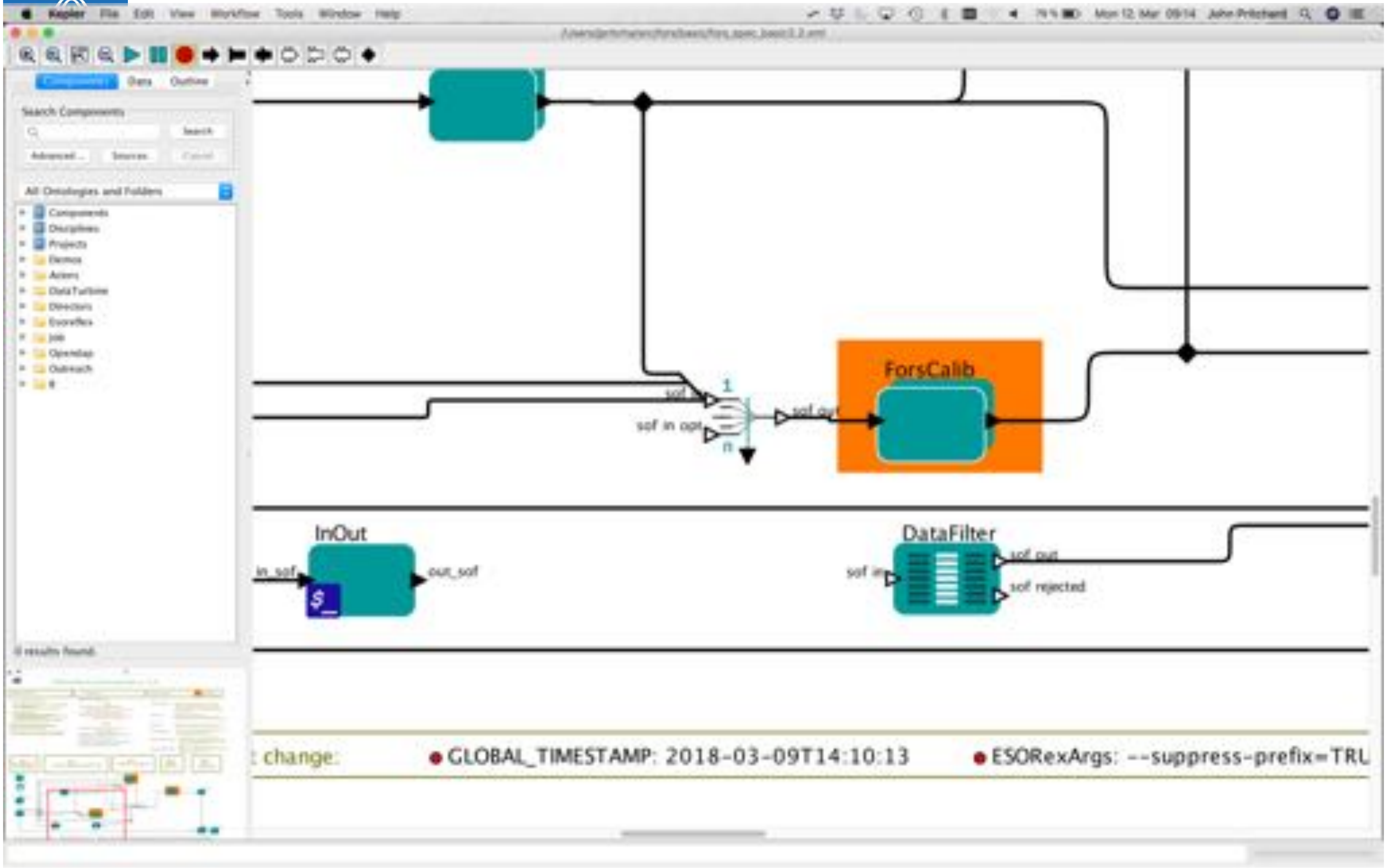

Kepler
File
Edit
View
Workflow
Tools
Window
Help
Mon 12. Mar 09:09
John Pritchard
Components
Data
Outline
Search Components
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Eworkflows
Job
Opendap
Outreach
R
0 results found

Step 3: Copy paste giraf CRC

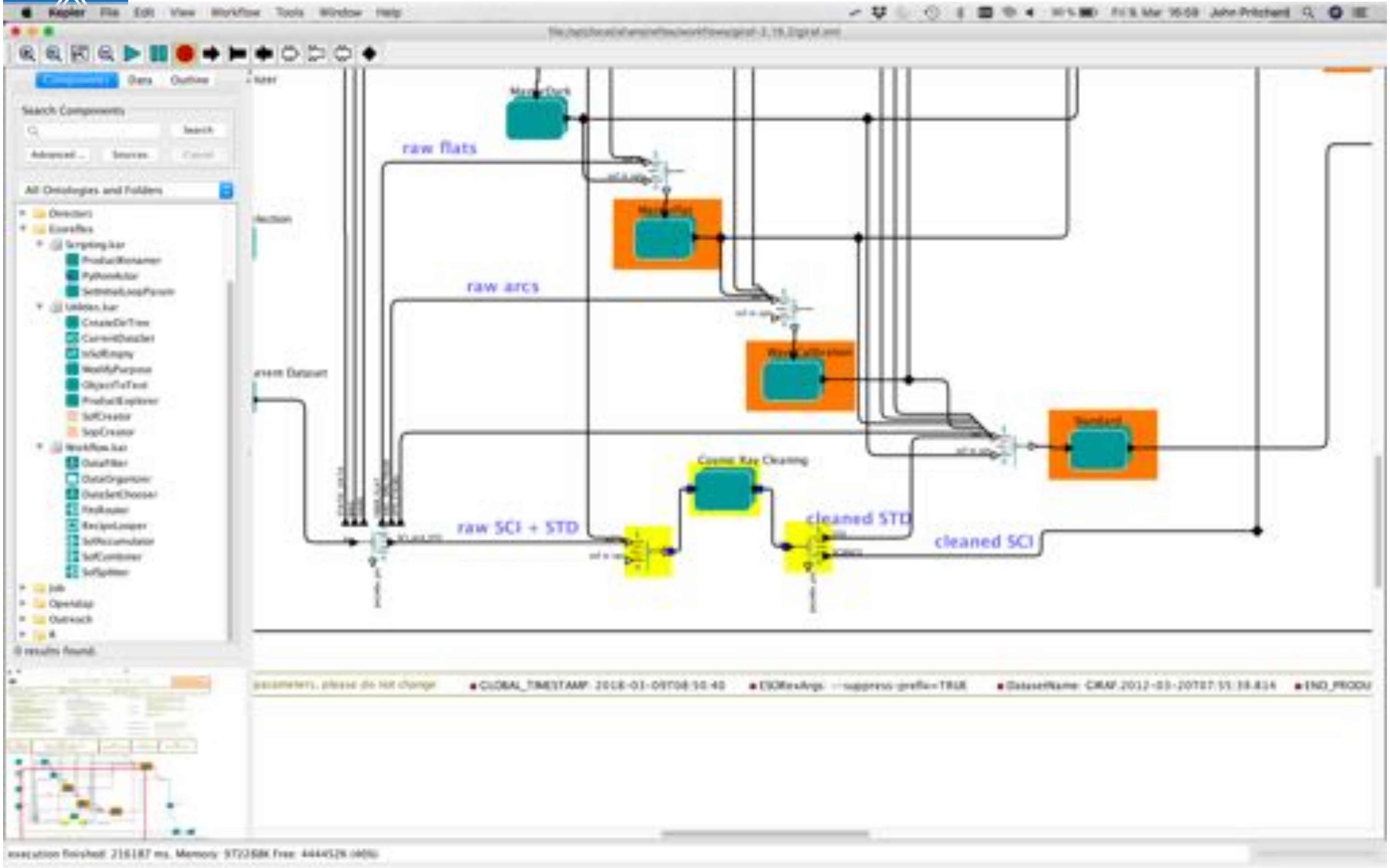
- Remove the fors_bias actor.
- Open the giraf workflow
- Select and copy the ‘Cosmic Ray Cleaning’, the SofCombiner and the FitsRouter actors just before and just after the CRC actor, and the connections between these three
- Paste into the fors_spec workflow, and move into positon below the DataFilter actor.

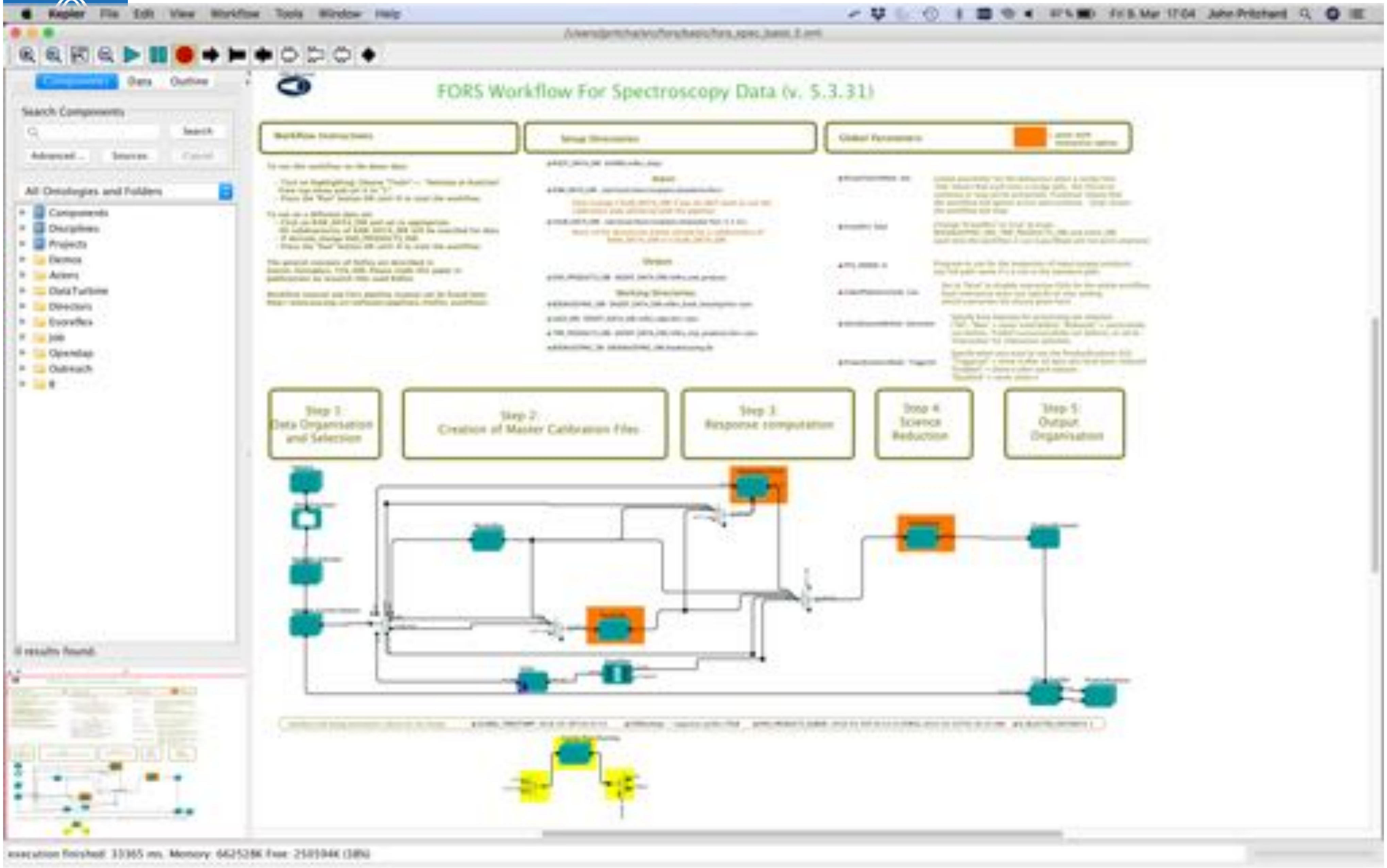
Step 3: Copy paste giraf CRC

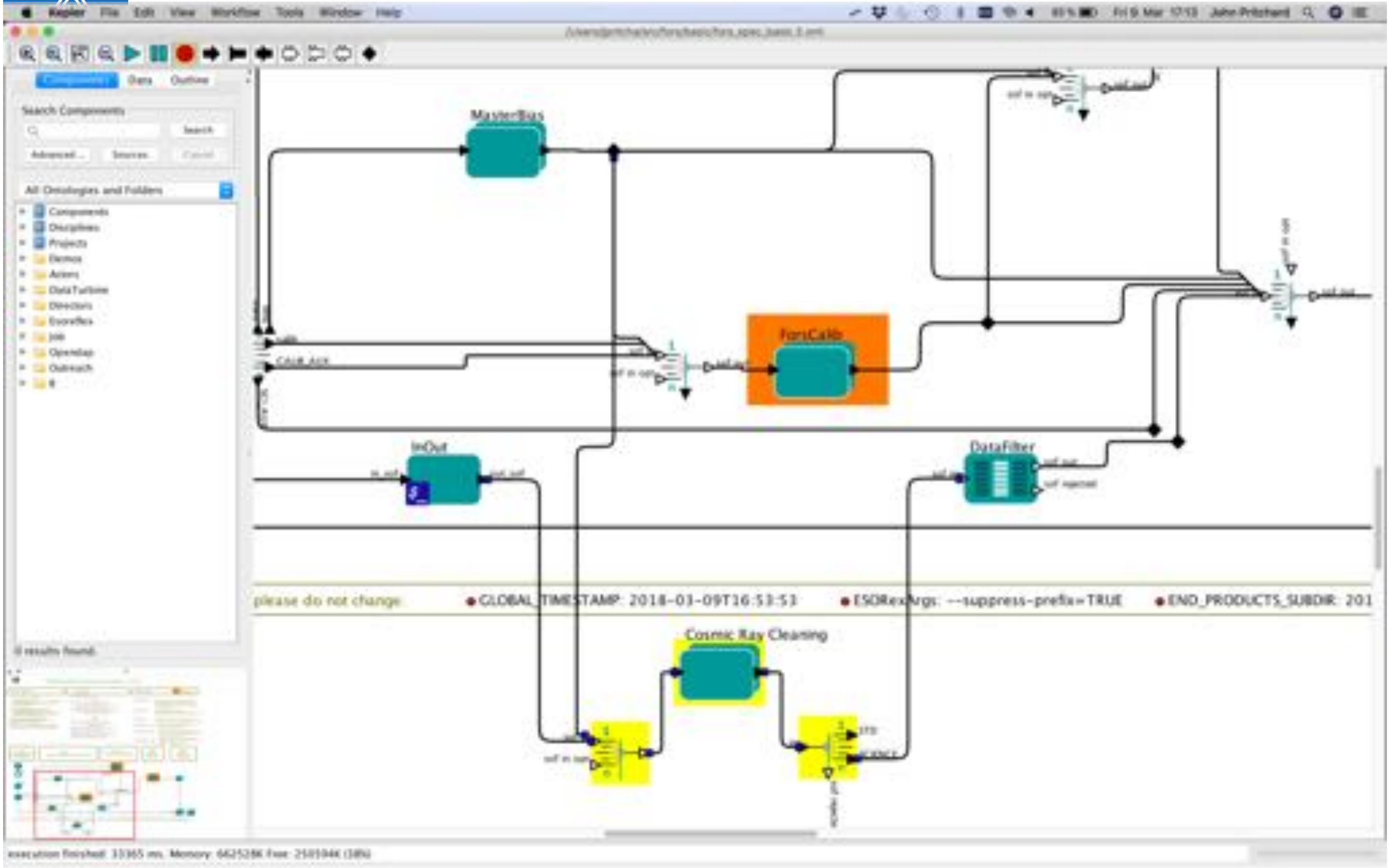
- Make a new connection from the ‘sof in’ port of the SofCombiner to the relation just after the MasterBias actor
- Reconnect the‘sof out’ port of the InOut actor to the ‘sof in’ port of the SofCombiner
- Make a new connection from the ‘SCIENCE’ port of the FitsRouter to the ‘sof in’ port of the DataFilter



change: ● GLOBAL_TIMESTAMP: 2018-03-09T14:10:13 ● ESORexArgs: --suppress-prefix=TRU


Kepler
File
Edit
View
Workflow
Tools
Window
Help
Components
Data
Outline
Search Components
Search
All Ontologies and Folders
MasterDark
raw flats
raw arcs
Cosmic Ray Cleaning
raw SCI + STD
cleaned STD
cleaned SCI
Standard


Kepler
File
Edit
View
Workflow
Tools
Window
Help
Components
Data
Outline
Search Components
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Eworkflex
Job
Opendap
Outreach
FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Global Parameters
Step 1:
Data Organisation
and Selection
Step 2:
Creation of Master Calibration Files
Step 3:
Response computation
Step 4:
Science
Reduction
Step 5:
Output
Organisation
0 results found.
execution finished: 33365 ms. Memory: 6625280K Free: 2505594K (38%)



please do not change ● GLOBAL_TIMESTAMP: 2018-03-09T16:53:53 ● ESORexArgs: --suppress-prefix=TRUE ● END_PRODUCTS_SUBDIR: 201

execution finished: 33365 ms. Memory: 662528K Free: 250594K (38%)

Step 4: adapt giraf to fors

- Ok so it ran past the CRC actor without crashing, but then the DataFilter is empty and continuing the science recipe crashes because there is no science file. So what’s wrong?
- In a nutshell, the GIRAF science data have category SCIENCE, while the FORS spec’ data have one of SCIENCE_MXU, SCIENCE_MOS or SCIENCE_LSS. So this needs to be fixed in the workflow and the python script.
- Check the categories of the science coming out of the master FitsRouter

Step 4: adapt giraf to fors

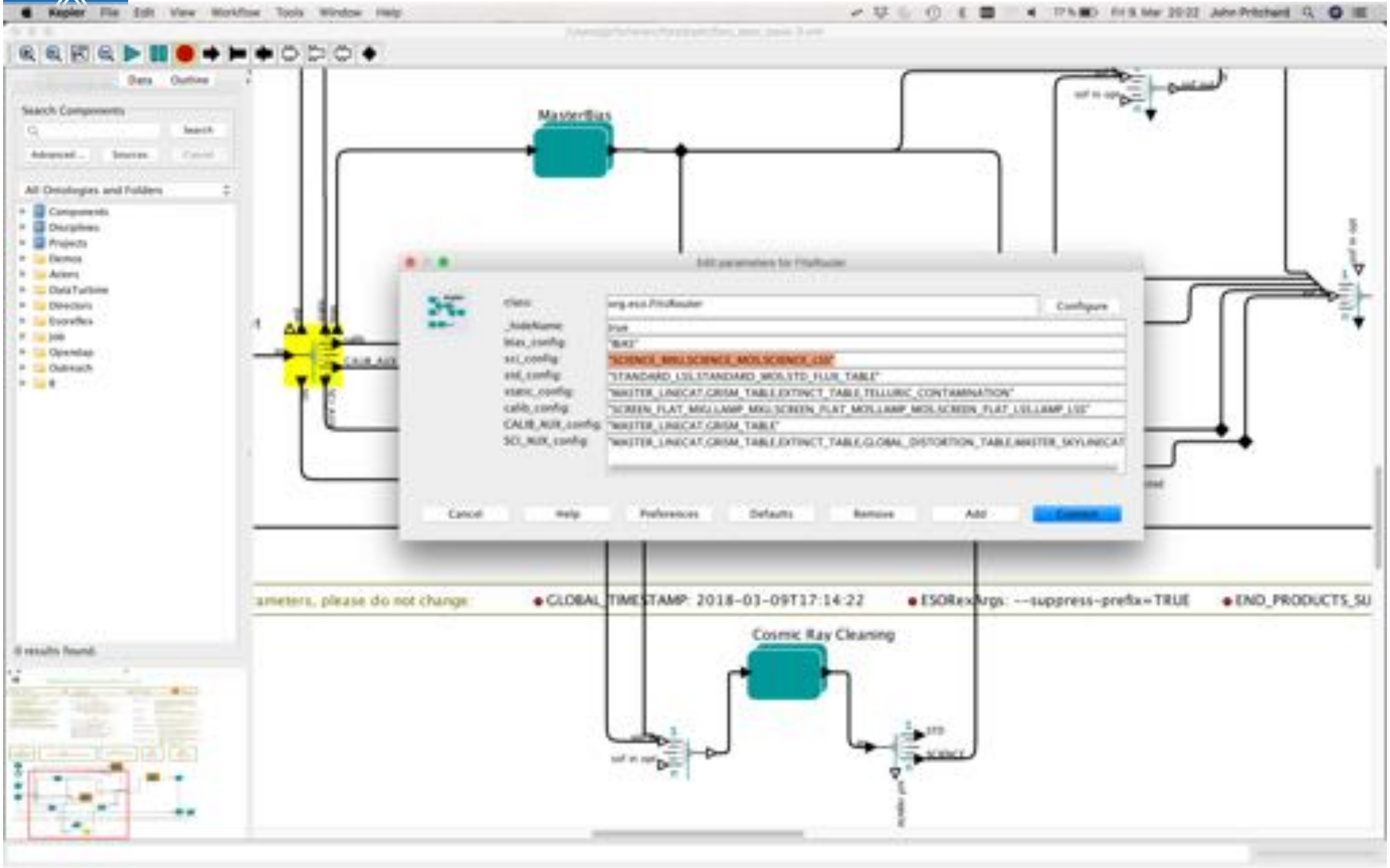
- Edit the parameters of the FitsRouter just after the CRC actor
- Add a parameter:
  - Class = ptolemy.data.expr.Parameter
  - Name = SCIENCE_config (the name of the port plus "_config")
  - Default value = "SCIENCE_MXU,SCIENCE_MOS,SCIENCE_LSS"
- Open (not edit) the CRC actor and do the same to the FitsRouter actor just after the Input port

Step 4: adapt giraf to fors

- Edit the parameters of the Astroscrappy actor
- Copy the name of the Python script.
- In a terminal, copy that python script file to the local directory:

```
cp /opt/local/share/esopipes/giraf-2.16.2/reflex/giraf_astroscrappy_multi.py fors_astroscrappy_multi.py
```

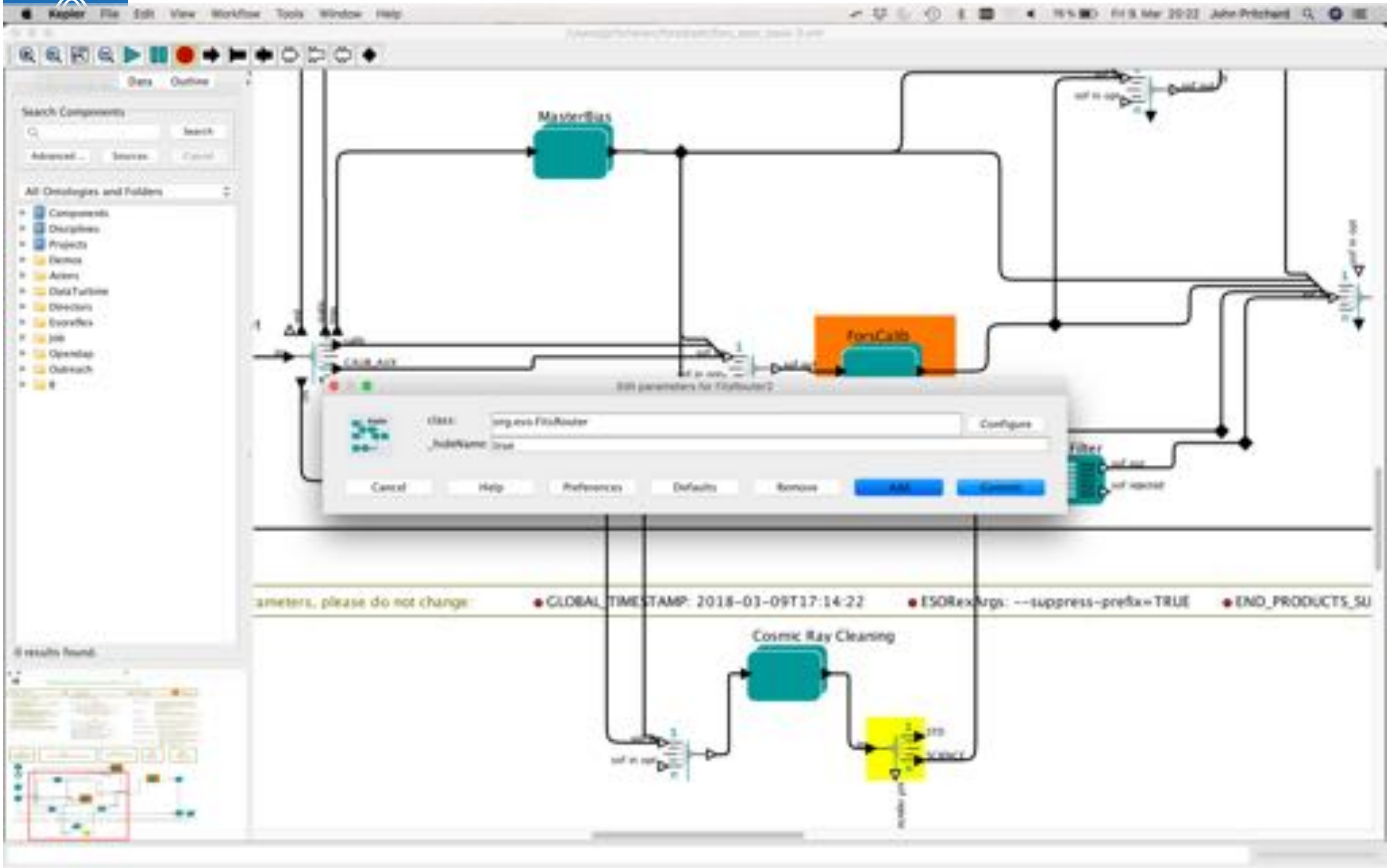
- Adapt the python script appropriately… ☺
- Set the Python script to the fors_astroscrappy_multi.py, make sure that it has the full path to the file.



Edit parameters for FitsRouter

class: org.eso.FitsRouter
_hideName: true
bias_config: "BIAS"
sci_config: "SCIENCE_MXU,SCIENCE_MOS,SCIENCE_LSS"
std_config: "STANDARD_LSS,STANDARD_MOS,STD_FLUX_TABLE"
static_config: "MASTER_LINECAT,GRISM_TABLE,EXTINCT_TABLE,TELLURIC_CONTAMINATION"
calib_config: "SCREEN_FLAT_MXU,LAMP_MXU,SCREEN_FLAT_MOS,LAMP_MOS,SCREEN_FLAT_LSS,LAMP_LSS"
CALIB_AUX_config: "MASTER_LINECAT,GRISM_TABLE"
SCI_AUX_config: "MASTER_LINECAT,GRISM_TABLE,EXTINCT_TABLE,GLOBAL_DISTORTION_TABLE,MASTER_SKYLINECAT

ameters, please do not change
● GLOBAL_TIMESTAMP: 2018-03-09T17:14:22
● ESORexArgs: --suppress-prefix=TRUE
● END_PRODUCTS_SU



Edit parameters for FitsRouter2

class: org.eso.FitsRouter

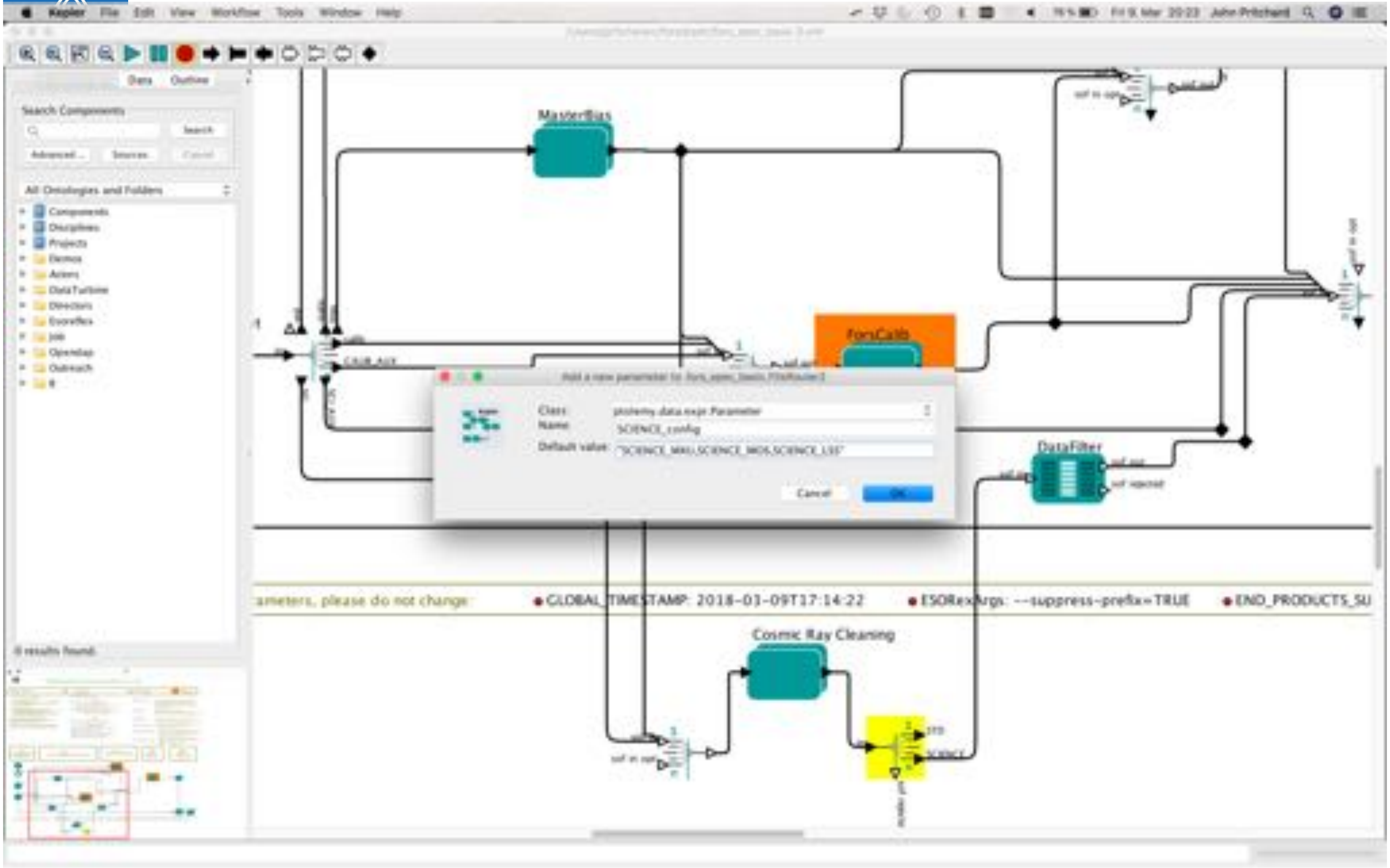
_hideName: true

ameters, please do not change

● GLOBAL_TIMESTAMP: 2018-03-09T17:14:22

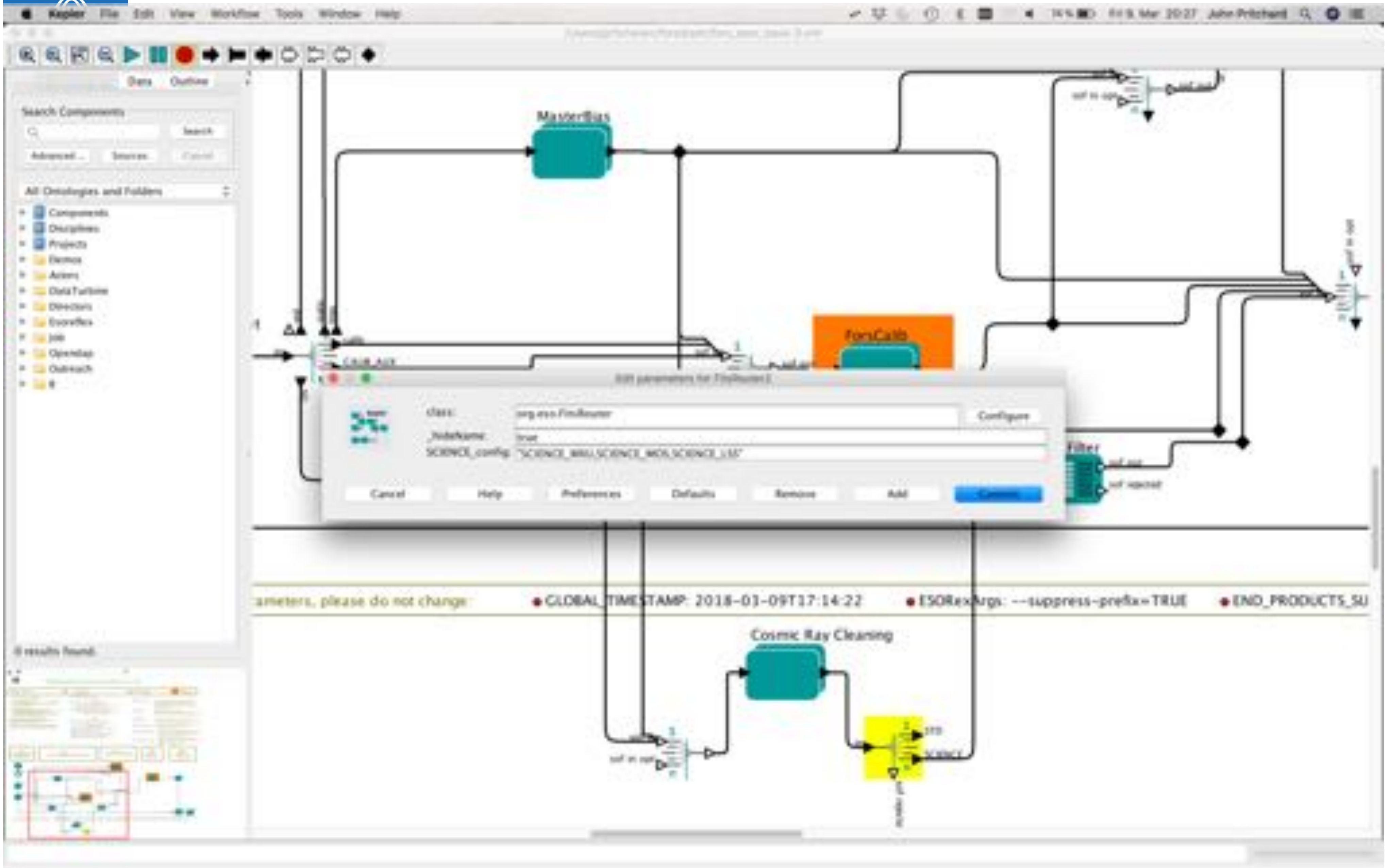
● ESORexArgs: --suppress-prefix=TRUE

● END_PRODUCTS_SU



Class: ptolemy.data.expr.Parameter
Name: SCIENCE_config
Default value: "SCIENCE_MXU,SCIENCE_MOS,SCIENCE_LSS"

ameters, please do not change
● GLOBAL_TIMESTAMP: 2018-03-09T17:14:22
● ESORexArgs: --suppress-prefix=TRUE
● END_PRODUCTS_SU

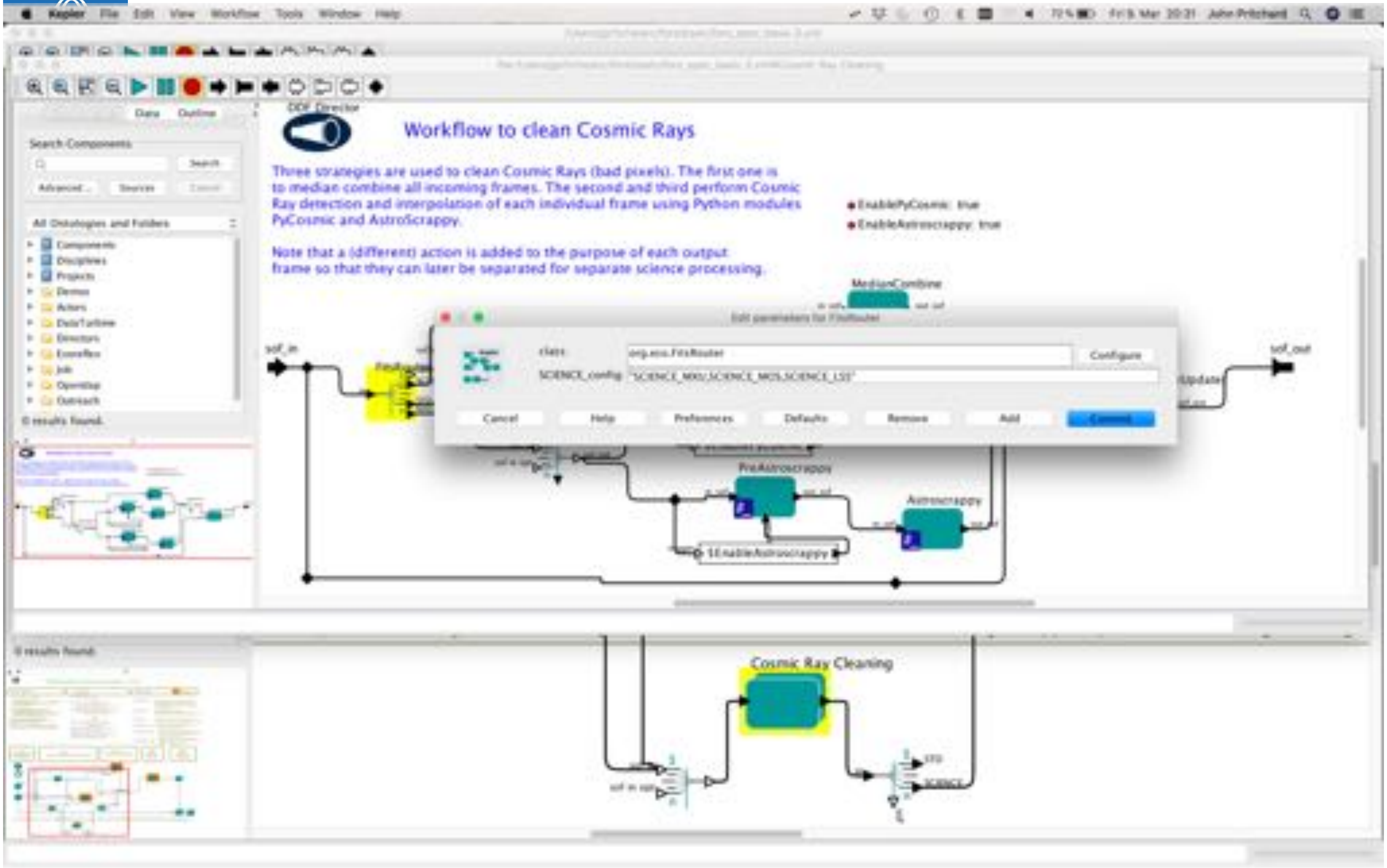


class: org.eso.FitsRouter

_hideName: true

SCIENCE_config: "SCIENCE_MXU,SCIENCE_MOS,SCIENCE_LSS"

ameters, please do not change: ● GLOBAL_TIMESTAMP: 2018-03-09T17:14:22 ● ESORexArgs: --suppress-prefix=TRUE ● END_PRODUCTS_SU



Workflow to clean Cosmic Rays

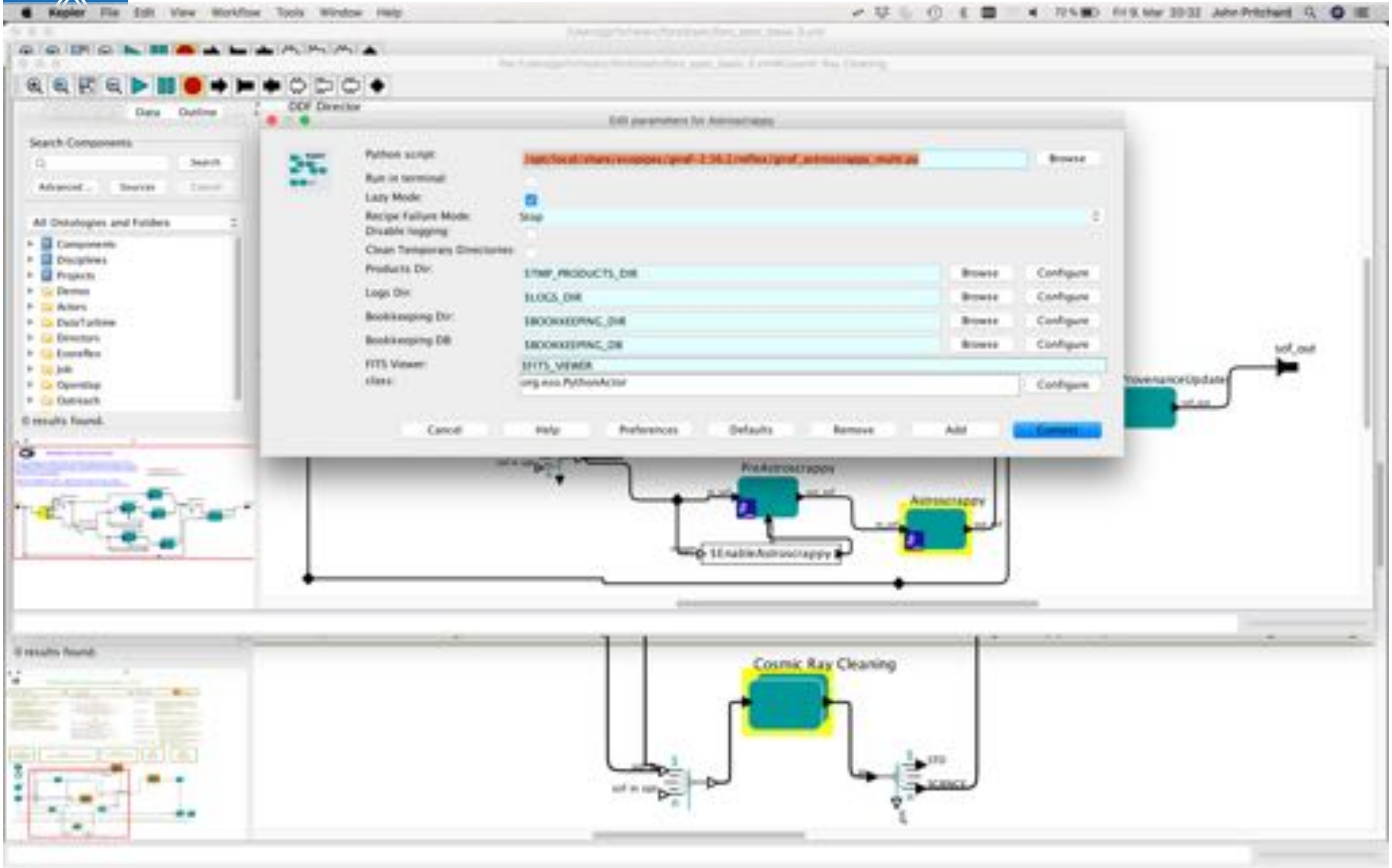
Three strategies are used to clean Cosmic Rays (bad pixels). The first one is to median combine all incoming frames. The second and third perform Cosmic Ray detection and interpolation of each individual frame using Python modules PyCosmic and AstroScrappy.

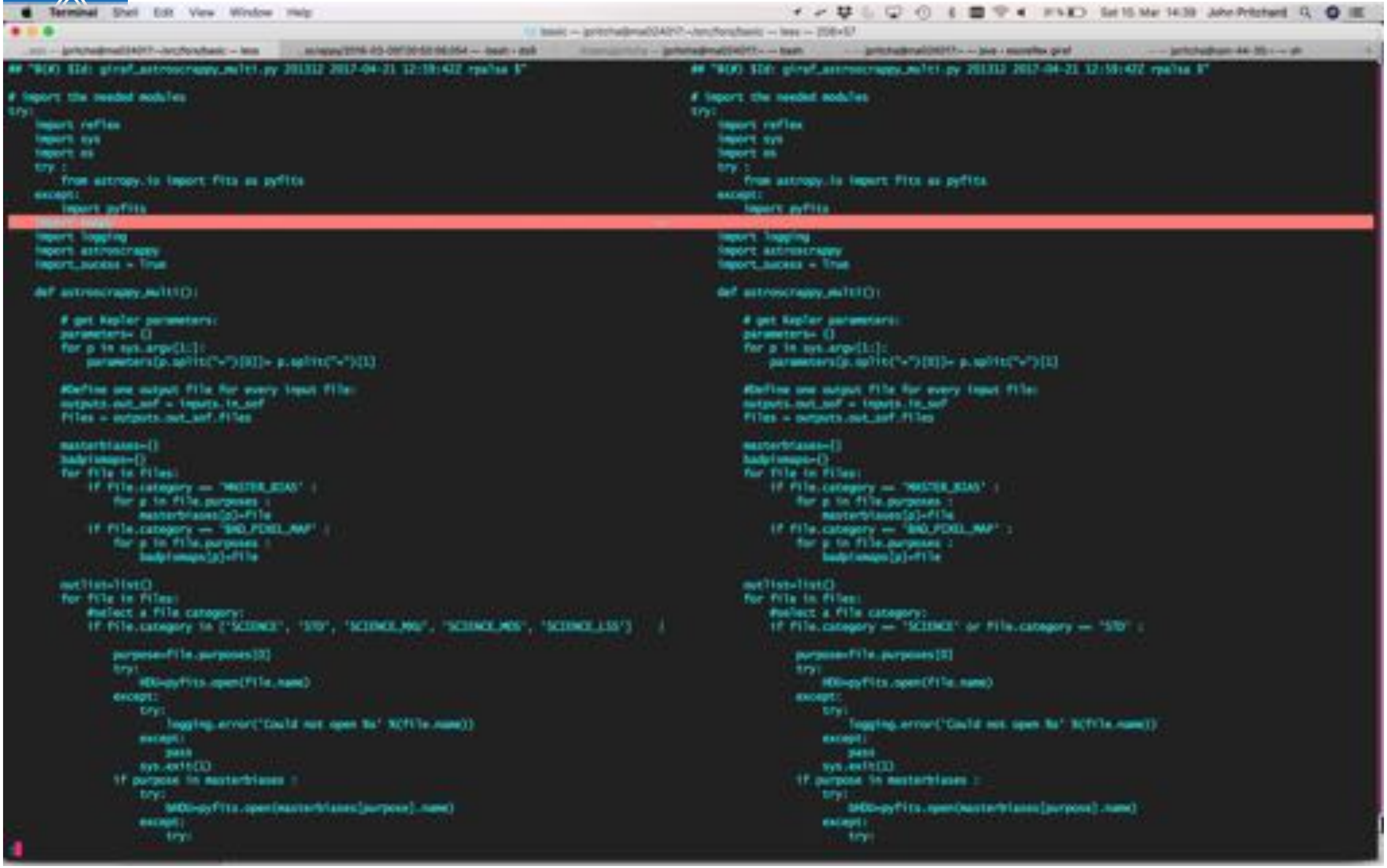
Note that a (different) action is added to the purpose of each output frame so that they can later be separated for separate science processing.

- EnablePyCosmic: true
- EnableAstroscrappy: true

class: org.eso.FitsRouter

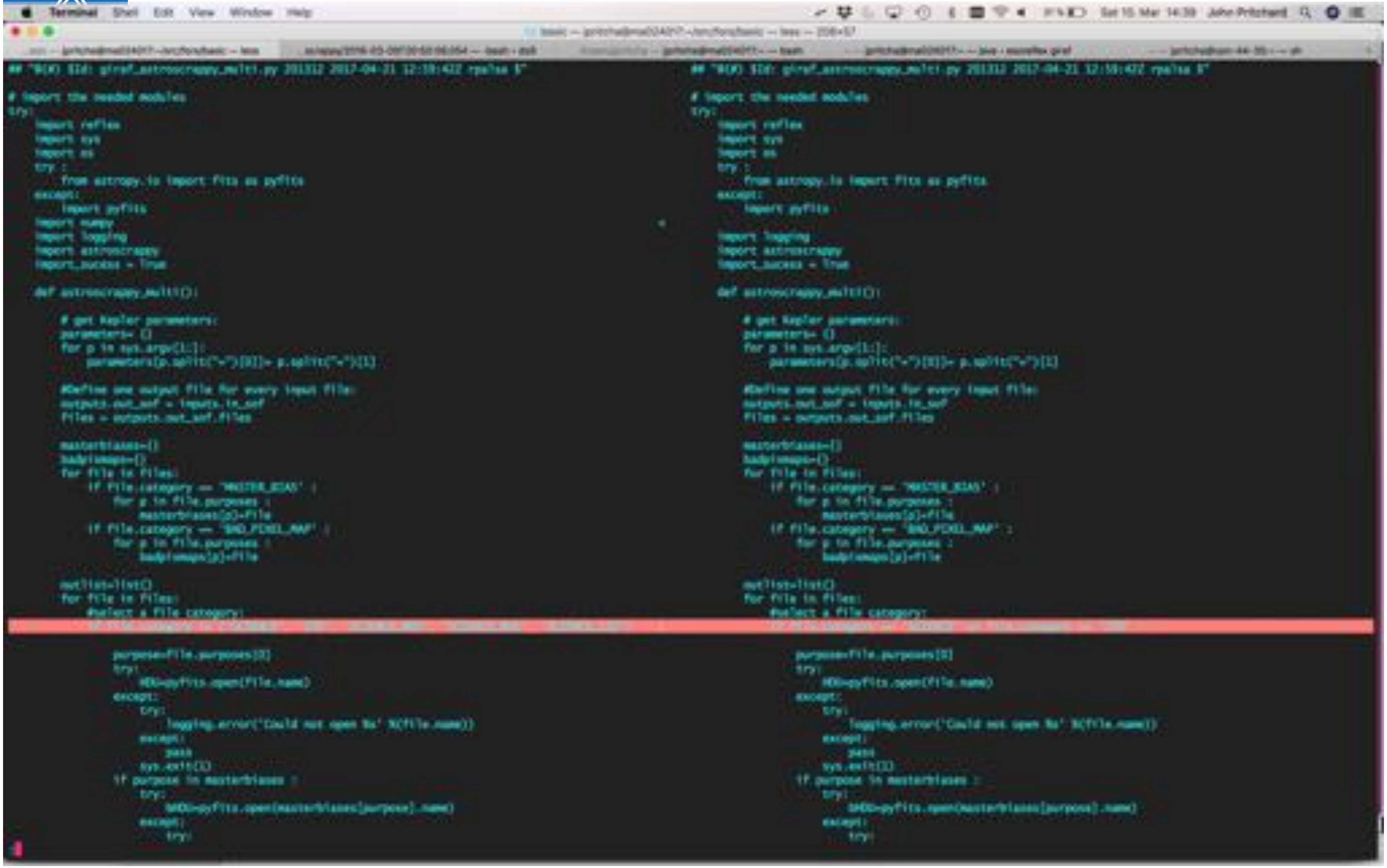
SCIENCE_config: "SCIENCE_MXU,SCIENCE_MOS,SCIENCE_LSS"





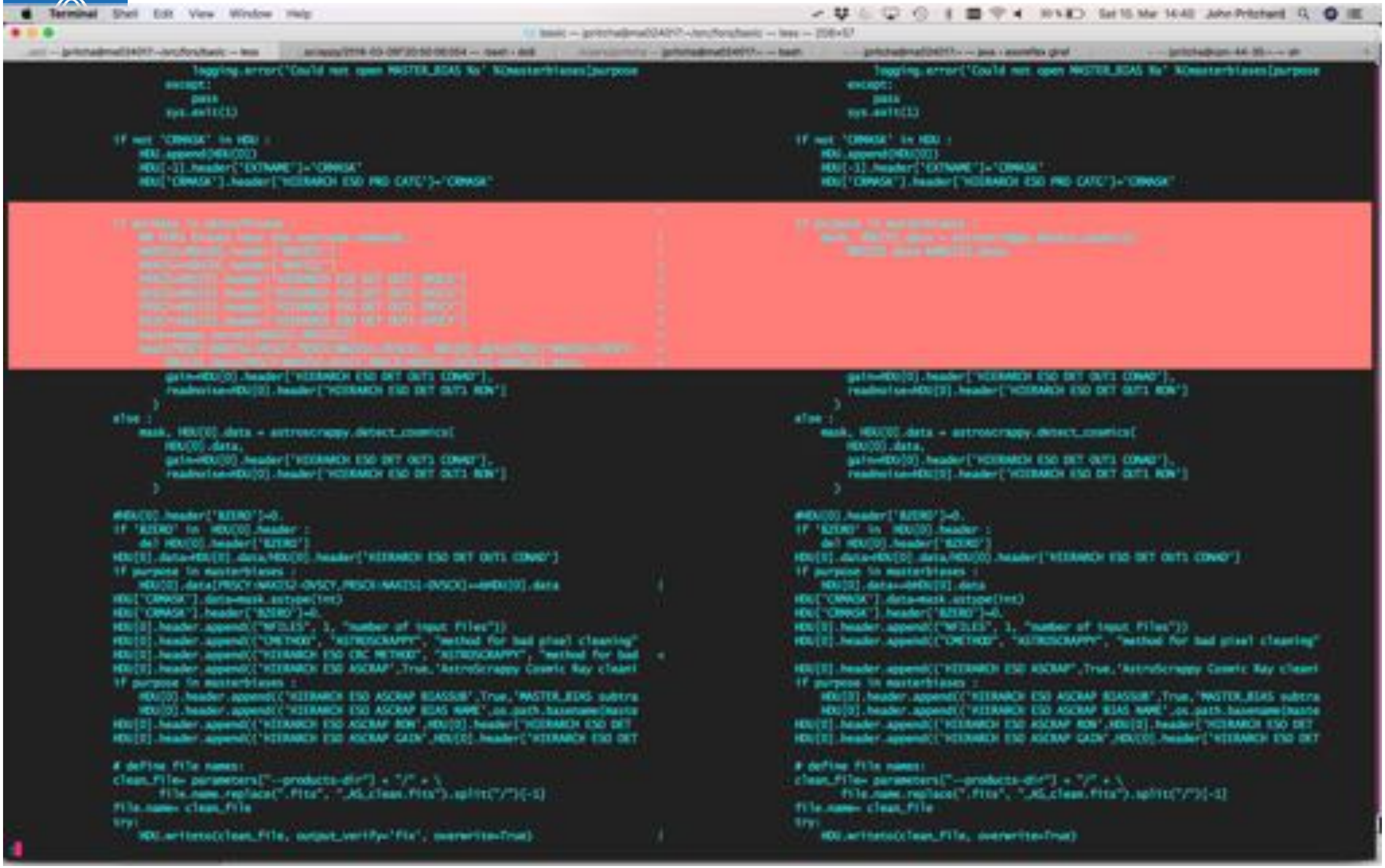
```
## "@(#) $Id: giraf_astroscrappy_multi.py 201312 2017-04-21 12:59:42Z rpalsa $"

# import the needed modules
try:
    import reflex
    import sys
    import os
    try :
        from astropy.io import fits as pyfits
    except:
        import pyfits
    import numpy
    import logging
    import astroscrappy
    import_success = True

    def astroscrappy_multi():

        # get Kepler parameters:
        parameters= {}
        for p in sys.argv[1:]:
            parameters[p.split("=")[0]]= p.split("=")[1]

        #Define one output file for every input file:
        outputs.out_sof = inputs.in_sof
        files = outputs.out_sof.files

        masterbiases={}
        badpixmaps={}
        for file in files:
            if file.category == 'MASTER_BIAS' :
                for p in file.purposes :
                    masterbiases[p]=file
            if file.category == 'BAD_PIXEL_MAP' :
                for p in file.purposes :
                    badpixmaps[p]=file

        outlist=list()
        for file in files:
            #select a file category:
            if file.category in ['SCIENCE', 'STD', 'SCIENCE_MOU', 'SCIENCE_MOS', 'SCIENCE_LSS']    :

                purpose=file.purposes[0]
                try:
                    HDU=pyfits.open(file.name)
                except:
                    try:
                        logging.error('Could not open %s' %(file.name))
                    except:
                        pass
                    sys.exit(1)
                if purpose in masterbiases :
                    try:
                        bHDU=pyfits.open(masterbiases[purpose].name)
                    except:
                        try:
```

```
## "@(#) $Id: giraf_astroscrappy_multi.py 201312 2017-04-21 12:59:42Z rpalsa $"

# import the needed modules
try:
    import reflex
    import sys
    import os
    try :
        from astropy.io import fits as pyfits
    except:
        import pyfits

    import logging
    import astroscrappy
    import_success = True

    def astroscrappy_multi():

        # get Kepler parameters:
        parameters= {}
        for p in sys.argv[1:]:
            parameters[p.split("=")[0]]= p.split("=")[1]

        #Define one output file for every input file:
        outputs.out_sof = inputs.in_sof
        files = outputs.out_sof.files

        masterbiases={}
        badpixmaps={}
        for file in files:
            if file.category == 'MASTER_BIAS' :
                for p in file.purposes :
                    masterbiases[p]=file
            if file.category == 'BAD_PIXEL_MAP' :
                for p in file.purposes :
                    badpixmaps[p]=file

        outlist=list()
        for file in files:
            #select a file category:
            if file.category == 'SCIENCE' or file.category == 'STD' :

                purpose=file.purposes[0]
                try:
                    HDU=pyfits.open(file.name)
                except:
                    try:
                        logging.error('Could not open %s' %(file.name))
                    except:
                        pass
                    sys.exit(1)
                if purpose in masterbiases :
                    try:
                        bHDU=pyfits.open(masterbiases[purpose].name)
                    except:
                        try:
```



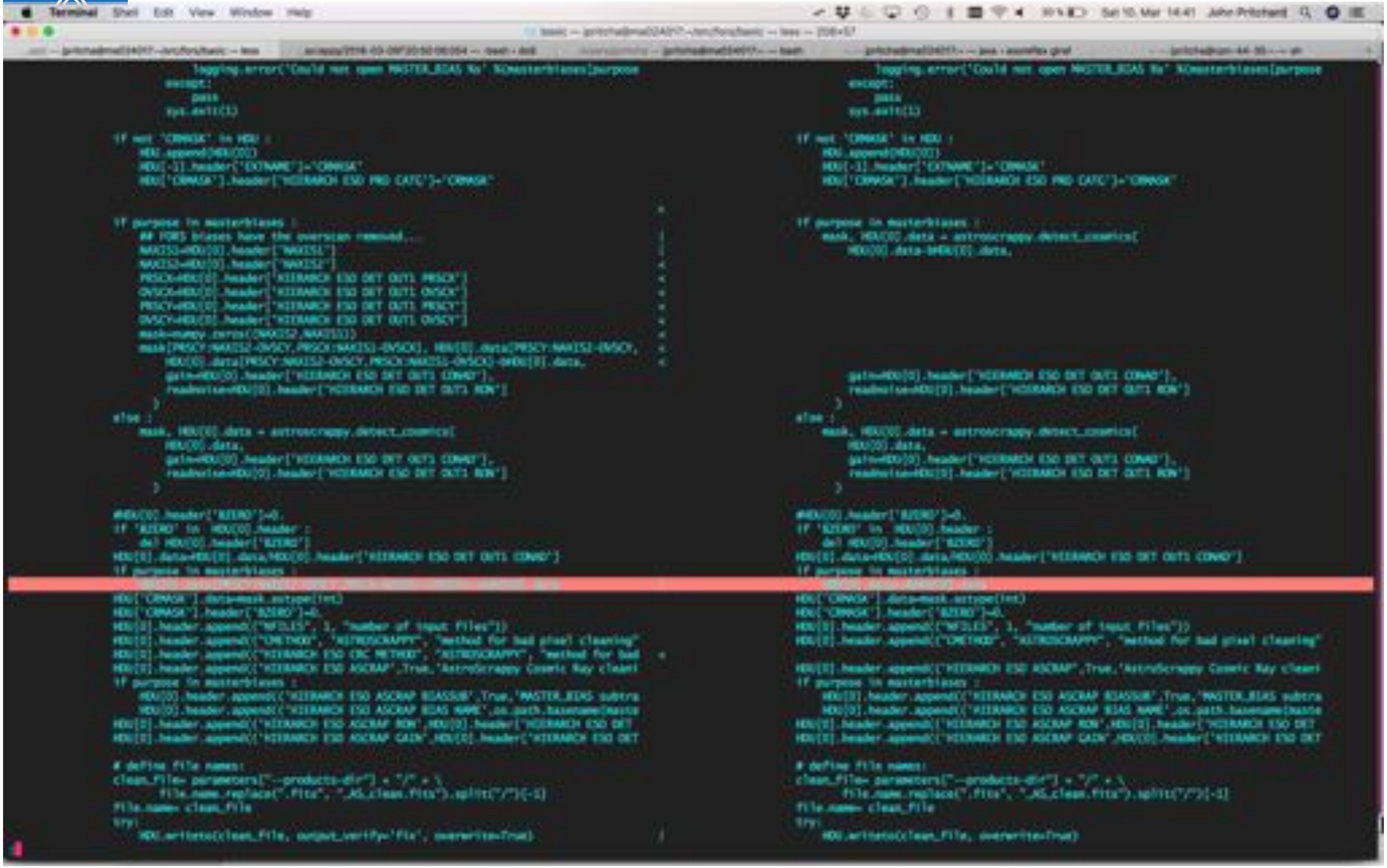
```
## "@(#) $Id: giraf_astroscrappy_multi.py 201312 2017-04-21 12:59:42Z rpalsa $"

# import the needed modules
try:
    import reflex
    import sys
    import os
    try :
        from astropy.io import fits as pyfits
    except:
        import pyfits
    import numpy
    import logging
    import astroscrappy
    import_success = True

    def astroscrappy_multi():

        # get Kepler parameters:
        parameters= {}
        for p in sys.argv[1:]:
            parameters[p.split("=")[0]]= p.split("=")[1]

        #Define one output file for every input file:
        outputs.out_sof = inputs.in_sof
        files = outputs.out_sof.files

        masterbiases={}
        badpixmaps={}
        for file in files:
            if file.category == 'MASTER_BIAS' :
                for p in file.purposes :
                    masterbiases[p]=file
            if file.category == 'BAD_PIXEL_MAP' :
                for p in file.purposes :
                    badpixmaps[p]=file

        outlist=list()
        for file in files:
            #select a file category:
            [illegible]

                purpose=file.purposes[0]
                try:
                    HDU=pyfits.open(file.name)
                except:
                    try:
                        logging.error('Could not open %s' %(file.name))
                    except:
                        pass
                    sys.exit(1)
                if purpose in masterbiases :
                    try:
                        BHDU=pyfits.open(masterbiases[purpose].name)
                    except:
                        try:
```

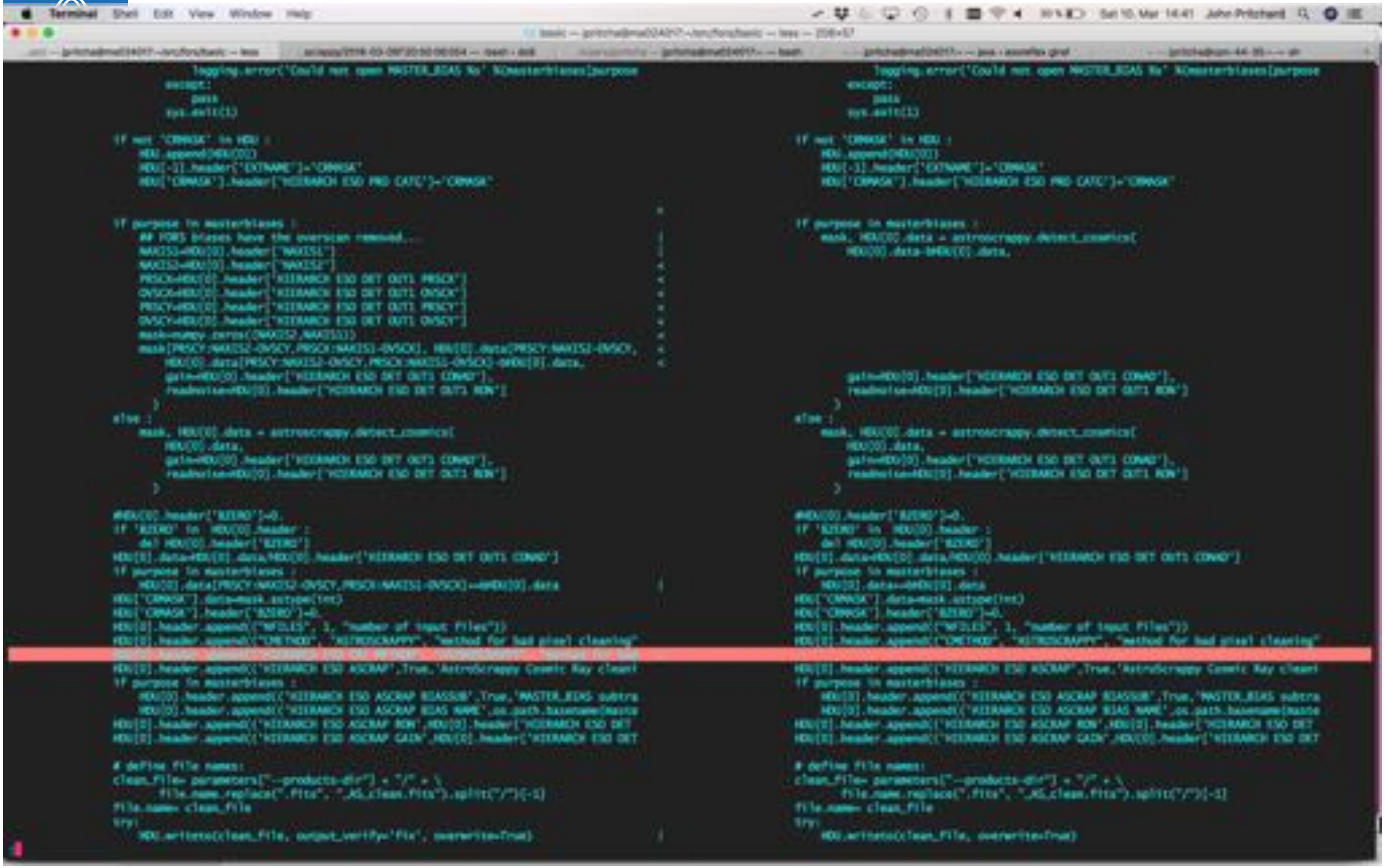
```
## "@(#) $Id: giraf_astroscrappy_multi.py 201312 2017-04-21 12:59:42Z rpalsa $"

# import the needed modules
try:
    import reflex
    import sys
    import os
    try :
        from astropy.io import fits as pyfits
    except:
        import pyfits

    import logging
    import astroscrappy
    import_success = True

    def astroscrappy_multi():

        # get Kepler parameters:
        parameters= {}
        for p in sys.argv[1:]:
            parameters[p.split("=")[0]]= p.split("=")[1]

        #Define one output file for every input file:
        outputs.out_sof = inputs.in_sof
        files = outputs.out_sof.files

        masterbiases={}
        badpixmaps={}
        for file in files:
            if file.category == 'MASTER_BIAS' :
                for p in file.purposes :
                    masterbiases[p]=file
            if file.category == 'BAD_PIXEL_MAP' :
                for p in file.purposes :
                    badpixmaps[p]=file

        outlist=list()
        for file in files:
            #select a file category:
            [illegible]

                purpose=file.purposes[0]
                try:
                    HDU=pyfits.open(file.name)
                except:
                    try:
                        logging.error('Could not open %s' %(file.name))
                    except:
                        pass
                    sys.exit(1)
                if purpose in masterbiases :
                    try:
                        BHDU=pyfits.open(masterbiases[purpose].name)
                    except:
                        try:
```



```
            logging.error('Could not open MASTER_BIAS %s' %(masterbiases[purpose
        except:
            pass
        sys.exit(1)

    if not 'CRMASK' in HDU :
        HDU.append(HDU[0])
        HDU[-1].header['EXTNAME']='CRMASK'
        HDU['CRMASK'].header['HIERARCH ESO PRO CATG']='CRMASK'

[illegible]
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )
    else :
        mask, HDU[0].data = astroscrappy.detect_cosmics(
            HDU[0].data,
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )

    #HDU[0].header['BZERO']=0.
    if 'BZERO' in  HDU[0].header :
        del HDU[0].header['BZERO']
    HDU[0].data=HDU[0].data/HDU[0].header['HIERARCH ESO DET OUT1 CONAD']
    if purpose in masterbiases :
        HDU[0].data[PRSCY:NAXIS2-OVSCY,PRSCX:NAXIS1-OVSCX]-=mHDU[0].data
    HDU['CRMASK'].data=mask.astype(int)
    HDU['CRMASK'].header['BZERO']=0.
    HDU[0].header.append(("NFILES", 1, "number of input files"))
    HDU[0].header.append(("CMETHOD", "ASTROSCRAPPY", "method for bad pixel cleaning"
    HDU[0].header.append(("HIERARCH ESO CRC METHOD", "ASTROSCRAPPY", "method for bad
    HDU[0].header.append(('HIERARCH ESO ASCRAP',True,'AstroScrappy Cosmic Ray cleani
    if purpose in masterbiases :
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
    HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
    HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

    # define file names:
    clean_file= parameters["--products-dir"] + "/" + \
            file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
    file.name= clean_file
    try:
        HDU.writeto(clean_file, output_verify='fix', overwrite=True)
```

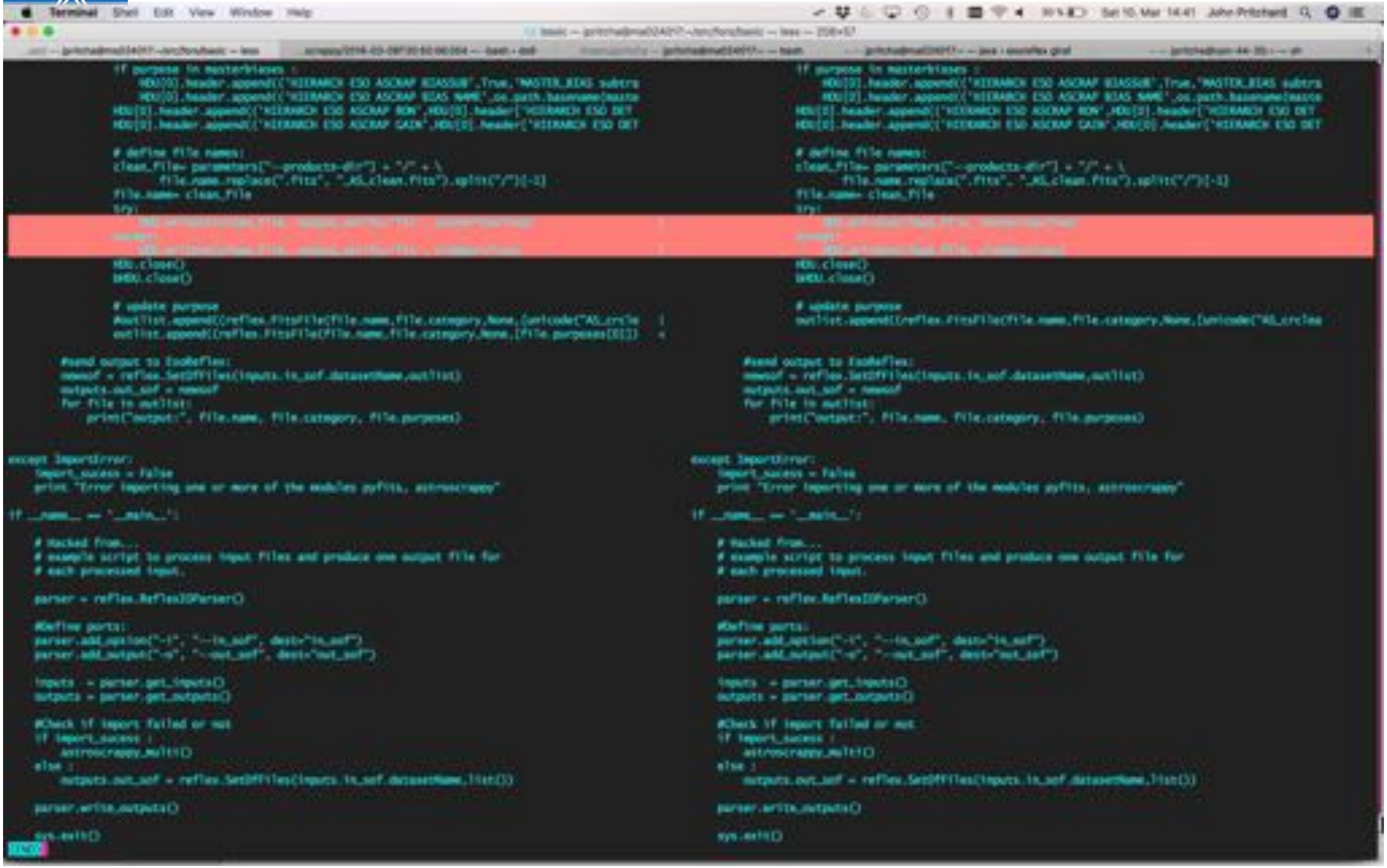
```
            logging.error('Could not open MASTER_BIAS %s' %(masterbiases[purpose
        except:
            pass
        sys.exit(1)

    if not 'CRMASK' in HDU :
        HDU.append(HDU[0])
        HDU[-1].header['EXTNAME']='CRMASK'
        HDU['CRMASK'].header['HIERARCH ESO PRO CATG']='CRMASK'

[illegible]
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )
    else :
        mask, HDU[0].data = astroscrappy.detect_cosmics(
            HDU[0].data,
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )

    #HDU[0].header['BZERO']=0.
    if 'BZERO' in  HDU[0].header :
        del HDU[0].header['BZERO']
    HDU[0].data=HDU[0].data/HDU[0].header['HIERARCH ESO DET OUT1 CONAD']
    if purpose in masterbiases :
        HDU[0].data-=mHDU[0].data
    HDU['CRMASK'].data=mask.astype(int)
    HDU['CRMASK'].header['BZERO']=0.
    HDU[0].header.append(("NFILES", 1, "number of input files"))
    HDU[0].header.append(("CMETHOD", "ASTROSCRAPPY", "method for bad pixel cleaning"

    HDU[0].header.append(('HIERARCH ESO ASCRAP',True,'AstroScrappy Cosmic Ray cleani
    if purpose in masterbiases :
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
    HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
    HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

    # define file names:
    clean_file= parameters["--products-dir"] + "/" + \
            file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
    file.name= clean_file
    try:
        HDU.writeto(clean_file, overwrite=True)
```





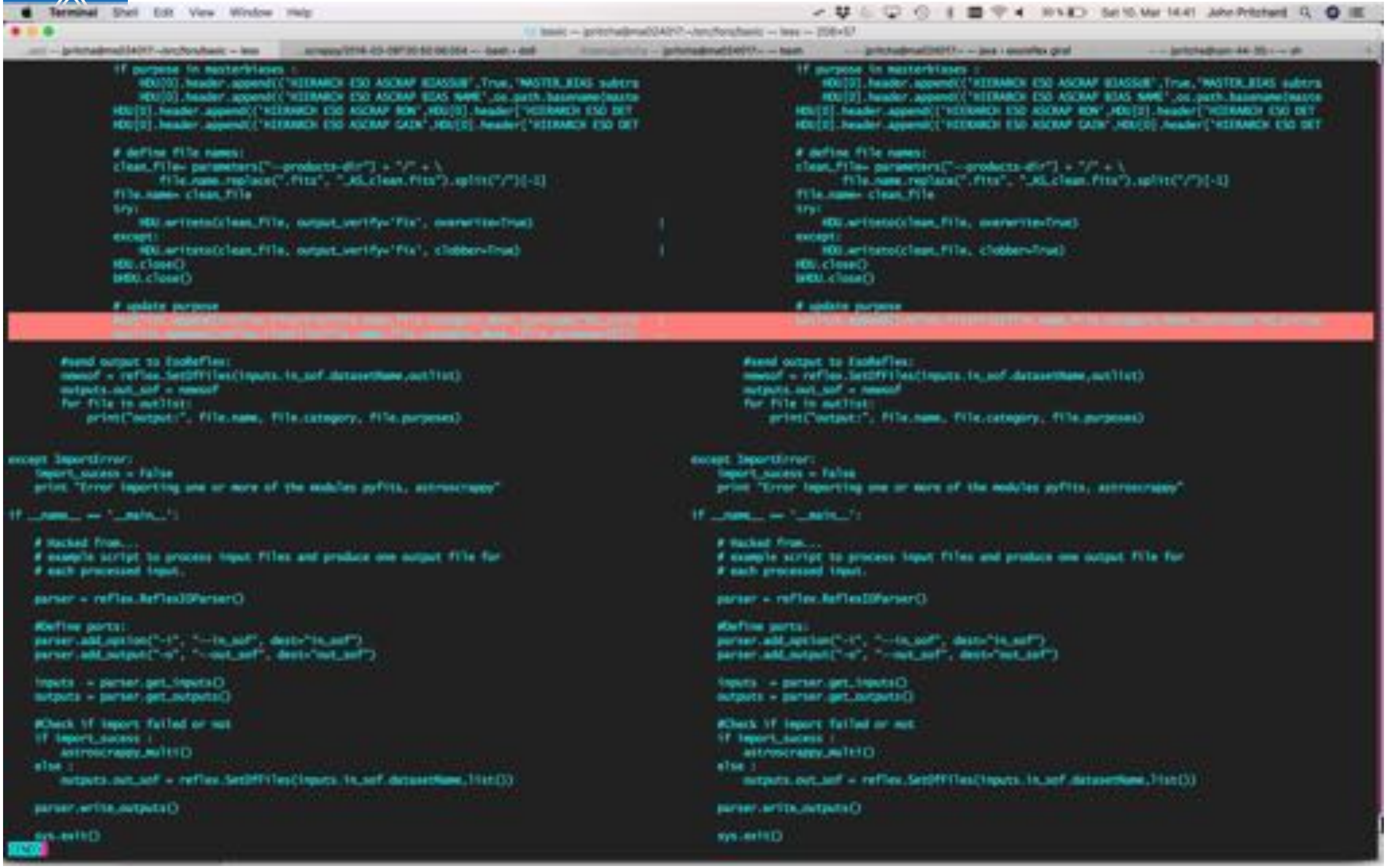
```
            logging.error('Could not open MASTER_BIAS %s' %(masterbiases[purpose
        except:
            pass
        sys.exit(1)

    if not 'CRMASK' in HDU :
        HDU.append(HDU[0])
        HDU[-1].header['EXTNAME']='CRMASK'
        HDU['CRMASK'].header['HIERARCH ESO PRO CATG']='CRMASK'

    if purpose in masterbiases :
        ## FORS biases have the overscan removed...
        NAXIS1=HDU[0].header['NAXIS1']
        NAXIS2=HDU[0].header['NAXIS2']
        PRSCX=HDU[0].header['HIERARCH ESO DET OUT1 PRSCX']
        OVSCX=HDU[0].header['HIERARCH ESO DET OUT1 OVSCX']
        PRSCY=HDU[0].header['HIERARCH ESO DET OUT1 PRSCY']
        OVSCY=HDU[0].header['HIERARCH ESO DET OUT1 OVSCY']
        mask=numpy.zeros((NAXIS2,NAXIS1))
        mask[PRSCY:NAXIS2-OVSCY,PRSCX:NAXIS1-OVSCX], HDU[0].data[PRSCY:NAXIS2-OVSCY,
            HDU[0].data[PRSCY:NAXIS2-OVSCY,PRSCX:NAXIS1-OVSCX]-bHDU[0].data,
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )
    else :
        mask, HDU[0].data = astroscrappy.detect_cosmics(
            HDU[0].data,
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )

    #HDU[0].header['BZERO']=0.
    if 'BZERO' in  HDU[0].header :
        del HDU[0].header['BZERO']
    HDU[0].data=HDU[0].data/HDU[0].header['HIERARCH ESO DET OUT1 CONAD']
    if purpose in masterbiases :
        HDU[0].data[PRSCY:NAXIS2-OVSCY,PRSCX:NAXIS1-OVSCX]+=bHDU[0].data
    HDU['CRMASK'].data=mask.astype(int)
    HDU['CRMASK'].header['BZERO']=0.
    HDU[0].header.append(("NFILES", 1, "number of input files"))
    HDU[0].header.append(("CMETHOD", "ASTROSCRAPPY", "method for bad pixel cleaning"
    HDU[0].header.append(("[illegible]", "ASTROSCRAPPY", "method for bad
    HDU[0].header.append(('HIERARCH ESO ASCRAP',True,'AstroScrappy Cosmic Ray cleani
    if purpose in masterbiases :
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
    HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
    HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

    # define file names:
    clean_file= parameters["--products-dir"] + "/" + \
            file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
    file.name= clean_file
    try:
        HDU.writeto(clean_file, output_verify='fix', overwrite=True)
```

```
            logging.error('Could not open MASTER_BIAS %s' %(masterbiases[purpose
        except:
            pass
        sys.exit(1)

    if not 'CRMASK' in HDU :
        HDU.append(HDU[0])
        HDU[-1].header['EXTNAME']='CRMASK'
        HDU['CRMASK'].header['HIERARCH ESO PRO CATG']='CRMASK'

    if purpose in masterbiases :
        mask, HDU[0].data = astroscrappy.detect_cosmics(
            HDU[0].data-bHDU[0].data,








            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )
    else :
        mask, HDU[0].data = astroscrappy.detect_cosmics(
            HDU[0].data,
            gain=HDU[0].header['HIERARCH ESO DET OUT1 CONAD'],
            readnoise=HDU[0].header['HIERARCH ESO DET OUT1 RON']
        )

    #HDU[0].header['BZERO']=0.
    if 'BZERO' in  HDU[0].header :
        del HDU[0].header['BZERO']
    HDU[0].data=HDU[0].data/HDU[0].header['HIERARCH ESO DET OUT1 CONAD']
    if purpose in masterbiases :
        HDU[0].data+=bHDU[0].data
    HDU['CRMASK'].data=mask.astype(int)
    HDU['CRMASK'].header['BZERO']=0.
    HDU[0].header.append(("NFILES", 1, "number of input files"))
    HDU[0].header.append(("CMETHOD", "ASTROSCRAPPY", "method for bad pixel cleaning"

    HDU[0].header.append(('HIERARCH ESO ASCRAP',True,'AstroScrappy Cosmic Ray cleani
    if purpose in masterbiases :
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
        HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
    HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
    HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

    # define file names:
    clean_file= parameters["--products-dir"] + "/" + \
            file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
    file.name= clean_file
    try:
        HDU.writeto(clean_file, overwrite=True)
```



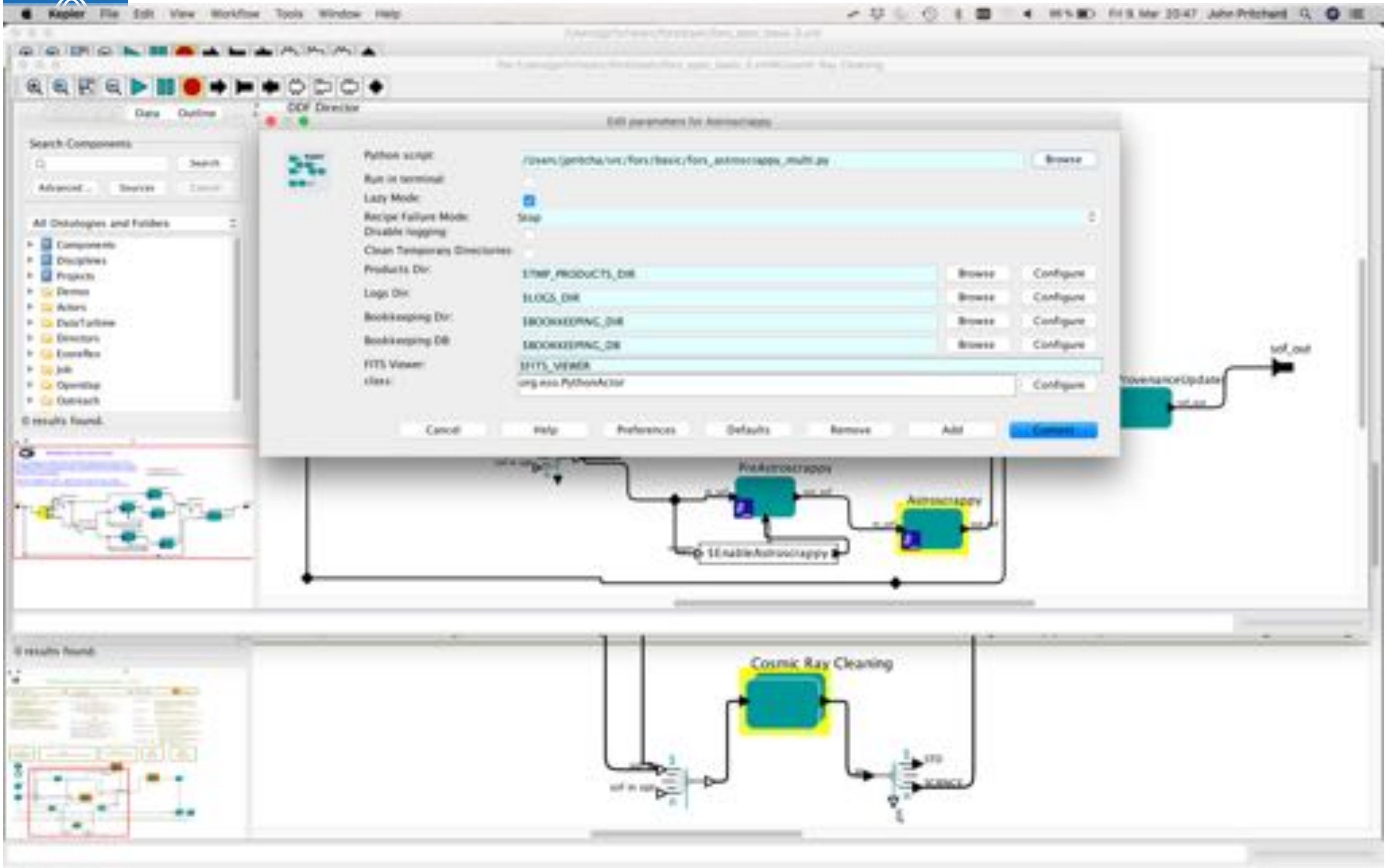
```
                if purpose in masterbiases :
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
                HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
                HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

                # define file names:
                clean_file= parameters["--products-dir"] + "/" + \
                       file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
                file.name= clean_file
                try:
                    [illegible]
                [illegible]
                    [illegible]
                HDU.close()
                BHDU.close()

                # update purpose
                #outlist.append((reflex.FitsFile(file.name,file.category,None,[unicode("AS_crcle
                outlist.append((reflex.FitsFile(file.name,file.category,None,[file.purposes[0]]))

        #send output to EsoReflex:
        newsof = reflex.SetOfFiles(inputs.in_sof.datasetName,outlist)
        outputs.out_sof = newsof
        for file in outlist:
            print("output:", file.name, file.category, file.purposes)


except ImportError:
    import_success = False
    print "Error importing one or more of the modules pyfits, astroscrappy"

if __name__ == '__main__':

    # Hacked from...
    # example script to process input files and produce one output file for
    # each processed input.

    parser = reflex.ReflexIOParser()

    #Define ports:
    parser.add_option("-i", "--in_sof", dest="in_sof")
    parser.add_output("-o", "--out_sof", dest="out_sof")

    inputs  = parser.get_inputs()
    outputs = parser.get_outputs()

    #Check if import failed or not
    if import_success :
        astroscrappy_multi()
    else :
        outputs.out_sof = reflex.SetOfFiles(inputs.in_sof.datasetName,list())

    parser.write_outputs()

    sys.exit()
(END)
```

```
                if purpose in masterbiases :
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
                HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
                HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

                # define file names:
                clean_file= parameters["--products-dir"] + "/" + \
                       file.name.replace(".fits", "_AS_clean.fits").split("/")[-1]
                file.name= clean_file
                try:
                    [illegible]
                [illegible]
                    [illegible]
                HDU.close()
                BHDU.close()

                # update purpose
                outlist.append((reflex.FitsFile(file.name,file.category,None,[unicode("AS_crclea


        #send output to EsoReflex:
        newsof = reflex.SetOfFiles(inputs.in_sof.datasetName,outlist)
        outputs.out_sof = newsof
        for file in outlist:
            print("output:", file.name, file.category, file.purposes)


except ImportError:
    import_success = False
    print "Error importing one or more of the modules pyfits, astroscrappy"

if __name__ == '__main__':

    # Hacked from...
    # example script to process input files and produce one output file for
    # each processed input.

    parser = reflex.ReflexIOParser()

    #Define ports:
    parser.add_option("-i", "--in_sof", dest="in_sof")
    parser.add_output("-o", "--out_sof", dest="out_sof")

    inputs  = parser.get_inputs()
    outputs = parser.get_outputs()

    #Check if import failed or not
    if import_success :
        astroscrappy_multi()
    else :
        outputs.out_sof = reflex.SetOfFiles(inputs.in_sof.datasetName,list())

    parser.write_outputs()

    sys.exit()
```



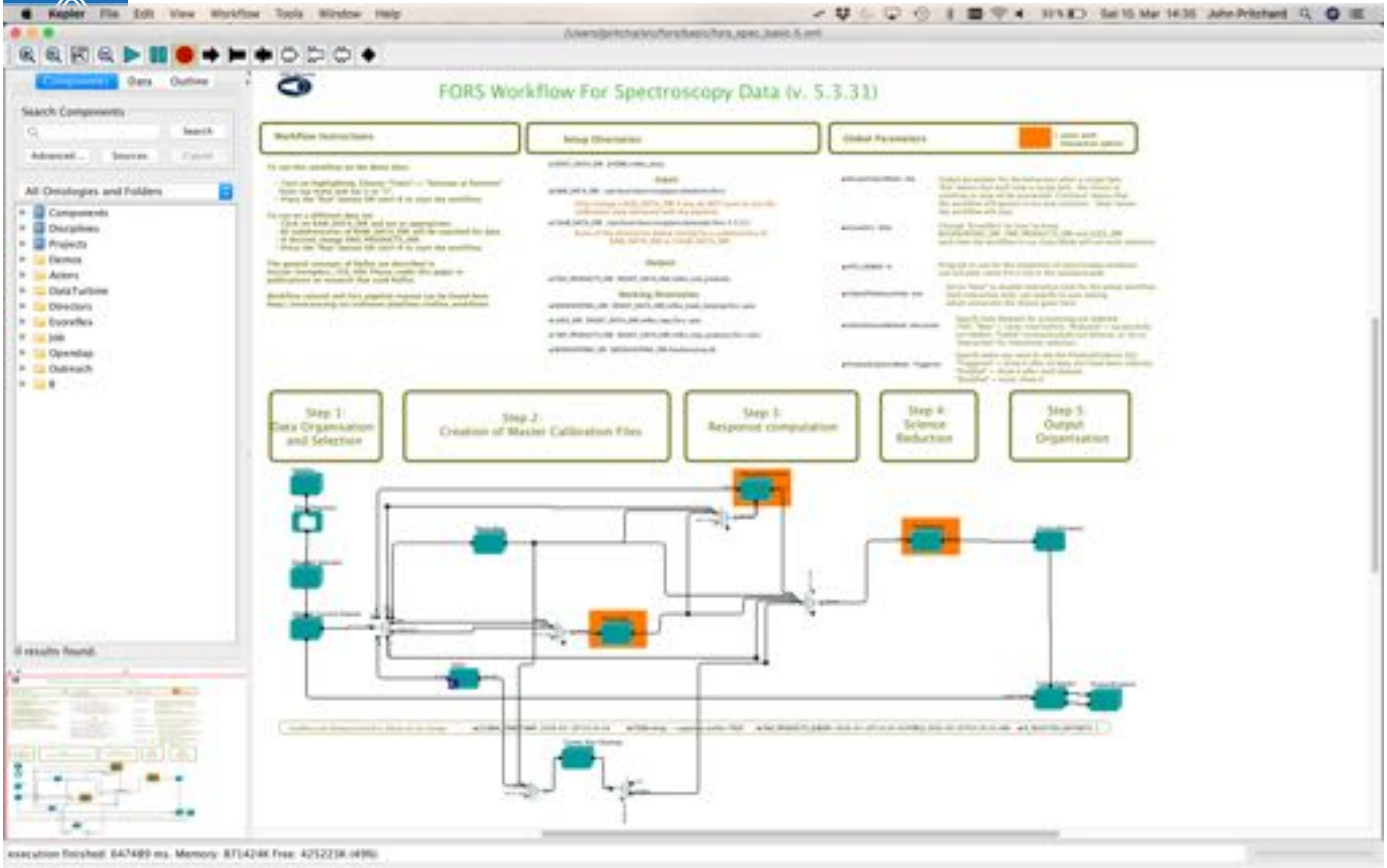
```
                if purpose in masterbiases :
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
                HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
                HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

                # define file names:
                clean_file= parameters["--products-dir"] + "/" + \
                       file_name.replace(".fits", "_AS_clean.fits").split("/")[-1]
                file.name= clean_file
                try:
                    HDU.writeto(clean_file, output_verify='fix', overwrite=True)
                except:
                    HDU.writeto(clean_file, output_verify='fix', clobber=True)
                HDU.close()
                BHDU.close()

                # update purpose
                [illegible]
                [illegible]

        #send output to EsoReflex:
        newsof = reflex.SetOfFiles(inputs.in_sof.datasetName,outlist)
        outputs.out_sof = newsof
        for file in outlist:
            print("output:", file.name, file.category, file.purposes)

except ImportError:
    import_success = False
    print "Error importing one or more of the modules pyfits, astroscrappy"

if __name__ == '__main__':

    # Hacked from...
    # example script to process input files and produce one output file for
    # each processed input.

    parser = reflex.ReflexIOParser()

    #Define ports:
    parser.add_option("-i", "--in_sof", dest="in_sof")
    parser.add_output("-o", "--out_sof", dest="out_sof")

    inputs  = parser.get_inputs()
    outputs = parser.get_outputs()

    #Check if import failed or not
    if import_success :
        astroscrappy_multi()
    else :
        outputs.out_sof = reflex.SetOfFiles(inputs.in_sof.datasetName,list())

    parser.write_outputs()

    sys.exit()
```

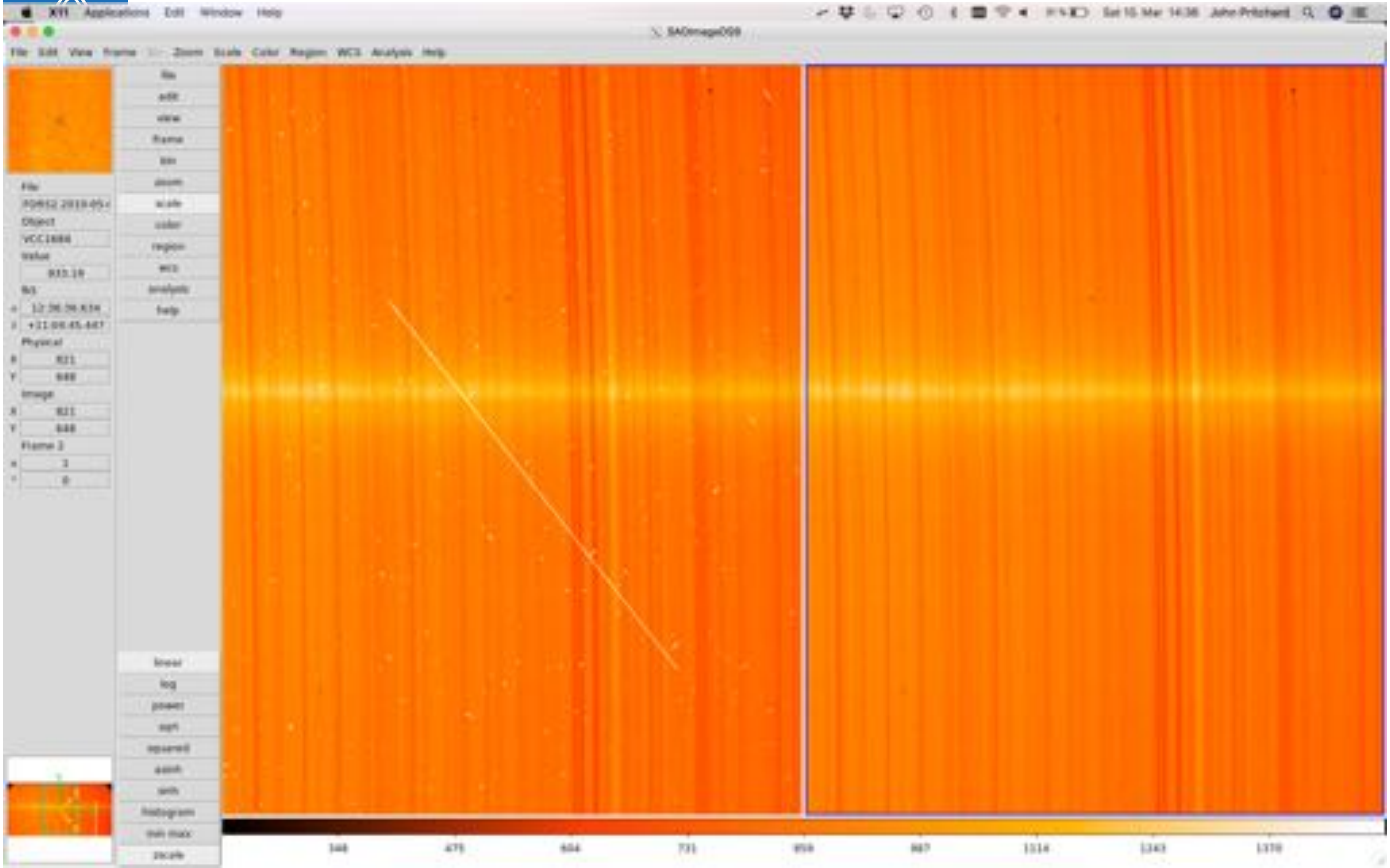
```
                if purpose in masterbiases :
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIASSUB',True,'MASTER_BIAS subtra
                    HDU[0].header.append(('HIERARCH ESO ASCRAP BIAS NAME',os.path.basename(maste
                HDU[0].header.append(('HIERARCH ESO ASCRAP RON',HDU[0].header['HIERARCH ESO DET
                HDU[0].header.append(('HIERARCH ESO ASCRAP GAIN',HDU[0].header['HIERARCH ESO DET

                # define file names:
                clean_file= parameters["--products-dir"] + "/" + \
                       file_name.replace(".fits", "_AS_clean.fits").split("/")[-1]
                file.name= clean_file
                try:
                    HDU.writeto(clean_file, overwrite=True)
                except:
                    HDU.writeto(clean_file, clobber=True)
                HDU.close()
                BHDU.close()

                # update purpose
                [illegible]

        #send output to EsoReflex:
        newsof = reflex.SetOfFiles(inputs.in_sof.datasetName,outlist)
        outputs.out_sof = newsof
        for file in outlist:
            print("output:", file.name, file.category, file.purposes)

except ImportError:
    import_success = False
    print "Error importing one or more of the modules pyfits, astroscrappy"

if __name__ == '__main__':

    # Hacked from...
    # example script to process input files and produce one output file for
    # each processed input.

    parser = reflex.ReflexIOParser()

    #Define ports:
    parser.add_option("-i", "--in_sof", dest="in_sof")
    parser.add_output("-o", "--out_sof", dest="out_sof")

    inputs  = parser.get_inputs()
    outputs = parser.get_outputs()

    #Check if import failed or not
    if import_success :
        astroscrappy_multi()
    else :
        outputs.out_sof = reflex.SetOfFiles(inputs.in_sof.datasetName,list())

    parser.write_outputs()

    sys.exit()
```

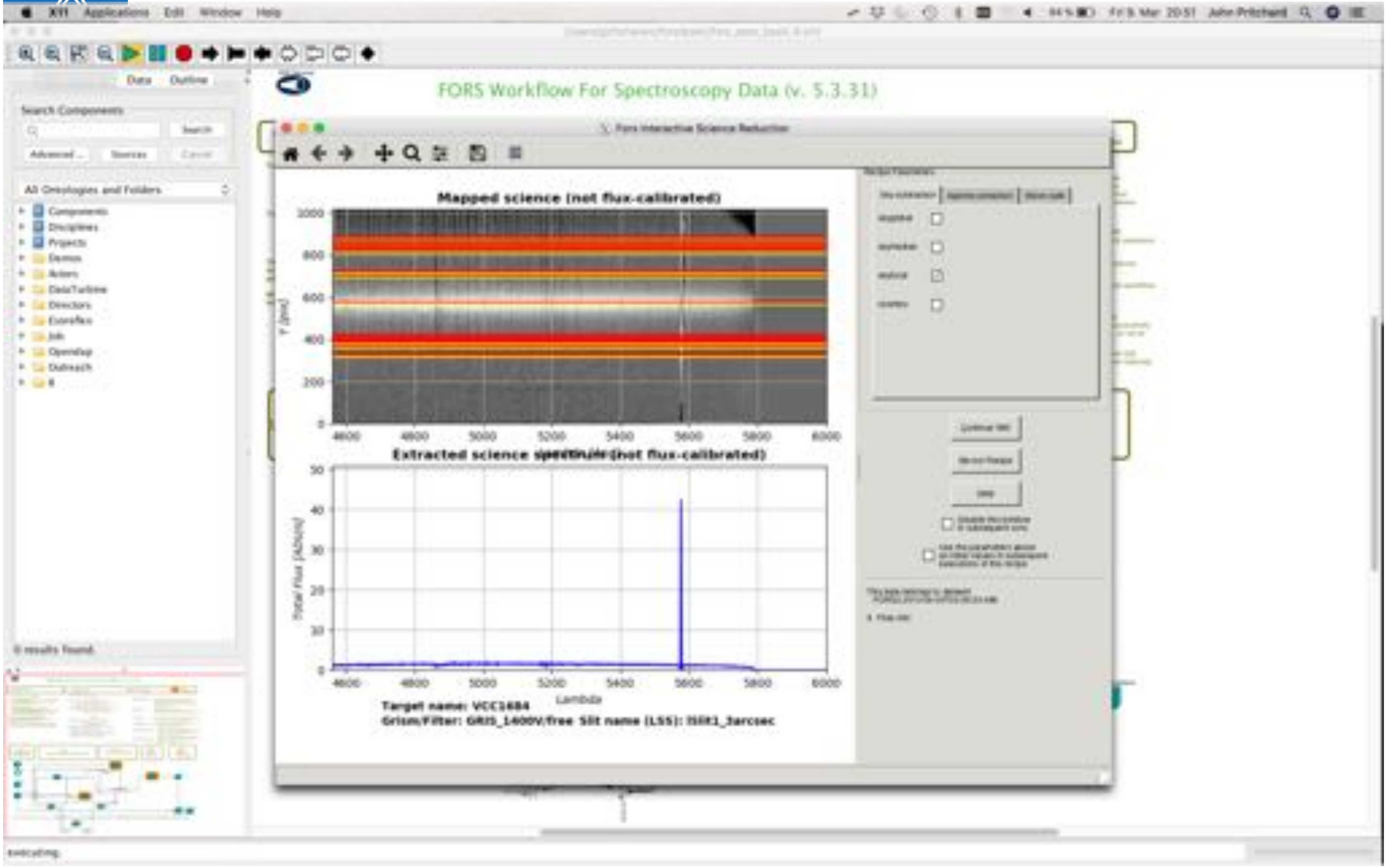


Edit parameters for Astroscrappy

- Python script: /Users/jpritcha/src/fors/basic/fors_astroscrappy_multi.py
- Run in terminal: ☐
- Lazy Mode: ☑
- Recipe Failure Mode: Stop
- Disable logging: ☐
- Clean Temporary Directories: ☐
- Products Dir: $TMP_PRODUCTS_DIR
- Logs Dir: $LOGS_DIR
- Bookkeeping Dir: $BOOKKEEPING_DIR
- Bookkeeping DB: $BOOKKEEPING_DB
- FITS Viewer: $FITS_VIEWER
- class: org.eso.PythonActor

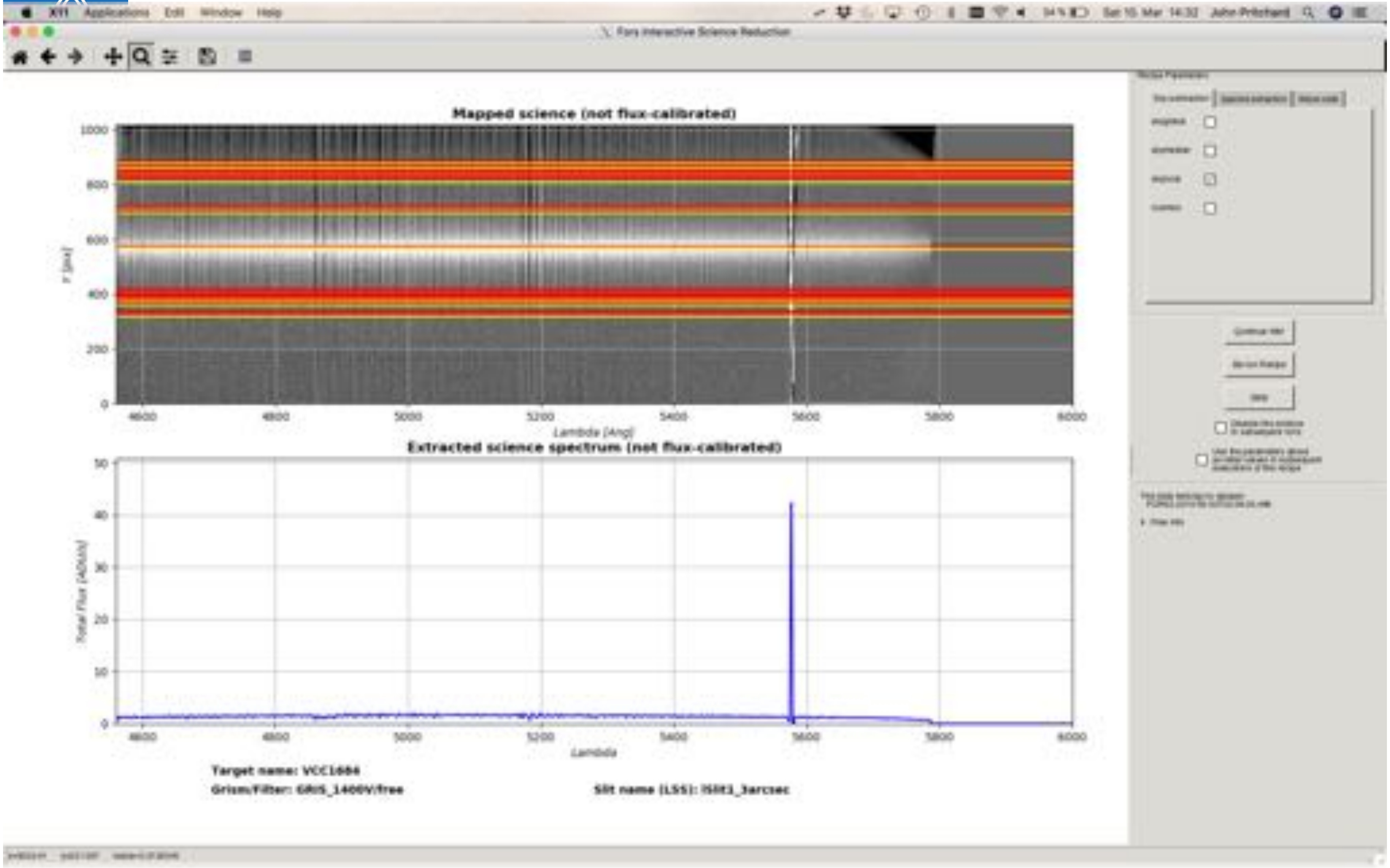

Kepler File Edit View Workflow Tools Window Help
Components
Data
Outline
Search Components
Search
Advanced...
Sources
Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Directors
Eworkflex
Job
Opendap
Outreach
0 results found.
FORS Workflow For Spectroscopy Data (v. 5.3.31)
Workflow Instructions
Setup Directories
Global Parameters
Step 1
Data Organisation and Selection
Step 2
Creation of Master Calibration Files
Step 3
Response computation
Step 4
Science Reduction
Step 5
Output Organisation


SAOImageDS9
File
Edit
View
Frame
Zoom
Scale
Color
Region
WCS
Analysis
Help
file
edit
view
frame
bin
zoom
scale
color
region
wcs
analysis
help
linear
log
power
sqrt
squared
asinh
sinh
histogram
min max
zscale
File
Object
Value
Physical
Image



FORS Workflow For Spectroscopy Data (v. 5.3.31)

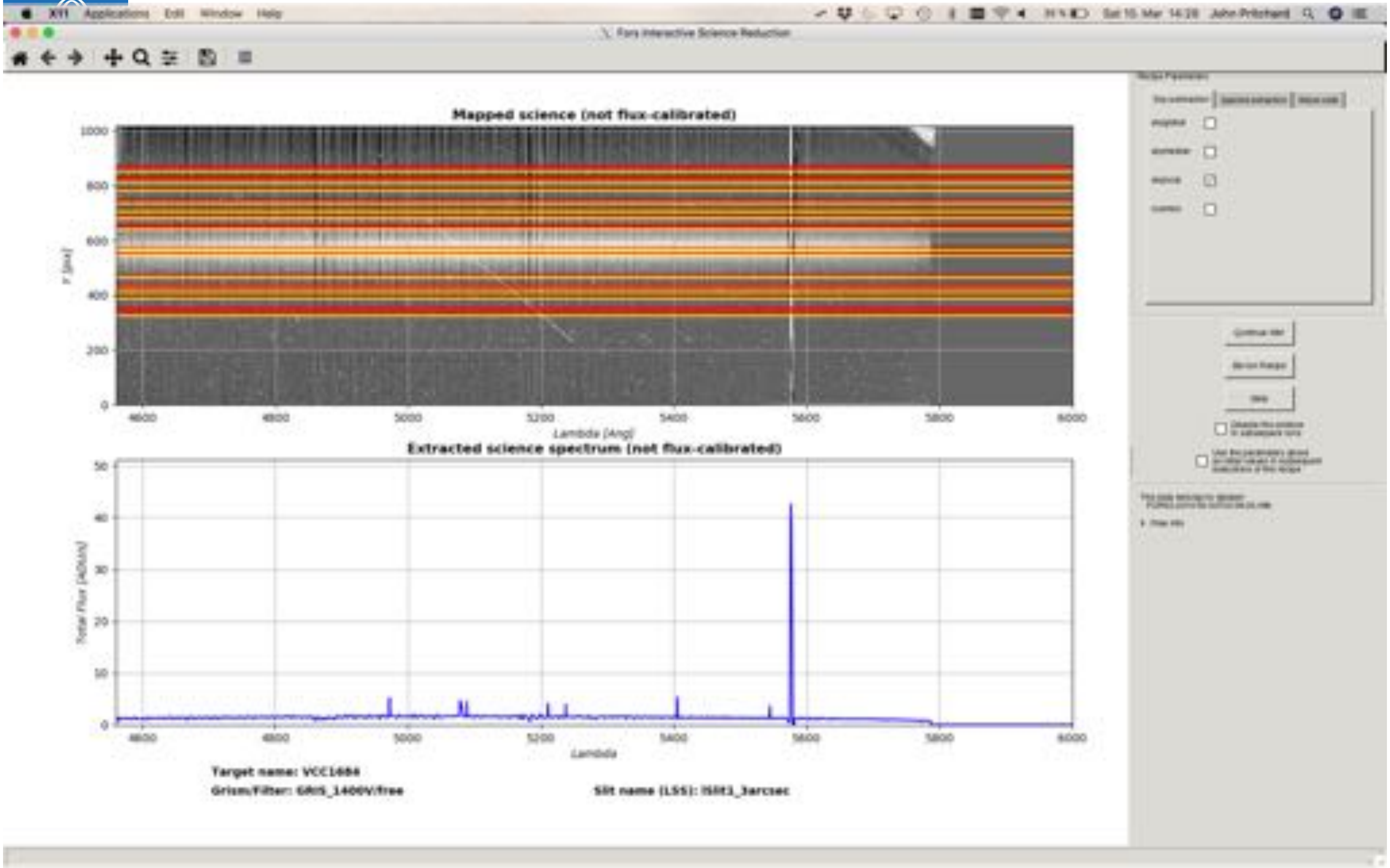
Target name: VCC1684
Grism/Filter: GRIS_1400V/free Slit name (LSS): lSlit1_1arcsec



Target name: VCC1684

Grism/Filter: GRIS_1400V/free

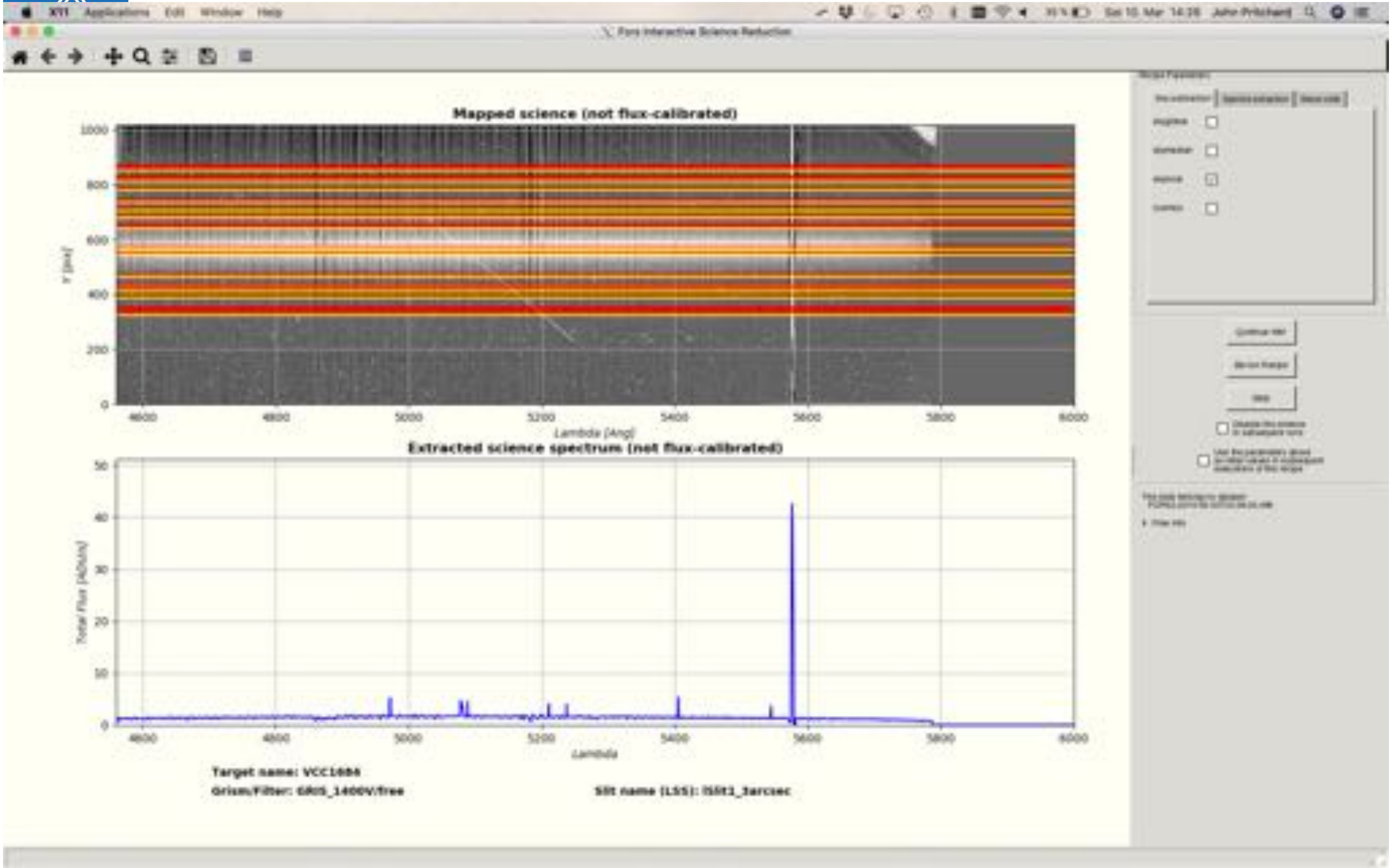
Slit name (LSS): lSlit1_3arcsec



Target name: VCC1684

Grism/Filter: GRIS_1400V/free

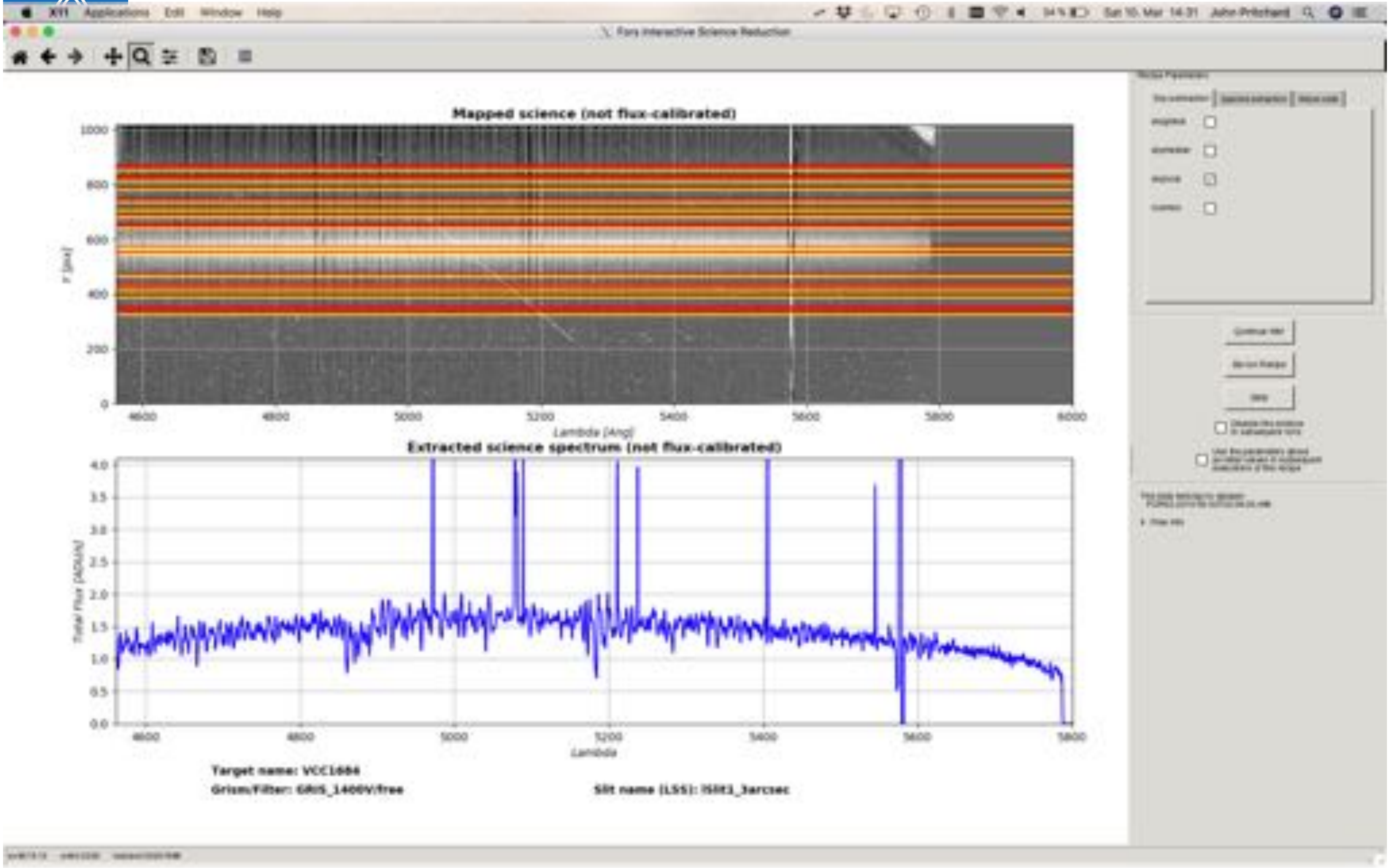
Slit name (LSS): lSlit1_3arcsec



Target name: VCC1084

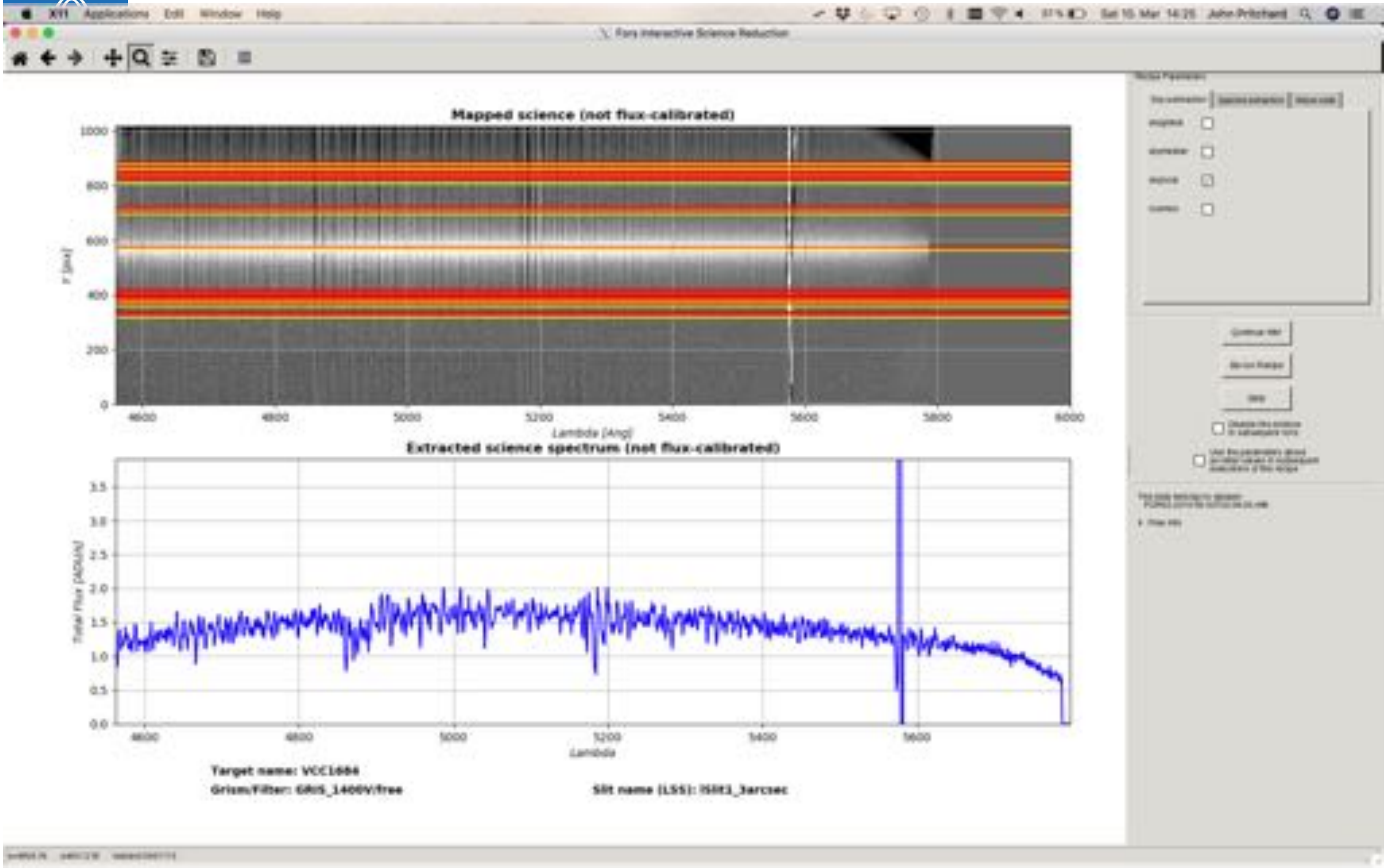
Grism/Filter: GRIS_1400V/free

Slit name (LSS): lSlit1_3arcsec

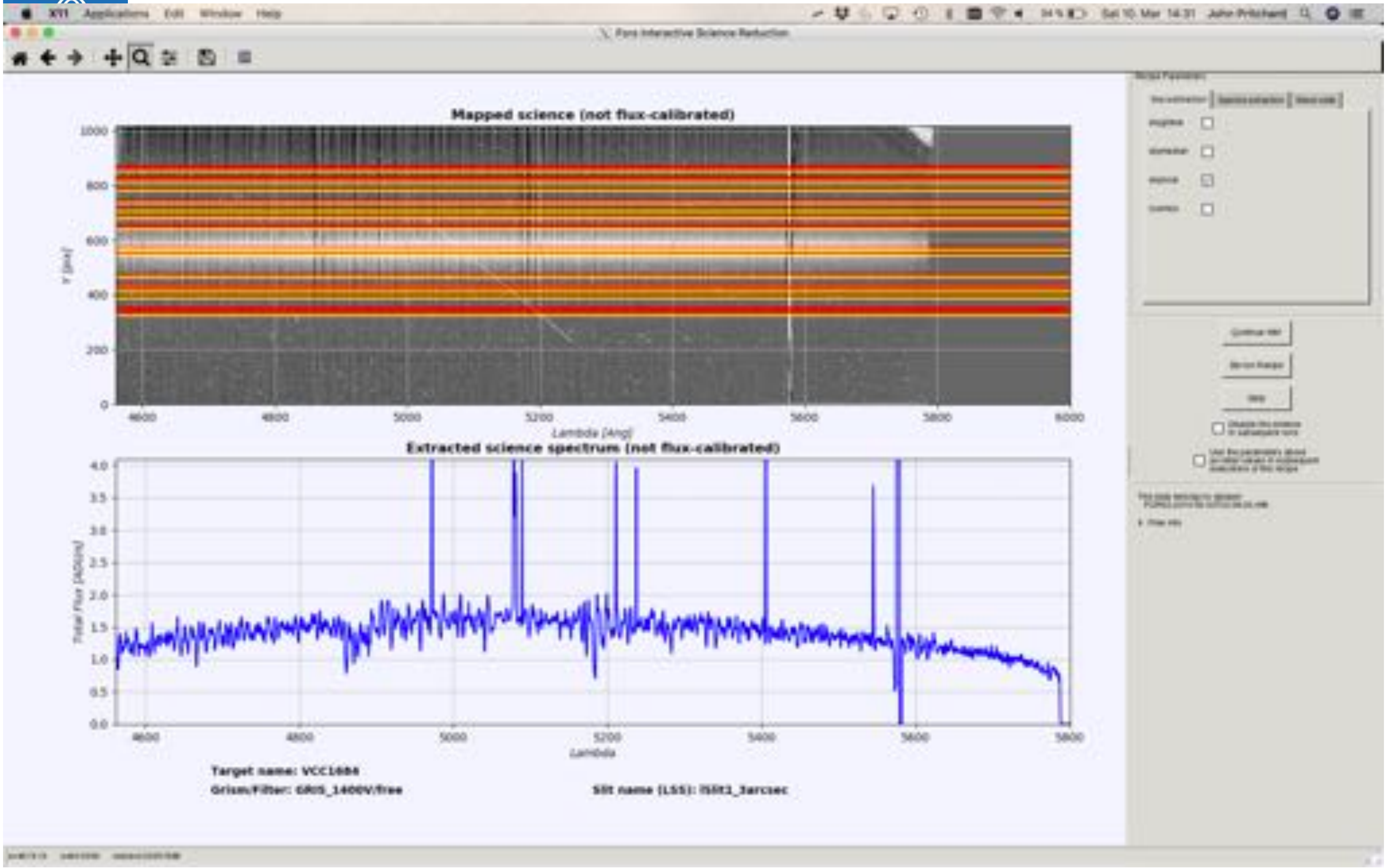


Target name: VCC1684
Grism/Filter: GRIS_1400V/free

Slit name (LSS): lSlit1_3arcsec



Target name: VCC1684
Grism/Filter: GRIS_1400V/free
Slit name (LSS): lSlit1_3arcsec



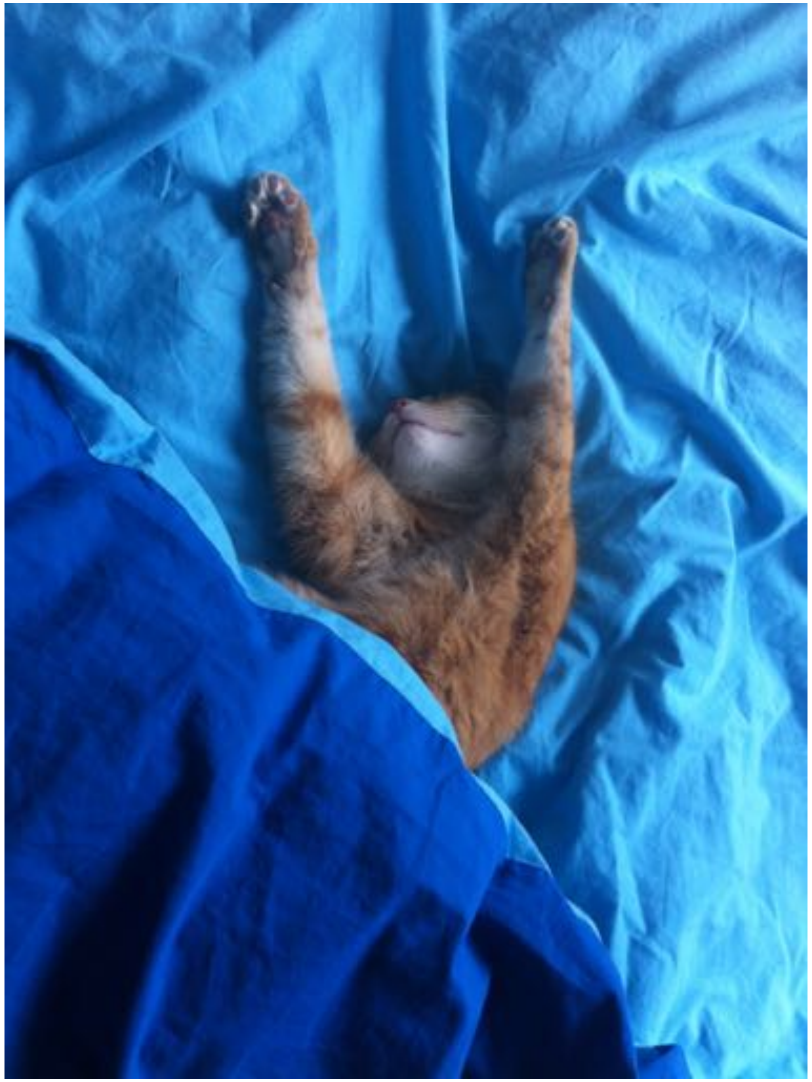
Target name: VCC1684

Grism/Filter: GRIS_1400V/free

Slit name (LSS): lSlit1_3arcsec

The end…really this time…

- And that really is “How to modify a workflow…”



Oh, and one more thing…

- Exercises for the workshop participants…
  - Implement the PyCosmic and Median+Pass-Thru
  - Implement the handling of multiple versions of the same RAW file in the ForsScience actor
  - Implement the handling of multiple versions of the same RAW file in the ForsScience actor GUI interface

Actually I've done all this (except the last step) but it is a bit beyond the scope of today's presentation… but I hope it will find its way – when finished with the last step – into the public release of the workflow (if not for 2018, then at least for 2019).