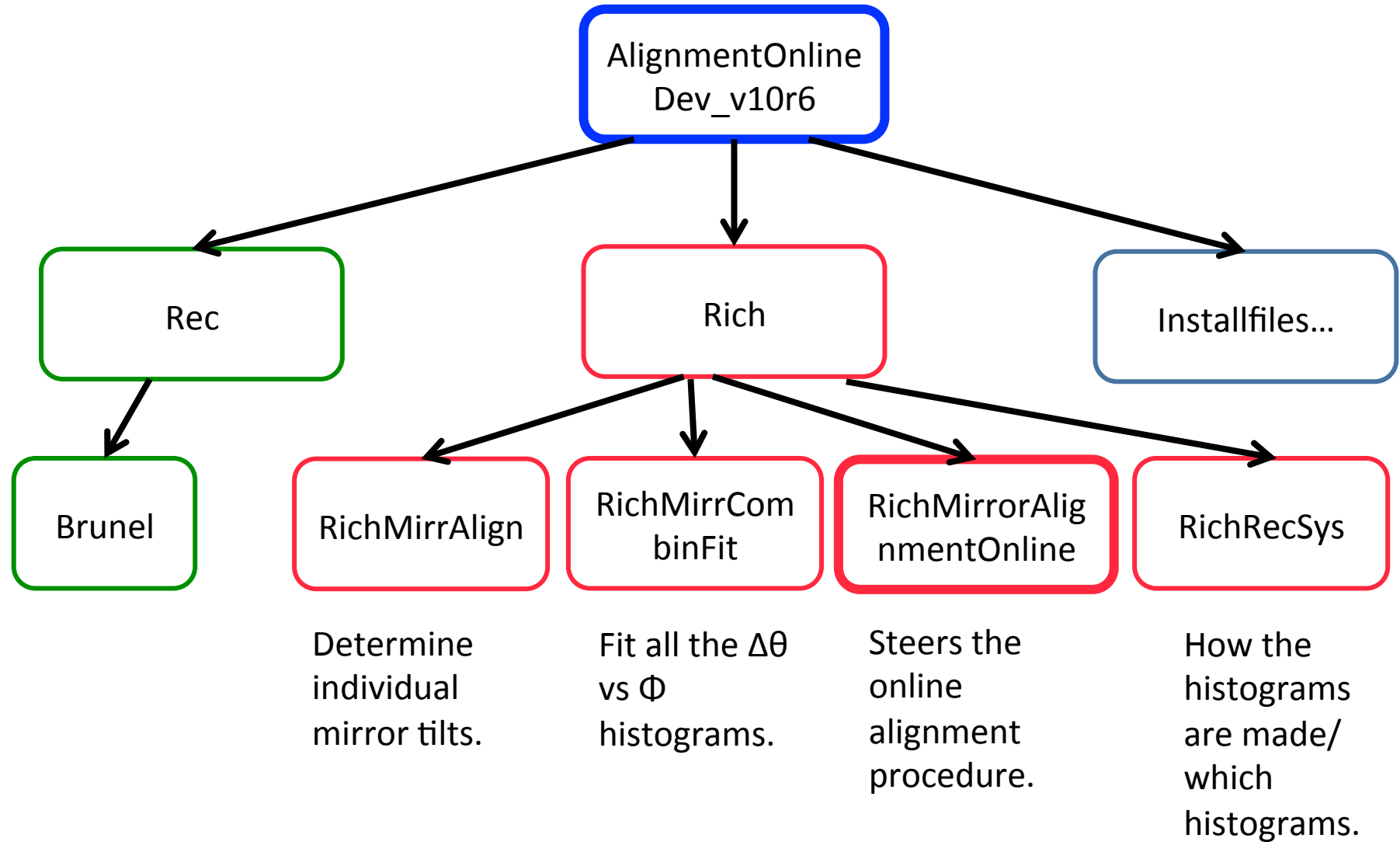


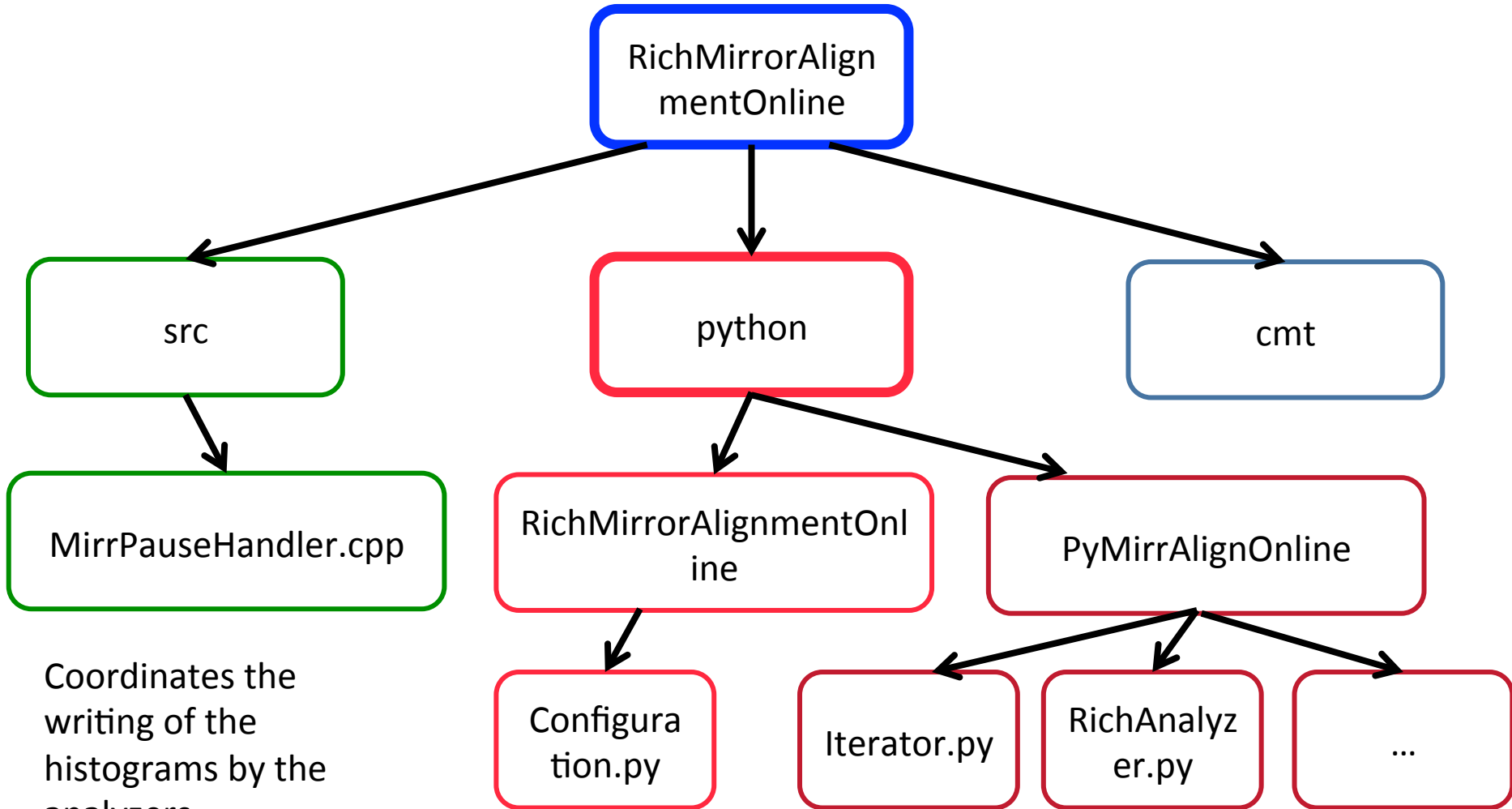
# Rich Mirror Alignment Code

Claire Prouve

# AlignmentOnlineDev\_v10r6



# RichMirrorAlignmentOnline



Coordinates the writing of the histograms by the analyzers.

# Configuration file (1/2)

**Location:** /group/rich/sw/cmtuser/AlignmentOnlineDev\_v10r6/Rich/  
RichMirrorAlignmentOnline/python/RichMirrorAlignmentOnline/Configuration.py

```
# @package RichMirrorAlignmentOnline
# @author Claire Prouve <Claire.Prouve@cern.ch>
# @date 08/07/2015

__author__ = "Claire Prouve <Claire.Prouve@cern.ch>"

from Gaudi.Configuration import *
import GaudiKernel.ProcessJobOptions
from Configurables import ( LHCbConfigurableUser)

class Rich1MirrAlignOnConf(LHCbConfigurableUser):
    __used_configurables__ = [ ]
    __slots__ = {
        "MajItStart" : 0
        , "MinItStart" : 0
    }
...

```

## 1. Rich1MirrAlignOnConf

```
class Rich2MirrAlignOnConf(LHCbConfigurableUser):
    __used_configurables__ = [ ]

    __slots__ = {
        "MajItStart" : 0
        , "MinItStart" : 0
    }
...

```

## 2. Rich2MirrAlignOnConf

# Configuration file (2/2)

## Configuration-variables explained below the listing:

```
_propertyDocDct = {
  "MajItStart"      : "" Start the alignment at this major iteration. Please make sure this is consitent with MinItStart. ""
  , "MinItStart"    : "" Start the alignment at this minor iteration. Please make sure this is consitent with MajItStart. ""
  , "Rich"          : "" Rich1 or Rich2. ""
  , "HistoDir"       : "" Directory where the savesets are being written to. ""
  , "WorkDir"        : "" Directory in which all the output will be written. ""
  , "coeffCalibTilt" : "" Mirror-tilts applied for the calculation of the magnification coefficients. ""
  , "minAverageBinPop": "" Demanded minimal entries per bin in x-y-bin; atm 6: for 20 phi-bins, 4.8: for 25 phi-bins. ""
  , "phiBinFactor"   : "" Factor by which the number of phi-bins is reduced. The histograms should now come with 60 phi bins, fa
o 20 bin in the fit.""
  , "deltaThetaWindow": "" dTheta-range in the histograms; 4.0 for Rich1 and 3.0 for Rich2. ""
  , "combinFitVariant": "" Method for fitting the 2D histograms; 1: first fit the slices of dTheta with a Gaussian and then fit t
mean of the Gaussian, 3: fit a 2D function ""
}
```

## After a change: **compile!**

```
cd /group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6
```

```
export OnlineDev_DIR=/group/online/dataflow/cmtuser/OnlineDev_v5r29/InstallArea/$CMTOPT
```

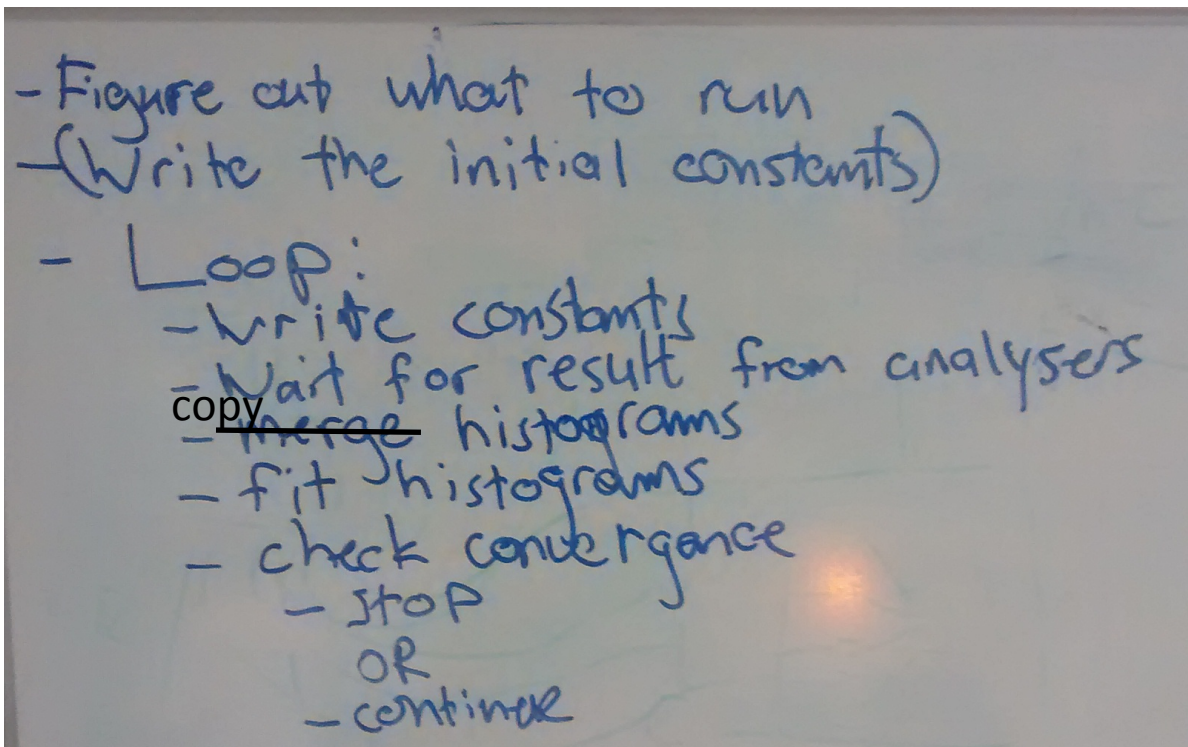
```
make -j 8 install
```



Or the corresponding version of OnlineDev listed in  
/group/rich/sw/cmtuser/AlignmentOnlineDev\_v10r6/CMakeLists.txt

# The Iterator (1/2)

Here is basically what the iterator does:

- 
- A photograph of a whiteboard with handwritten text in blue ink. The text is organized into a list of steps for an iterator process. The steps are: '- Figure out what to run', '- (Write the initial constants)', '- Loop:', '- write constants', '- Wait for result from analysers', '- merge histograms', '- fit histograms', '- check convergence', '- stop OR', and '- continue'. The word 'copy' is written in small text next to the 'merge histograms' step.
- Figure out what to run
  - (Write the initial constants)
  - Loop:
    - write constants
    - Wait for result from analysers
    - merge histograms
    - fit histograms
    - check convergence
    - stop OR
    - continue

The **Iterator** contains most of the code that was previously in RichMirrorAlignment.py. It sets up all the xml-files, takes the rootfiles and executes RichMirrCombinFit and RichMirrAlign and makes the decision whether or not the alignment converged.

Iterator does all his work while he is in state *paused*. When his thing is done he sets his own state to *running* and then the analyzers know to run too.

Iterator code: /group/rich/sw/cmtuser/AlignmentOnlineDev\_v10r6/Rich/RichMirrorAlignmentOnline/python/PyMirrAlignOnline/Iterator.py

# The Iterator (2/2)

Everything happens in the ***run()*** method (more comments can be found in the code):

1. *whichRich* is provided by the choosing Rich1 or Rich2 in the run-control while running the alignment.
2. The configuration is read in and variables are set.
3. The while loop begins.
4. At the start of the first iteration the work-directory will be emptied out.
5. The starting xml-file is gotten from '/group/online/alignment/Rich' + str(whichRich) + '/MirrorAlign/'
5. The xml-files for the mirror-tilts for the magnification coefficients are made.
6. The xml-file for the first iteration put in the place where the analyzers will pick it up.
7. Set Iterator state to READY which will make the analyzers run their jobs.
8. When the analyzers are READY the iterator will get the command "pause" and will check if one major iteration has been finished already:
  - if not then it just copied the xml file for the next iteration to the place where the analyzers can pick it up
  - if yes it will perform the fits and the alignment and check weather or not is has converged. If yes it's finished, if no it starts again at 7.

# The Analyzer

The Analyzer is the entity that will execute the Brunel jobs and produce the histogram files.

The Analyzer code contains the **options!!!** for the Brunel jobs, the code that you can see in the RichAnalyzer.py is only executed once when the program is 'configured'.

The analyzer is quite dumb, it can only pick up the xml-file for the reconstruction in one place (which means before each reconstruction the Iterator needs to put it in the right place in [/group/online/AligWork/Rich1/CondDB\\_Rich1.xml](#) or [/group/online/AligWork/Rich2/CondDB\\_Rich2.xml](#) ).

Analyzer code: [/group/rich/sw/cmtuser/AlignmentOnlineDev\\_v10r6/Rich/RichMirrorAlignmentOnline/python/PyMirrAlignOnline/RichAnalyzer.py](#)



# The Helper-Files

A lot of functions and methods the iterator uses are outsourced into “Helper-files”:  
RichAlignmentHelper.py, SetupHelper.py, HistoHelper.py, XMLFileHelper.py

SetupHelper: cleans out the work-directory, write summary file at the end of alignment...

XMLFileHelper: handles the XML files, gets the starting XML, makes the tilted XML-files for magnification coefficients...

HistoHelper: contains classes that wait for and retrieve the SaveSets (= output from the Analyzers) and copies them into working directory

RichAlignmentHelper: executes the methods needed for/using the RichMirrCombinFit and RichMirrAlignpackage

# RICH Mirror Alignment Code

The online alignment code is in

`/group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6`

This project contains everything the online alignment needs and needs to be setup exactly this way.

All the Rich-specific stuff will be in

`/group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6/Rich`

The code that drives the alignment is in

`/group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6/Rich/RichMirrorAlignmentOnline/  
python/PyMirrAlignOnline/`

and the configuration file is in

`/group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6/Rich/RichMirrorAlignmentOnline/  
python/RichMirrorAlignmentOnline/Configuration.py`

The actual alignment is started in (this 'belongs' to everyone)

`/group/online/dataflow/cmtuser/OnlineDev_v5r29/Online/FarmConfig/job/AligDrv.sh`

# RICH Mirror Alignment Code

The output of the alignment will be in (including the histograms)

`/group/online/AligWork/Rich1`

`/group/online/AligWork/Rich2`

The MirrPauseHandler (the one that coordinates the writing of the histograms)

`/group/rich/sw/cmtuser/AlignmentOnlineDev_v10r4/Rich/RichMirrorAlignmentOnline/  
MirrPauseHandler.cpp`

Original Histogram output location:

`/hist/Savesets/2015/LHCbA/AligWrk_Rich1`

`/hist/Savesets/2015/LHCbA/AligWrk_Rich2`

# Compiling

Compiling the code in [AlignmentOnlineDev\\_v10r6](#):

```
cd /group/rich/sw/cmtuser/AlignmentOnlineDev_v10r6  
export OnlineDev_DIR=/group/online/dataflow/cmtuser/OnlineDev_v5r29/InstallArea/  
$CMTOPT  
make configure  
make -j 8 install
```

# Monitoring

Problem: monitoring runs on the monitoring farm, the alignment runs on the alignment farm → cannot use DIM to communicate between the alignment and the monitoring task

→ Use zeroMQ to send the name of the newest histograms to the monitoring task (this part already works)

Monitoring tasks awaits information (in that order): iteration (int), Histogram file(string), RichMirrAlign\_out file (string).

1. From the histogram file the Cherenkov angle is fitted and filled into monitoring-histogram with resolution as y and iteration number as x
2. From the RichMirrAlign\_Out file the mirror tilts for the current iteration are extracted and filled into 2D histogram

Special feature: at the beginning of the task the monitor searches for the last converged alignment and put the last Cherenkov-angle resolution into the histogram under iteration -1