

## *MIPP JobControl framework: What is it?*

JobControl is a package that organizes programs that look at the “raw” data and perform analysis and reconstruction. It provides a very simple executable (“anamipp,” in the case of MIPP) which can be extended at run time to any bit of user reconstruction and analysis.

```
enrico2% anamipp --help
usage: anamipp [options] file1.root file2.root ...
options are:
-i file.root,--input=file.root    : Add input data file
-o file.root,--output=file.root   : Set output data file
-g file.root,--histo=file.root    : Set output histogram file
-n #, --nevt=#                   : Set number of events to process
-s #, --nskip=#                  : Skip number of events at start
-r #, --run=#                     : Skip to run number at start
-e #, --event=#                  : Skip to event number at start
-x #, --xml=file.xml             : Parse XML file
-p #, --xmlpath=dir               : Add directory to XML search path
-d #, --xmldir=dir                : Read all XML files in directory
-m #, --memory-report=#          : Summarize resources used energy n events
-l file.txt,--event-list=file    : Read event list from text file
-L #, --size-limit=#             : Largest acceptable file size (in MB)
```

anamipp only runs in batch mode. Interactive mode is supported via the event display

## *Organization of a job*

The basic unit of analysis and reconstruction is called a JobModule (more on these later). Modules are attached to a job as a “node” and nodes are grouped together into “Sequences”, and “Sequences” are grouped together to form “Jobs”. This is done via and XML configuration file:

```
<jobdoc>
  <xmlfile> TPCReco.xml TrkRBase.xml SPSegAssn.xml SPFit.xml </xmlfile>
  <link> NumericalMethods Swimmer SPFit </link>

  <job name="Tracking">
    <node sequence="TPCReco" filter="off"/>
    <node sequence="TrkRBase" filter="off"/>

    <node module="SPSegAssn" config="default" reco="1" ana="0" filter="off"/>
    <node module="SPFit" config="default" reco="1" ana="0" filter="off"/>
    <node module="SPVertexFind" config="default" reco="1" ana="0" filter="off"/>
  </job>

</jobdoc>
```

In this case, the job is extended by the NumericalMethods Swimmer and SPFit packages. The sequence “TPCReco” is a stand in for a long series of modules that perform all the stages of TPC reconstruction (clustering, hit finding,\ hit fitting, track finding, track fitting, etc. etc.) The sequence is loaded from the “TPCReco.xml” file. The user modules added to the job are “SPSegAssn”, “SPFit”, and “SPVertexFind”.

Here, for example, is what the TPCReco.xml file looks like:

```
<?xml version="1.0"?>
<jobdoc>

<link>
    Geometry
    RecoBase
    Bfield
    TPCRecoJP
</link>

<xmlfile>
    TPCR2DClusterFind.xml
    TPCRHitFind.xml
    TPCRTTrackFind.xml
    TPCRVertexFind.xml
</xmlfile>

<sequence name="TPCReco">
    <node module="TPCR2DClusterFind" config="default" reco="1" ana="0" filter="on"/>
    <node module="TPCRHitFind" config="default" reco="1" ana="0" filter="on"/>
    <node module="TPCRTTrackFind" config="default" reco="1" ana="0" filter="on"/>
    <node module="TPCRVertexFind" config="default" reco="1" ana="0" filter="off"/>
</sequence>

</jobdoc>
```

# Summary of job attached to end of log file:

```
** Thu May  4 09:38:48 2006 End job Tracking **
+-Tracking sequence (1/0/1 events 0.54u/0.05s seconds)
 | -TPCReco sequence (1/0/1 events 0.35u/0.05s seconds)
   +TPCR2DClusterFind/default/
     +reco/00001/00000/00001 2.50e-01/4.00e-02
     -ana /00000/00000/00000 0.00e+00/0.00e+00
   +TPCRHitFind/default/
     +reco/00001/00000/00001 2.00e-02/1.00e-02
     -ana /00000/00000/00000 0.00e+00/0.00e+00
   +TPCRTTrackFind/default/
     +reco/00001/00000/00001 8.00e-02/0.00e+00
     -ana /00000/00000/00000 0.00e+00/0.00e+00
   TPCRVertexFind/default/
     +reco/00001/00000/00001 0.00e+00/0.00e+00
     -ana /00000/00000/00000 0.00e+00/0.00e+00
 |-TrkRBase sequence (0/1/1 events 0u/0s seconds)
  +TrigDoT0/default/
    +reco/00000/00001/00001 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkWireFilter/default/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkMakeClust/default/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkMakeCross/default/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkSegBuilder/BC/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkSegBuilder/C123/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkSegBuilder/C456/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  TrkCandBuilder/default/
    +reco/00000/00000/00000 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  SPSegAssn/default/
    +reco/00001/00000/00001 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  SPFit/default/
    +reco/00001/00000/00001 1.90e-01/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
  SPVertexFind/default/
    +reco/00001/00000/00001 0.00e+00/0.00e+00
    -ana /00000/00000/00000 0.00e+00/0.00e+00
```

# JobModules

JobModules actually do the work of analysis or reconstruction.  
They optionally implement the following methods:

```
// Event processing
virtual JobCResult Reco(EDMEventHandle& evt);
virtual JobCResult Ana(const EDMEventHandle& evt);

// Start/end of files
virtual void NewFile(const char* filename);
virtual void EndFile(const char* filename);

// Start/end data runs
virtual void NewRun(int run, int subrun);
virtual void EndRun(int run, int subrun);

// Start/end sub runs
virtual void NewSubRun(int run, int subrun);
virtual void EndSubRun(int run, int subrun);

// Status and reporting
virtual void Reset();
virtual void Report();
```

The Reco and Ana methods make filter decisions on an event. The job can be configured to either ignore, apply, or reverse these decisions.  
All communication between modules is done via the event. Modules are independent of each other.

## *Configuration*

All the parameters modules use to make their decisions are configured at run via a package called “Config”. Configurations are lists of named parameters grouped together under a collective name and version tag.

Default configurations are provided via a set of XML files.

In interactive mode (ie. in the event display) configurations can be edited (using a text-based or gui tool)

JobModules inherit from a class called “CfgObserver” and watch specific versions of configurations of a set name. Notification of changes to the configuration are sent to the JobModule via the “Update” method.

MIPP hard codes the relationship between a module's name and the configuration it uses. (SPFitTrack module uses configuration SPFitTrack)

# Sample configuration document

```
<configdoc>

<config name="RICHRadFit" version="default">

    <param name="Debug"><int> 1 </int>
        Debug flag: 0 - no debugging, 1 - some debugging, >1 verbose
    </param>

    <param name="MaxItr"><int> 3 </int>
        Number of fitting iterations to perform. At each iteration PMT's
        are selected based on the fit found in the previous iteration.
    </param>

    <param name="ItrChiCut"><float> 25.0 16.0 9.0 </float>
        At each iteration only include PMT's in the fit which contribute
        less than this amount to chi^2
    </param>

    <param name="Sigma0"><float> 1.50 </float>
        Assumed Gaussian width of RICH ring radii
    </param>

    <param name="MinPMTperRing"> <int> 5 </int>
        Minimum number of PMT's per ring.
    </param>

</config>

</configdoc>
```

## *Dependencies*

[1] From MIPP:

JobControl, Config, MippXML, EventDataModel

[2] ROOT (minimal, actually)

[3] SRT (needs to build search path for XML files)

[3] From IBM:

xerces-c

## *MIPP JobControl and Related: Do we want to use these?*

I've started working with the SoCal code and I'm not convinced a job-control-like interface is needed right now. Is it better to delay these "framework" decisions until we have more of a plan in front of us and focus on putting reconstruction code together in the SoCal system? Or do people want to try this out now?

If we do decide to try this out, I think I will need to tweak the build system to have a predictable environment for finding configuration files...

Code posted here for the curious:

<http://enrico1.physics.indiana.edu/mipp/doxygen/offline/html/>