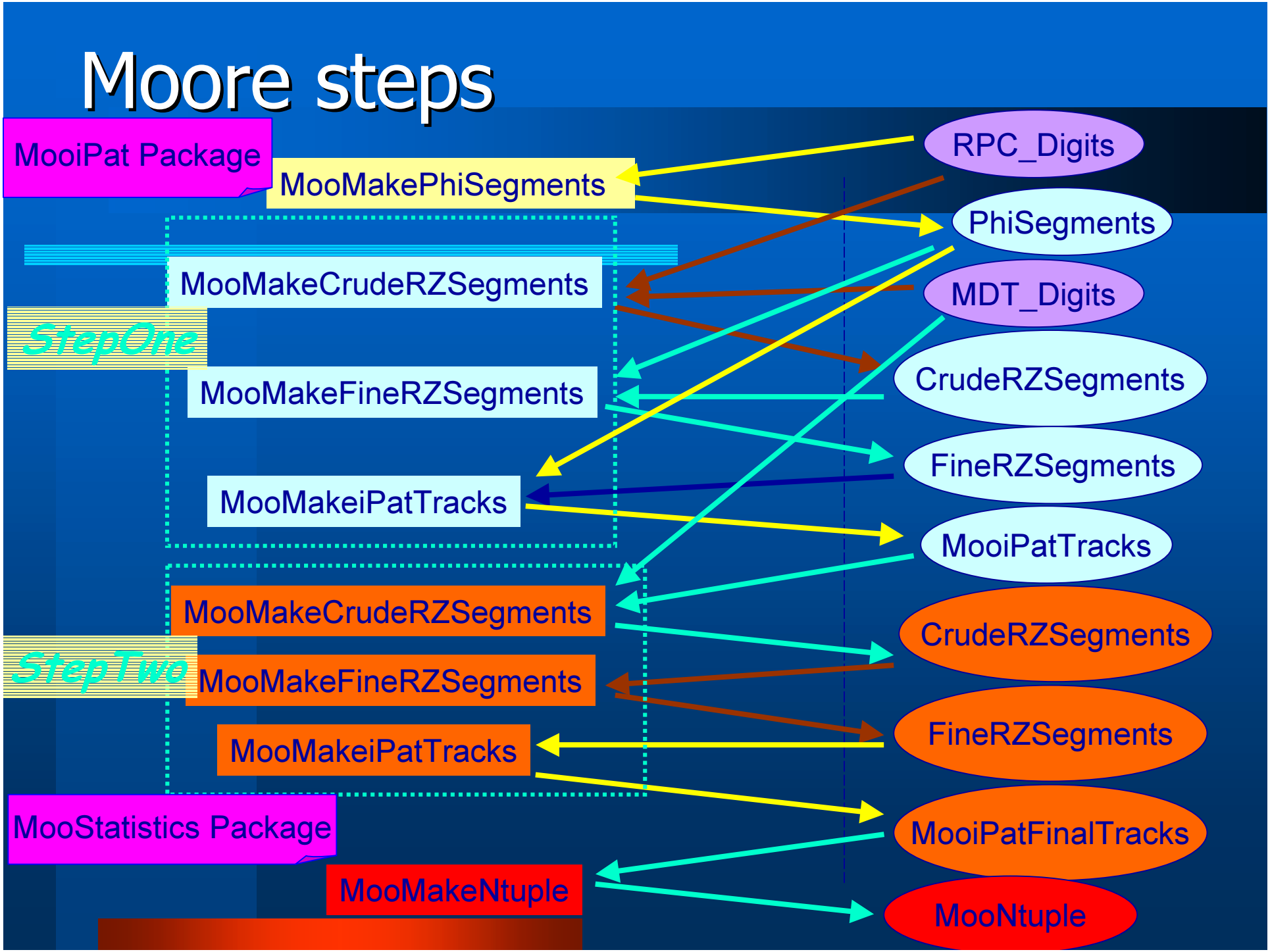
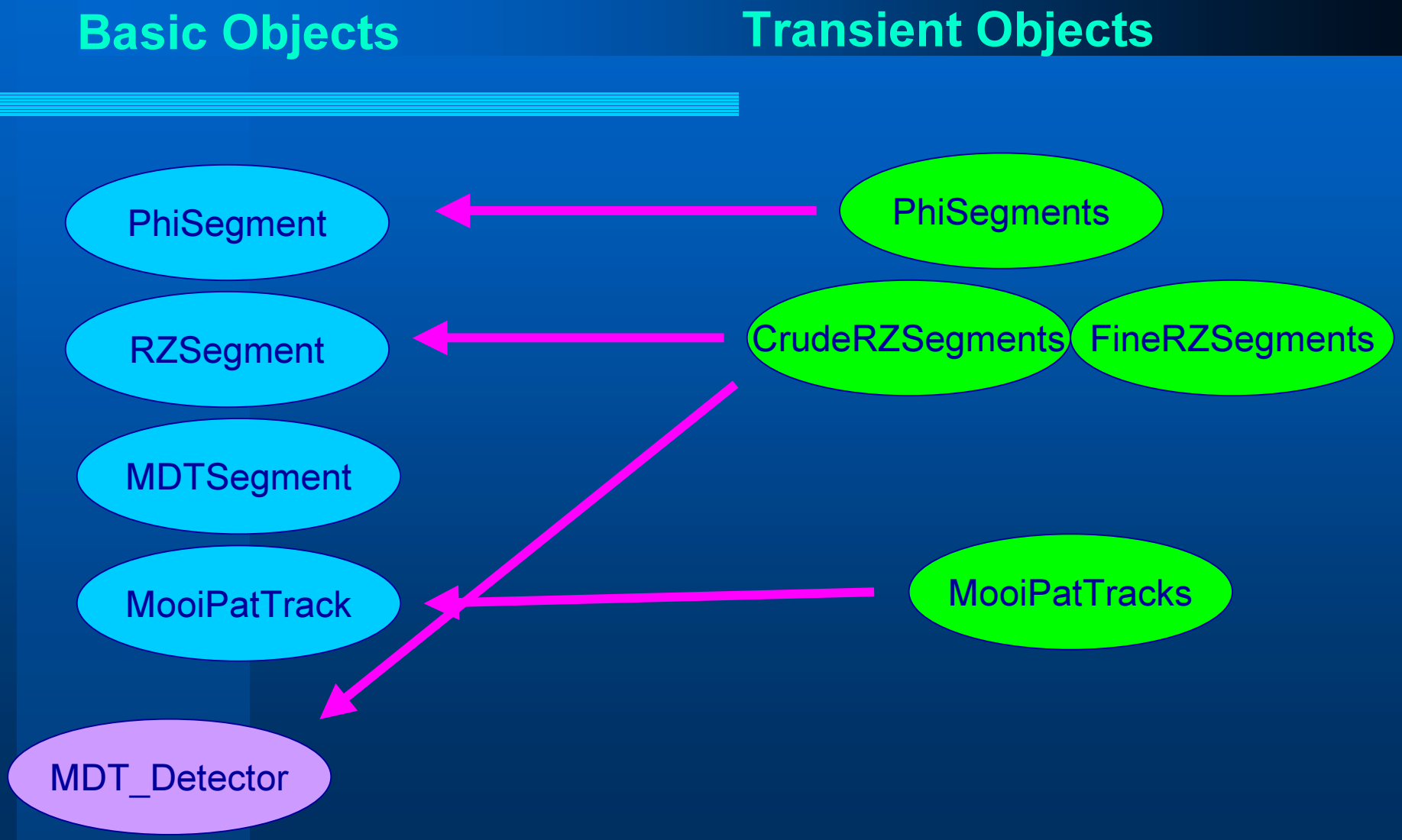
Status of Moore Makers Scheme

Michela Biglietti
Università di Napoli Federico II
INFN - Napoli

Moore steps


MooiPat Package
MooMakePhiSegments
MooMakeCrudeRZSegments
StepOne
MooMakeFineRZSegments
MooMakeiPatTracks
MooMakeCrudeRZSegments
StepTwo
MooMakeFineRZSegments
MooMakeiPatTracks
MooStatistics Package
MooMakeNtuple
RPC_Digits
PhiSegments
MDT_Digits
CrudeRZSegments
FineRZSegments
MooiPatTracks
CrudeRZSegments
FineRZSegments
MooiPatFinalTracks
MooNtuple

MooEvents

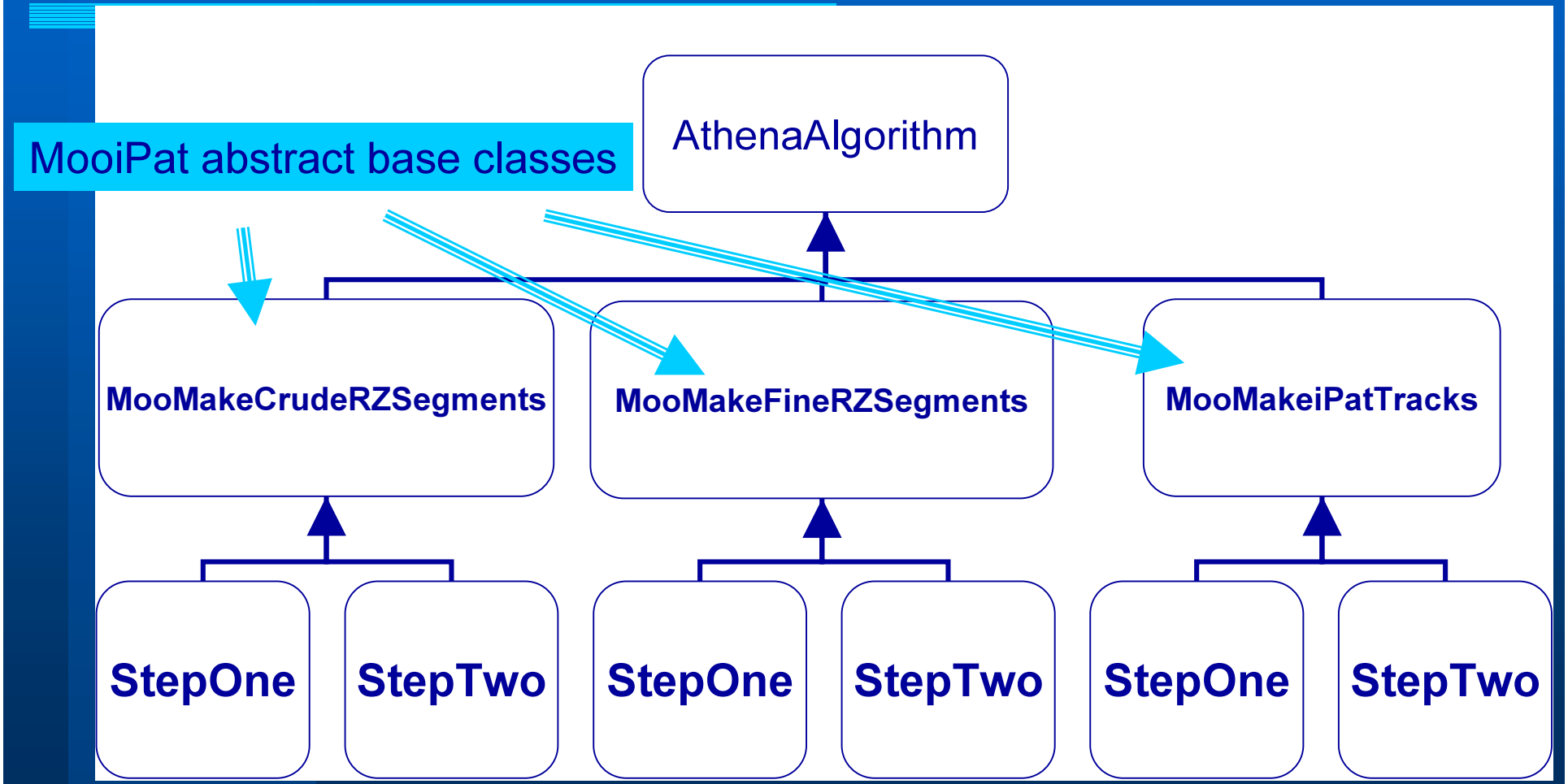

Basic Objects
Transient Objects
PhiSegment
RZSegment
MDTSegment
MooiPatTrack
MDT_Detector
PhiSegments
CrudeRZSegments
FineRZSegments
MooiPatTracks

MooiPat Algos

- StepOne – StepTwo modules are very similar
- The same task/interface
- Some differences in
  - I/O transient objects
  - value of external parameters
  - few implementations

Use of inheritance

MooiPat Algos



MooiPat Classes

- Base classes have
  - public:
    - initialize() execute() finalize()
  - private:
    - virtual my_execute() = 0
    - virtual retrieve()/record() = 0
  - protected:
    - virtual helper methods (implemented when are shared between the concrete classes)
    - members shared between concrete classes (StoreGateSvc* m_ SGevent,

      int m_print_level …)

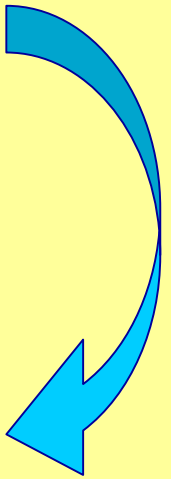
```
execute ()
{
retrieve();          //implemented
my_execute();     //in the deriv
record();            // class
}
```

MooiPat Classes

- In general concrete classes have
- (at least)
  - my_execute()
  - retrieve()/record()
- (eventually)
  - use of base classes methods/members
  - overloaded methods
  - own methods/members

Example - MooMakeFineRZSegments.h base class

```
public:
StatusCode initialize();
StatusCode execute();      // just a call to retrieve/make_fine_segments_execute()/record()
StatusCode finalize();
private:
virtual StatusCode retrieve() = 0;                          // implemented in the derived class
virtual StatusCode make_fine_segments_execute() = 0;        // possibility to use different I/O objects
virtua StatusCode record() = 0;                             // access to the "helper" class
int m_print_level                    // shared parameters but specialized for each
…                                    // derived class
protected:
StoreGateSvc* SGEvent;
…
virtual vector<RZSegment* > make_fine_segments(RZSegment,double);
virtual pair<double,double> drift_distance_and_error(double, MDT_Digit*);
...
```



Example – MooMakeCrudeRZSegments.h base class

```
class MooMakeCrudeRZSegments: public Algorithm
{
public:

    MooMakeCrudeRZSegments(const std::string& name, ISvcLocator* pSvcLocator);
    virtual ~MooMakeCrudeRZSegments();

    StatusCode  initialize();
    StatusCode  execute();
    StatusCode  finalize();
private:
    //
    // class members
    //
    int m_print_level;
    StoreGateSvc* m_SGevent;
    const DataHandle<ZebraTDREvent>*   m_tdr_event;
    RZSegmentMap* m_crude_RZSegments;
    //
    // virtual class method to be implemented in the concrete class
    //
    virtual StatusCode make_crude_segment_execute() = 0;
    //
    // helper virtual class methods to be implemented in the concrete class
    //
    virtual StatusCode retrieve() = 0;
    virtual std::vector<RZSegment*>  make_crude_segments(MDT_Detector*)    ;
    virtual StatusCode record() = 0;
};
```

Example – MooMakeCrudeRZSegments.cxx base class

```
MooMakeCrudeRZSegments::
MooMakeCrudeRZSegments(const std::string& name, ISvcLocator* pSvcLocator) :
  Algorithm(name, pSvcLocator),
  m_print_level(0)
{
  declareProperty("print_level",                m_print_level);
}

StatusCode
MooMakeCrudeRZSegments::execute()
{
  StatusCode status = StatusCode::SUCCESS;

  MsgStream exe_log(messageService(), name());
  if (m_print_level > 0) exe_log << MSG::INFO << "Executing" << endreq;

  status = make_crude_segment_execute();
  if ( status.isFailure() )
    {
      exe_log << MSG::FATAL
              << "Could not execute make-crude-segments " << endreq;
      return StatusCode::FAILURE;
    }
  return status;
}
```

Example – MooMakeCrudeRZSegmentsStepOne.h concrete class

```
class MooMakeCrudeRZSegmentsStepOne: public MooMakeCrudeRZSegments
{
 public:

    MooMakeCrudeRZSegmentsStepOne
      (const std::string& name, ISvcLocator* pSvcLocator);

    ~MooMakeCrudeRZSegmentsStepOne();

private:
    //
    // class members
    //
    int m_the_thr;
    //*****************************************************************
    // concrete class method from base class executed by Athena
    //
    StatusCode make_crude_segment_execute();
    //*****************************************************************
    //
    // helper concrete class method from base class
    //
    StatusCode retrieve();
    std::vector<RZSegment*>  make_crude_segments(MDT_Detector*);
    StatusCode record();
    //
    // class methods
    //
    CollectDetectorVisitor<EventDetectorElement> associated_trigger_chambers(Id\
entifier);
};
```

Example – MooMakeCrudeRZSegmentsStepOne.cxx

```
static const AlgFactory<MooMakeCrudeRZSegmentsStepOne>    Factory;
const IAlgFactory& MooMakeCrudeRZSegmentsStepOneFactory = Factory;

MooMakeCrudeRZSegmentsStepOne::MooMakeCrudeRZSegmentsStepOne(const std::string&\
 name, ISvcLocator* pSvcLocator) :
  MooMakeCrudeRZSegments(name, pSvcLocator)
{
  declareProperty("theta_thr_bin",                m_the_thr);
}


StatusCode
MooMakeCrudeRZSegmentsStepOne::make_crude_segment_execute()
{
  StatusCode status = StatusCode::SUCCESS;


    status = retrieve();
    if(status.isFailure()) return status;

    // Create segments on those MDTs

    RZSegmentMap * m_crude_RZSegments = new RZSegmentMap;
    for (std::vector<MDT_Detector*>::const_iterator  det = mdt_col.detectors_be\
gin();
         det != mdt_col.detectors_end();
         ++det)
    {
      std::vector<RZSegment*> segments = make_crude_segments(*det);
      if( segments.size() > 0 )
        m_crude_RZSegments->insert(RZSegmentMap::value_type(*det, segments));

    }

    status = record();
    if(status.isFailure()) return status;

     if (m_print_level > 1)
       {
         // print-outs
       }
```