The EventView Analysis Framework

Akira Shibata, Queen Mary University of London

3.May.2007 @ DA Tutorial in Valencia

Outline

1. Event Data Model and Analysis Model
    - Where we are.
    - Current issues.
2. What is EventView?
    - The EventView EDM class.
    - EventView framework.
    - PhysicsView packages.
3. Where is EventView?
    - What is the scope of EventView in PAT.
    - The scope of EventView in DA.
    - Current use cases.
4. Future developments
    - Future developments in EDM
    - Future developments in PANDA

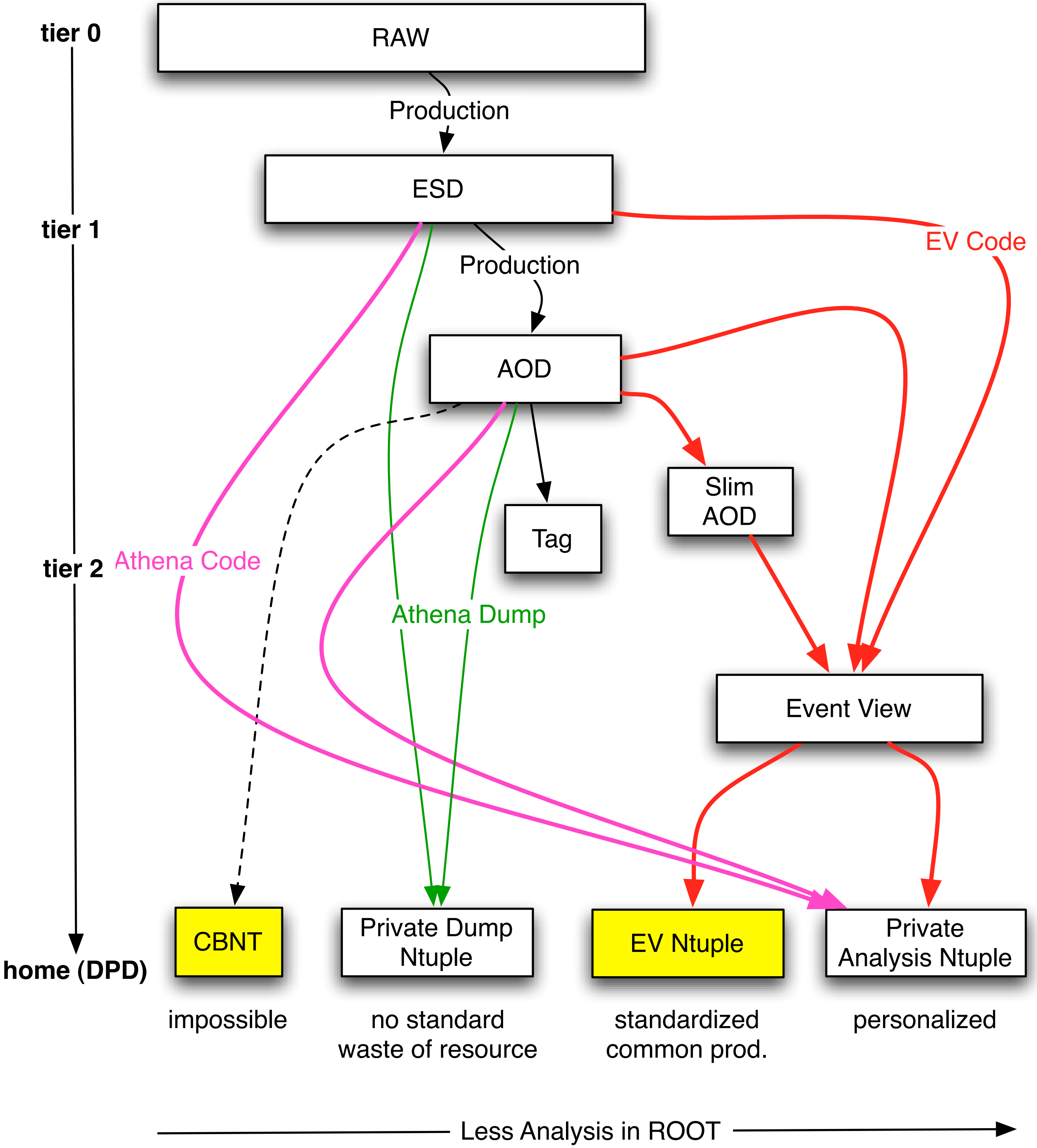
Our Goal

- Is to do physics analysis to the level that we can publish our result.
- Great deal of effort went in. Lots of trial and error and it is still evolving very fast. (Computing Service Commissioning, CSC is one such place.)
- There are thousands of people trying to analyse huge amount of data and we need the right environment to do this.
- Physics analysis tools and distributed analysis are trying to solve the problems.

Acronyms

- Athena: The Atlas Software (algorithms in C++, steering with Python). Used for Monte Carlo, Simulation, Recosntruction, and Analysis. Also used by the High Level Trigger.
- RAW: Events as output by the Event Filter for reconstruction. The computing model assumes it is only at Tier0 site (1.6MB/event at rate of 200Hz).
- CBNT: "ComBined NTuple" summarizes output of reconstruction. Read into Root, can't be read in Athena. Not part of the computing model.
- ESD: Event Summary Data, is the output of Reconstruction can be read into Athena. Only to be stored at Tier1 sites. (500kB/event).
- AOD: Analysis Object Data, distributed to every one and meant for analysis, can be readin Athena(100kB/event).
- DPD: Derived Physics Data, included in computing model, something analogous to an ntuple.
- Tags: Event-level metadata used to quickly select "interesting events" (1kB/event).

This talk focuses mainly on DPD production from AOD.



tier 0

tier 1

tier 2

home (DPD)

RAW

Production

ESD

EV Code

Production

AOD

Slim AOD

Tag

Athena Code

Athena Dump

Event View

CBNT

Private Dump Ntuple

EV Ntuple

Private Analysis Ntuple

impossible

no standard
waste of resource

standardized
common prod.

personalized

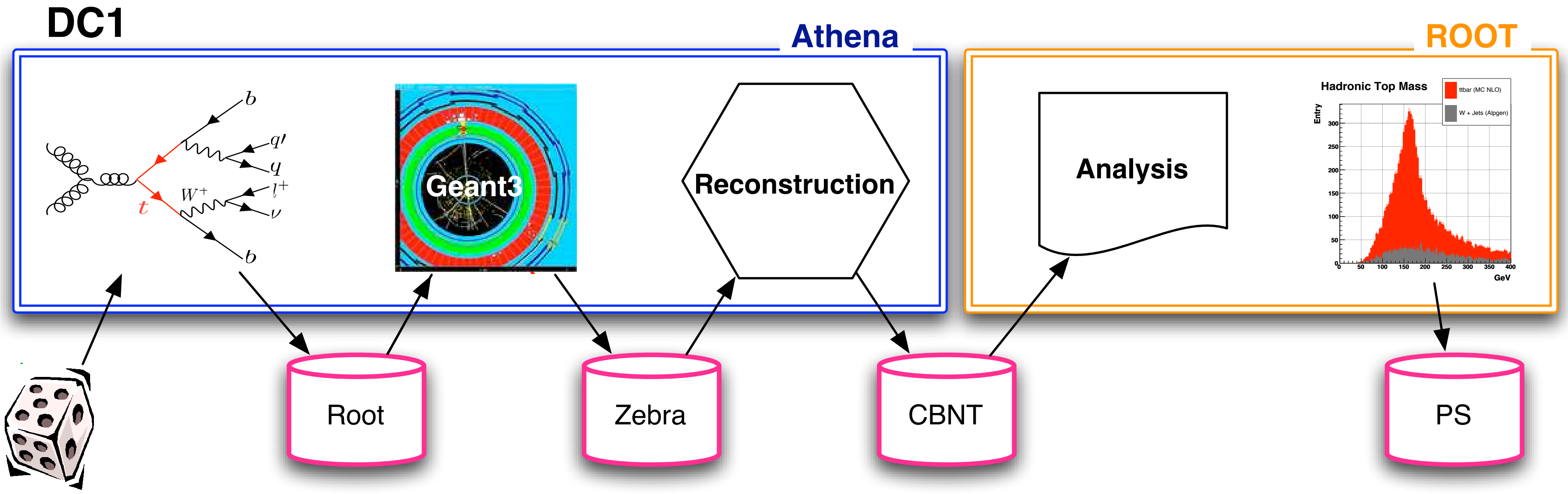
Less Analysis in ROOT

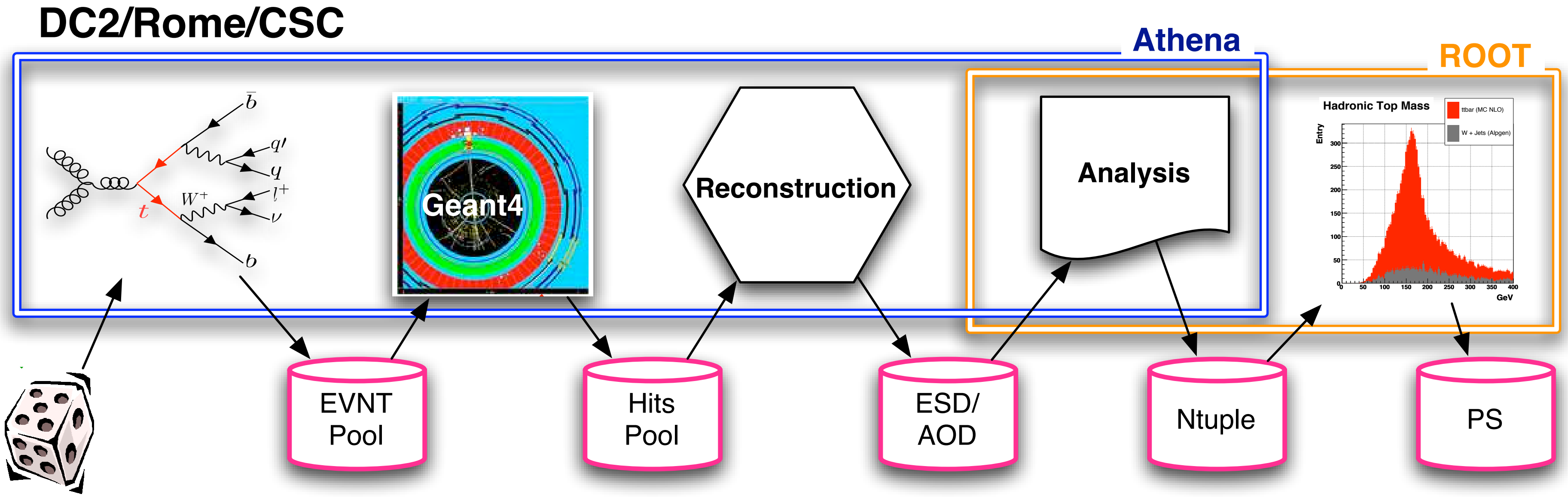
Part I

Event Data Model and Analysis Model

Event Data Model

History of our data format and
history of our analysis.





- CBNT
  - Flat ntuple with all information from reconstruction.
  - No common look and feel.
  - No official mechanism to share activity after reconstruction.
- AOD/ESD is nice!
  - It has common interface to all objects.
  - With it, we can do analysis using Athena.
  - We need to publish and share analysis code for cross check and reproduction of results.
- But why Athena?
  - All of our reconstruction alg is in Athena.
  - We'll need to be able to re-run clustering, vertexing, track extrapolation, calibration etc on AOD.
- How about Root
  - We always need to use Root for our final analysis.
  - Athena is good for event-level processing but not sample level.
- Two stage analysis
  - We cannot do everything in one place. Some in Athena, rest in Root.
  - Support flexibility in analysis. But what to do where?

Where to do Analysis?

Root lovers / Athena purists
physicists / computing specialists

1. Pre-physics analysis: reconstruction.

2. Baseline analysis: preselection & overlap removal, further processing such as jet re-calibration etc.

3. Event level analysis: object reconstruction, event variable calculation etc.

4. Sample level analysis: plot histogram, fitting templates, toy MC etc.

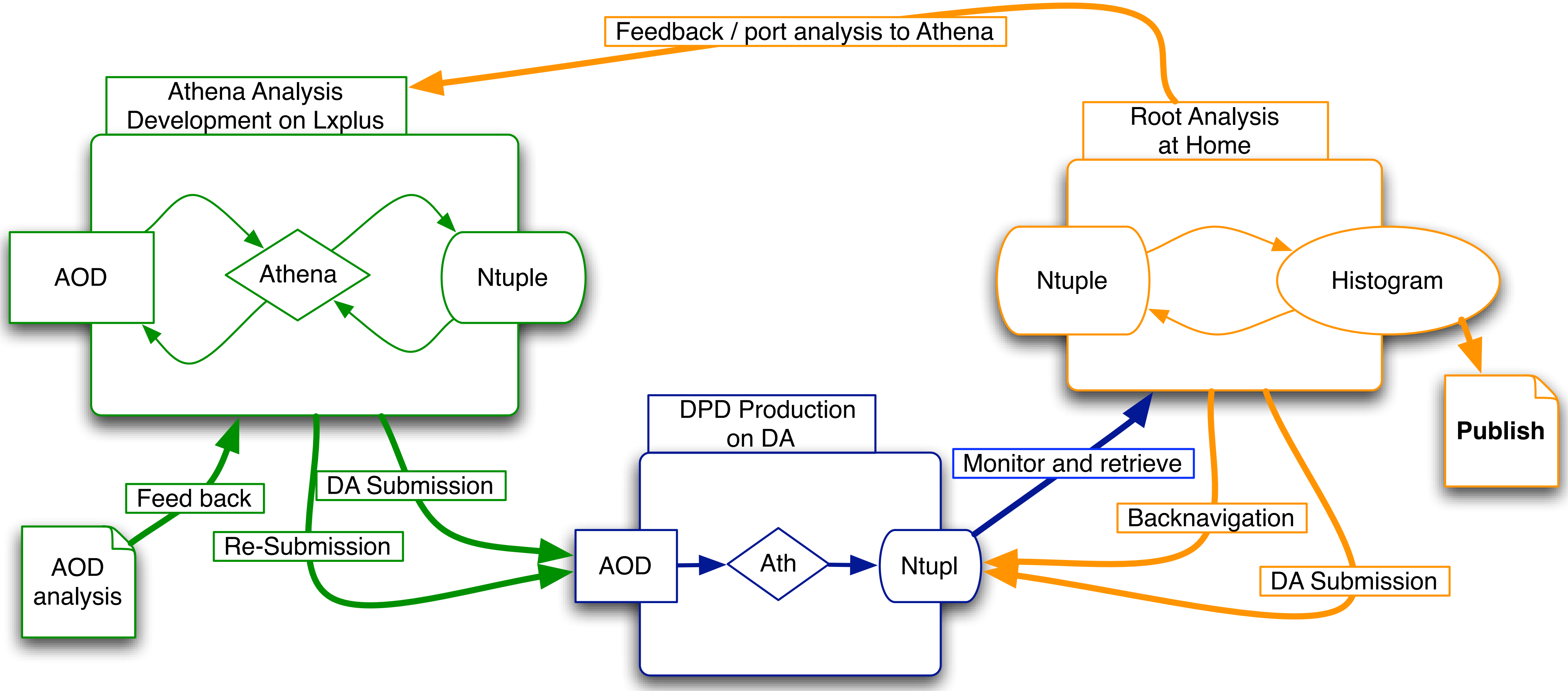
Athena Central Production

Athena Analysis on Distributed Ana

Root Analysis Local or DA

Two-Stage Analysis

A global view



Scope of this workshop:
Athena analysis and distributed analysis

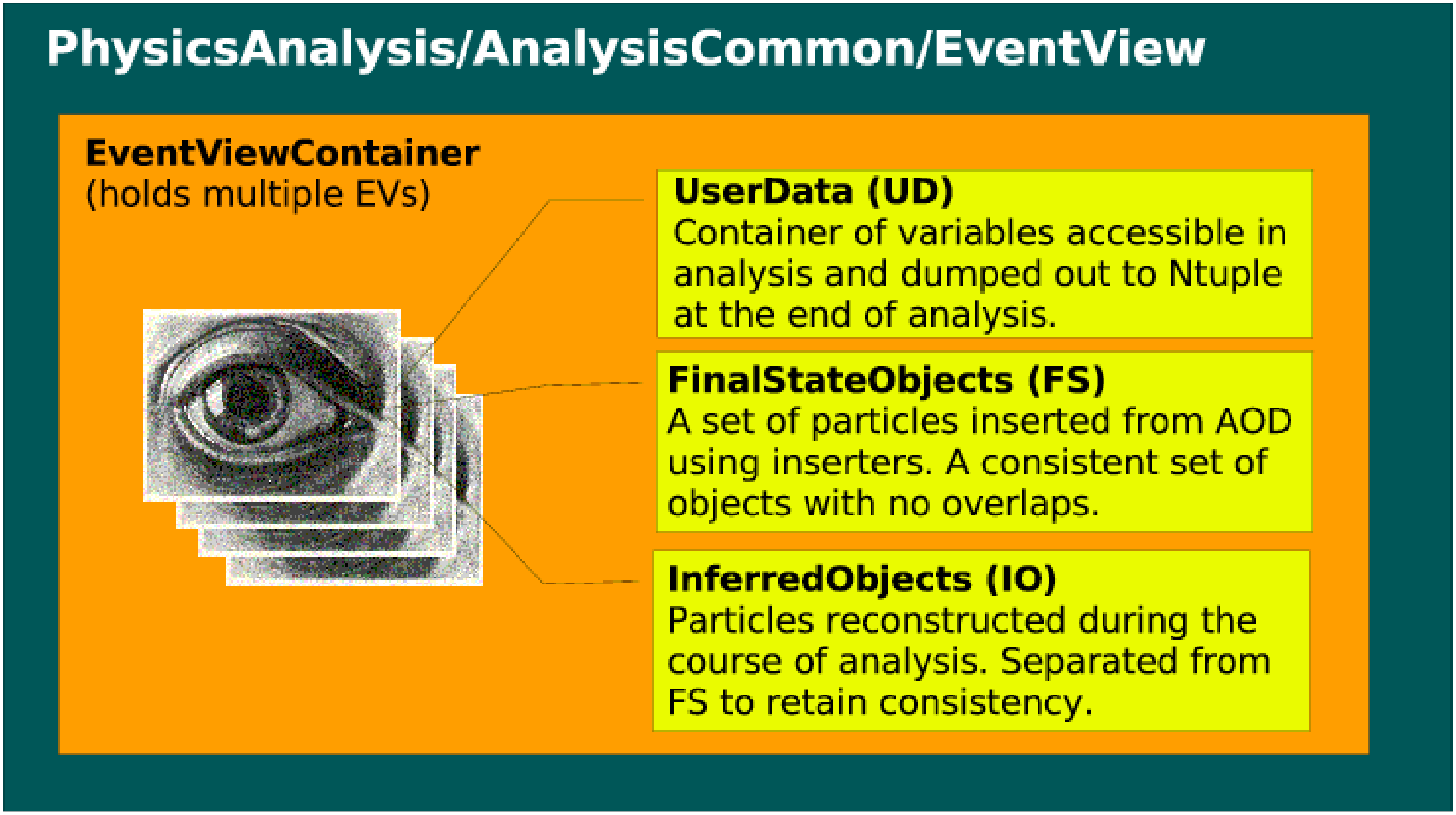
- Athena analysis should offer:
  - The official place for common analysis.
  - Should be easy enough for physicists to use!
  - Simple mechanism for sharing code - modular analysis.
  - Should be more than an ntuple dumper, need to do "analysis". (e.g. tune PID, re-calibrate)
- AOD object definition is loose (needs to be):
  - It has overlapping objects (e.g. electrons and jets).
  - It has loose object selection (e.g. no isolation cut).
  - It has multiple algorithms (e.g. Staco/Muid muons).
- First step in analysis: Define your "view" of events:
  - Do this by constructing EventView.
  - Further baseline and event-level analysis using EventView tools and modules.
- Produce Derived Physics Data, DPD (ntuple) for Root analysis:
  - Private ntuple.
  - Group standard DPD.

Part 2

What is EventView?

What is EventView?

EventView is a container which holds objects (or links to them) needed in physics analysis.



PhysicsAnalysis/AnalysisCommon/EventView

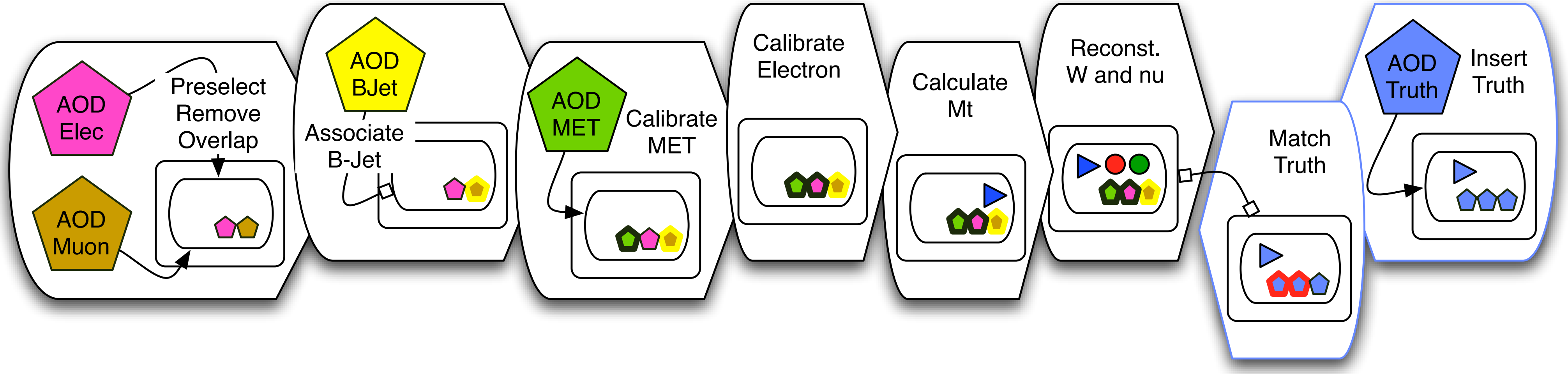
EventViewContainer
(holds multiple EVs)

UserData (UD)
Container of variables accessible in analysis and dumped out to Ntuple at the end of analysis.

FinalStateObjects (FS)
A set of particles inserted from AOD using inserters. A consistent set of objects with no overlaps.

InferredObjects (IO)
Particles reconstructed during the course of analysis. Separated from FS to retain consistency.

EventView Modular Analysis Framework



Once we have defined our container, now analysis can be divided into pieces.

Each step of analysis divided into separate tools.

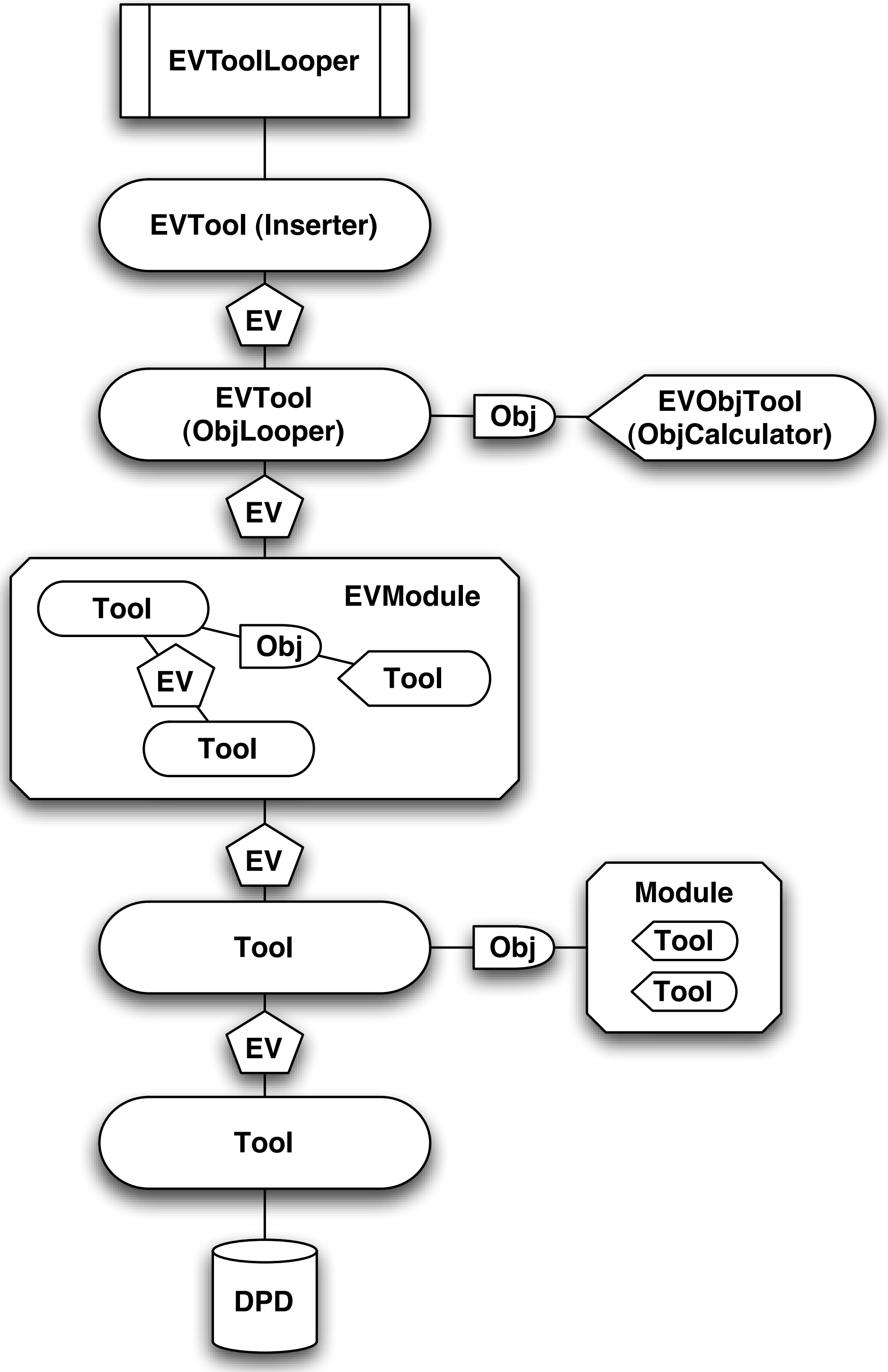
Each tool is made independent and general as much as possible.

Analysis information is propagated through EventView.

Each tool receives EventView and execute something on EventView.

Gradually build up EventView with more information.

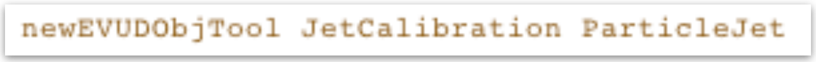
Components of EV Framework



- EVToolLooper (or just looper)
  - Schedule all EV components.
- EVTools (or just Tools)
  - Smallest component in EV.
  - Written in C++.
  - Mainly EV tool and EV obj tool.
  - Other types of tools too.
- EVModules (or just Modules)
  - Set of tools and configurations.
  - Written in Python.
  - Can be EV level or obj level.

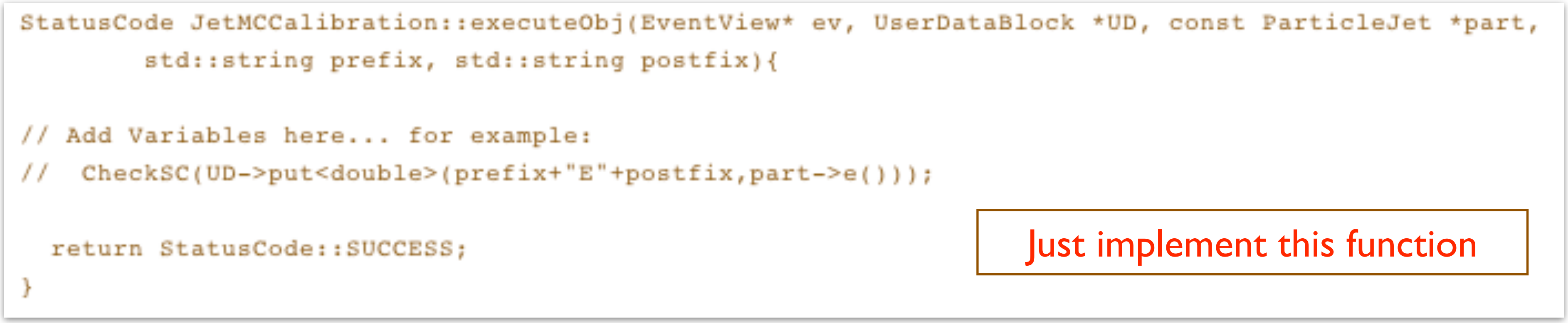
Extending the Framework

- It is crucial that the framework is easily extendable.
  - EventView framework is a collection of tools, extendibility is its natural feature.
- Minimize work needed to add components.
  - Subdivide usefulness of the tools into different types.
  - Interface and functionality inherited through class hierarchy.
- Skeleton code generated for typical type of tools.



```
newEVUDObjTool JetCalibration ParticleJet
```

Generate skeleton

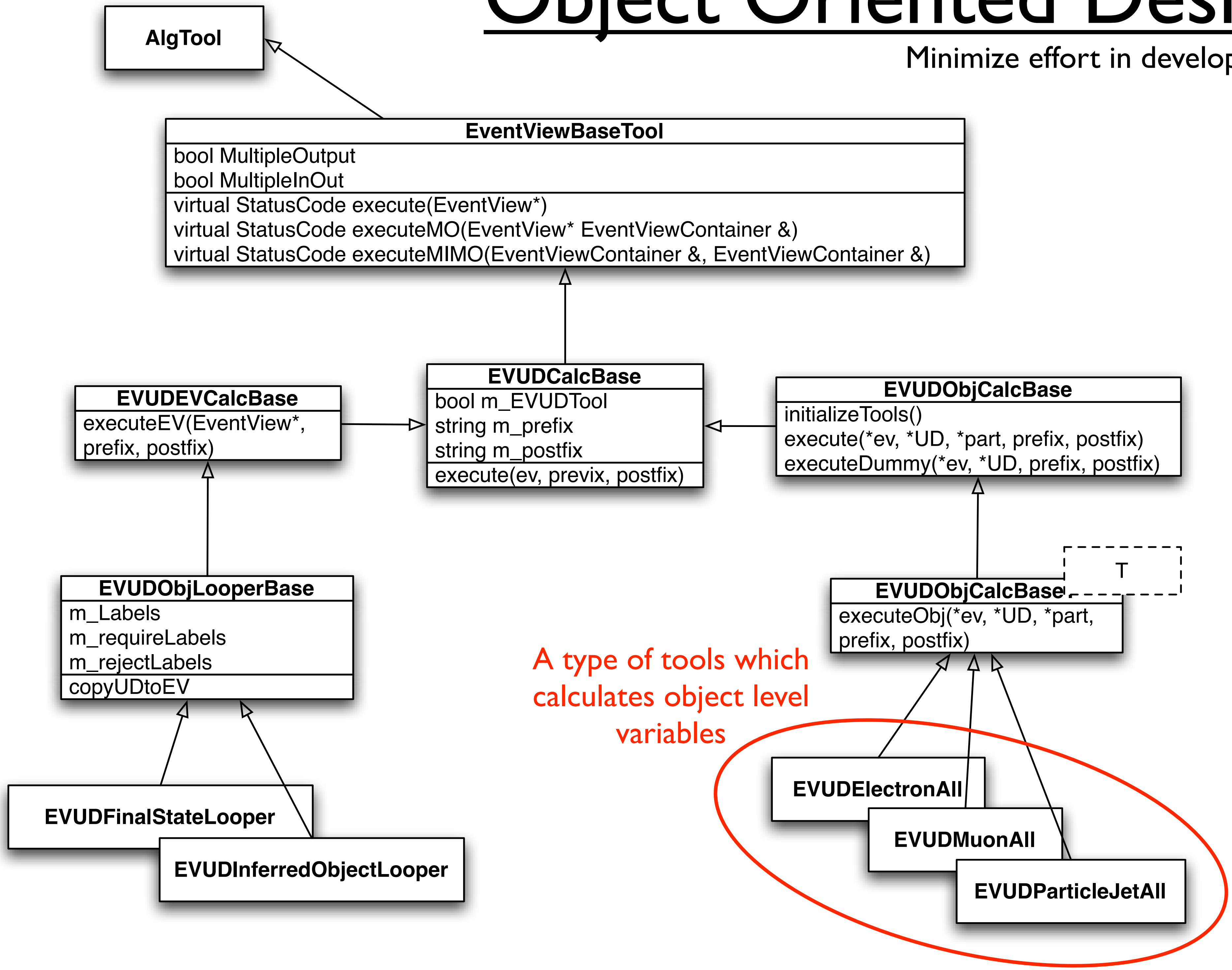


```
StatusCode JetMCCalibration::executeObj(EventView* ev, UserDataBlock *UD, const ParticleJet *part,
        std::string prefix, std::string postfix){

// Add Variables here... for example:
//  CheckSC(UD->put<double>(prefix+"E"+postfix,part->e()));

  return StatusCode::SUCCESS;
}
```

Just implement this function

Object Oriented Design

Minimize effort in development



AlgTool

EventViewBaseTool

bool MultipleOutput
bool MultipleInOut

virtual StatusCode execute(EventView*)
virtual StatusCode executeMO(EventView* EventViewContainer &)
virtual StatusCode executeMIMO(EventViewContainer &, EventViewContainer &)

EVUDEVCalcBase

executeEV(EventView*, prefix, postfix)

EVUDCalcBase

bool m_EVUDTool
string m_prefix
string m_postfix

execute(ev, previx, postfix)

EVUDObjCalcBase

initializeTools()
execute(*ev, *UD, *part, prefix, postfix)
executeDummy(*ev, *UD, prefix, postfix)

EVUDObjLooperBase

m_Labels
m_requireLabels
m_rejectLabels

copyUDtoEV

EVUDObjCalcBase T

executeObj(*ev, *UD, *part, prefix, postfix)

EVUDFinalStateLooper

EVUDInferredObjectLooper

EVUDElectronAll

EVUDMuonAll

EVUDParticleJetAll

A type of tools which calculates object level variables

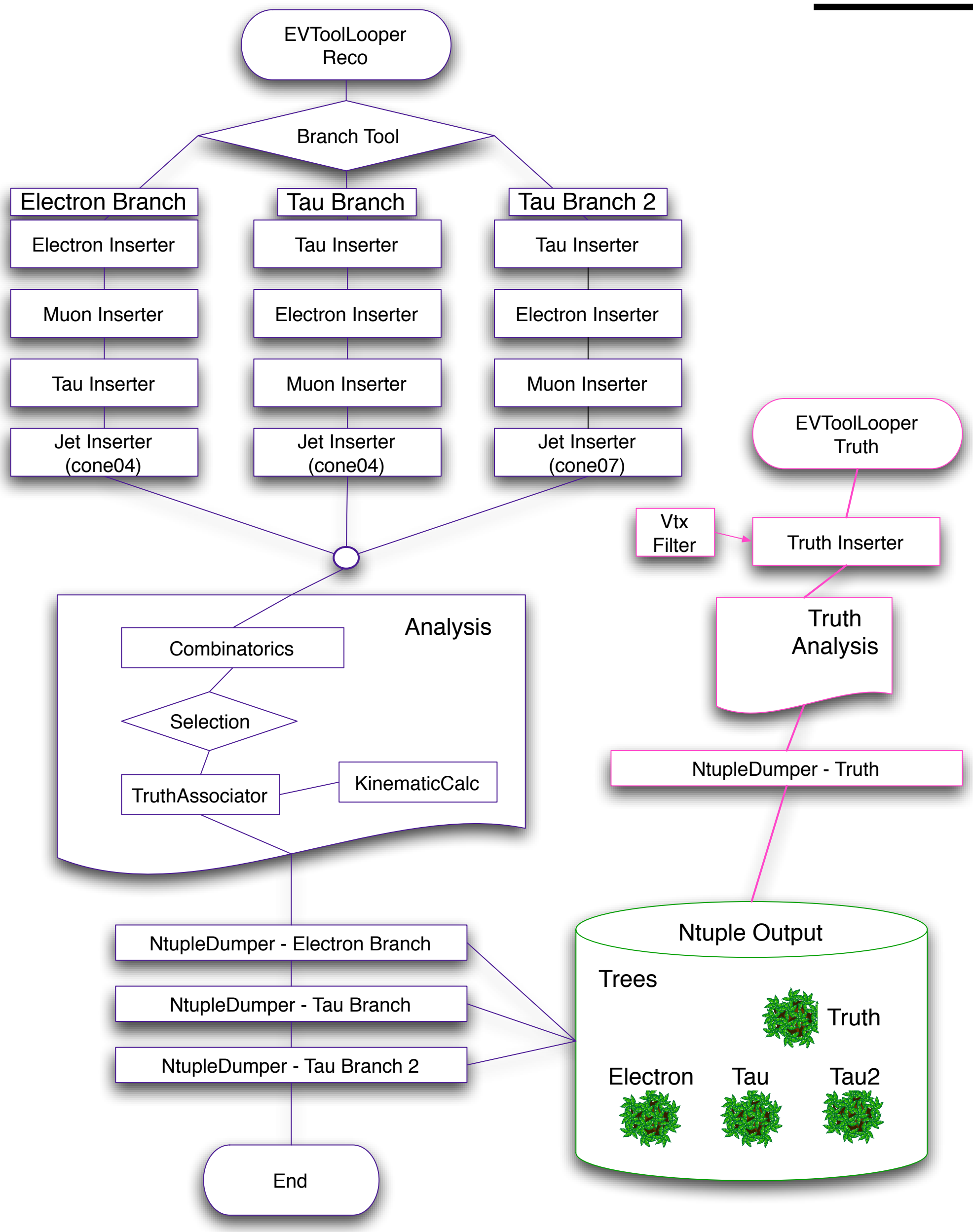
Standard EventView Tool/Module Library

- EventViewBuilder (Container package)
    - EventViewBuilderUtils: Base classes of all tools
    - EventViewBuilderAlgs: Tool loopers
    - EventViewConfiguration: High level configuration interface
    - EventViewInserters: Tools for selection and overlap removal
    - EventViewDumpers: Output to screen, ntuple, and atlantis
    - EventViewUserData: Calculators and associators
    - EventViewTrigger: Tools for reading trigger info
    - EventViewPerformance: Performance study modules
    - EventViewCombiners: Combinatorics tools
    - EventViewSelectors: Event selection tools

List growing thanks to contributions from people but we need more people to work on them.

Multiple View (Branch)

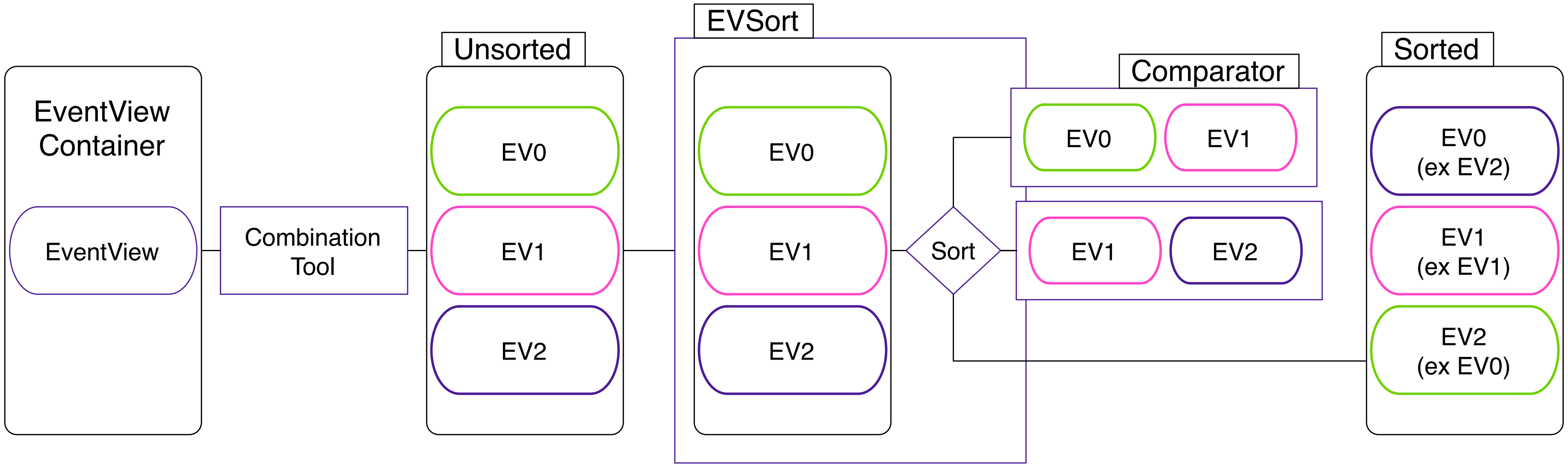
An example EV analysis



- One can define multiple views of an event:
  - e.g. use different jet in each view.
  - e.g. use different object selection in each view.
- Same analysis executed on each EventView.
- Each view can talk to each other:
  - compare views.
  - order views.
  - match views.
- Each view saved into separate Root trees in ntuple.

Multiple View (Combinatorics)

Another example analysis

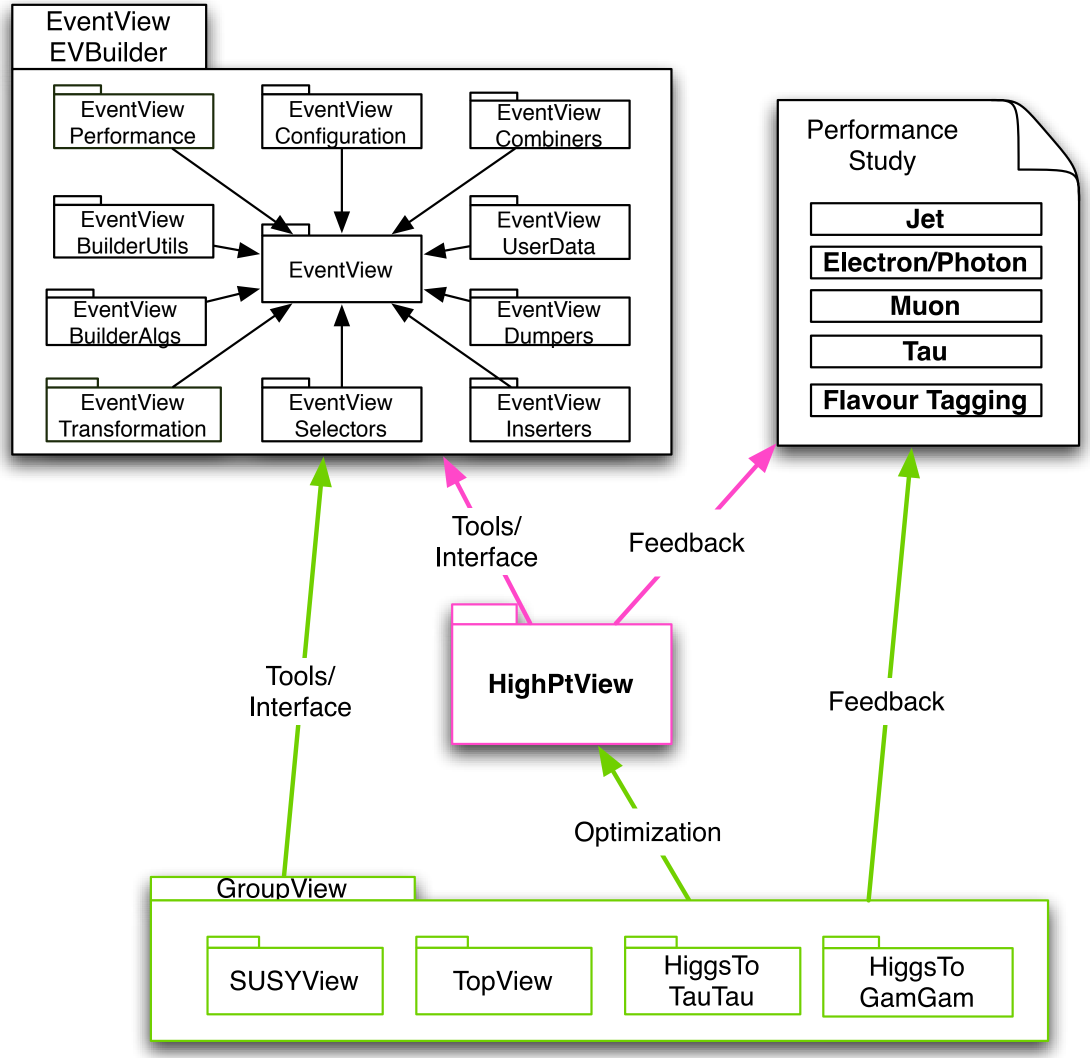


- Multiple EV produced from combinatorics:
  - e.g. take all combinations of dijets.
  - e.g. combine one W and a b-jet to obtain top quark.
- Manipulation of multiple EV through Multi Input (MI) tools:
  - e.g. sort EV according to the pt of reconstructed top.
  - e.g. remove EVs not satisfying condition.
- Each view saved to separate tree or all views in one tree.

DPD Production

- EV analysis can easily produce custom ntuple with desired amount of information.
- It is not just copying information but a lot more needs to be done:
  - Selection of objects
  - Slimming contents
  - Adding derived variables
  - Calibration / correction
  - you name it.
- Various physics groups started making their own ntuple:
  - They are based on the same tools but the format is slightly different.
  - Need common package to provide interface for producing DPD. HighPtView.
  - Provide same look and feel but still support difference in contents.
  - Central production of default ntuples. Good starting point and reference point.

HighPtView



- HighPtView implements a set of configurations to manipulate tools/modules in EventViewBuilder.
- Provides common interface for DPD building for all PhysicsView packages.
- Default physics contents (selection cuts) come from performance study.
- Physics groups may have their own “view” packages (eg TopView) for optimization (override configuration) and extra functionality (eg tools for analysis).
- Group view packages will look similar inheriting interface from HighPtView.

Part 3

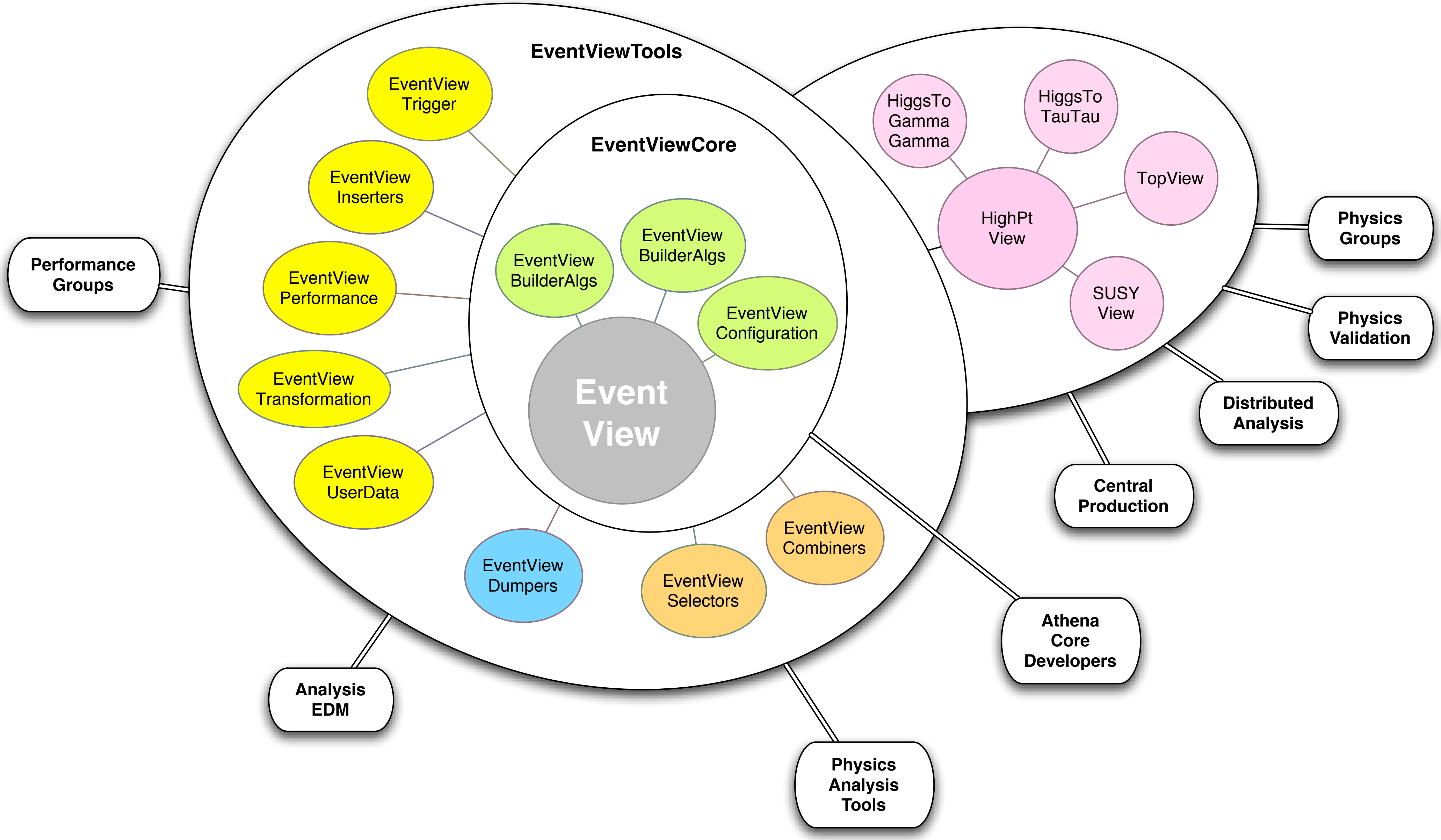
Where is EventView?

People Involved

This is a VERY incomplete list of people.
There are lots of other people contributing!

- Core Developers:
  - Kyle Cranmer (BNL)
  - Amir Farbin (CERN)
  - Akira Shibata (QMUL)
- Package Developers:
  - Attila Krasznahorkay (EventViewTrigger)
  - Jamie Boyd (EventViewPerformance)
  - Lorenzo Feligioni (Btagging)
  - Liza Mijovic (Development for release 13)
- GroupArea management:
  - Akira (Lxplus), Fabien Tarrade (BNL), Sandrine Laplace (Lyon)
- We have a lot of interaction with
  - **physics groups** and **performance groups**
  - **distributed analysis** and **Athena core** developers.

Where is EventView?



- Interaction with various groups is VERY important to maintain a functional framework:
  - Performance groups: Maintenance of relevant tools and input of physics contents (object selection, common performance study.)
  - EDM: Compatibility of the tools to developing EDM. Optimal DPD format for the future
  - Physics analysis tools: tool development and maintenance.
  - Core developers: support for new feature request and help on core development.
  - Central production: production of HighPtView ntuple.
  - Distributed analysis: support for EV analysis.
  - Physics validation: validation on standard DPD and RTT on Physics View packages.
  - Physics groups: development of group tools and modules in PhysicsVies packages.
- These things are just beginning to happen. More input and collaboration is required for:
  - Development and maintenance of PhysicsView packages.
  - Standardised performance study and selection tools.

DA and EventView

EventView has a long experience with PANDA distributed framework because

It is highly integrated with data management tool (dq2).

It supports GroupArea and configurables.

It runs athena jobs seamlessly and no book keeping is necessary.

1. Local test with sample fetched with DQ2.

```
athena TopView_Top_jobOptions.py
```

2. Set up grid/DQ2.

```
source GridSetupLxplus.sh
```

3. Send job (replace “athena” by “pathena”):

```
pathena TopView_Top_jobOptions.py \
--inDS csc11.005200.T1_McAtNlo_Jimmy.recon.AOD.v11004103 \
--outDS user.top_wg.TopView21.T1_McAtNlo_Jimmy.001
```

TopView Use Case

Athena Analysis Developmnt

1. Develop analysis based on EventView tools under TopView runtime package: Write C++ tools, write python modules and write job options.
2. Run tests on lxplus with test samples fetched with DQ2 end user tool.
3. Commit changes to CVS and tag.

(iterate 1 and 2 until I'm happy; use HN if I need help)

Athena Ana. Submission on DA

4. Send PANDA job specifying input/output dataset and a few other parameters.
5. Check status on the web.

(if job failed, report Savannah bug and iterate 4 and 5)

Post-Athena Analysis

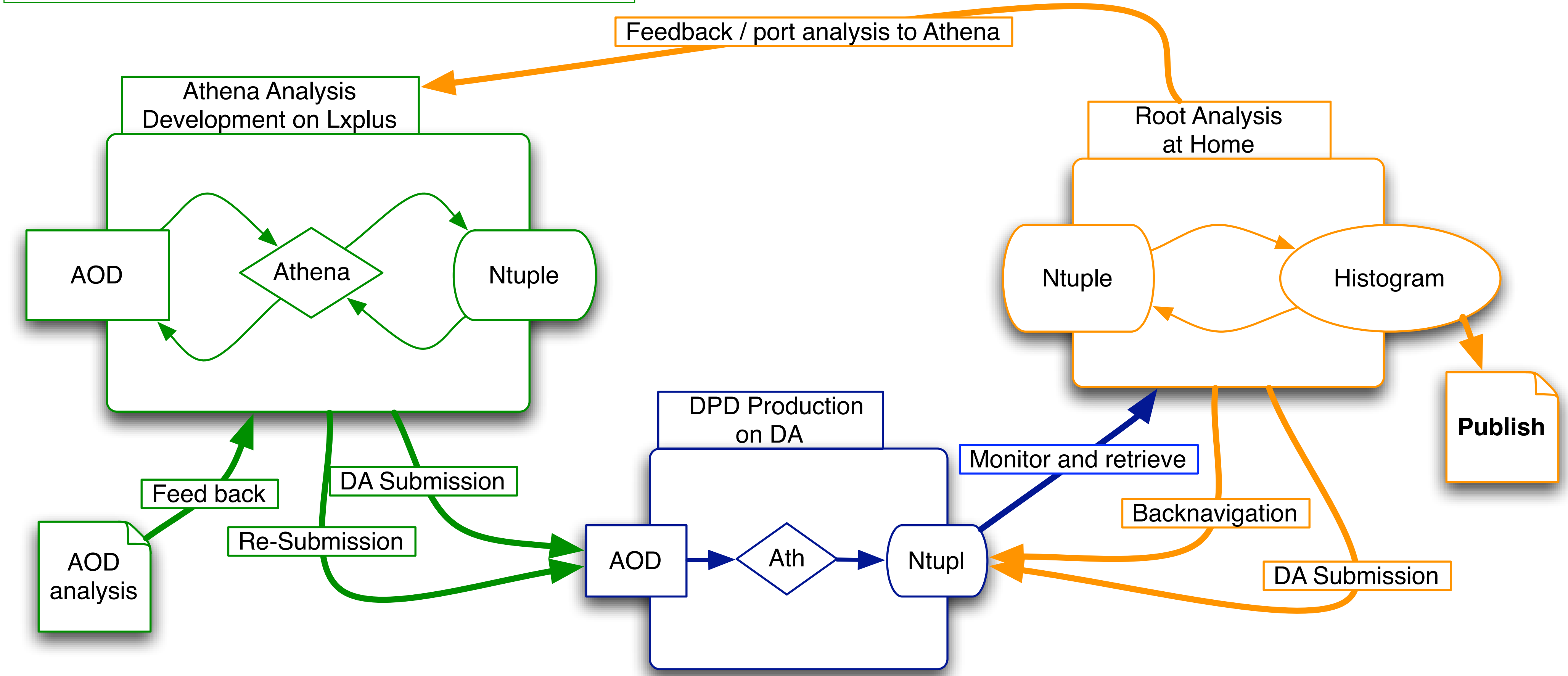
6. Fetch results (AANTuple) using DQ2 end user tool and analyze with ROOT locally.
7. Go back to the AOD/ESD using the AANT and do further analysis. (in future).

Two-stage Iterative Analysis Process

Again!

1st stage: Define common optimal preparation for a physics group. Includes selection, calibration, b-tagging. Use modular analysis framework and feedback/tools from performance study. Parameterise using common tools and python configuration.

2nd stage: Interactive analysis in ROOT with quick turnaround. Use TMVA, Roofit and other tools outside framework. Produce final plots to be published. Back-navigate through DA if necessary.



Group-wise central production on DA system.

Part 4

Future Developments

EDM in Release 13

- Faster AOD access:
  - Factor of ~10 in I/O speed expected thanks to transient persistent separation.
  - Broadens the use case of Athena analysis.
- AOD access from root or structured DPD:
  - Different technology now under test: pAOD, SAN, and new Scott's recipe.
  - Broadens the use case of Root analysis and lowers the wall between Athena and Root: higher portability of analysis code.
  - We will still need common framework just like before! EventView will write out structured format.
- ESD/AOD merger
  - ESD / AOD classes are now merged: identical interface for ESD and AOD analysis.
  - Can use ESD based tools on AOD objects: e.g. particle identification.

Future developments in PANDA

- Root analysis support on PANDA
  - Using PROOF or otherwise
- Support for more sites with improvements in operation
  - BNL has been fully operational for a long time.
  - Added lcg sites: ANALY_CERN, ANALY_LYON, ANALY_RAL, ANALY_FZK, ANALY_CNAF, ANALY_PIC, ANALY_SARA

Tutorial?

Let’s see how this all works.
But maybe after lunch?