PAT Tutorial

PAT Task Force contributors

Wolfgang Adam, Volker Adler, Jeremy Andrea, Christian Autermann, Colin Bernet, Freya Blekman, Christophe Delaere, Gregory Hammad, Benedikt Hegner, Liz Sexton-Kennedy, Slava Krutelyov, Steven Lowette, Luca Lista, Gheorghe Lungu, Petar Maksimovic, Sudhir Malik, Petra Van Mulders Benedikt Mura, Giovanni Petrucciani, Charles Plager, Sal Rappoccio, Tanja Rommerskirchen, Frederic Ronga, Roberto Tenchini, Eric Vaandering, Jean-Roch Vlimant, Clemens Zeidler, you??

Based on other similar tutorials

Outline

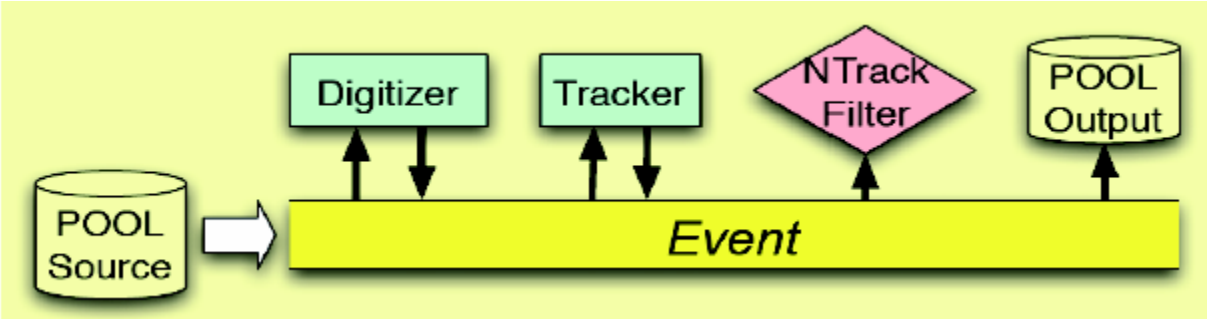
- The CMS analysis environment
- Overview of Physics Analysis Toolkit (PAT)
- The Starter Kit
- Summary

Outline

- The CMS analysis environment
  - the CMS Event Data Model
  - what the framework provides you with
  - the different software components
  - data access with CMSSW, FWLite and ROOT
  - analysis within the CMS Computing Model
  - the AOD event content
- Overview Physics Analysis Toolkit (PAT)
- The Starter Kit
- Summary

The CMS Event Data Model

- the core of the CMS software is often referred to as "the framework"
- based upon the Event Data Model (EDM)
  - the central concept is the "event"
  - an event is immutable: existing information in the event cannot be changed
  - exchange of information between modules happens only via the event
  - the event is processed along a "path", an ordered list of modules
- working with modules
  - piece (or component) of CMSSW code that can be plugged into the CMSSW executable cmsRun
- EDProducer, EDFilter, EDAnalyzer, OutputModule (and EDLooper)
- modules to be arranged in sequences and to go into path
- https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookCMSSWFramework



What the Framework provides you with

- a binary, to which all modules are linked
  - → cmsRun
- job configuration
  - → config file language (python) + tools to read in job configurations into code
  - → avoids the need to recompile when parameters change
  - → Python tutorial
    - http://indico.cern.ch/conferenceDisplay.py?confId=37576
- provenance tracking
  - → datafiles always contain a full history of the used configurations
- EventSetup and services
  - → calibrations, logging, random numbers,...
- debugging and error handling
- reading in data PAT Layer-1
  - → reading in files or streams
  - → retrieving data from the event
- producing and writing data
  - → creating your own objects and dataformats

The different software components

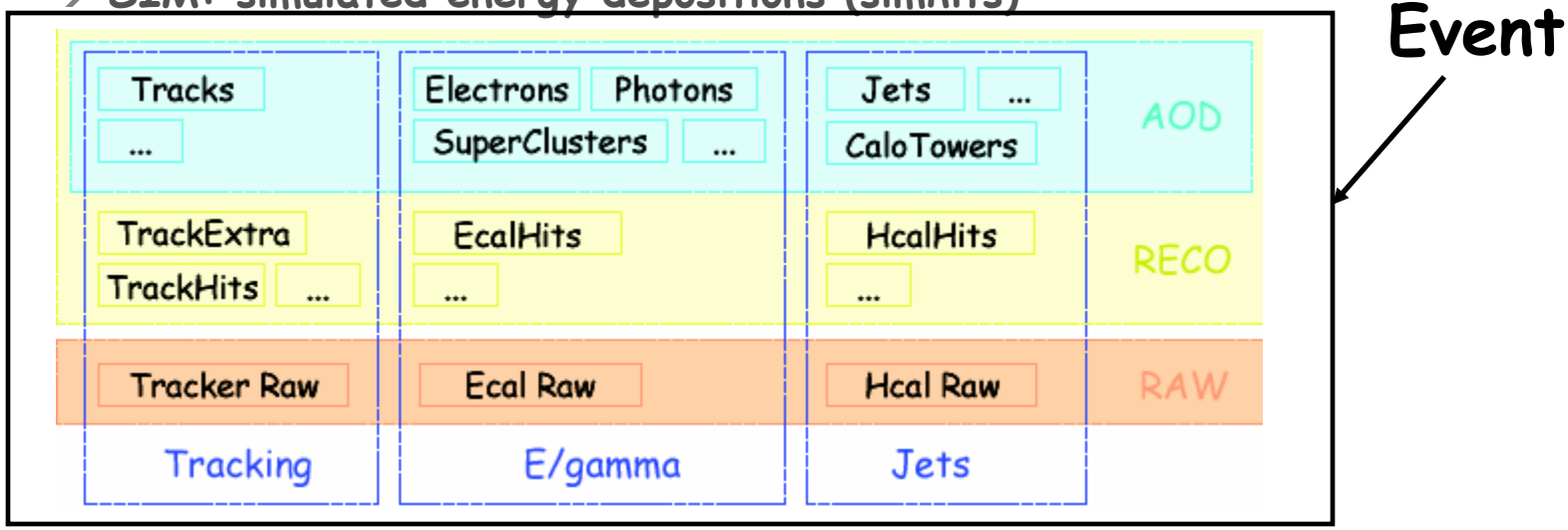
- the main CMS software: CMSSW
  - http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/
  - contains everything from the core framework to analysis code
  - also contains simulation, FWLite, Fireworks, Iguana
- daily workhorse: ROOT
- code compilation and external software configuration: SCRAM
- code management and reference: CVS, LXR, Doxygen
  - http://cmslxr.fnal.gov/lxr/source/
  - http://cms-cpt-software.web.cern.ch/cms-cpt-software/General/gendoxy-doc.php
- job submission to the grid: CRAB
- data storage: ROOT, CORAL(COmbining Relations And Logic Language)
- ...

Data access with CMSSW, FWLite and ROOT

- CMS event data is stored in ROOT files
- storing C++ classes into ROOT's TTrees happens through so called dictionaries
- ways of doing analysis on these ROOT files
  - full framework, using cmsRun: full flexibility, but a bit cumbersome and a rather slow turn-around
  - FWLite: access to the data formats and their methods in rootfiles, without the burden (and benefits) from the full framework; can be compiled in FW or ROOTor run interactively in ROOT
- sometimes both within the same analysis at different steps of complexity reduction

Analysis within the CMS Computing Model

- event data is organized in so-called data-tiers
  - RAW: output HLT; primary archive format; input to offline reconstruction
  - RECO: offline reconstructed objects
  - FEVT: RAW + RECO
  - AOD: format for physics analysis; subset of RECO
  - PAT: Physics Analysis Tools
  - GEN: Monte-Carlo information
  - SIM: simulated energy depositions (simhits)



https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookDataFormats

The AOD event content

- AOD is by definition a subset of RECO, and contains many small containers that are linked through associations
  - very useful in computing: smaller than RECO
  - very useful for development
  - analysis within the CMS Computing Model
  - logical consequence of the EDM "event" paradigm
  - less problems of backwards incompatibility
  - complicates an analyser's life significantly
  - time-to-first-plot is too long; learning curve is too steep
- more on the AOD and RECO content

https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideRecoDataFormat

Outline

- The CMS analysis environment
- Overview of the Physics Analysis Toolkit
  - PAT: what is it about?
  - PAT: what it is not about
  - the PAT in layers
  - Output of PAT
- The Starter Kit
- Summary

PAT: what is it about?

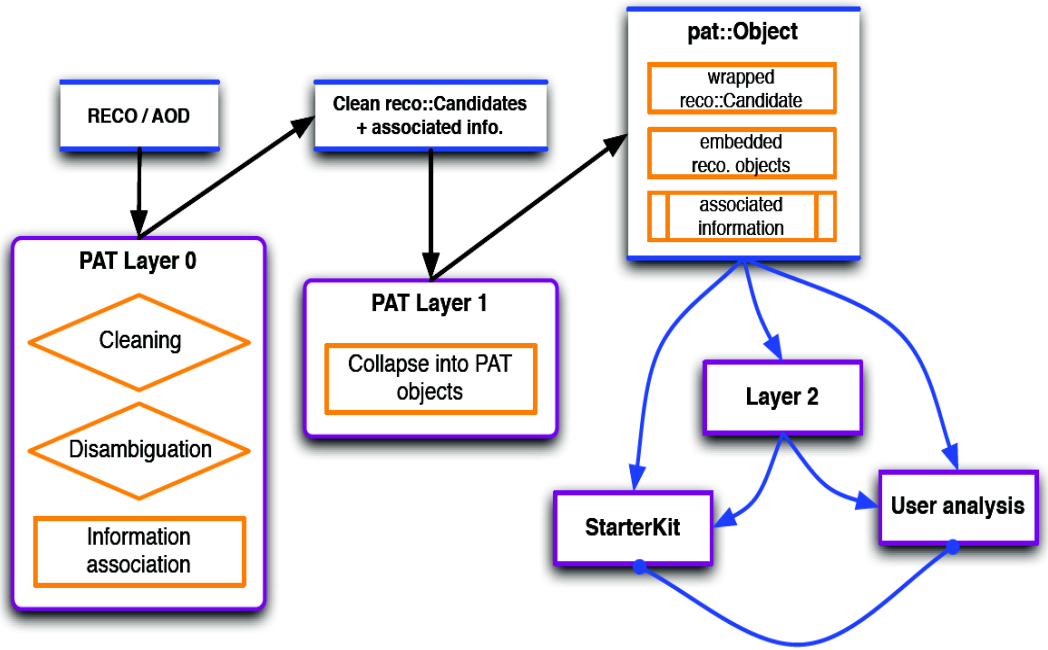
- bridge to close the gap between the rough AOD and physics plots
  - → we start from the needs from the physics user's perspective
- common set of basic tools for almost any physics analysis in CMS
  - → no code duplication and faster development with a common tool
  - → easier, faster and shared problem solving and debugging
  - → code reliability enhanced -> easier approvals of the results
  - → easy to switch from one to another analysis
- provide sensible defaults and configurations
  - → always have a consistent workflow through the PAT with very few includes
  - → allows to quickly bootstrap an analysis chain
- full FWLite compatibility
- PAT – Physics Tools in CVS
  - → http://cmssw.cvs.cern.ch/cgi-bin/cmssw.cgi/CMSSW/PhysicsTools/

PAT: what it is not about

- **we don't re-invent the wheel**
  - → we don't rewrite the AOD objects, we stay in the Candidate model
  - → POG specific tasks remain with POGs, we provide an interface to POG's
  - → PAG specific algorithms remain within the PAG code-base (but we can provide infrastructure)
- **the PAT is not a framework on it's own**
  - → fully embedded in the EDM: no private "rootuples" are written
  - → profit from framework's persistency and provenance tracking
- **we don't claim to cover all analysis use cases**
  - → but if you need/have extra tools, let us know and contribute!

The PAT in layers

- a multi-layered approach for maximal flexibility and user-friendliness within the constraints of the EDM
- event interpretation
  - "Layer 0": cleaning and disambiguation -> event interpretation + additional analysis level tasks (e.g. MC+trigger matching)
  - "Layer 1": creation of big objects that collapse externally associated information -> no algorithmic tasks
- event hypothesis
  - "Layer 2": event hypothesis dependent tasks -> provide possibility for re-tuning event interpretation
- analysis
  - "Starter Kit": for data exploration and plotting
  - "paste-your-analysis-here"



https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATLayer0
https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePATLayer1

Output of PAT

The input of the PAT is AOD/RECO.
The output of the PAT is your business

- We give tools for many output options:
  - Skimming – dropping events
  - Slimming – dropping candidates – taus, electrons...
  - Thinning – making object smaller i.e.dropping info
- We do not constrain any of them. It is entirely between you, your PAG/POG, and the other members of your analysis

This is not intended to be another large scale layer of production but it is becoming one

Outline

- The CMS analysis environment
- Overview Physics Analysis Toolkit (PAT)
- The Starter Kit
  - Goals
  - Components
  - Overview
  - The Starter Kit in practice
- Summary

Goals

- a plotting package for standard plots of standard objects the CMS Event Data Model
  - allows quick exploration of data without lots of coding and bookkeeping
  - goal is to provide several analysis examples, monitored for physics changes
  - make histograms and simple "rootuples" out of PAT Layer-1 or Layer-2 objects
  - user can easily add plots (requiring some coding, obviously)
- the Starter Kit is ideal to make you step into CMS analysis and get to do some physics
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKit

Components

PatAnalyzerSkeleton

- "VerySimplePatAnalysis" plugin
- Shows how to pull out PAT objects and loop over them simply

PatAnalyzerKit

- "StarterKit" plugin
- Plots everything in the event and has a simple ntuple interface

CompositeKit

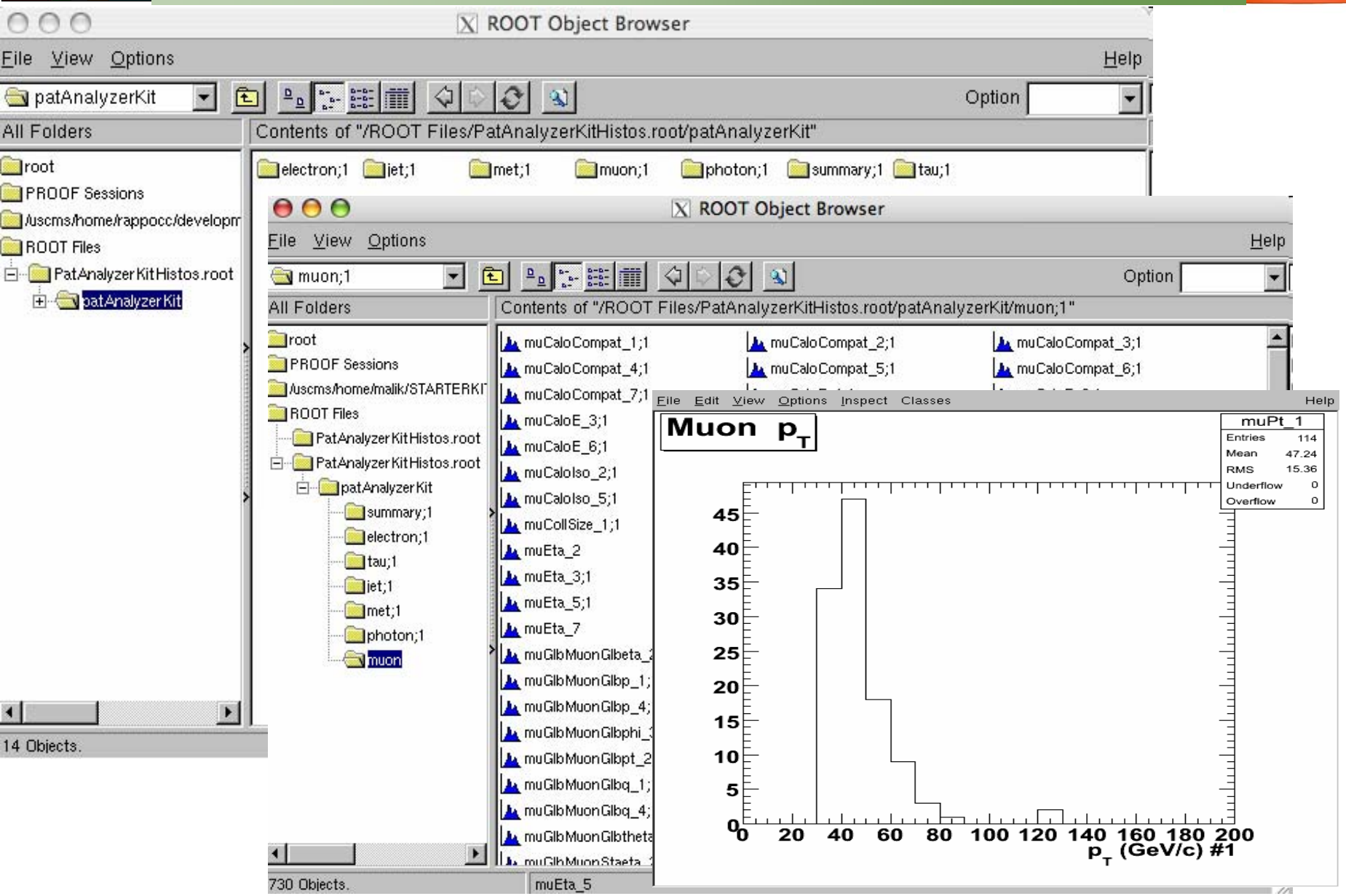
- Plots an input collection of composite candidates

The Starter Kit in practice (it's a package NOT a module)

- examples in the CMS Workbook
- what it does for you
  - automatically plot four vector and object ID variables for all objects in the event
  - sensible default axis limits
  - also simple rootuples of these variables can be added
  - driven by configuration files
- check it out in the tutorial!

The Starter Kit

Physics Analysis Toolkit



Outline

- The CMS analysis environment
- Overview of Physics Analysis Toolkit (PAT)
- Summary
  - Conclusions
  - Documentation
  - Tutorial

Conclusions

Physics Analysis Toolkit

- the PAT is a vital project
  - started from and unifies the several CMS analysis frameworks
  - emphasis on both flexibility and user-friendliness
  - core PAT functionality is in good shape
- the PAT is ready for users
  - out-of-the-box examples available
  - documentation coming into place
  - embedded successfully in Layer-2 and Starter Kit
  - users/testers from all PAGs coming in
  - first analysis results being produced with PAT
- there is still a to-do list of course
  - we keep a prioritized list linked to the twiki
  - users are encouraged to contribute!
    - -> not all tasks have names assigned
    - -> add your own tools for existing/missing features!

Documentation and support (get used to twikis and book mark them)

- documentation
  - currently PAT's highest priority!
  - with help from the CMS documentation people (Kati et al.)
  - central wiki page in the CMS Software Guide:
    - https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePAT
      and many links therein
  - Starter Kit workbook to get you started
    - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKit
  - PAT page in the CMS workbook: contains the PAT tutorial
    https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookPAT
  - see also the more general Physics Tools page:
    - https://twiki.cern.ch/twiki/bin/view/CMS/SWGuidePhysicsTools
  - FWLite
    - https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideFWLiteAnalysis
  - Fireworks
    - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookFireworks
- For submitting jobs on grid please go through CRAB pre tutorial
  - https://twiki.cern.ch/twiki/bin/view/Main/EricVaanderingCRABPreTutorialJTerm
- Look for data in DBS
  - https://cmsweb.cern.ch/dbs_discovery/
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookDataSamples
- Write your own analyser
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookWriteFrameworkModule
- support
  - all questions about PAT should currently go to the PhysicsTools hypernews
    - **hn-cms-physTools@cern.ch**

Tutorial

There will now be a step-by-step tutorial session to put this into practice!

- Starter Kit tutorial
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKit
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKitCodeDescription
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookPAT

- Also try to attend the following two talks tomorrow for more general overview/status
  - 9:00 CMS Software - Elizabeth Sexton-Kennedy (Fermilab)
  - 9:25 Physics Tools and Analysis Model – Salvatore Rappoccio (Johns Hopkins)

Tutorial

- Please login to your cmslpc account
- Open the first twiki and we will go through it step by step
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookAnalysisStarterKit
  - On the twiki all the text in blue boxes is coding

```
# set the number of events
process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(200)
)
```

  - All the command lines and output is in brown boxes

```
cd StarterKit/CMSSW_2_1_9/src
cmsenv
cd PhysicsTools/StarterKit/test
cmsRun PatAnalyzerSkeleton_cfg.py >& skoutput.txt &
```

  - Further questions malik@fnal.gov

Backup Slides

Quote from the PAT mandate

The present situation causes substantial and unnecessary duplication of work and results in ever-increasing confusion of the users. In particular, analysis beginners (who constitute the vast majority of the CMS physicists at this point in time) do not know where to start from to get their input for their analysis, and as a result their possible involvement towards physics studies is delayed.

A related issue of concern is the analysis of the (first) data. In the absence of a unified PAT, analyses rely heavily on the availability of experts from the various groups, resulting in unnecessary delays between data taking and running of the analyses. These delays would slow down the understanding of potential problems with the detector, the reconstruction chain, the analysis programs and could even have dire effects on the competitiveness of CMS.

Design considerations

- remember we want to reconcile maximal flexibility vs. maximal ease-of-use
- every action in the following steps is required to be fully configurable from config files
  - ➔ so also switched on/off
- every algorithm in the following steps needs to be factorized from the CMSSW framework, such that it can be re-used within FWLite
  - ➔ this means that e.g. no interface to the algorithms should involve the configuration language, and no database access is required

PAT Layer-0

Object-dependent ID and cleaning

- object identification (ID) variables
  - produce if not on AOD yet
  - using POG recommendations as defaults
  - interface multiple algorithms
  - provide possibility for custom ID
- object flagging and cleaning
  - technically: a cleaner (EDProducer) calls a FW-independent selector
    -> this factorizes algorithmic code from the rest
  - use the ID variables to choose "clean objects"
  - technical implementation with flags written in the 'status' of the object
    -> also used in the next step of event-wide ambiguity resolution
  - possibility to keep "clean", "non-clean" and all objects
    -> e.g. isolated vs. non-isolated electrons

PAT Layer-0

Cross-object ambiguity resolution

- happens logically in the same processing steps as the Layer-0 cleaners
- both "clean", "not-clean" and "all" collections can be used to check overlaps
- in photon cleaning
  - → electron removal "bySeed" (default) or "bySuperCluster"
- in tau cleaning
  - → functionality there, but all switched off by default
- in jet cleaning
  - → current default: only electrons (but in addition e.g. taus can be used)
- sequence of cleaners determines the order of ambiguity resolving
  - → default order: muons, electrons, photons, taus, jets
- once cleaning disambiguation is done MET can be easily recomputed

PAT Layer-0

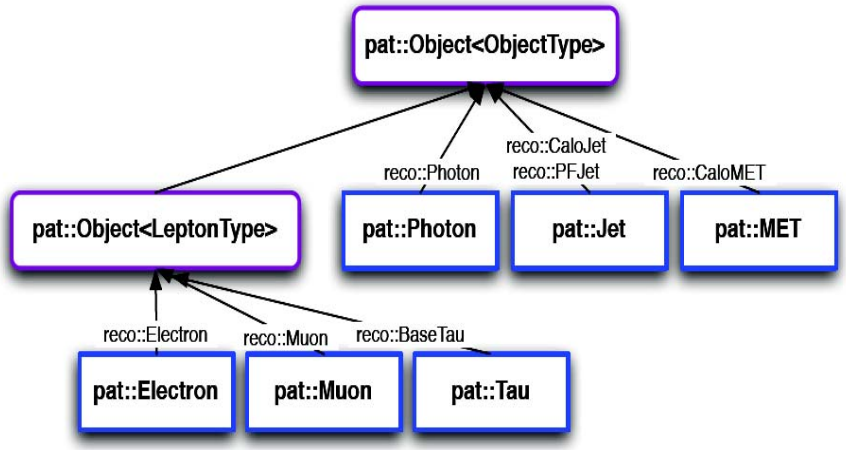
Production of additional associated information

- at the end of the Layer-0 processing, additional analysis-level tasks can be performed to produce information not present yet in the RECO/AOD
- this should involve all remaining tasks depending on the framework
  - leave no obstacles to use FWLite on Layer-0 output
- some of these tasks don't involve an event interpretation (can possibly run before the actual Layer-0)
  - retrieval jet energy scale factors
  - parton flavour determination for jets
  - jet-track association and jet charge
  - additional b- and tau-tagging
  - ...
- some tasks depend on the event interpretation
  - MC matching, trigger primitive matching
  - external MET recalculation?
  - ...

PAT Layer-1

PAT objects: enriched reco dataformats

- PAT Layer-1 brings lots of related information together in "big" objects
  - easy one-entry user interface to previously associated information
  - 1-object approach without remaining associations makes it easy for users to play with the objects: event selection, scaling and smearing, vetoing, kinematic transformatic
- PAT Layer-1 objects
  - implemented: **Electron**, **Muon**, **Tau**, **Jet**, **Photon**, **MET**
  - inherit from reco dataformats
  - add additional members and methods for accessing the extra information
  - only store what is asked for by the user



- will provide a uniform interface to Particle Flow objects

PAT Layer-1

PAT object interfaces

- **all PAT objects**
  - → have methods to access their MC and trigger matches
  - → have methods to retrieve resolutions obtained from MC
- **pat::Photon, pat::Electron, pat::Muon and pat::Tau**
  - → contain methods to access isolation and ID variables
- **pat::Jet**
  - → provides access to b-tagging information
  - → methods for handling jet corrections
  - → access to jet flavour from MC
  - → interface to associated tracks and jet charge
- **pat::MET**
  - → MC match is in this case the generated MET

Introducing event hypotheses

- up to now no assumptions have been made about the final state
  - ➔ in principle the final state hypothesis belongs more to the analysis
  - ➔ but tools and examples can be provided to users on how to easily deal with their specific final state
- User has a "ph… d:

```
Z -> muon + muon
```

- Want to support s… ic this intuitive picture

```
muon1 : muon
muon2 : muon
Z = muon1 + muon2
```

- Can s…

```
H -> (Z -> muon + muon) + (Z -> electron + electron)
```

```
muon1 : muon
muon2 : muon
electron1 : electron
electron2 : electron
Z1 = muon1 + muon2
Z2 = electron1 + electron2
H = Z1 + Z2
```

PAT Layer-2

Flat event hypothesis

- Provided by a simple list that associates strings to objects
- Access data members through those strings
  - → Supports PERL-syntax regexpr

```
CandLooper looper<Muon> =
hyp.loop("muon.");

while( !looper() ) {
   const Muon & mu = *looper;
   fill( mu.pt() );
   looper++;
}
```

Hierarchical event hypothesis

- Uses a "named" Composite Candidate to store decay structures in a hierarchy
- Access data members through Candidate linked list utilities
- Added functionality: can access daughters by role string instead of index

```
const Candidate * Z1 = H.daughter("Z1");
```

PAT Layer-2

"Hard coded" event hypotheses

- **Top Quark Analysis Framework (TQAF)**
  - Has hard coded ttbar hypothesis

```
pat::Jet getHadb() const;
pat::Jet getHadp() const;
pat::Jet getHadq() const;
pat::Jet getLepb() const;
pat::Muon getMuon() const { return *muon_; };
pat::Electron getElectron() const { return *electron_; };
pat::MET getNeutrino() const { return *neutrino_; };
```

  - Recently have interfaced TtSemiEvtSolution with NamedCompositeCandidates to provide generic access as a prototype

```
const reco::NamedCompositeCandidate & getRecoHyp() const { return recoHyp_; }
```

- **Others are out there**

????

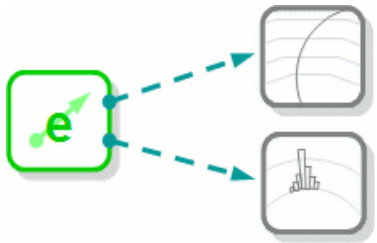
- Ignore warning during scramv1 b
- Version – data compatibility
- Ask questions at hn-cms-physTools@cern.ch
- How to get files of the web (wget)
- Look for data in DBS
  - https://cmsweb.cern.ch/dbs_discovery/
- Write your own analyser
  - https://twiki.cern.ch/twiki/bin/view/CMS/WorkBookWriteFrameworkModule
- For advance tutorial, go through CRAB pre-tutorial
  - https://twiki.cern.ch/twiki/bin/view/Main/EricVaanderingCRABPreTutorialJTerm

Overview of the Candidate Model

What is a Candidate?

- Represented by a vertex and a four-vector
- Has particle ID information
- Anything that has a four-vector and a vertex inherits from Candidate
  - Jets
  - Electrons
  - Photons
  - Muons
  - Missing Et

- Candidates can have detector objects as data members



- Support for lists of candidates

- Can also have composite candidates

Overview of the Candidate Model

Navigation

- Candidates have "mothers" and "daughters"
- For instance, Monte Carlo particles are Candidates (GenParticle), have access to full linked list of ancestry and progeny

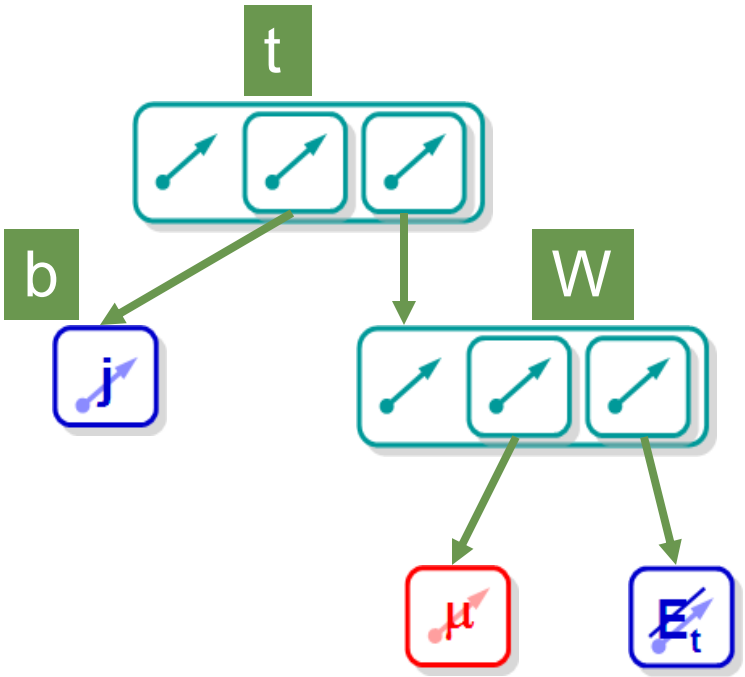
```
const GenParticle & higgs = getHiggsFromSomewhere();
for( size_t i = 0; i < higgs.numberOfDaughters(); ++ i ) {
        const Candidate * Z = higgs.daughter( i );
}
```

- Composite candidates "act" exactly the same way!
  - ➔ In above example, replace GenParticle with CompositeCandidate,

```
const CompositeCandidate & higgs = getReconstructedHiggs();
for( size_t i = 0; i < higgs.numberOfDaughters(); ++ i ) {
        const Candidate * Z = higgs.daughter( i );
}
```

Overview of the Candidate Model

- Hierarchical structure
  - Access hypotheses in an intuitive way
  - Easy interface for I/O of your objects
  - Can have shallow clones for storing kinematics of fitted objects
  - Utilize common plotting utilities to save you time



- Ultimately a time saver!

Overview of the Candidate Model

Changes to candidate types

- Previously Candidate landscape was a little cluttered
- We changed that! (>=2.0.9, >=2.1.0pre6)
- The only things you need to worry about are
  - "by value": Candidate (base Candidates)
  - "by pointer": Ptr<Candidate> (smart pointers to Candidates)
  - "by shallow clone pointer": ShallowClonePtrCandidate (Ptr<Candidate> which has different four-vector and vertex, suitable for fitting, etc)
- Composite Candidates have also subsumed functionality of "NamedCompositeCandidates"
  - Can now add names and roles to CompositeCandidate, NamedCompositeCandidate is deprecated

Overview of the Candidate Model

Physics Analysis Toolkit

Candidate Selectors

- Many selectors overall
  - Pt, eta, delta R, pdg id, etc
- Mostly limited to things available to candidate
- Can use the same functionality to make more advanced selectors
  - This is a little cumbersome to do

Where's the code?

- Some examples:
  - http://cmslxr.fnal.gov/lxr/source/PhysicsTools/CandSelectors/plugins/
  - http://cmslxr.fnal.gov/lxr/source/PhysicsTools/RecoAlgos/plugins/
- The real drivers:
  - http://cmslxr.fnal.gov/lxr/source/PhysicsTools/UtilAlgos/interface/SingleObjectSelector.h
  - http://cmslxr.fnal.gov/lxr/source/PhysicsTools/UtilAlgos/interface/ObjectSelector.h

```
module goodMuons = CandSelector {
   InputTag src = allMuons
   string cut = "pt > 5.0"
 }
```

Overview of the Candidate Model

Candidate Combiners

- **CandViewCombiner**
  - Combines by "value"
    - Outputs View<CompositeCandidate> with value clones of leaves
- **CandViewShallowClonePtrCombiner**
  - Combines by "shallow clone pointer"
    - Outputs View<CompositeCandidate> with shallow clones of leaves

Where's the code?

- **PhysicsTools/CandAlgos/interface/CandCombiner.h**
  - plugin definition
- **PhysicsTools/CandUtils/interface/CandCombinerBase.h**
  - Workhorse for the actual combinations
- **PhysicsTools/CandUtils/interface/CandCombiner.h**
  - Driver for workhorse

```
module zToMuMu = CandViewShallowCloneCombiner {
   string decay = "muons@+ muons@-"
   string cut = "50 < mass < 120"
   string name = "zToMuMu"
   vstring roles = {'mu1', 'mu2'}
 }
```