

C++ Developments @ H1

H100

A new physics analysis framework

Andreas Meyer
University of Hamburg
Andreas.Meyer@desy.de



Seminar Datenverarbeitung in der Hochenergiephysik, 6 May 2002

Motivation

- Reorganize central physics analysis tools
 - Reunify cuts - establish common particle concept
 - Make expert knowledge persistent - reusable
- Integrated analysis environment:
 - One tool for data storage and interactive analysis
 - Common and unique code reference
 - Code portability between private analysis and common H1 software
- OO-paradigms to organize software design and maintenance

Choice: C++, RooT (and custom-built extensions)

Scope

- H1 is a running experiment
- Reconstruction/simulation code remains stable
- Scope: reorganize the quickly changing part - ie. physics analysis code and data
- Constraint: sustain quality & precision - do not disturb ongoing physics analyses
- Development: core group effort - continuous support of and feed back from users

Nice and easy-to-use modern set of physics tools

Overview

Development Strategy

Project roadmap

Data Storage Model

ODS / mODS / HAT

Data Access Tools

'H1Tree' / 'RunCatalog'

Physics Algorithms

coherent / modular / portable

Physics Analyses

Some examples

Event Display

Integrated with analysis

Performance

Precision & speed

Documentation

'Spreading the message'

Project Roadmap

Four-Phase Development:

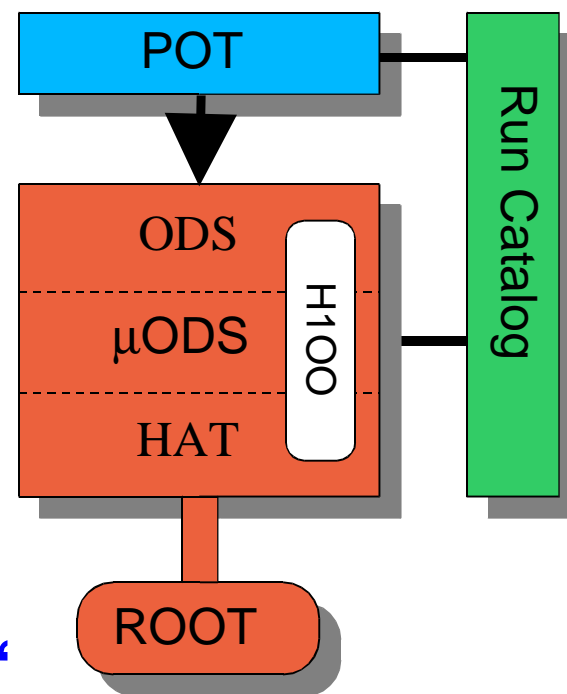
- Development and Release Environment Jan.2001
- Implementation of Data Model, Rel. 1.0.0 May 2001
- Portable C++ Filling Code, Rel. 2.0.0 Feb.2002
- Consolidation Ongoing
 - Continuously improve framework and physics contents
 - Implement full precision (calibration)
 - Speed (remove bottlenecks)
 - Initiate and support new analyses in H100 (P.R. & documentation)

Organization

- Core Development Team:
 - Order of 8 Ph.D. Stud., 5 PostDocs, 2 Engineers
 - Weekly developers' meeting
- Code management: O(100k lines in 30 CVS Packages)
 - Clear package task and hierarchy (1 responsible/pkg)
 - Keep packages small
 - Common pkg-structure and generic makefiles
- Code Releases (deadline-driven with well-defined goal)
 - Every two weeks: "Release often and listen to users"
- Project Review Board:
 - Regular reviews on progress and plans

Data Storage Model

- Based on ROOT
- Three hierarchical layers
 - Reconstruction: 'ODS' (Object Data Storage, 15 kB/evt)
 - Particles: ' μ ODS' (1.5 kB/evt)
 - Event Tag: 'HAT' (0.5 kB/evt)
- Additional layer: 'User Tree'
 - RunCatalog: Retrieve data by run and event #

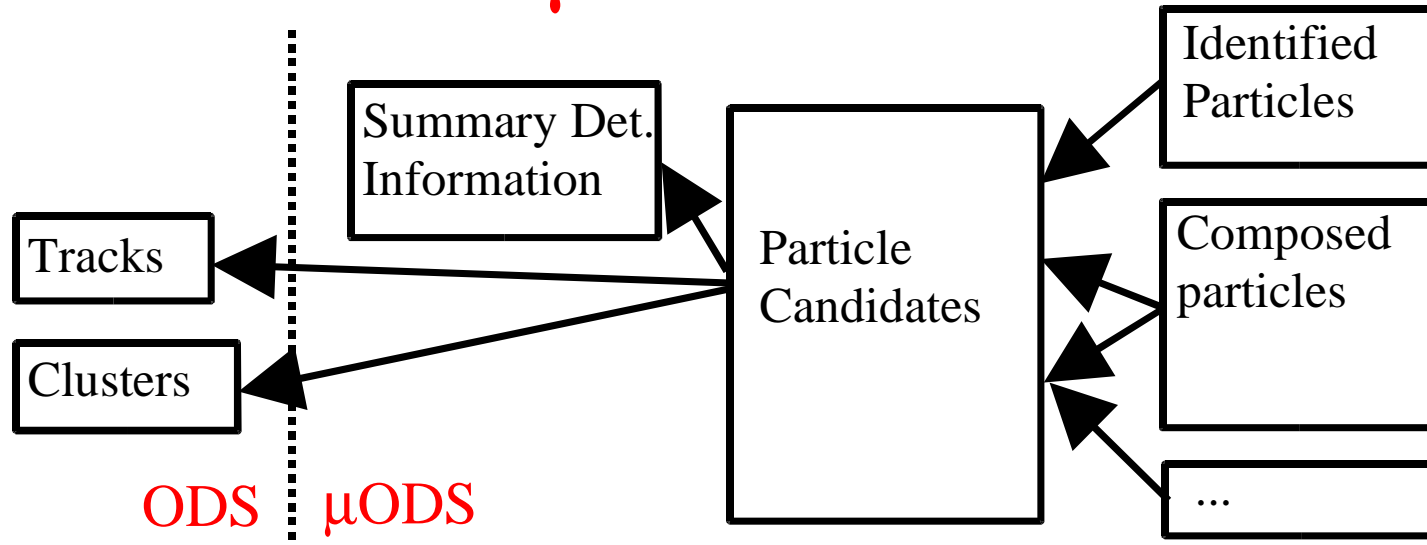


Common environment for both H1 and user data

Three Layer Data Storage

- ODS: reconstruction level
 - Tracks, clusters, detector information
 - 1-1 correspondence with former DST
 - Backward compatibility:
Existing analysis software remains functional
- μ ODS: particle level
 - 4-vectors of particles, charge, ID, navigation ptrs.
 - "Intelligent" accessors
- HAT: event level
 - Triggers, event kinematics, particle multiplicities etc.
 - Ntuple: only floats and ints
 - -> Fast event selection

μ ODS



- **Particle Candidates:** particle kinematics
Pointers to reco-level tracks and/or clusters (ODS)
- **Identified/Composed Particle Lists:**
Pointers to specific, selected particle candidates (extendable)
- **In addition: Summary Detector Information:**
Extra information for specific tracks and clusters

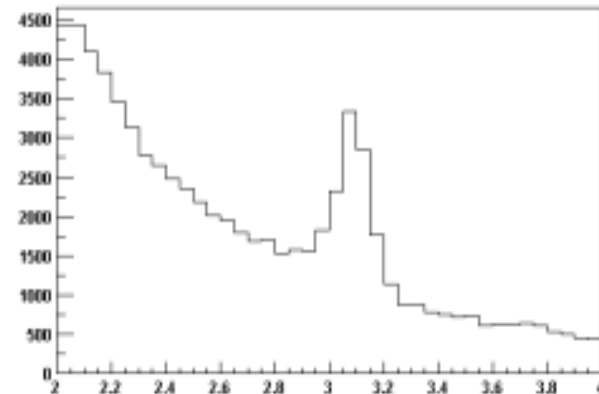
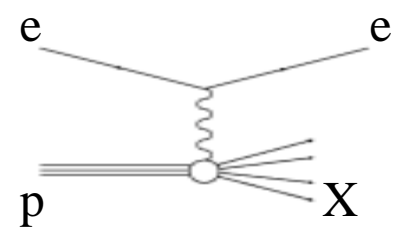
Quick navigation over particles (μ ODS)

Direct access to tracks and clusters (ODS)

H1 Particle Candidates

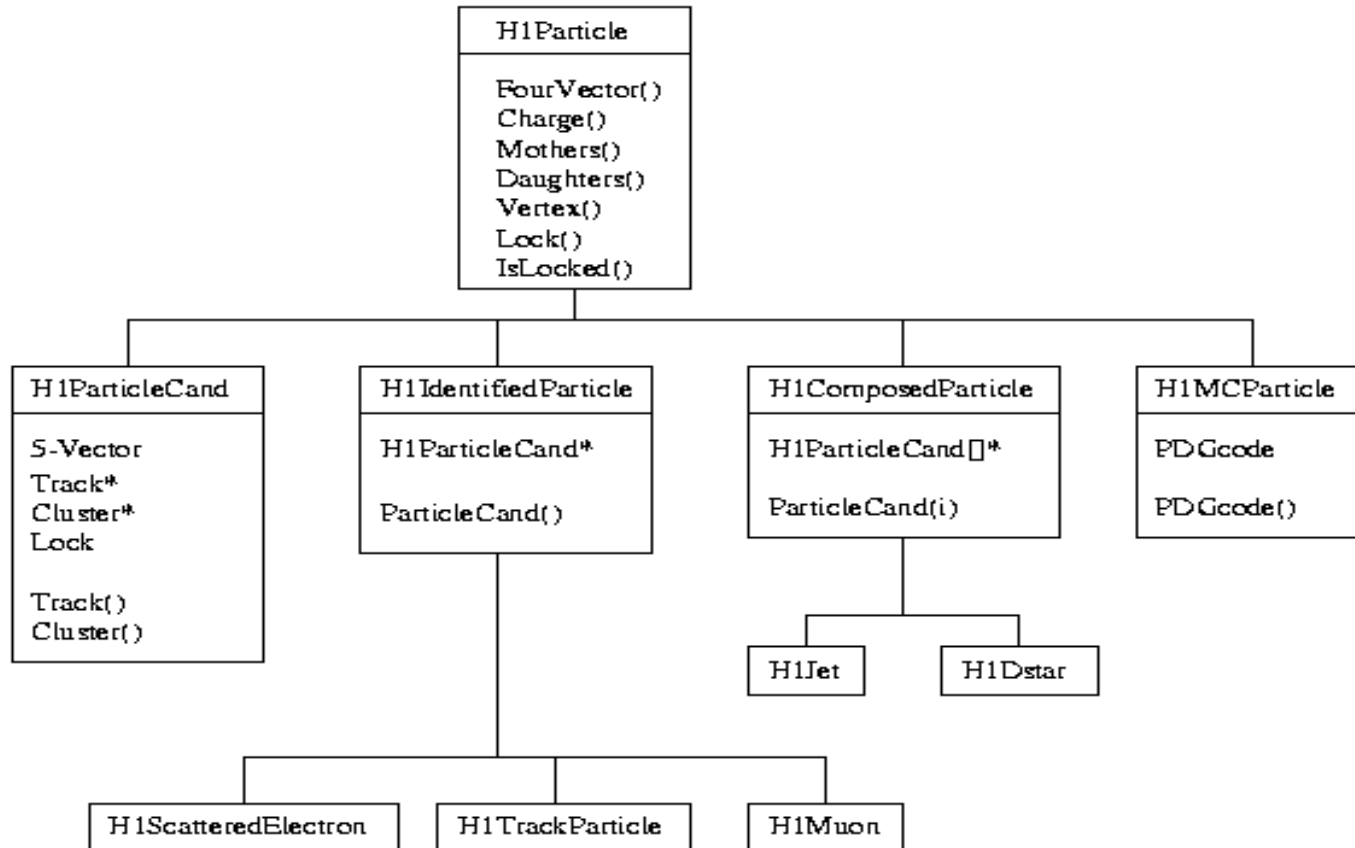
Persistent storage of physics output on μ ODS

- Charged Particles (Tracks)
- Electrons (incl. scattered electron)
- Muons
- Hadronic Final State Objects
- Jets (Kt, Jade)
- $J/\Psi \rightarrow ll$ / $D^* \rightarrow K\pi\pi$
- ... / leading p / $\rho^0 \rightarrow \pi\pi$ / $\pi^0 \rightarrow \gamma\gamma$ / ...



Each track and/or cluster object counted only once
with possibly multiple ID hypotheses

Particle Class Hierarchy

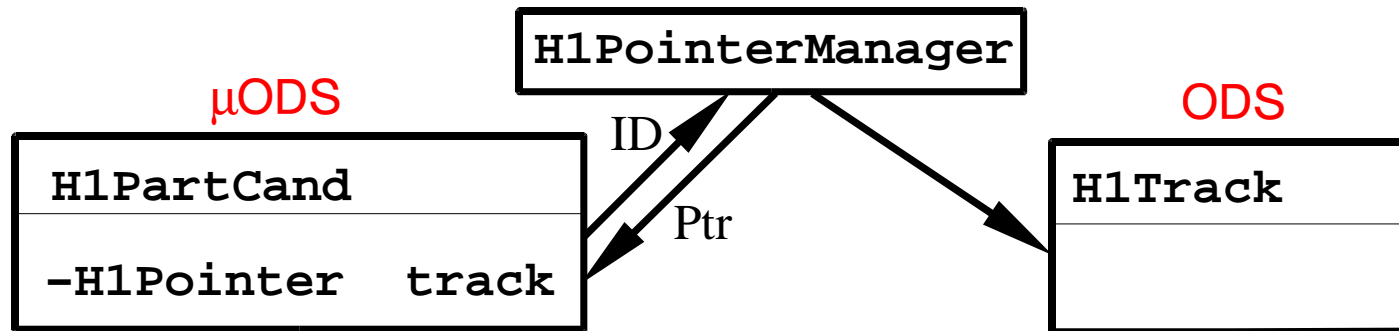


Inheritance of accessor methods: Enforce same interface for all particle types

Particle Navigation

H1OO Framework Class 'H1Pointer':

- Persistent storage of class relations (also across different files, e.g. μ ODS \rightarrow ODS)
- Direct access from particles (μ ODS) to complete reco-level information (ODS)



- Track and/or Cluster data automatically loaded when H1Pointer dereferenced

H1Pointer extends present functionality of RooT

Data Access

H100 Framework Classes 'H1Tree' and 'RunCatalog'

- H1Tree:
 - File handling (3 layers + user tree)
 - Event (pre-)selection on HAT -> event list
 - Direct access to selected events
 - Partial reading of data (subset of events and variables)
- Runcatalog (mySQL database)
 - contains relations between run/event# and files

Data Access fully encapsulated
Event delivery according to individual user selection

μ ODS/HAT Production Sequence

H100 Finder Modules

Modular: abstract finder-class defines (polymorphic) interface for concrete finder classes -> bookkeeping for particle list creation

Coherent: no double-counting of energy and momenta

Portable: develop and/or re-run finders during analysis (Root)

- Create Particle Candidates -> μ ODS Sel. Trks / EM-Parts / μ / HFS
- Find Composed Particles -> μ ODS Jets / D^* / J/Ψ /
- Calculate Event Level Info -> HAT Multiplicities, Kinematics, Trigger & Sel. Flags

Make new μ ODS and HAT when new algorithms are available (every 2-3 months), 99/2000 production takes $O(1)$ week).

Analysis Sequence

- Fast Event Selection using HAT level information:
 - `H1Tree->SelectHat("selectionstring")`
 - Retrieve eventlist (index) or eventfiles (subset)
- Iterate on selected events (index or subset)
 - HAT and μ ODS (1.5 kB/event)
 - direct access to ODS (optional, via H1Pointer)
 - store additional info using User-Tree (optional)
- Accessor methods provided by container classes
 - In addition: high level analysis tool box 'calculator'
 - calculation of non-persistent quantities
 - variation of syst. errors etc.

Ultimate goal: Full interactive use of inclusive data set

Event Display

Completely integrated in analysis framework

- e.g. event selection on HAT
- display events from within analysis code
- Modern GUI
- Pick and inspect
- Software from previous display largely re-used
- Can display data in RooT and BOS format
- Also accepts commands/macros of previous command-line based event display

An Event

H1Cluster		H1Cluster.0	H1 cluster class
Member Name	Value	Title	
fNCellsPerCluster	49	<i>Total number of cells</i>	
fEnFIL	170.42	<i>Energy at final level</i>	
fEnEML	170.42	<i>Energy at e.m. level</i>	
fEnOL	168.628	<i>Energy at e.m. level before dead material cor</i>	
fBarycenter.fX	-106.747		
fBarycenter.fY	23.4398		
fBarycenter.fZ	167.682		
fBarycenter.fUniqueID	0	<i>object unique identifier</i>	

Analysis Results

Number of H100 physics analyses continuously increasing

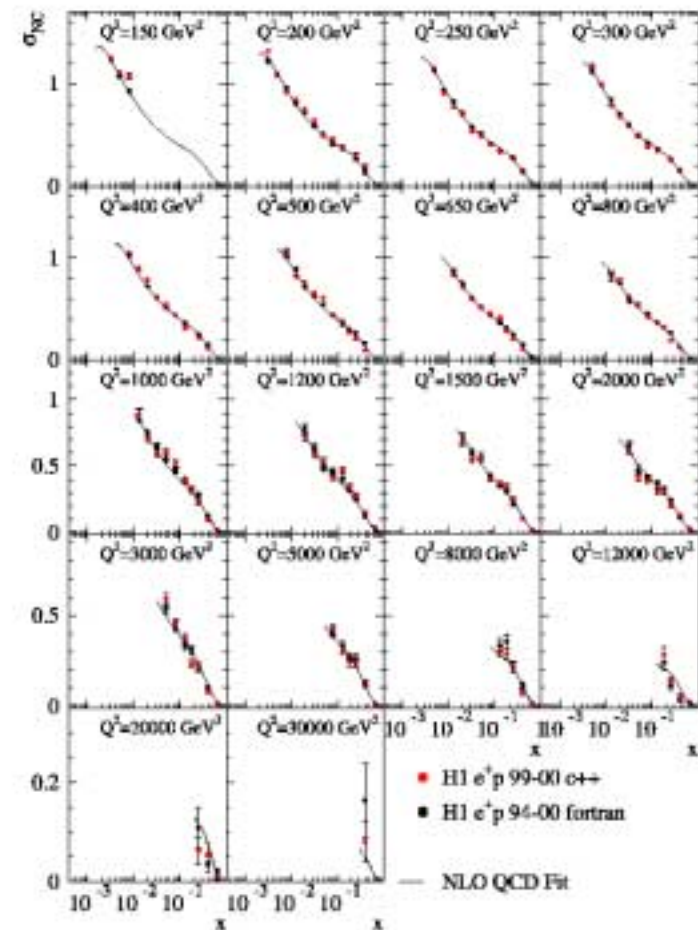
O(30) Users at present

Examples shown here:

- F_2 at high Q^2
- J/Ψ in DIS
- High- P_t Leptons (W-Events)

Also used for technical studies

- Fast Track Trigger
H1 HERA-II upgrade



Precision

Calibration Framework is coming last ... in time

- Late implementation on purpose:
 - First: Clean & modular implementation of algorithms
 - Then: Use one common calibration for all algorithms
- Main calibration issues:
 - Beam tilts (for kinematic variables)
 - Calorimeter (fine-)calibration (corrections of few %)
- Interface with existing reconstruction database
 - Provide snapshot of needed database contents in RooT format (accessible from RooT-session)

Speed

The essential live-or-die issue...

... and one of the difficulties of object-oriented programming

Present Status:

- HAT (event summary ntuple with floats and ints):
 - ✓ Full RooT performance available by construction
 - x Some H1OO selection tools are presently factor of O(10) slower than RooT
- mODS (list of identified particles):
 - x Scales with length of particle list (ave. of about 30 parts./evt)
 - => Split of particle list into different branches (being implemented)
 - Re-organisation of container classes for faster access (being considered)
- ODS:
 - x Access to clusters too slow:
 - => Cluster and cell container classes being revised

Bottlenecks in H1Tree and H1Pointer being identified and widened

Numbers from an Analysis

Example analysis of jets in diffraction

- 99e+: 24 million events in total
- First Iteration: `H1Tree->SelectHat()`
 - Select subset of 22k events
 - Store μ ODS and HAT files locally on desktop PC

Speed comparison with HBOOK:

- H1OO: 1-2 mins for 22k events on 350 MHz machine
 - HBOOK: 2-3 mins for 100k events on 500 MHz machine
- => H1OO presently factor of two slower than HBOOK**

**Still serious drawback esp. for inclusive analyses
Hope/need to overcome bottlenecks soon**

Spreading the Message ...

Precision, flexibility and speed:

... sine-qua-non requirements for success

- Exceed capacities and performance of previous physics tools
 - Initiate physics studies (e.g. update of track-cluster matching alg.)
 - Collect and implement useful so-far-private analysis code
- P.R., documentation and communication
 - Workshops on physics algorithms
 - Aim for H1-wide unification of algorithms and terminology
 - Documentation, tutorials and presentations
 - Individual support (direct personal contact)
- Online production of ODS, mODS and HAT with HERA-II
 - HAT-files replace Data-Quality ntuples

Summary

- A New Object-Oriented Physics Analysis Framework has been established:
 - based on RooT
 - persistent storage of expert analysis results
 - fast access and navigation of events, particles, reco-objects
 - coherent data formats, physics code portability
 - unified and reorganized physics analysis algorithms
- A number of extensions to RooT:
 - H1Tree/RunCatalog: file handling / event delivery
 - H1Pointer: persistent relations across files
- A fully integrated event display

Conclusions

- Number of physics analyses is slowly, but continuously increasing
- Critical items at present and in the future:

Physics analysis capabilities:

- ✓ Up-to-date Analysis Algorithms
- ✓ Precision
- ✓ Speed

H1 Physics is starting to profit from this effort