Software for future colliders

Juraj Smieško (CERN)

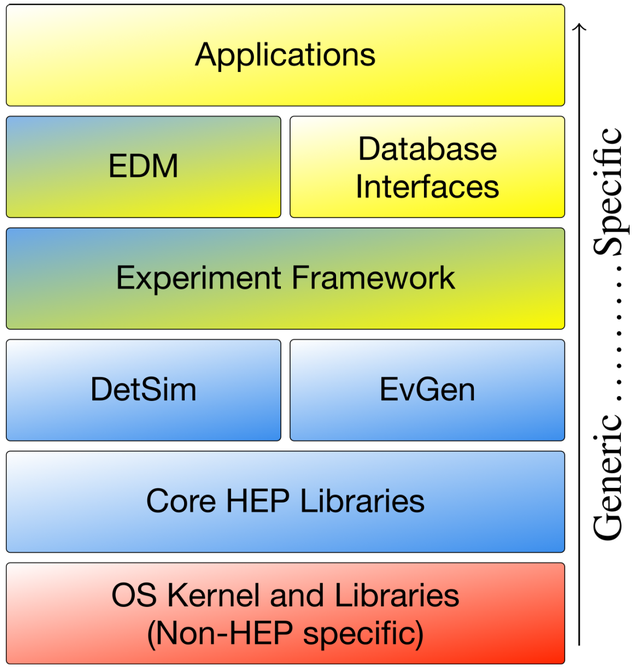
3rd ECFA workshop on e+e- Higgs, Electroweak and Top Factories

Campus des Cordeliers, Paris, FR

09-11 October 2024

Requirements on Software for Future Colliders

Provide future experiments with a ready-to-use software ecosystem supporting all required workflows



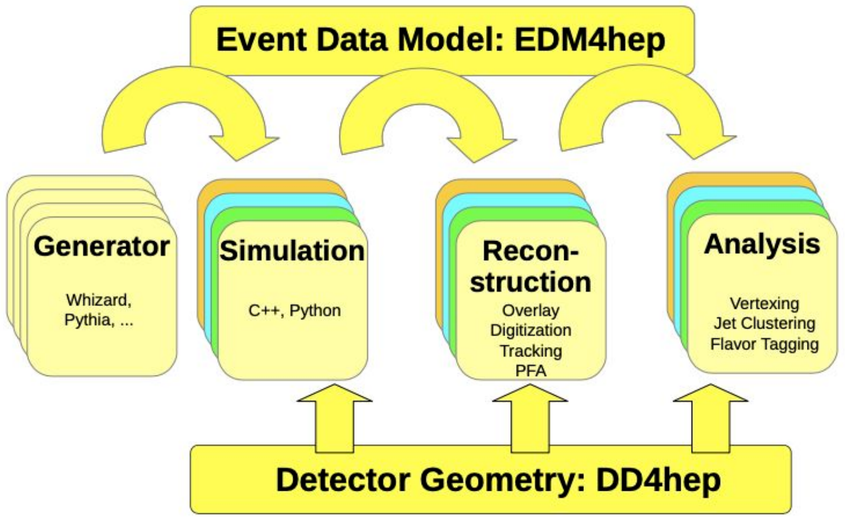
source: 10.1051/epjconf/202024510002

- Allow for quick estimations, but also detailed performance studies
- Aid in detector design and optimization
- Support interoperability among the tools
- Allow different usage modes
  - Local running of Analysis, Simulation, Reconstruction, ...
  - Bulk processing / large productions
- Encourage developments and their quick distribution

Key4hep

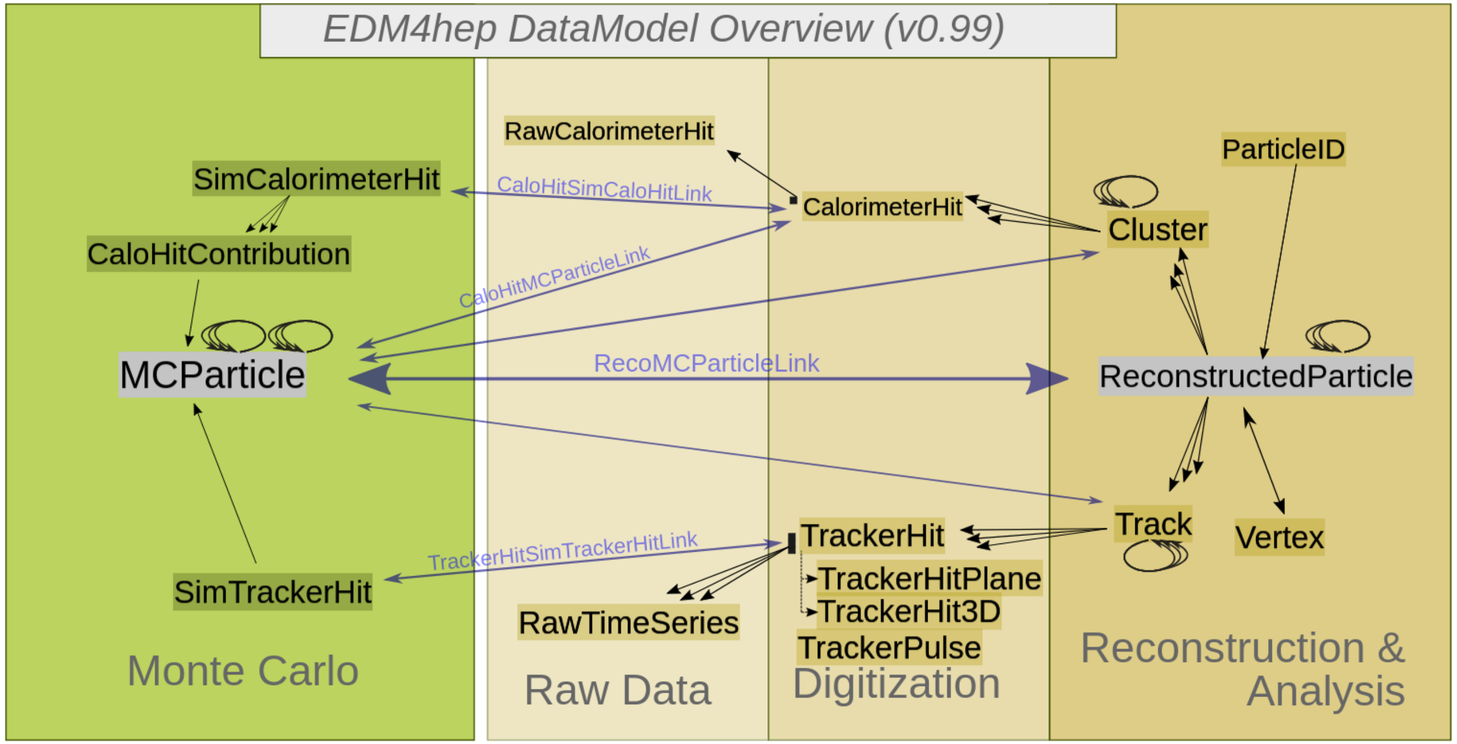
Coherent set of packages, tools, and standards for different collider Concepts

- Common effort from FCC, CLIC/ILC, EIC, CEPC, Muon Collider, ...
  - Preserves and adds onto existing functionality from iLCSoft, FCCSW, CEPCSW, ...
  - Builds on top of the experience from LHC experiments and results of targeted R&D (AIDA, ...)
  - Many institutes involved: CERN, DESY, IHEP, INFN, IJCLab, ...
- Each project rebases its stack on top of Key4hep
- Having common building blocks enables synergies across collider communities
- Main ingredients:
  - Event data model: EDM4hep, based on PODIO, AIDA project
  - Event processing framework: Gaudi, used in LHCb, ATLAS, ...
  - Detector description: DD4hep, AIDA project
  - System to build, test and deploy: Spack, suggested by HSF + CVMFS



EDM4hep

Common "language" for processing and persistifying data



- Specification in a single YAML file
  - Describes standard data structures and relations between them
- Generated by PODIO (developed as part of AIDA R&D)
- Challenge: efficiency and thread safeness

- Created by consensus
- Trade-off between being generic and preserve compactness
- First stable LTS version (v1.0) almost ready

PODIO / EDM4hep Highlights

Version v1.0 includes support for

- Schema evolution
  - Events read through reader will be updated on the fly
- Interface classes
  - Useful for point to a class of collections with common members
  - Example: `TrackerHit`
- Links/associations can created between any two collection types
- Improved support for MC event generators' information
  - Ensured mapping of hepmc to EDM4HEP
- Python / Julia* bindings
  - Enable quick analysis
- Ready to support RNTuple when released
  - New ROOT data structure expected to replace TTree soon

Interfaces example:

```yaml
interfaces:
  edm4hep::TrackerHit:
    Description: "Tracker hit interface class"
    Author: "Thomas Madlener, DESY"
    Members:
      - uint64_t cellID // ID of the sensor that created this h
      - int32_t type // type of the raw data hit
      - int32_t quality // quality bit flag of the hit
      - float time [ns] // time of the hit
      - float eDep [GeV] // energy deposited on the hit
      - float eDepError [GeV] // error measured on eDep
      - edm4hep::Vector3d position [mm] // hit position
    Types:
      - edm4hep::TrackerHit3D
      - edm4hep::TrackerHitPlane
```

PODIO quick event loop example:

```python
from podio.root_io import Reader
reader = Reader("input.edm4hep.root")
for evnt in reader.get("events"):
  hits = evnt.get("hits")
  for hit in hits:
    # ...
```

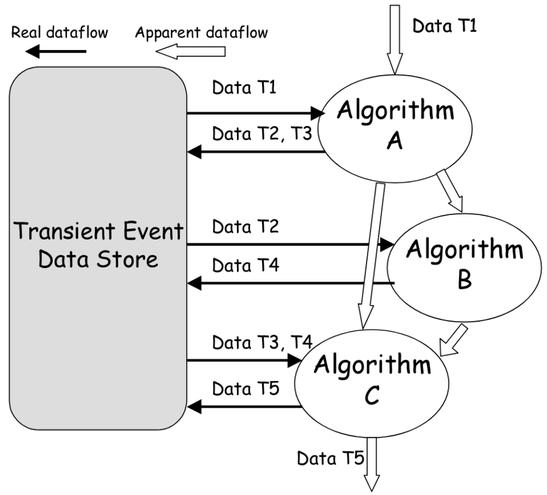
---

* Julia is a new programming language being evaluated for HEP, addressing the two language problem. As performant as C/C++ while remaining as scriptable as Python.

Gaudi and Key4hep

Gaudi is battle tested event processing framework

- Reminder about Gaudi — Event processing framework
  - Connecting and steering the work of the various algorithms together
  - Controlling event loop
  - Managing transient and persistent store (I/O)
- Meant to cover all event processing tasks
  - Supports multi-threading through Gaudi::Functional
  - Dual language: Python for configuration, C++ for algorithms
- Used by operating LHC experiments: ATLAS, LHCb, and others: Belle2, ...



Key4hep / k4FWCore

- Gaudi components are controlled through k4FWCore
  - Provides input and output file handling, but also I/O among algorithms
    `IOSvc`, `DataHandle`, `MetaDataHandle`
- External packages interfaced through dedicated converter/wrapper algorithms
  - Wrappers for MC Generators, Geant4, Delphes inherited from FCCSW
  - k4MarlinWrapper allows reuse of iLCSoft algorithms
  - Recent additions: k4CLUE, clustering algorithm developed for CMS HGCAL
  - Under development: k4GaudiPandora, k4ActsTracking, ...
- Ongoing work: Move to `Gaudi::Functional` for multithreading support

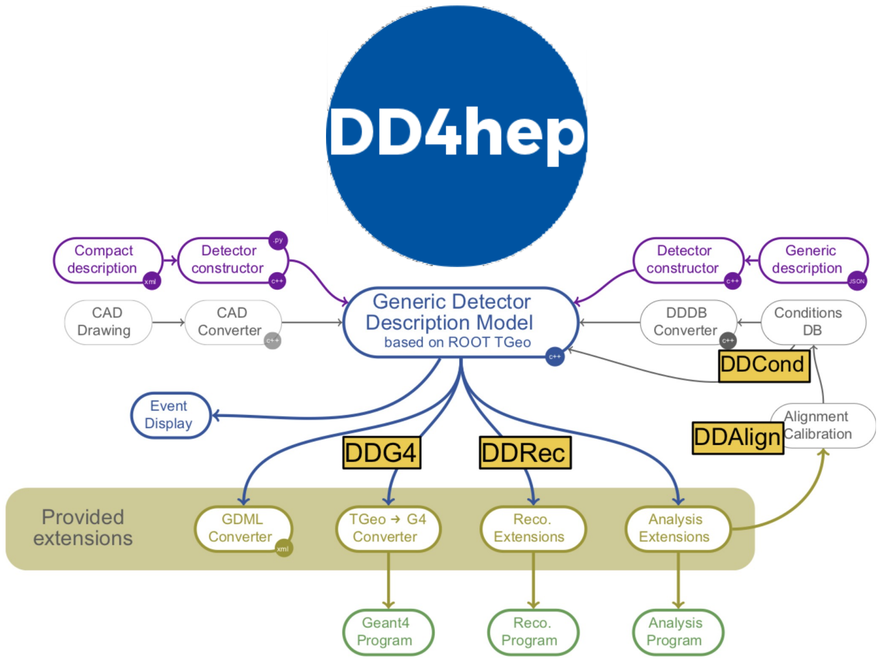
Hello World in Gaudi:

```
from Gaudi.Configuration import *
from Configurables import HelloWorldEx

alg = HelloWorldEx()

ApplicationMgr(
    EvtMax = 10,
    EvtSel = 'NONE',
    HistogramPersistency = 'NONE',
    TopAlg = [alg],
)
```

Source: Gaudi

DD4hep

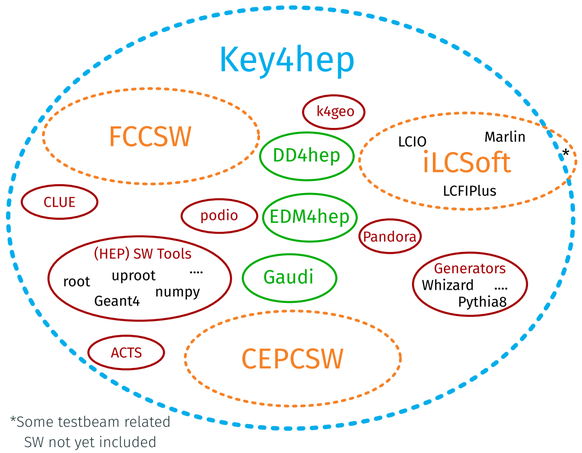
Single source for complete detector description



- Description provided through C++ drivers configured through XML compact file(s)
  - Organized in hierarchical structure, enabling Plug-and-Play
- Specialized data can be attached to each sub-detector at runtime
- Provides components to interface to Geant4 (DDG4), to reconstruction programs (DDRec), and others
- Standalone executable DDSim to steer simulation via DDG4

Build, test and deploy

- Builds and tests are managed with Spack, a package manager recommended by HSF
  - Designed and used for supercomputing centers
- Fully Python based, packages build recipe is a python script
  - No separation between main package repository and spack code
- For Key4hep, packages are registered in two repositories
  - Upstream Main Spack repository and dedicated Key4hep repository
- Compiled packages are published on CVMFS
  - More than 500 packages
  - Release: `source /cvmfs/sw.hsf.org/key4hep/setup.sh`
  - Nightlies: `source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh`



source: T. Madlener

Generation

Theoretical efforts for ee generators are ramping up

Example of k4GeneratorsConfig YAML:

```yaml
Generators:
  - Sherpa
  - Whizard
  - Madgraph
  - KKMC
  - Pythia

OutputFormat: hepmc3
OutDir: Run-Cards
Events: 10000
EventMode: unweighted

SqrtS: 91.2
Model: SM
ISRMode: 0

Processes:
  Muon:
    Final: [13, -13]
    Order: [2,0]

  Tau:
    Final: [15, -15]
    Order: [2,0]

Sherpa:
  Run:
    EW_SCHEME: 3


ParticleData:
  23:
    mass: 91.1876
    width: 2.4952
```

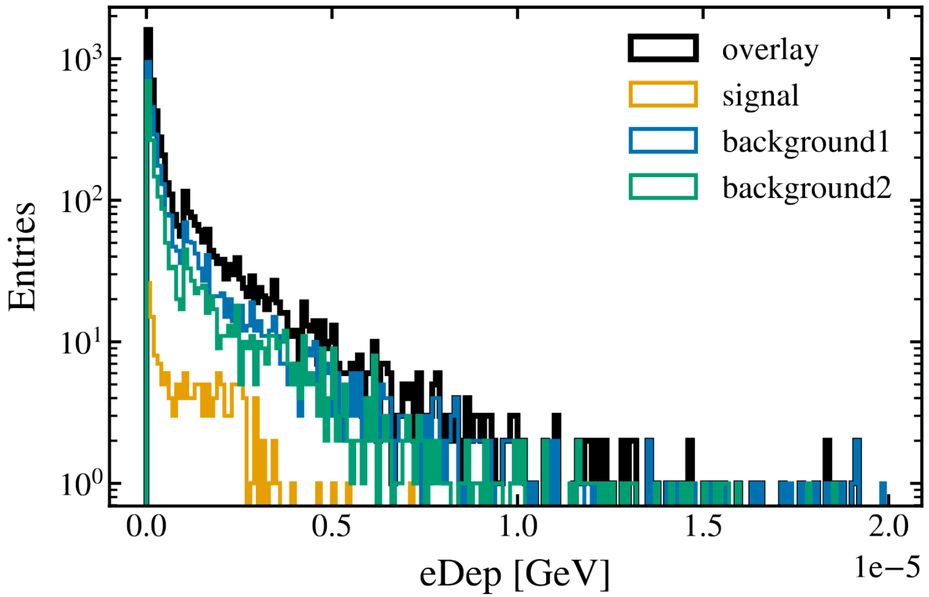
- Most of the generators already packaged in Key4hep
  - MadGraph5_aMC@NLO, Pythia6/8, Herwig3, Whizard, BabaYaga, KKMCee, Guinea-Pig, Sherpa, EvtGen, ...
- Set of Gaudi algorithms and helpers packaged in k4Gen
  - Particle gun, particle filters, vertex smearing, ...
- New effort for unified generator configuration packaged in k4GeneratorsConfig
  - Integrated: BabaYaga, KKMC, MadGraph, Pythia, Sherpa, Whizard
  - Users write one YAML file and datacards are generated for each generator
  - A script to run the generation step is provided
    - Runs the generator (output: hepmc{2,3}, LHEF) and converts to EDM4HEP afterwards
  - Packaged in Key4hep stack
  - More details in A. Price's talk on Thursday
- Preferred formats: HepMC3 and EDM4hep
  - EDM4hep is now more suitable for generators
- Events can be filtered based on the JIT compiled rules acting on the MC particle tree

More details about generators itself in talk from C. Calame on Friday

Simulation

- Parametrized (Delphes) simulation integrated in Key4hep via k4SimDelphes package
  - Particle identification: Time-of-flight, cluster counting
  - FastJet integration with the e+e- clustering algorithms
- Full simulation done using `DDSim` (part of DD4hep)
  - Takes any established MC generator file format (HepMC{2,3}, hepevt, stdhep, ...)
- Integration of Geant4 with event processing framework k4SimGeant4 and `Gaussino` on back burner
  - Approaches of ATLAS/LHCb
- ILC/CLIC/FCC-ee detector descriptions collected in k4geo
- Ongoing work on detector description of the three FCC-ee detector concepts IDEA, CLD and ALLEGRO almost complete
  - Effort now shifting from detector description towards Digitization and Reconstruction
  - And comparisons between Full and Parametrized simulation
- Background Overlay Algorithm combines collections from signal and background events
  - MCParticles, SimTrackerHits, SimCalorimeterHits

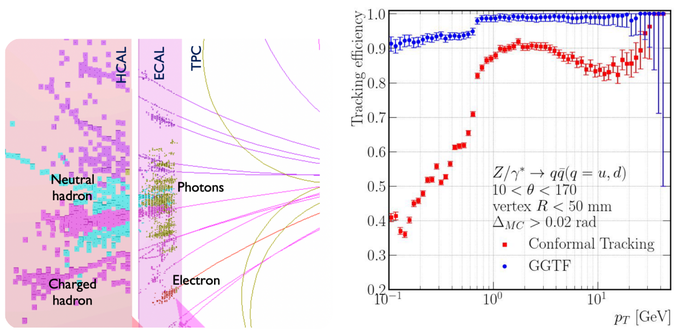
Overlay of SimTrackerHits:



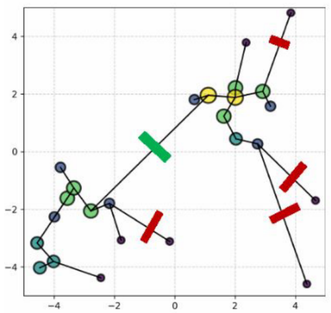
More details about Simulation in A. Delgado's talk on Thursday

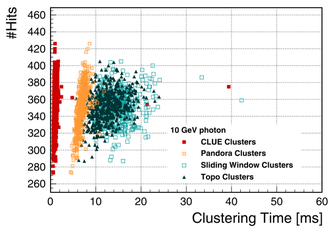
Reconstruction

Work in full swing on integration of multitude of reconstruction solutions



Pandora illustration and ML track reconstruction in CLD





CLUE clustering and its performance

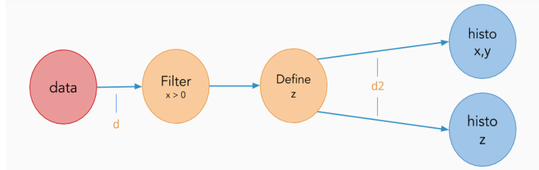
- Efforts are packaged per sub-detector type, for example
  - kRecCalorimeter: Reconstruction of Noble Liquid based calorimeter
  - k4RecTracker: vertex and tracker reconstruction as well as tracking
  - kReco: Common Gaudi native reconstruction algorithms
- Or per reconstruction solution, e.g.
  - k4GaudiPandora: Direct wrapping of Pandora in Gaudi
    - More details in S. Sasikumar's talk on Thursday
  - k4Clue: Clustering algorithm from HGCAL
- Ongoing efforts include
  - Machine learning based track reconstruction
    - More details in D. Garcia's talk on Thursday
  - Integration of ACTS tracking into Key4hep
  - Particle identification with Array of RICH Cells (ARC)

More details in F. Gaede's talk on Thursday

Analysis

PODIO and ROOT DataFrame got closer

- Simple C++/Python analysis by reading ROOT/SIO files through PODIO Reader
- Python bindings of PODIO through ROOT's cppyy
- Julia has standalone EDM4hep ROOT files reader
- Podio::DataSource now allows to work with full fledged EDM4hep objects in RDataFrame
- Set of level functions under development
  - Plotting/printing kinematic variables, sorting, ...
- Analysis framework FCCAnalyses offers:
  - Higher level analyzer functions/functors
  - Management of input samples
  - Running of the dataframe: locally or on HTCondor
  - Analysis Catalogs:
    - FCCeePhysicsPerformance
    - FCChhPhysicsPerformance




```
// d2 is a new data-frame, a transformed version of d
auto d2 = d.Filter("x > 0")
           .Define("z","x*x + y*y");
// make histograms out of it
auto hz = d2.Histo1D("z");
auto hxy = d2.Histo2D({"hxy","hxy",16,-1,1,64,-1,1},"x","y");
```

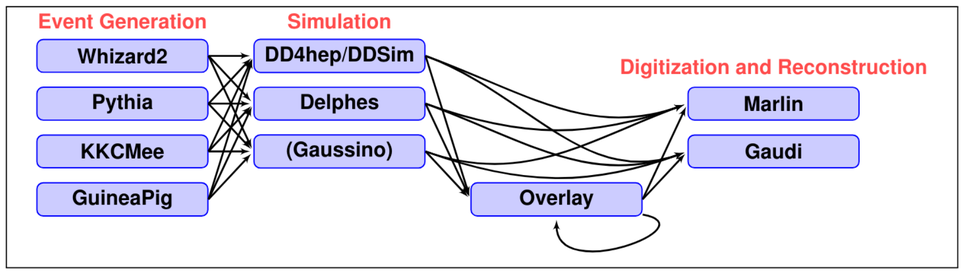
Case studies (evolving list)

1. Electroweak physics at the Z peak
2. Tau Physics
3. Flavour physics
4. WW threshold
5. QCD measurements
6. Higgs physics
7. Top physics
8. Direct searches for new physics

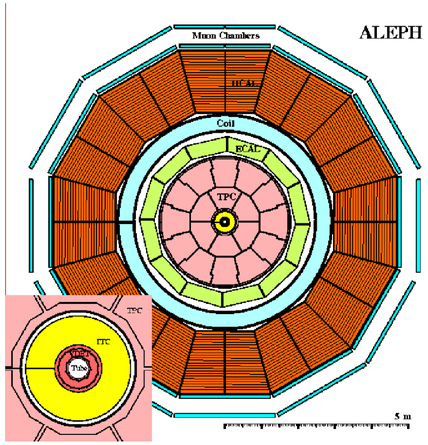
Centralized Productions and LEP data

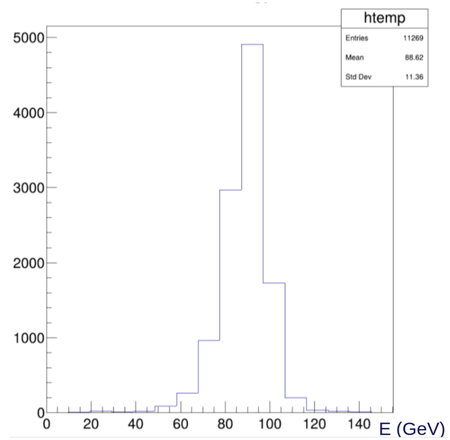
iLCDirac

- DIRAC* extension for future lepton colliders
- ILC, CALICE and FCC VO (virtual organization)
- New workflow modules
  - Monte Carlo generators, Delphes param. simulation
- Config file interface for FCC production managers
- FCC metadata agent
- First FCC-ee Full Sim productions launched on the GRID



More details in A. Sailer's talk on Thursday





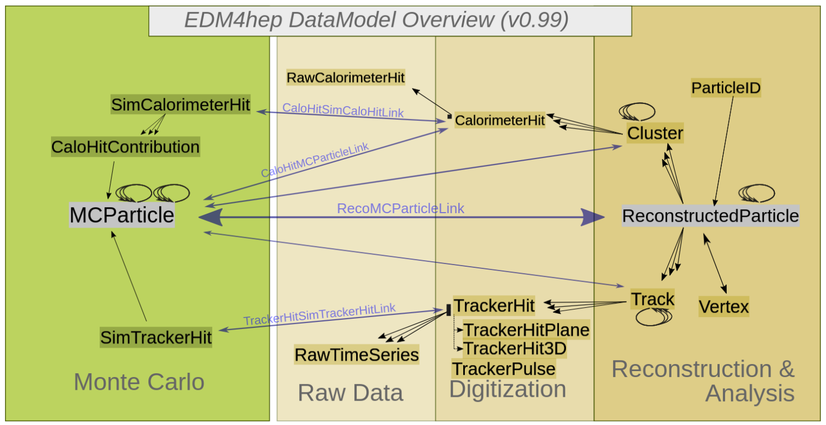
ALEPH data in EDM4hep

- Data from LEP experiments still preserved but difficult to work with
- Opportunity to train, develop and validate algorithms on real data
- Test of EDM4hep itself
- Conversion and validation chains put in place

---

* DIRAC is an interware for distributed computing on the GRID

Conclusions



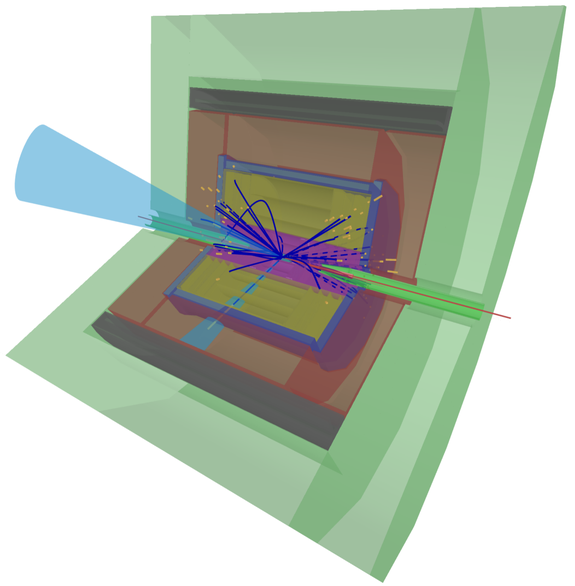
- Key4hep stack project is becoming established and is delivering results
  - Many case studies, detector performance investigations, ...
- Quest for integration and interoperability continues
- EDM4hep datamodel is becoming mature — version 1.0 is very close
- Plenty of exciting work to be done in Simulation, Reconstruction and Analysis tools
  - Join us: Indico category
- ECFA Report: Software Ecosystem will be edited by Andre Sailer, Frank Gaede and Gerardo Ganis

Backup

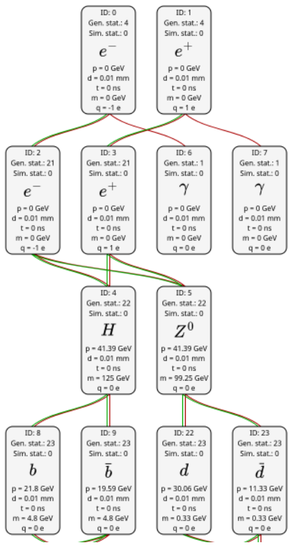
Visualization

Movement towards web based visualization

- Plenty of native solutions available:
  - CED, geoDisplay, Geant4 Qt Visualization, ...
- For detector geometry JSROOT works well
  - One needs to convert compact file to ROOT file
- Event display with Phoenix implemented in:
  - Phoenix-ILD
  - Phoenix@FCC
  - Muon Collider
- Explorer of the event contents: eede
  - Allows to browse MC particle tree, Reco particle, hits and clusters, ...

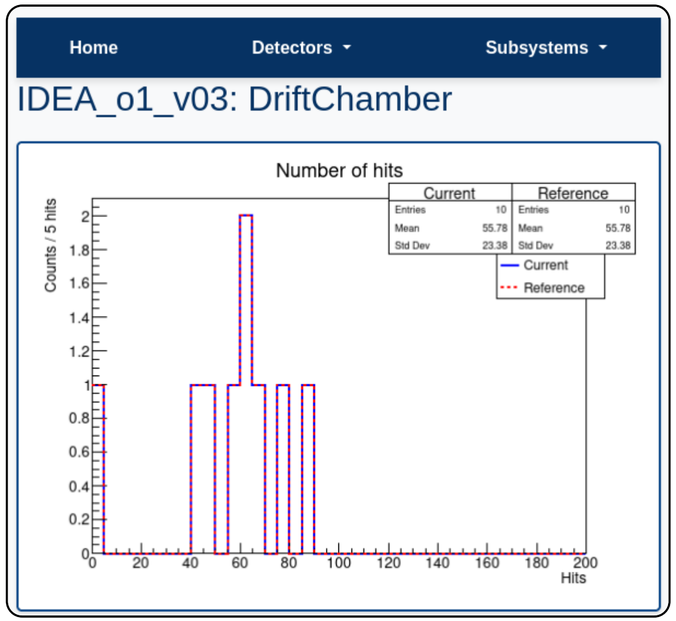


ttbar event in CLD



Pythia 8 | ee → ZH @ 240 GeV

Validation and Development



key4hep-validation.web.cern.ch

Key4hep Validation

- Validation of Simulation and Reconstruction with centralized tests
- Comparison of the nightlies stack agains a reference every night
- Reports on failure
- Detectors implemented:
  - `CLD_o3_v01`, `IDEA_o1_v03`, `ALLEGRO_o1_v03`

General steps for Key4hep package development

- Source Key4hep stack from CVMFS

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh
# or
source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
```

  - Use `-r` parameter to find out available stacks
- Clone and build package locally according to the instructions
  - Usually done with the help of CMake
- Activate your local version

```
k4_local_repo
```

- Documentation of Key4hep and its components is growing
  - Links also to FCC Software, iLCSoft, CEPCSW