Key4hep & EDM4hep

A small introduction

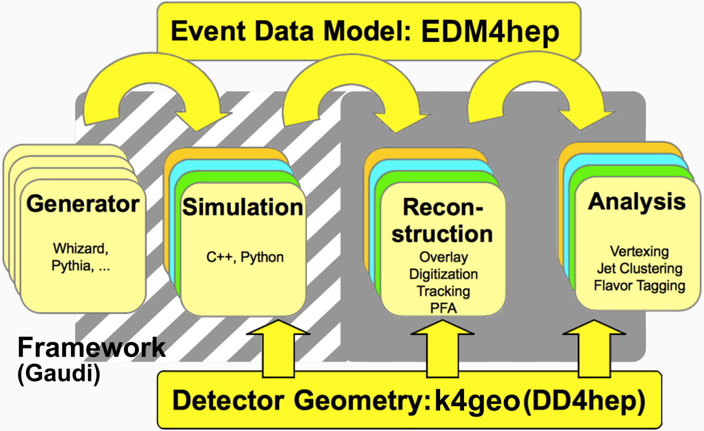
Thomas Madlener

FH Future Collider Day & SciComp Workshop

Dec 5, 2024

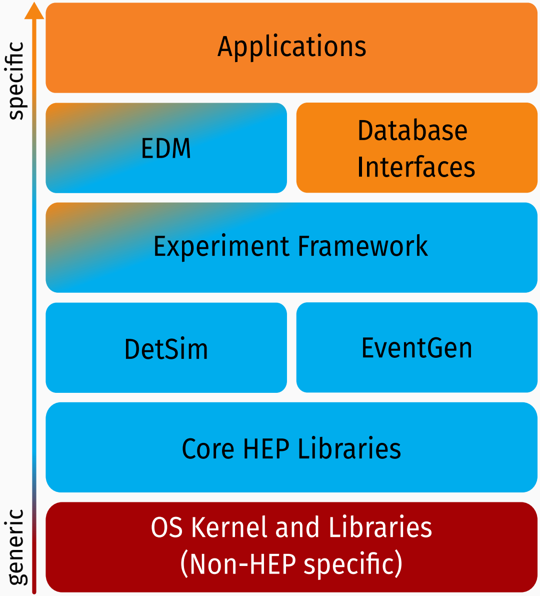
This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

From generation to analysis - the general workflow



- Many steps involved from generating events to analyzing them
- Hundreds of SW packages
  - Building & deploying
  - Consistency
  - Reproducibility
- Key4hep aims at facilitating interoperability and focuses on common approaches

HEP Software Stack



specific

generic

Applications

EDM

Database Interfaces

Experiment Framework

DetSim

EventGen

Core HEP Libraries

OS Kernel and Libraries (Non-HEP specific)

Dedicated algorithms (reconstruction, analysis, …)

Data access & representation

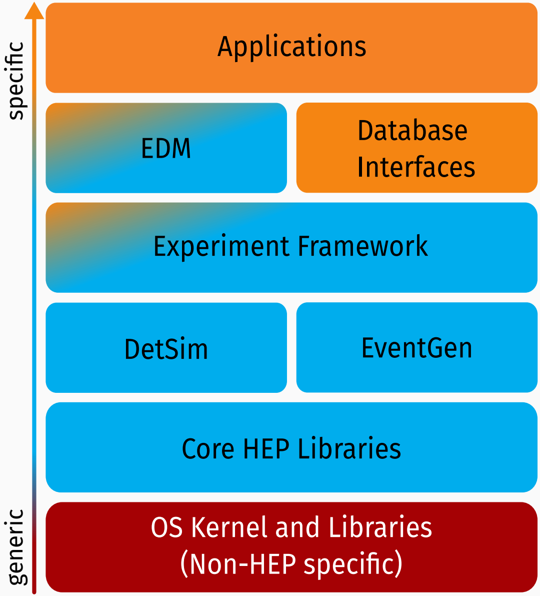
Algorithm / workflow orchestration

Specific components reused by many experiments (DD4hep, Delphes, Pythia, …)

Commonly used HEP core libraries (ROOT, Geant4, CLHEP, …)

Python, Compilers, CMake, boost, …

HEP Software Stack



- Pieces of software are not living in isolation
- Ecosystem of interacting components
- Compatibility between different elements doesn’t come for free
  - Common standards can help a lot
- Building a consistent stack of software for an experiment is highly non-trivial
  - Benefits can be gained from using common approaches

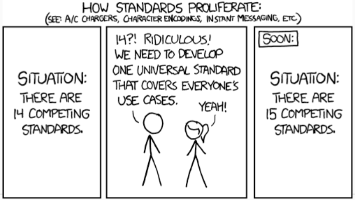
Key4hep goals

- Provide and maintain a consistent SW stack that allows to do physics studies for **all projects**
- Ensure interoperability of the necessary building blocks
- Reuse existing solutions where possible
  - A lot of experience from LHC experiments and LC communities
- Focus new developments on EW/Higgs factory specifics
- Share knowledge, processes, workflows and resources
  - Best practices, tutorials, documentation, …

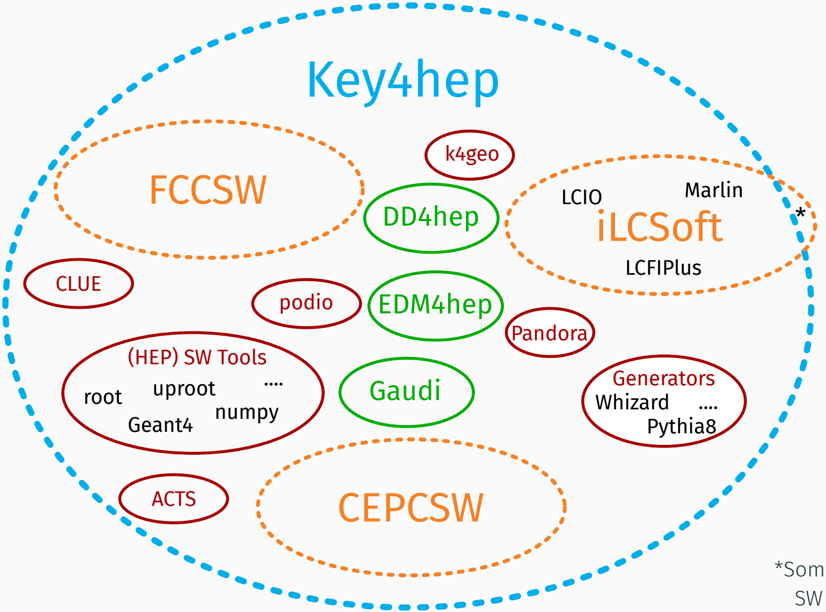
Photo by Stewart B. / CC-BY

Non-goal

- Develop and maintain project specific software and workflows


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.
xkcd.com/927

Key4hep (simplified) overview



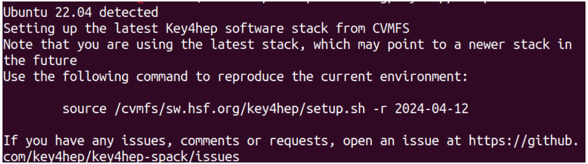
*Some testbeam related SW not yet included

Keyhep releases and nightlies

- (Rolling) latest release of the complete Key4hep software stack
  - Full stacks for AlmaLinux9, Ubuntu22.04

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

- Documentation
  - key4hep.github.io/key4hep-doc
  - Includes tutorials & How-tos
- **Release early and release often**
  - Make fixes available early
  - Discover problems and collect feedback as early as possible
- Biweekly, alternating meetings for Key4hep & EDM4hep
  - indico.cern.ch/category/11461/



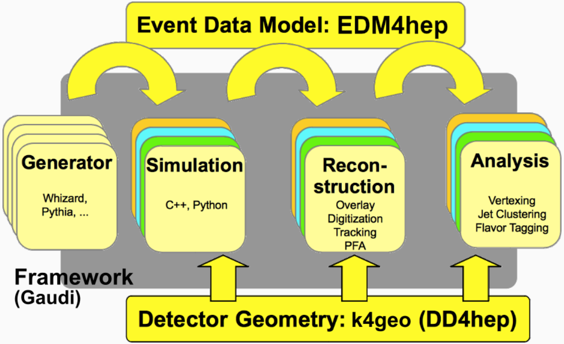
```
Ubuntu 22.04 detected
Setting up the latest Key4hep software stack from CVMFS
Note that you are using the latest stack, which may point to a newer stack in
the future
Use the following command to reproduce the current environment:

        source /cvmfs/sw.hsf.org/key4hep/setup.sh -r 2024-04-12

If you have any issues, comments or requests, open an issue at https://github.
com/key4hep/key4hep-spack/issues
```

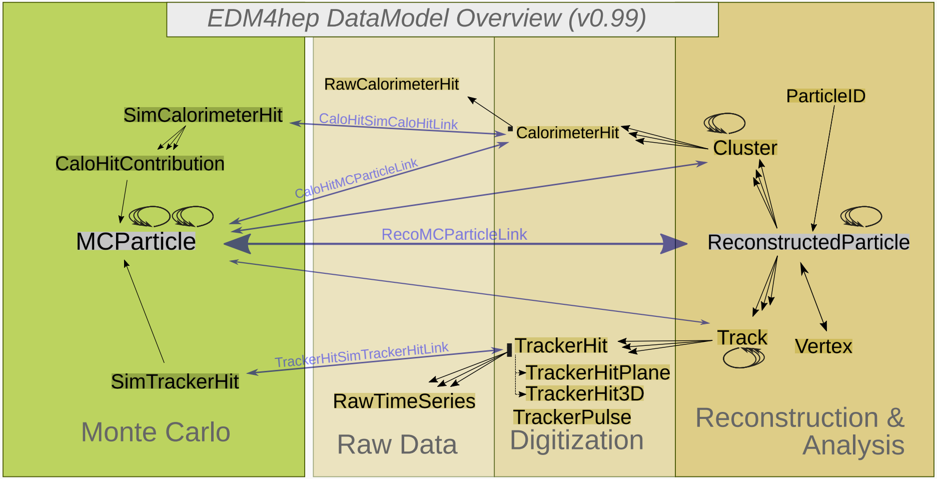
The EDM at the core of HEP software

- Key4hep aims to provide a common SW stack for future collider projects



- Different components of experiment software have to exchange data
- The event data model defines structure and language - also for users

EDM4hep - The EDM for Key4hep



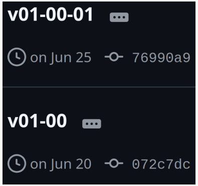
- Heavily inspired by *LCIO* and *FCC-edm*
- Focus on usability in reconstruction and analysis

key4hep/EDM4hep

edm4hep.web.cern.ch

The podio EDM toolkit

- Implementing a performant event data model (EDM) is non-trivial
- Use `podio` to generate code starting from a high level description
- Provide an easy to use interface to the users
- **v1.0 available!** 🎉

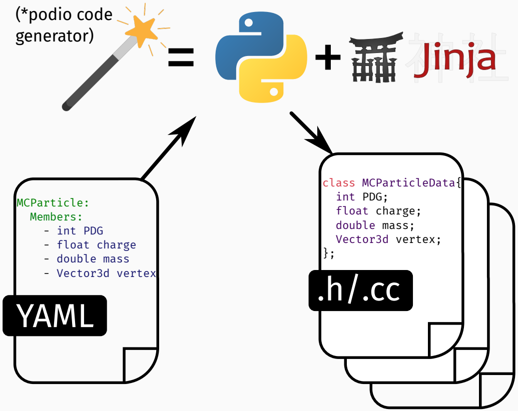


v01-00-01

on Jun 25 76990a9

v01-00

on Jun 20 072c7dc



```
MCParticle:
  Members:
    - int PDG
    - float charge
    - double mass
    - Vector3d vertex
```

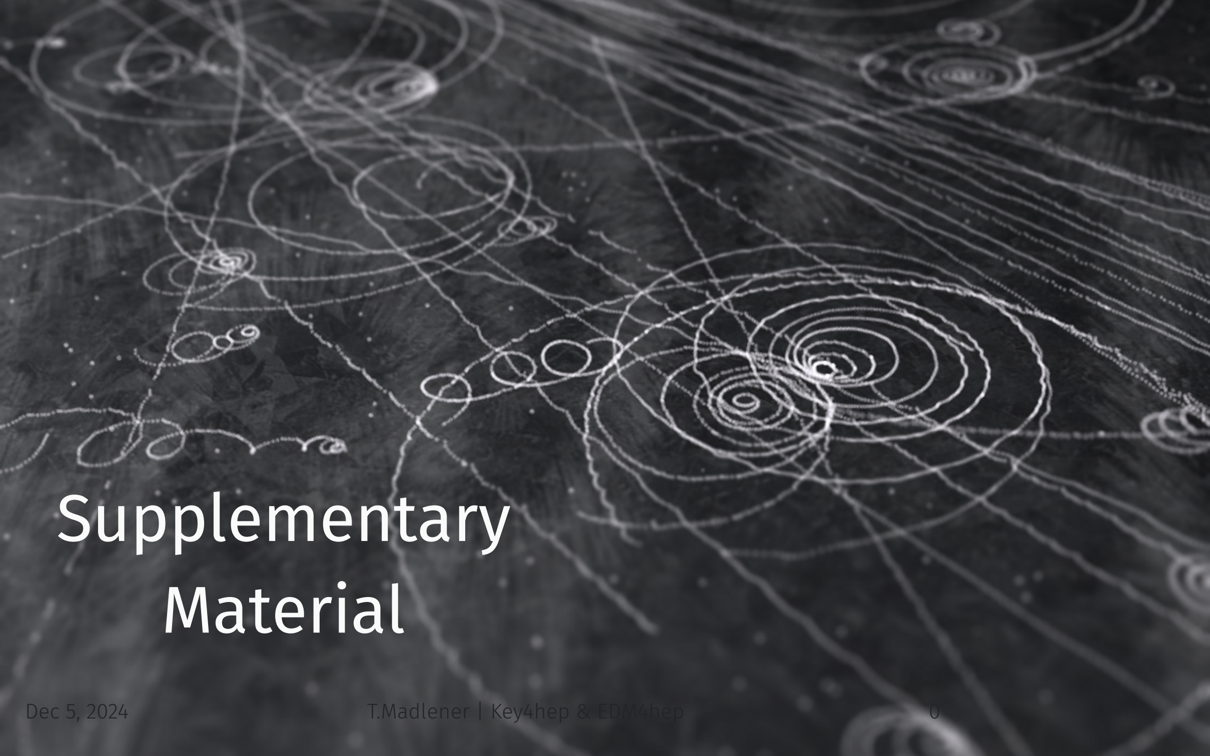
```
class MCParticleData{
  int PDG;
  float charge;
  double mass;
  Vector3d vertex;
};
```

AIDASoft/podio

key4hep.web.cern.ch/podio

`edm4hep::MCParticle` definition in YAML file (abbr.)

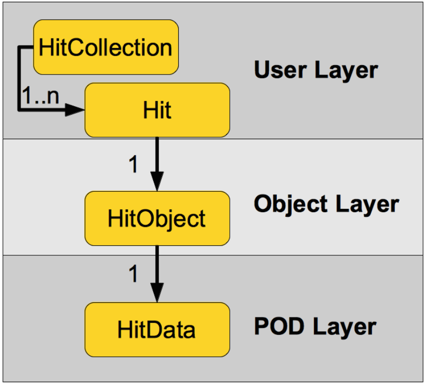
```yaml
edm4hep::MCParticle:
  Members:
    - int32_t PDG                                 // PDG code of the particle
    - int32_t generatorStatus                     // status of the particle as defined by the generator
    - int32_t simulatorStatus                     // status of the particle from the simulation program - use BIT constants below
    - float charge                                // particle charge
    - float time [ns]                             // creation time of the particle wrt. event, (pre-assigned decays, or decays in flight)
    - double mass [GeV]                           // mass of the particle
    - edm4hep::Vector3d vertex [mm]               // production vertex of the particle
    - edm4hep::Vector3d endpoint [mm]             // endpoint of the particle
    - edm4hep::Vector3d momentum [GeV]            // particle 3-momentum at the production vertex
    - edm4hep::Vector3d momentumAtEndpoint [GeV]  // particle 3-momentum at the endpoint
    - edm4hep::Vector3f spin                      // spin (helicity) vector of the particle
    - edm4hep::Vector2i colorFlow                 // color flow as defined by the generator
  OneToManyRelations:
    - edm4hep::MCParticle parents                 // The parents of this particle
    - edm4hep::MCParticle daughters               // The daughters this particle
  ExtraCode:
    declaration: "
      // define the bit positions for the simulation flag\n
      static const int BITCreatedInSimulation = 30;\n
      static const int BITBackscatter = 29 ;\n
      static const int BITVertexIsNotEndpointOfParent = 28 ;  \n
      static const int BITDecayedInTracker = 27 ; \n
      static const int BITDecayedInCalorimeter = 26 ;   \n
      static const int BITLeftDetector = 25 ;     \n
      static const int BITStopped = 24 ;    \n
      static const int BITOverlay = 23 ;    \n"
```



Supplementary Material

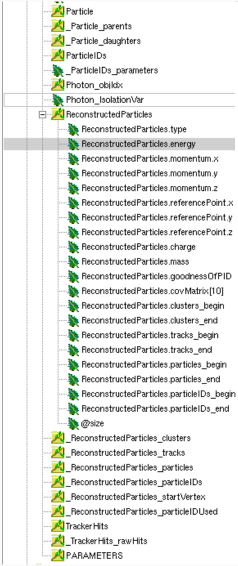
The three layers of podio

- podio favors **composition over inheritance** and uses **plain-old-data (POD)** types wherever possible
- Layered design allows for efficient memory layout and performant I/O implementation

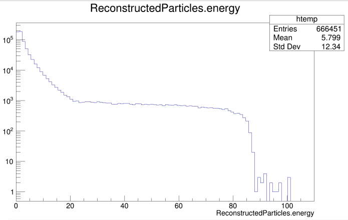


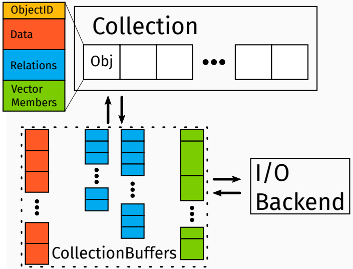
podio supports different I/O backends

- Default **ROOT** backend
  - Effectively flat `TTree` / `RNTuple`
  - Files can be interpreted **without EDM library**(!)
  - Can be used in `RDataFrame` (**FCCAnalyses**) or with `uproot`
  - Also with Julia
- Adding more I/O backends is possible
  - Alternative SIO backend exists
  - Working on `RDataSource` for better RDataFrame integration
- Generated interfaces provide many “convenience features”



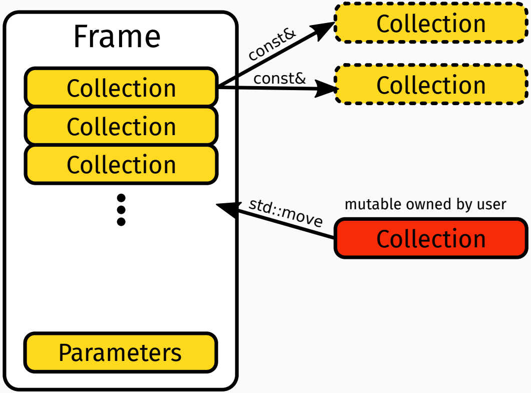
```
Particle
_Particle_parents
_Particle_daughters
ParticleIDs
_ParticleIDs_parameters
Photon_objIdx
Photon_IsolationVar
ReconstructedParticles
    ReconstructedParticles.type
    ReconstructedParticles.energy
    ReconstructedParticles.momentum.x
    ReconstructedParticles.momentum.y
    ReconstructedParticles.momentum.z
    ReconstructedParticles.referencePoint.x
    ReconstructedParticles.referencePoint.y
    ReconstructedParticles.referencePoint.z
    ReconstructedParticles.charge
    ReconstructedParticles.mass
    ReconstructedParticles.goodnessOfPID
    ReconstructedParticles.covMatrix[10]
    ReconstructedParticles.clusters_begin
    ReconstructedParticles.clusters_end
    ReconstructedParticles.tracks_begin
    ReconstructedParticles.tracks_end
    ReconstructedParticles.particles_begin
    ReconstructedParticles.particles_end
    ReconstructedParticles.particleIDs_begin
    ReconstructedParticles.particleIDs_end
    @size
_ReconstructedParticles_clusters
_ReconstructedParticles_tracks
_ReconstructedParticles_particles
_ReconstructedParticles_particleIDs
_ReconstructedParticles_startVertex
_ReconstructedParticles_particleIDUsed
TrackerHits
_TrackerHits_rawHits
PARAMETERS
```





The `Frame` - A generalized (event) data container

- *Type erased* container aggregating all relevant data
- Defines an *interval of validity* / category for contained data
    - Event, Run, readout frame, ...
- Easy to use and thread safe interface for data access
    - Immutable read access only
    - Ownership model reflected in API
- Decouples I/O from operating on the data



```
template<typename CollT>
const CollT& get(const std::string& name) const;

template<typename CollT, /*enable_if*/>
const CollT& put(CollT&& collection,
                 const std::string& name);
```

Spack for Key4hep

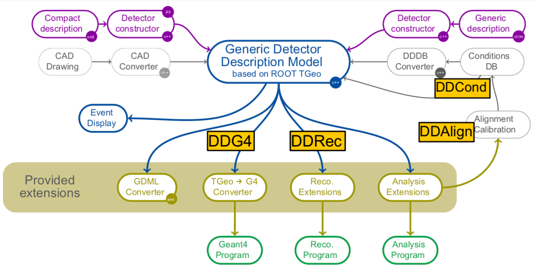
- Spack is a package manager
  - Independent of operating system
  - Builds all packages from source
- Originaly developed by the HPC community
  - Emphasis on dealing with **multiple configurations** of the same package
- Basic building block is a formalized build procedure → **spack recipe**
  - Build instructions, dependencies, versions and location of source code
  - ∼ 8000 packages currently available from spack
  - Many Key4hep packages in key4hep/key4hep-spack
- The whole Key4hep software stack can be built from scratch using spack

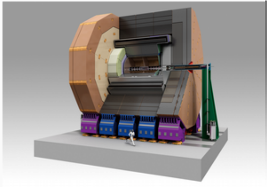
```
spack install key4hep-stack
```

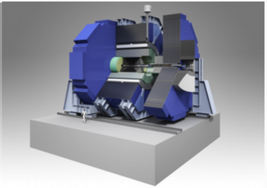
DD4hep - Detector description

dd4hep.web.cern.ch

- Complete detector description
  - Geometry, materials, visualization, readout, alignment, calibration, ...
- From a **single source of information**
  - Simulation, reconstruction, analysis
- Comes with a powerful plug-in mechanism that allows customization
- More or less “industry standard” now
  - FCC, ILC, CLIC, EIC, LHCb, CMS, ODD, ...
- `ddsim` - standalone simulation executable

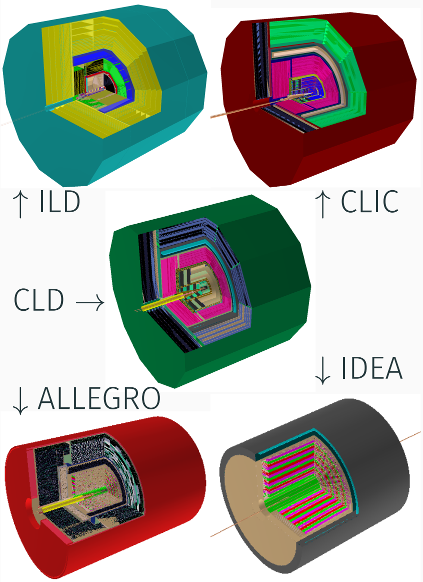






k4geo - The detector geometry repository

- Central repository for detector models
- Many existing detector models from LC studies
- Many recent developments for FCC detector concepts
- “Plug and play” approach for subdetectors
    - Use CLD inner tracker in ILD for TPC studies at FCC


↑ ILD
↑ CLIC
CLD →
↓ IDEA
↓ ALLEGRO