Key4hep - One software to rule them all

4th Future Colliders @ DESY meeting

Thomas Madlener

Oct 21, 2021

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Key4hep - One software to rule them all (... soon)

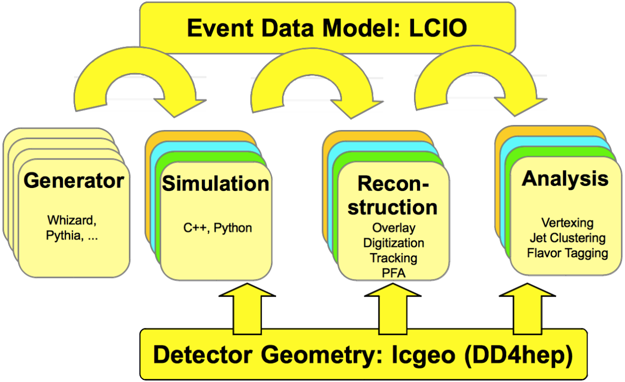
4th Future Colliders @ DESY meeting

Thomas Madlener

Oct 21, 2021

The general workflow (for future colliders)

From generation to analysis

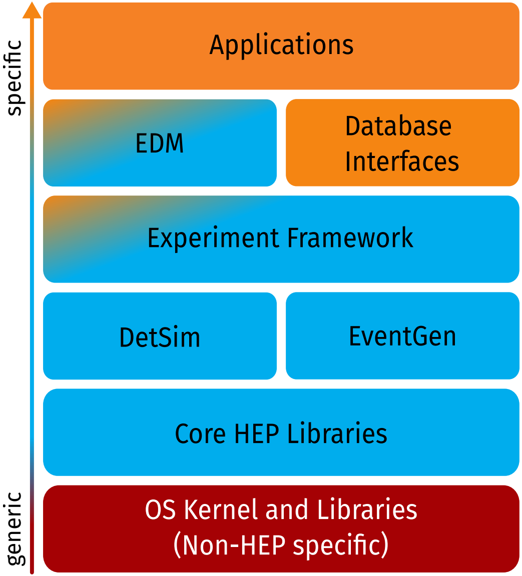


*example from iLCSoft

- Many steps involved from generating events to analyzing them
- This talk: Focus on what happens after events have been generated
- Give an introduction to Key4hep and the tools that are available

HEP Software Stack

Libraries all the way down...



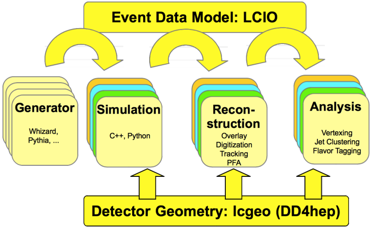
- Pieces of software are not living in isolation
- Ecosystem of interacting components
- Compatibility between different elements doesn’t come for free
  - Common standards can help a lot
- Building a consistent stack of software for an experiment is highly non-trivial
  - Benefits can be gained from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

Key4hep Motivation

- Future detector studies rely on well maintained software to properly study possible detector concepts and their physics reach and limitations
- Existing scattered landscape of HEP software
  - Dedicated tools for specific tasks
  - Integrated frameworks tailored to specific experiments
- Aim for a low maintenance common stack for future collider projects with ready to use “plug-ins” to develop detector concepts
- A consensus to develop such a common software stack has been reached among all communities for future colliders in the “2019 Bologna Future Collider Software Workshop”
- **Regular contributions from FCC, ILC, CLIC, CEPC, …**
- Part of the CERN Strategic R&D Programme on Technologies for Future Experiments

Key4hep Goals

- Connect and extend individual packages towards a complete data processing framework
  - Convert a set of disconnected packages into a **turnkey** system
  - Share as many components as possible to reduce overhead for all users
- Re-use existing tools as much as possible
  - e.g. from ILC/CLIC and FCC studies
- Easy to use for librarians, developers and users
  - Easy to deploy (e.g. CVMFS, containers)
  - Easy to set up
  - Easy to extend
- Provide full functionality for different use cases
- Provide examples and documentation for simulation, reconstruction, …



Event Data Model: LCIO

Generator
Whizard, Pythia, ...

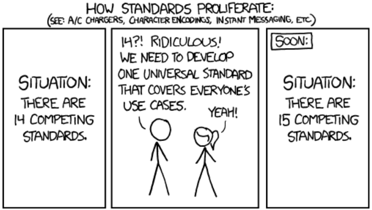
Simulation
C++, Python

Recon-struction
Overlay
Digitization
Tracking
PFA

Analysis
Vertexing
Jet Clustering
Flavor Tagging

Detector Geometry: lcgeo (DD4hep)

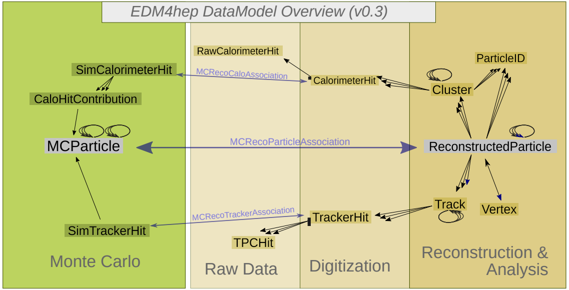
iLCSoft components here, but general scheme applies


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

EDM4hep

The EDM for Key4hep



key4hep/EDM4Hep

edm4hep.web.cern.ch

- To facilitate interoperability, different components should talk the same language
- In HEP this is the **Event Data Model**
  - Describes the structure of HEP Data
- Defining an EDM is not entirely trivial
  - Is it possible that lepton and hadron colliders share an EDM?
- **Heavily inspired by `LCIO`**
  - `LCIO` has been very succesfully shared by CLIC and ILC
- EDM4hep can be very easily analysed with ROOT
- Ongoing work, first version already used for physics studies

Experiment Framework

Conducting all the different pieces

- Traditionally HEP has not done too well with sharing efforts towards a common experiment framework
  - Notable exception is `Marlin` used by ILC and CLIC
- `Gaudi`, originally developed by LHCb, now also used by ATLAS, FCCSW and smaller experiments
  - Supports concurrency
  - “Battle-proven” from data taking during LHC operations
- Key4hep has decided to adapt `Gaudi` as its experiment framework
  - Contribute to its development where necessary
- Integration and migration of iLCSoft algorithms into Key4hep with the help of a `Marlin`→`Gaudi` wrapper
  - Allows to use `Marlin` processors within the `Gaudi` framework
  - key4hep/k4MarlinWrapper

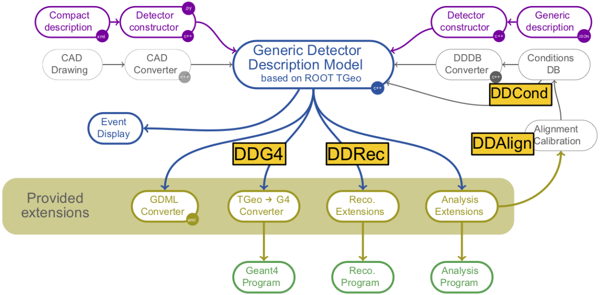
DD4hep

Detector Description Toolkit for HEP

AIDASoft/DD4hep

dd4hep.web.cern.ch

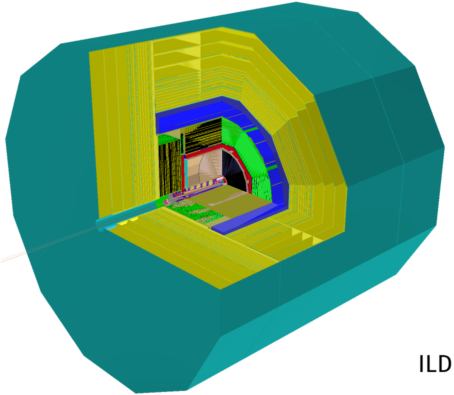
- Originally developed for ILC and CLIC but with all of HEP in mind
- Provides a complete detector description
  - Geometry, materials, visualization, readout, alignment, calibration, ...
- From a **single source of information**
  - Used in simulation, reconstruction, analysis
- Comes with a powerful plug-in mechanism that allows customization
- More or less “industry standard” now
  - ILC, CLIC, FCC, CEPC, LHCb, ...
  - CMS is switching to DD4hep



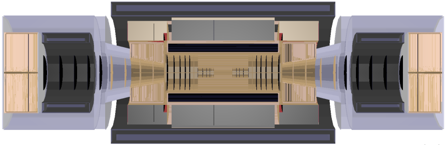
Available detector models

… and how to use them

- Detector concepts for FCC
  HEP-FCC/FCCDetectors
- Detector models for linear colliders
  iLCSoft/lcgeo
- `ddsim` is a standalone executable for running Geant4 simulations with DD4hep detector models
  - Can create outputs in LCIO or EDM4hep format
  - **Used in production for ILD studies!**
- Integration into Gaudi based Key4hep framework is ongoing



ILD



FCC-hh

Key4HEP Framework

Organization of packages

- `k4FWCore` key4hep/k4FWCore
  - Core Key4hep framework providing core functionality, e.g.
    - Data Service for EDM4hep inputs
    - Overlay for backgrounds
- `k4SimDelphes` for Delphes fast simulation key4hep/k4SimDelphes
- `k4MarlinWrapper` using Marlin processors in Gaudi key4hep/k4MarlinWrapper
- Migration of software from FCCSW to Key4hep ongoing
  - `k4SimGeant4` for Geant4 simulation integration HEP-FCC/k4SimGeant4
  - `k4Gen` for generic generator interface HEP-FCC/k4Gen
  - ...
- Ongoing work to collaborate more with Gaudi ecosystem (Gaussino)
- Ongoing work to integrate more components
  - ACTS tracking framework acts-project/acts
  - CLUE fast clustering algorithms .cern.ch/kalos/CLUE

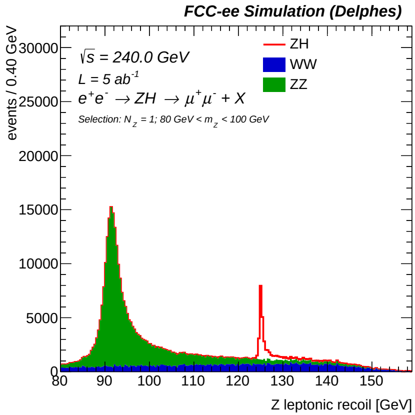
k4SimDelphes

First steps towards physics

key4hep/k4SimDelphes

DELPHES
fast simulation

- `k4SimDelphes` uses the Delphes fast simulation toolkit
- Creates output files in EDM4hep format
- **Quick way to get your hands dirty and do some physics with EDM4hep**
- Available as standalone executables
  - E.g. `DelphesPythia8_EDM4HEP`, `DelphesSTDHEP_EDM4HEP`, ...
- Prototype integration into Key4hep framework
- Use your favorite Delphes detector card unchanged
- Output configurable separately



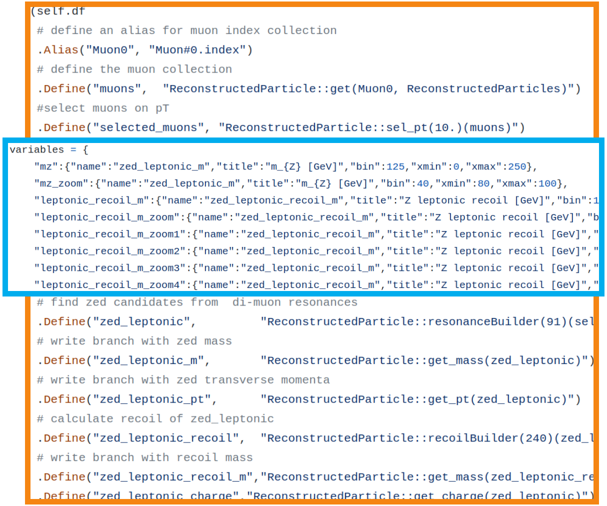
courtesy of C. Helsens

FCCAnalyses

HEP-FCC/FCCAnalyses

Analyze EDM4hep with RDataFrame

- `FCCAnalyses` is a python analysis framework based on `RDataFrame`
  - Comes with high level reco functionality
  - Extensible via C++
- **Not specific to FCC!** Can be run with EDM4hep inputs
- Declarative style of the analysis
  - Describe what you want
  - Framework will deal with the how
- More details here, on hep-fcc.github.io/fcc-tutorials or in this presentation



```
(self.df
 # define an alias for muon index collection
 .Alias("Muon0", "Muon#0.index")
 # define the muon collection
 .Define("muons",  "ReconstructedParticle::get(Muon0, ReconstructedParticles)")
 #select muons on pT
 .Define("selected_muons", "ReconstructedParticle::sel_pt(10.)(muons)")
```

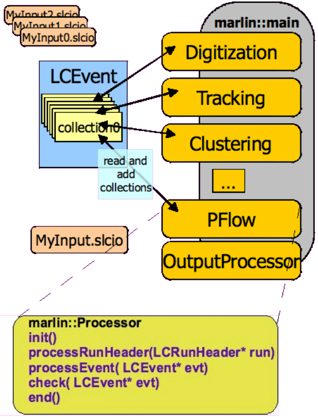
```
variables = {
    "mz":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":125,"xmin":0,"xmax":250},
    "mz_zoom":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":40,"xmin":80,"xmax":100},
    "leptonic_recoil_m":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","bin":1
    "leptonic_recoil_m_zoom":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","b
    "leptonic_recoil_m_zoom1":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","
    "leptonic_recoil_m_zoom2":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","
    "leptonic_recoil_m_zoom3":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","
    "leptonic_recoil_m_zoom4":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","
```

```
 # find zed candidates from  di-muon resonances
 .Define("zed_leptonic",         "ReconstructedParticle::resonanceBuilder(91)(sel
 # write branch with zed mass
 .Define("zed_leptonic_m",       "ReconstructedParticle::get_mass(zed_leptonic)")
 # write branch with zed transverse momenta
 .Define("zed_leptonic_pt",      "ReconstructedParticle::get_pt(zed_leptonic)")
 # calculate recoil of zed_leptonic
 .Define("zed_leptonic_recoil",  "ReconstructedParticle::recoilBuilder(240)(zed_l
 # write branch with recoil mass
 .Define("zed_leptonic_recoil_m","ReconstructedParticle::get_mass(zed_leptonic_re
 .Define("zed_leptonic_charge","ReconstructedParticle::get_charge(zed_leptonic)")
```

Reconstruction and Analysis with Marlin

"Standing on the shoulders of giants"

- `Marlin` framework from iLCSoft has been tried and tested in ILC and CLIC studies
  - Marlin *Processors* are the working units
- Complete (low level) reconstruction chain available in iLCSoft
  - Digitization, tracking, particle flow (Pandora), ...
- Many high level analysis algorithms for various tasks
  - Jet flavor tagging, isolated lepton finding, ...
- See it in action at this iLCSoft tutorial
- On a high level very similar to Gaudi framework
  - Differences emerge at various “lower” levels

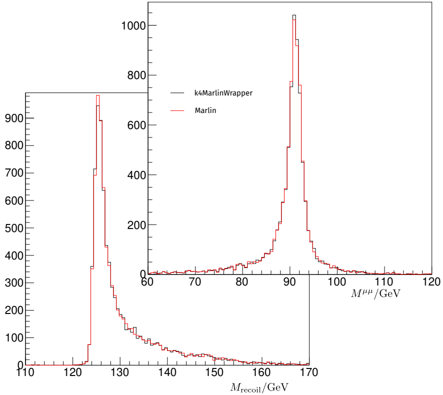


k4MarlinWrapper

key4hep/k4MarlinWrapper

Running Marlin processors in Gaudi

- Allows to run **Marlin processors unchanged** within the Gaudi based Key4hep framework
- Takes care of wrapping the processors and providing the input data in LCIO format
- Comes with an XML steering file to Python options file converter
- Provides tools for automatic in-memory, on-demand conversion between LCIO and EDM4hep
  - Possible to mix Marlin processors with Gaudi algorithms
- See P. Fernández talk at LCWS for more details
- **Validation ongoing**



courtesy of B. Stacey

Available resources and release

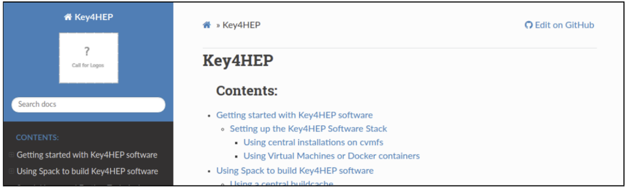
Where to go from here

- (Rolling) latest release of the complete Key4hep software stack

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

  - **Comes with FCCSW, CEPCSW, iLCSoft** (and more)!
  - “Batteries included” - $\mathcal{O}(100)$ software packages and libraries

- Documentation
  - key4hep.github.io/key4hep-doc (main documentation)



Key4HEP

?

Call for Logos

Search docs

CONTENTS:

Getting started with Key4HEP software

Using Spack to build Key4HEP software

» Key4HEP

Edit on GitHub

Key4HEP

Contents:

- Getting started with Key4HEP software
  - Setting up the Key4HEP Software Stack
    - Using central installations on cvmfs
    - Using Virtual Machines or Docker containers
- Using Spack to build Key4HEP software
  - Using a central buildcache

- Automated builds and continuous integration (CI) where possible
  - Regular nightly builds of the complete stack
- **Release early and release often**
  - Make fixes available early
  - Discover problems and collect feedback as early as possible

Current status and ongoing work

An incomplete list

- Integration of k4SimDelphes into the Gaudi framework
  - Prototype exists but still has some teething problems
- Integration of full detector simulation into the framework
  - Some work has already been done for FCC
- Full validation of the k4MarlinWrapper
  - Can we achieve “bit-by-bit” equivalence?
- (Finish) migration of existing software from FCC
- Schema evolution for EDM4hep
- Development and investigation of multithreaded workflows
- Integration of more software packages and libraries into the Key4hep stack
- … (your ideas welcome)

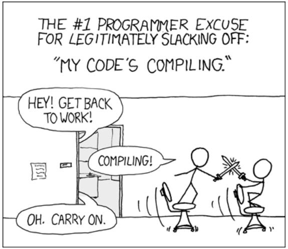
How to collaborate

At the moment more work than people!

- Active weekly meetings, alternating between EDM4hep and Key4hep
  https://indico.cern.ch/category/11461/
- Check out documentation at https://cern.ch/key4hep
  - Also contains some examples
- **Any feedback is welcome**, this should not just be an academic exercise!
- If you find any issues, **do not hesitate to report them**
  - Documentation not up to date?
  - Examples not working?
- **We also greatly appreciate pull requests**

Summary & Conclusions

- Future collider studies rely on a well maintained and consistent software stack
- The Key4hep project aims to provide a common software stack for **all future collider projects**
  - Regular contributions from ILC, CLIC, FCC, CEPC
  - **Common EDM4hep** format is already used in physics studies
- DD4hep detector models / concepts exist for all projects
  - Varying degrees of details yet
- Possibility to integrate existing software from iLCSoft into the Key4hep framework exists
- **Still a lot of work ahead!**



xkcd.com/303

Backup &
Supplementary Material

More information

… and how to get involved

- HEP-FCC/FCCeePhysicsPerformance - “Landing page” for FCCee performance studies
- iLDAnaSoft - Github organisation hosting ILD benchmark analysis and documentation
- twiki.cern.ch/twiki/bin/view/CLIC/Detector - “Landing page” for CLIC related studies and samples

Existing (and future) tutorials

- ilcsnowmass.org - Tutorials and links to existing miniDST samples for ILC for Snowmass
- agenda.linearcollider.org/category/273 - Ongoing tutorial series of the WG3 Software Group
- hep-fcc.github.io/fcc-tutorials - Tutorials and documentation on different aspects of FCC workflows
- key4hep.github.io/key4hep-doc - Documentation and tutorials for Key4hep

Pointers to software (re)sources

- Key4hep
  key4hep.github.io/key4hep-doc
  key4hep - github organisation
- EDM4hep
  key4hep/EDM4hep
  cern.ch/edm4hep
- DD4hep
  AIDASoft/DD4hep
  dd4hep.web.cern.ch
- k4SimDelphes
  key4hep/k4SimDelphes
- k4MarlinWrapper
  key4hep/k4MarlinWrapper

- iLCSoft
  iLCSoft - github organisation
  ilcsoft.desy.de

xkcd.com/138

Marlin vs Gaudi

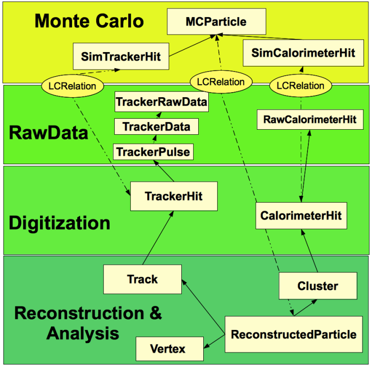
- Conceptually the two frameworks are very similar
- Most obvious differences in naming conventions
    - As always some differences emerge when looking at the details

	Marlin	Gaudi
language	C++	C++
working unit	Processor	Algorithm
config language	XML	Python
transient data format	LCIO	anything
set up function	`init`	`initialize`
work function	`processEvent`	`execute`
wrap up function	`end`	`finalize`

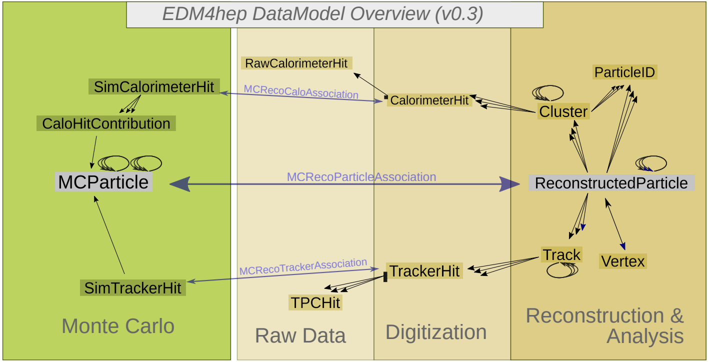
LCIO vs EDM4hep

A side-by-side comparison

LCIO



EDM4hep



- Since EDM4hep is based on LCIO the high-level structure is very similar
- Largest differences between the two are due to their implementations
- LCIO has over 15 years of usage. A lot of time to develop tools for it.
  - Not nearly as far with EDM4hep