The turnkey software stack Key4hep

Status and Plans - the non-FCC view

FCC-week 2021

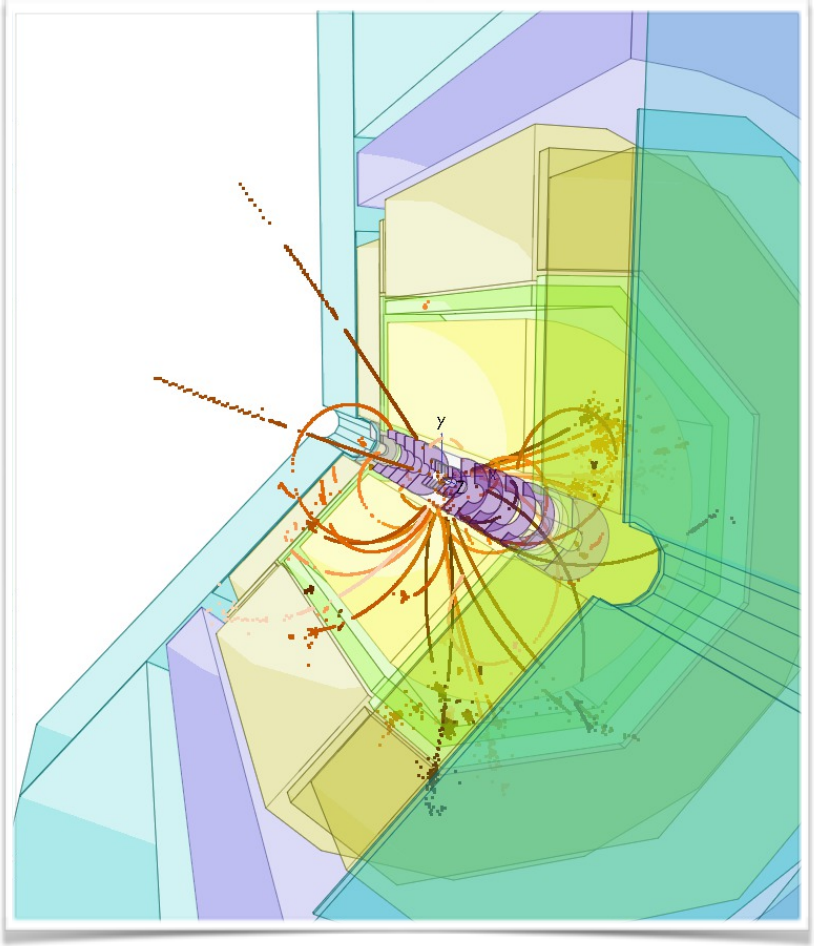
30.06.21

Frank Gaede, DESY
for the Key4hep team

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Outline

- Introduction
  - Key4hep - the vision
- the main ingredients of Key4hep
  - DD4hep, Gaudi, PODIO/EDM4hep
- simulation, reconstruction and analysis tools for LC community in iLCSoft
- migration scenarios from iLCsoft to Key4hep
  - for ILC, CLIC and CEPC
- status and plans
- Conclusion and Outlook



focus here on the non-FCC view - see G.Ganis's detailed plenary talk for Key4hep and FCC aspects

Key4hep

turnkey software stack for all future colliders

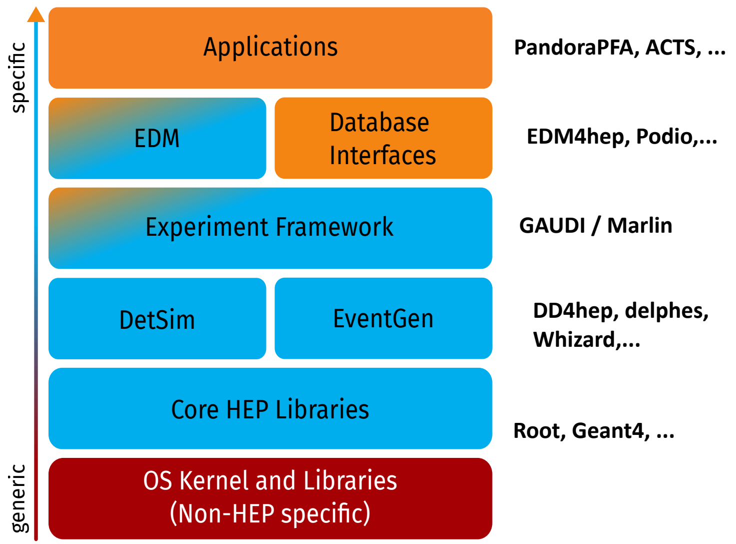
- HEP community decided to develop a **common turnkey software stack** – for future collider studies

- create a software ecosystem integrating in an o**ptimal way the best software components** to provide a **ready-to-use full-fledged solution** for data processing of **HEP** experiments (with initial focus on future colliders)

  - similar to what was done with **iLCSoft** for the linear collider community >15 years ago

  - supported by **HSF** and **CERN** EP-R&D and *AIDAinnova*

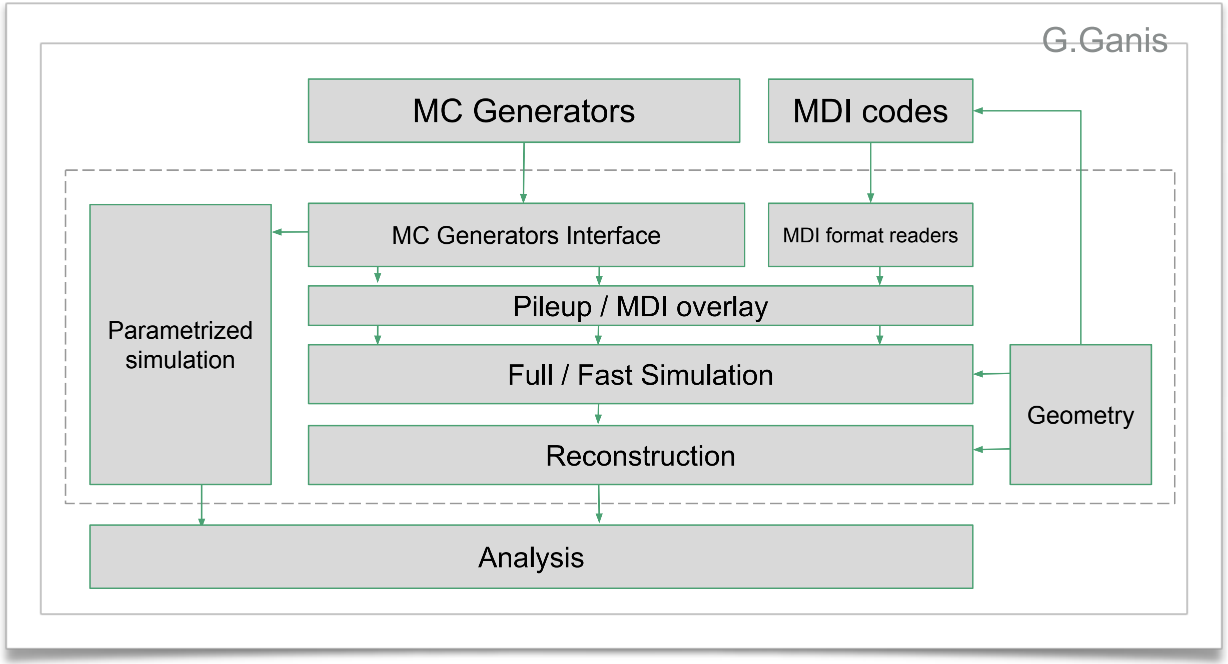
  - involved communities: CEPC, CLIC, FCC, ILC,…



- kick-off meeting: "Future Collider Workshop", Bologna, Jun 2019
- follow-up: "IAS: HEP Meeting", Hong Kong, Jan 2020

Workflows in HEP and Interoperability

the analyst and developers view



- top:typically workflows as seen by the analyst
- right: software techniques to enable these workflows in a turnkey stack

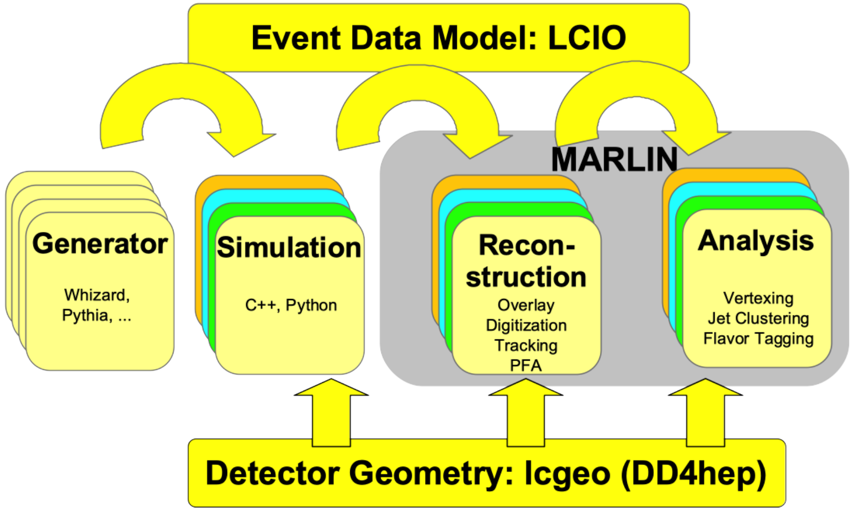
- Level 0 - **Common Data Formats**
  - Maximal interoperability, even on different hardware
- Level 1 - **Callable Interfaces**
  - Defined for one or more programming languages
  - Implementation quality of interfaced components important
  - Required to define plugins
- Level 2 - **Introspection Capabilities**
  - Software elements to facilitate the interaction of objects in a generic manner such as Dictionaries and Scripting interfaces
  - Language bindings, e.g. PyROOT
- Level 3 - **Component Level**
  - Software components are part of a common framework, optimal interplay
  - Common configuration, log and error reporting, plug-in management, ...

The common software vision

the high altitude view

- complete set of tools for
  - **generation, simulation, reconstruction, analysis**
  - build, package, test, deploy

- core ingredients of current **Key4hep**
  - **PODIO** for **EDM4hep** (based on LCIO and FCC-edm)
  - **Gaudi** framework, devel/used for (HL-)LHC
  - **DD4hep** for geometry
    - developed for LC adopted by LHC
  - **spack** package manager
    - lots of interest from LHC



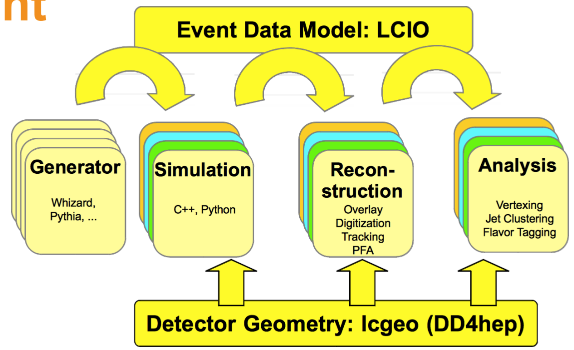
- generic scheme of iLCSoft framework and core tools
- very similar now in Key4hep:
  - Marlin -> Gaudi
  - LCIO -> EDM4hep
- should facilitate the migration/transition for LC community

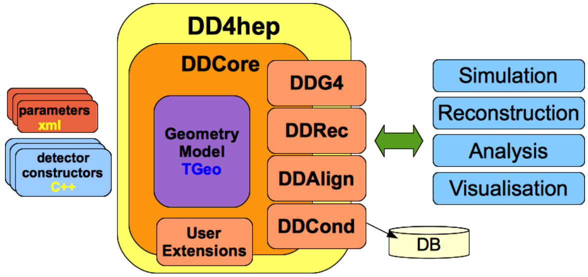
iLCSoft Core Tools

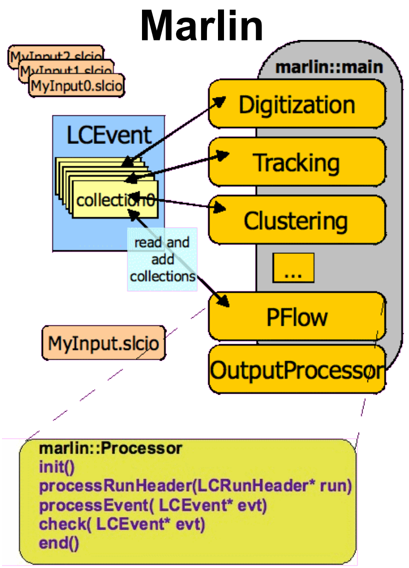
Setting the foundation for common software development

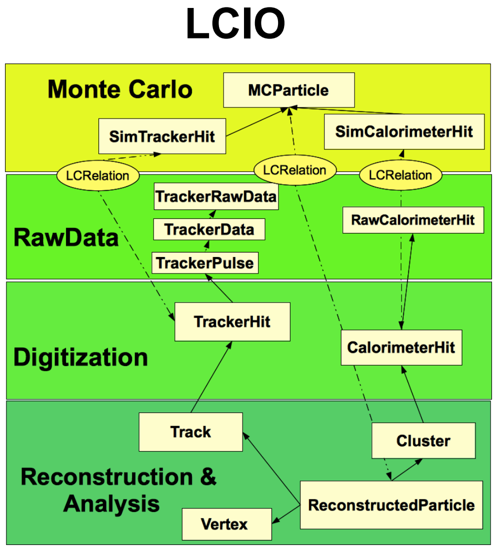
- the linear collider community had started to develop **common software** in 2003
- introducing a common event data model: **LCIO**
  - providing the ***language*** and a file format
- introducing a simple to use framework: **Marlin**
  - allow all LC groups to develop their algorithms in a common software ecosystem
- a detector geometry toolkit **DD4hep**

a flexible, generic and easy to use software ecosystem provided the foundation for future collider developments for many years
- **CEPC, CLICdp, ILC, (FCC)**
- CALICE, LC-TPC, EU-Telescope,…







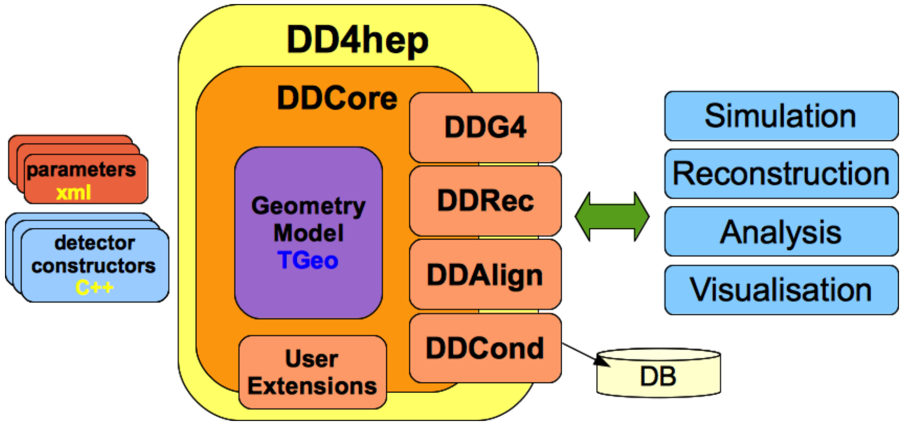


- moving from iLCSoft to key4hep offers great opportunity to **modernise the software stack**
- however need a well defined **migration strategy**

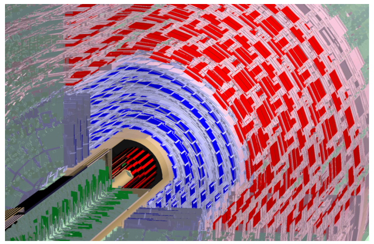
DD4hep geometry toolkit

defining the detector geometry and different views on it

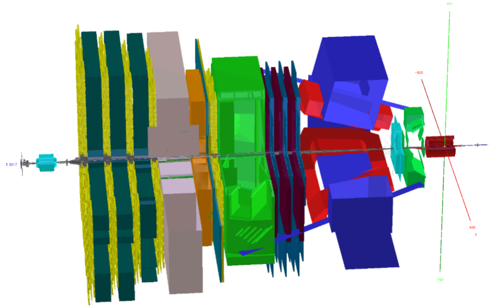
- LC community and CERN have developed a generic detector geometry system - based on best practises by ILC, CLIC, LHCb (*in AIDA, AIDA2020*)
- supporting the full life cycle of the experiment
- providing components and interfaces for
    - full simulation, reconstruction, conditions, alignment, visualisation and analysis
- adopted also by CMS and LHCb



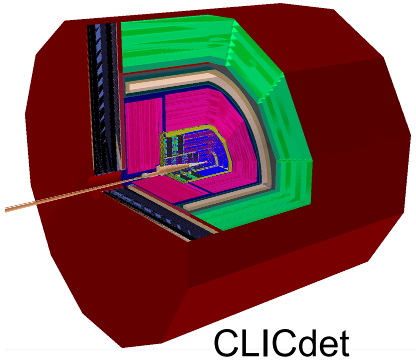
DD4hep: de facto industry standard



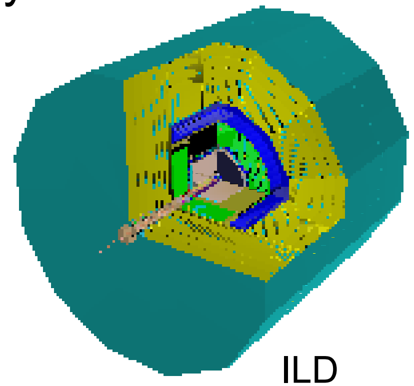
CMS



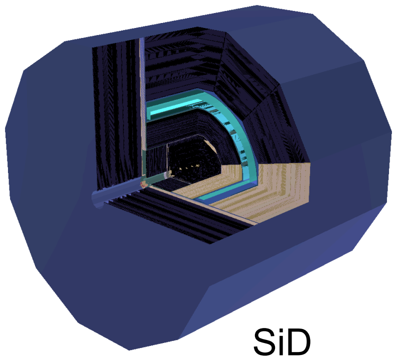
LHCb



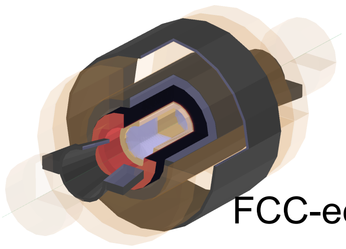
CLICdet



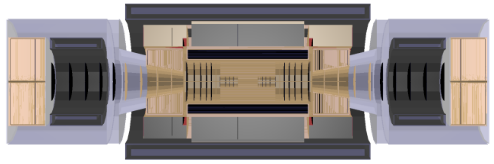
ILD



SiD



FCC-ee



FCC-hh

Gaudi

the application framework

- C++ application framework for HEP
- developed at CERN
- used in production for
  - LHCb and ATLAS (*battle-proven*)
  - FCC-SW and smaller experiments
  - and now in Key4HEP
- highly configurable
  - EDM, workflows (algorithms)
- allows parallelisation through multi-threading
- integration of heterogeneous resources
  - CPUs, GPUs, FPGAs,…

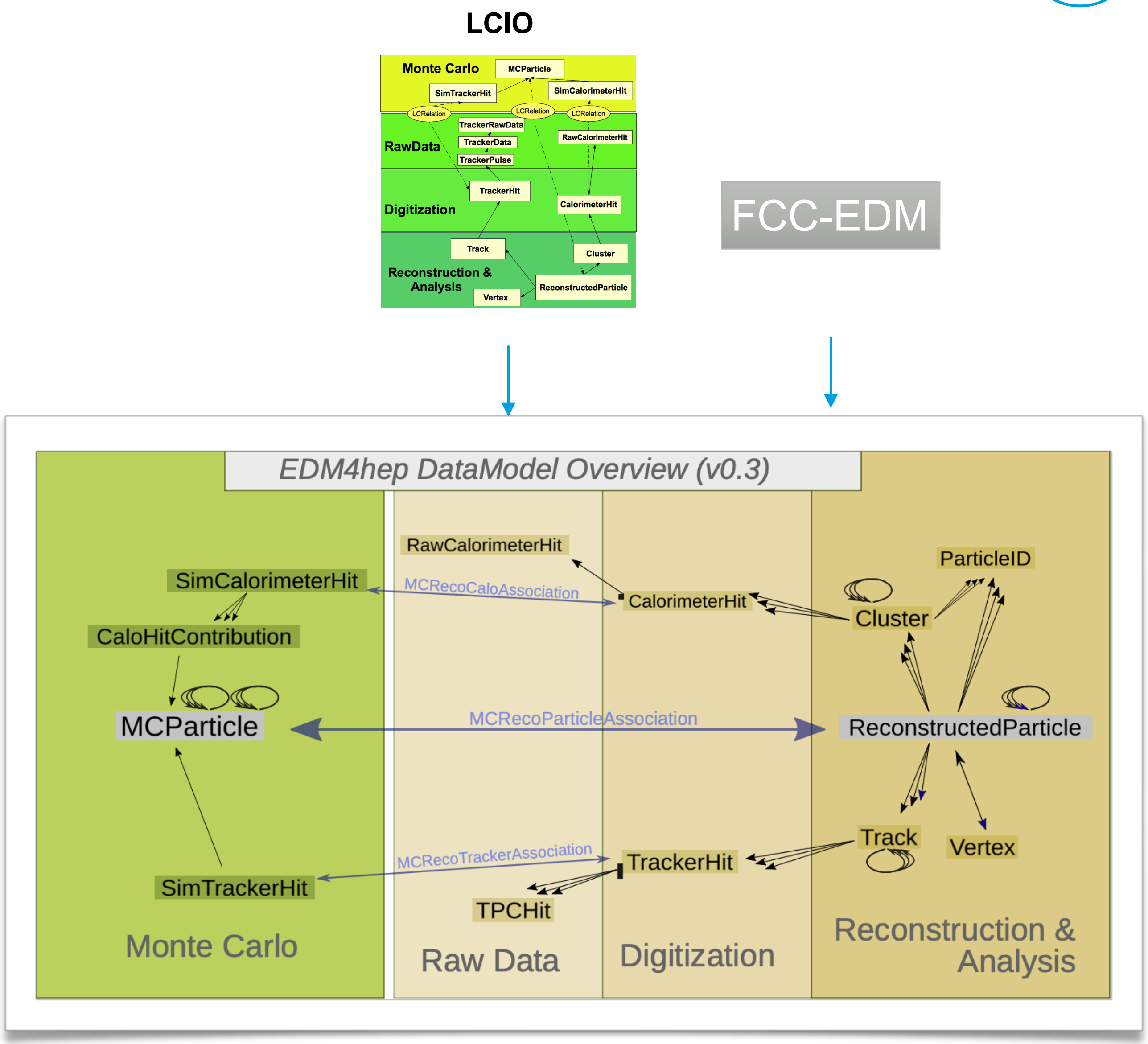
	Marlin	Gaudi
language	c++	c++
working unit	Processor	Algorithm
config language	XML	Python
transient data format	LCIO	anything
set up function	`init`	`initialize`
work function	`processEvent`	`execute`
wrap up function	`end`	`finalize`

similar to MARLIN framework
yet more powerful and larger user basis

EDM4hep

generic HEP event data model

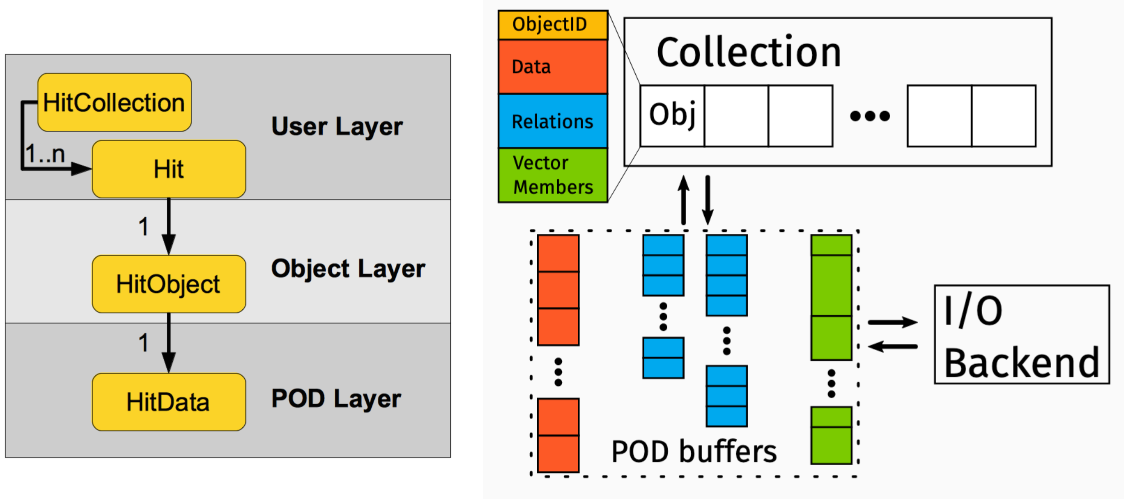
- the event data model defines the language for all data processing tasks in HEP
- EDM4hep aims at getting this right for all future collider projects - independent of the type of collider
  - (ee, mumu, pp,…)
- largely based on (battle-proven) LCIO and FCC-EDM
- first example analyses for FCC-hh successful
- EDM access and I/O part needs to be
  - fast and efficient
  - support multithreading
  - transparent choice of actual I/O system
- implemented with PODIO



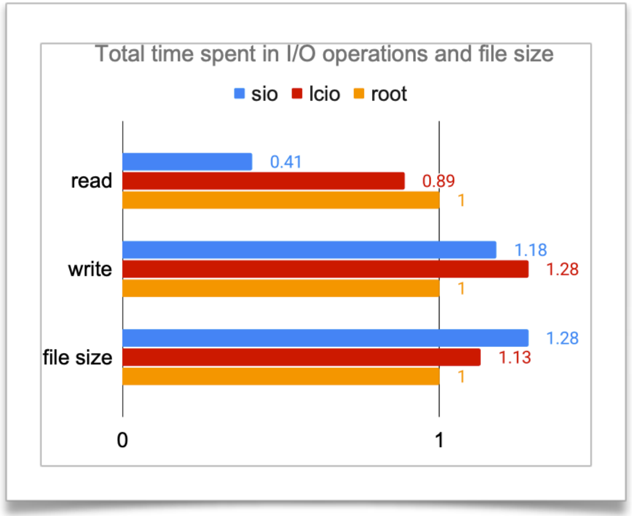
PODIO

EDM toolkit

- **PODIO**: event data model toolkit
    - based on storing PODs (plain old data structures)
    - developed in AIDA2020
- actual EDM defined in yaml files
- use jinja2 and Python scripts to create C++ and Python code
- features:
    - value semantics and
    - thread safety
    - multiple I/O implementations (ROOT, SIO,…)
    - support for parallel I/O (WIP)
    - schema evolution (WIP)



schematic view of the data layout and user classes

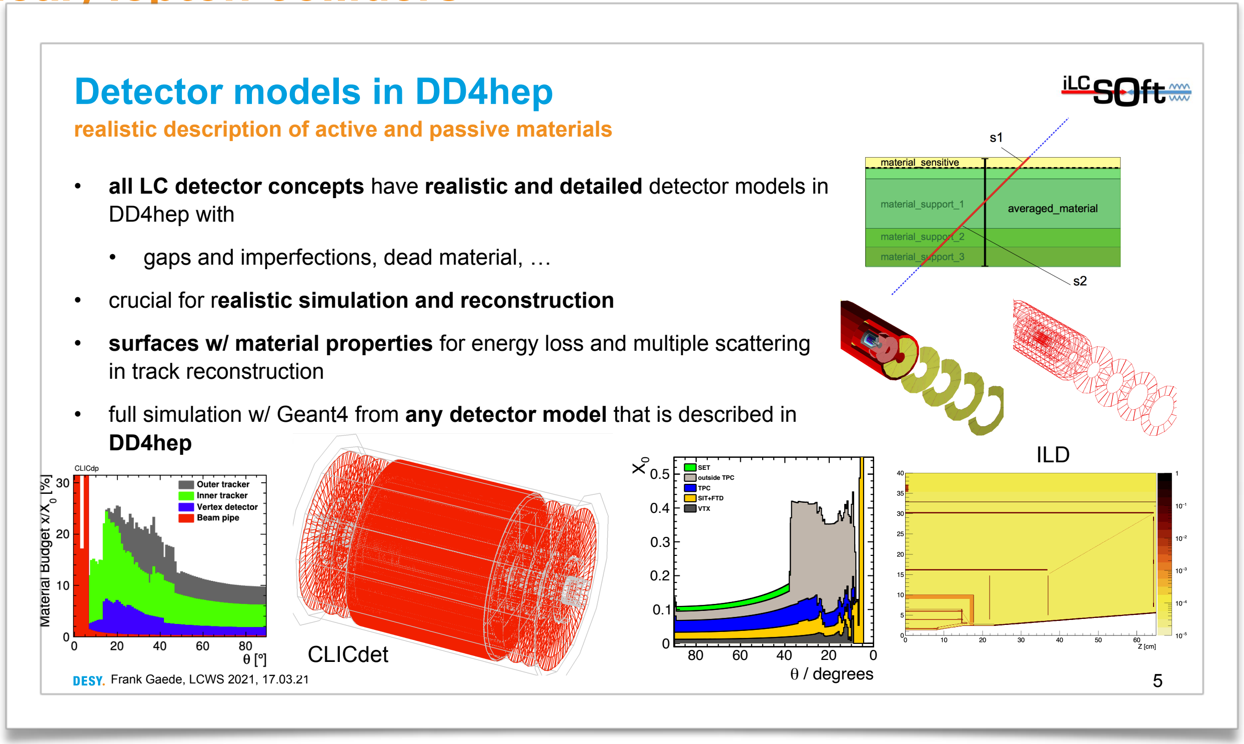


benchmark ROOT vs SIO w/ realistic ee-events from *k4SimDelphes*

large code base in iLCSoft

Simulation, Reconstruction and Analysis for (linear) lepton colliders

- realistic detector models for ILC, CLICdp and CEPC defined in DD4hep
  - incl. tracking/reconstruction geometry

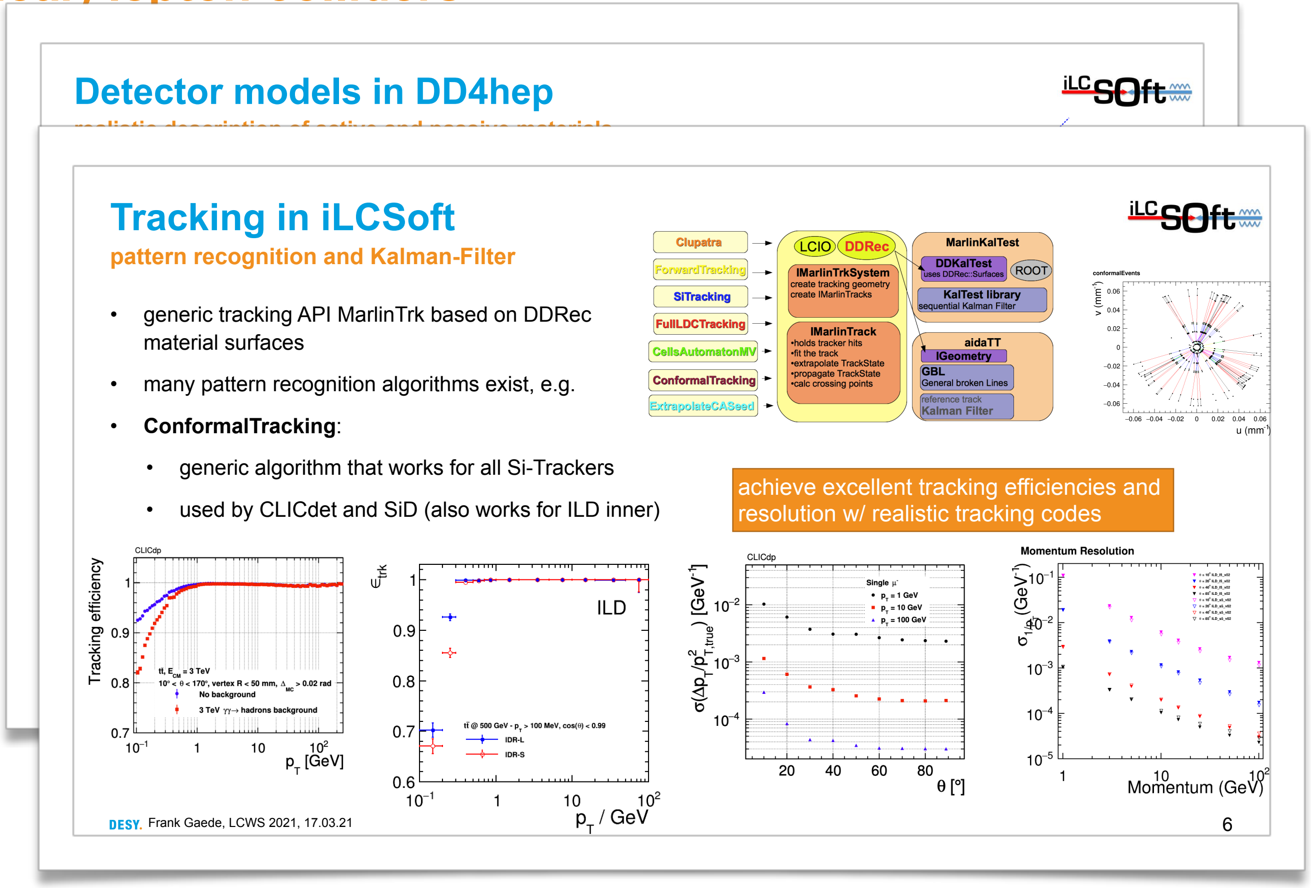


see talk on LC software at LCWS 2021

large code base in iLCSoft

Simulation, Reconstruction and Analysis for (linear) lepton colliders

- realistic detector models for ILC, CLICdp and CEPC defined in DD4hep
  - incl. tracking/reconstruction geometry
- track reconstruction
  - generic API for fitting algorithms
  - large number of pattern recognition algorithms

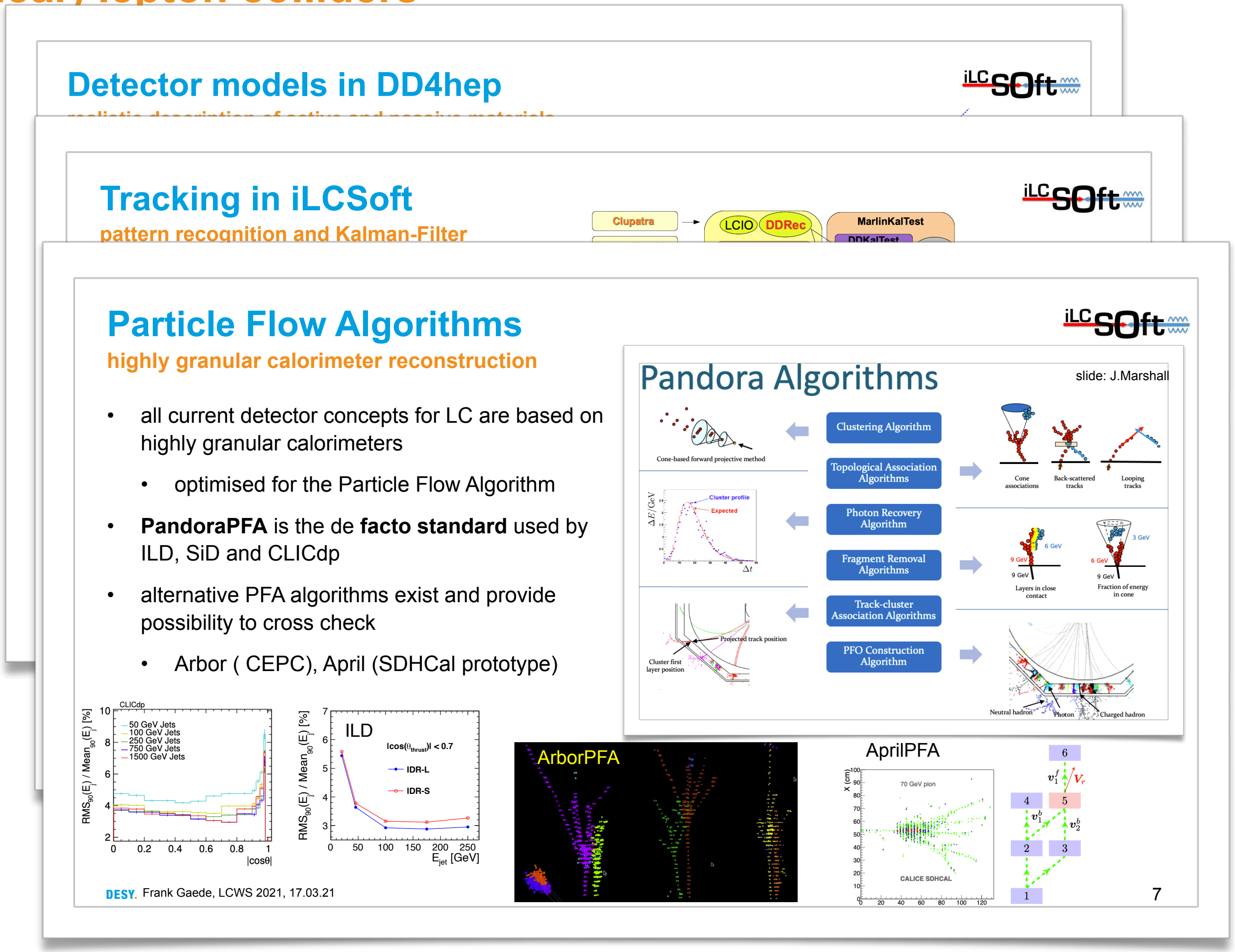


see talk on LC software at LCWS 2021

large code base in iLCSoft

Simulation, Reconstruction and Analysis for (linear) lepton colliders

- realistic detector models for ILC, CLICdp and CEPC defined in DD4hep
  - incl. tracking/reconstruction geometry
- track reconstruction
  - generic API for fitting algorithms
  - large number of pattern recognition algorithms
- particle flow algorithms
  - PandoraPFA ans Arbor, AprilPFA

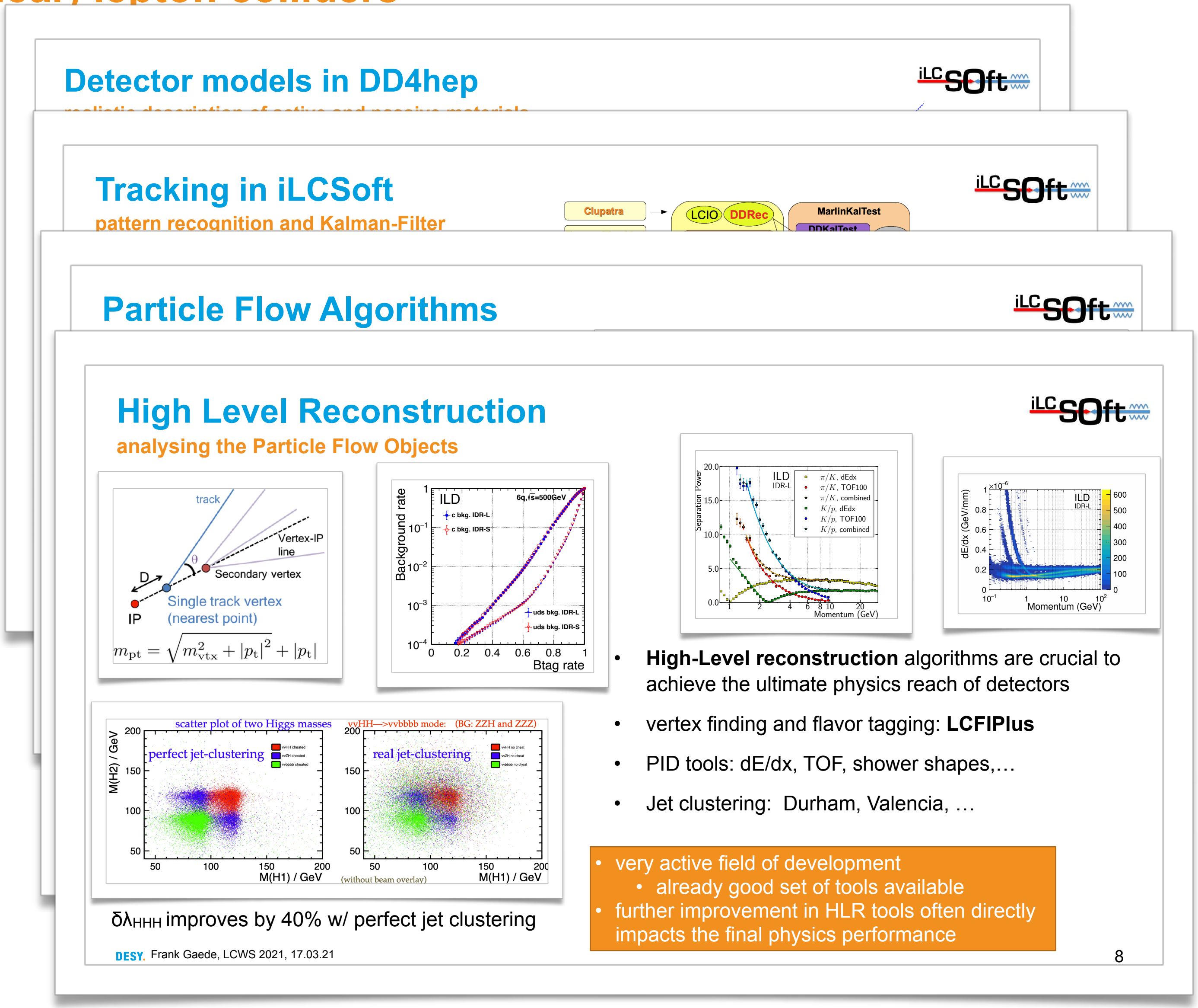


see talk on LC software at LCWS 2021

large code base in iLCSoft

Simulation, Reconstruction and Analysis for (linear) lepton colliders

- realistic detector models for ILC, CLICdp and CEPC defined in DD4hep
    - incl. tracking/reconstruction geometry
- track reconstruction
    - generic API for fitting algorithms
    - large number of pattern recognition algorithms
- particle flow algorithms
    - PandoraPFA ans Arbor, AprilPFA
- high level reconstruction
    - jet finding, flavor tagging, PID, TOF,…

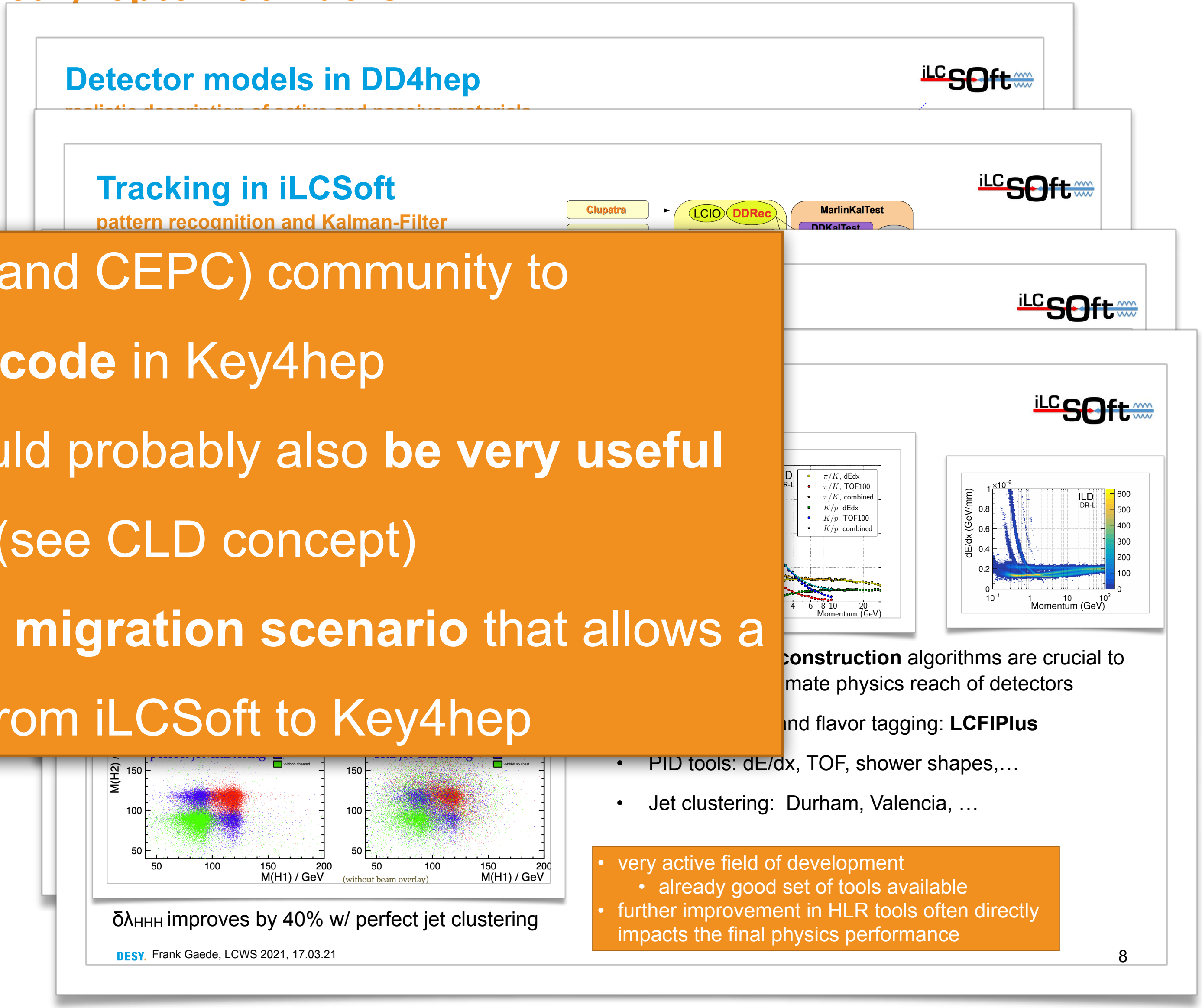


see talk on LC software at LCWS 2021

large code base in iLCSoft

Simulation, Reconstruction and Analysis for (linear) lepton colliders

- realistic detector models for ILC, CLICdp and CEPC defined in DD4hep
  - incl. tracking/reconstruction geometry
- track reconstruction
  - generic API for f
  - large number of algorithms
- particle flow algorith
  - PandoraPFA ar
- high level reconstruction
  - jet finding, flavor tagging, PID, TOF,…



- it is **vital** for the LC (and CEPC) community to **preserve all of this code** in Key4hep
- a lot of this code would probably also **be very useful** for FCC(ee) studies (see CLD concept)
- -> need to develop a **migration scenario** that allows a **smooth transition** from iLCSoft to Key4hep

see talk on LC software at LCWS 2021

k4MarlinWrapper

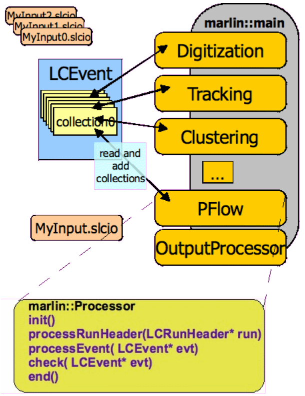
running Marlin processors in Gaudi (Key4hep)

- set of Gaudi algorithms that wrap Marlin processors
  - developed by CERN-SFT
  - automatic XML to Python steering file conversion
- tools for automatic in-memory, on-demand conversion between LCIO and EDM4hep
  - developed by IHEP and CERN
  - possibility to mix Marlin processors with genuine Gaudi algorithms

- this is the intended **working horse for a smooth transition** from iLCSoft to Key4hep
  - CLIC and ILD reco run as *proof-of-concept*
  - also adopted by SCT

```
MyTPCDigiProcessor = MarlinProcessorWrapper("MyTPCDigiProcessor")
MyTPCDigiProcessor.OutputLevel = INFO
MyTPCDigiProcessor.ProcessorType = "DDTPCDigiProcessor"
MyTPCDigiProcessor.Parameters = [
              "DiffusionCoeffRPhi", "0.025", END_TAG,
              "DiffusionCoeffZ", "0.08", END_TAG,
              "DoubleHitResolutionRPhi", "2", END_TAG,
              "DoubleHitResolutionZ", "5", END_TAG,
              "HitSortingBinningRPhi", "2", END_TAG,
              "HitSortingBinningZ", "5", END_TAG,
              "MaxClusterSizeForMerge", "3", END_TAG,
              "N_eff", "22", END_TAG,
              # ...
              ]
algList.append(MyTPCDigiProcessor)
```

```
from Gaudi.Configuration import *
CONSTANTS = {'BCReco': "3TeV",}
parseConstants(CONSTANTS)
# ...
read = LcioEvent()
InitDD4hep = MarlinProcessorWrapper("InitDD4hep")
Config = MarlinProcessorWrapper("Config")
VXDBarrelDigitiser = MarlinProcessorWrapper("VXDBarrel
VXDEndcapDigitiser = MarlinProcessorWrapper("VXDEndcap
# ...
algList.append(InitDD4hep)
algList.append(Config)
# algList.append(OverlayFalse)
# algList.append(Overlay350GeV_CDR)
algList.append(VXDBarrelDigitiser)
algList.append(VXDEndcapDigitiser)
# ...
```

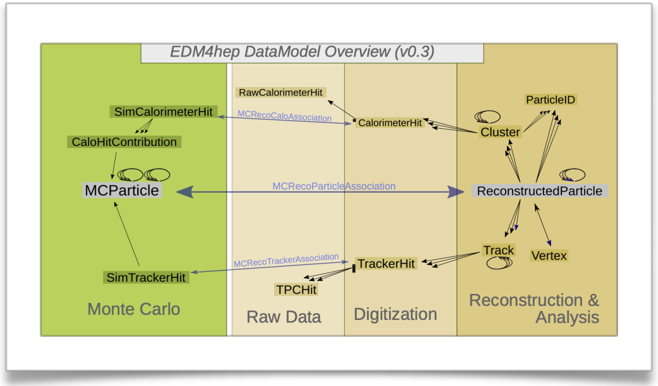


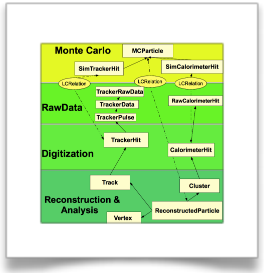
see talk P.Fernandez at LCWS

LCIO to EDM4hep

modernising the underlying language for all algorithms

- the actual EDM in EDM4hep is content wise very similar to the one in LCIO
    - every algorithm written in LCIO can - *in principle* - be converted to EDM4hep
- some technicalities in coding style
    - value semantics vs. pointer semantics
    - different handling of collections and ownership
    - …
- large set of **utility classes** developed in LCIO over the years, e.g:
    - navigation of relations, handling of evt/col - meta data, accessing bit fields, cellID encoding,…
    - started to **port these to EDM4hep**





```
auto* mc = new MCParticleImpl;
auto* mc2 = new MCParticleImpl;

// setting properties and relations
mc->setMass(3.096);
mc->addParent(mc2);

// looping over relations
for (auto* p : mc->getParents()) { /**/ }

// collections
auto* coll = new LCCollectionVec(LCIO::MCPARTICLE);
coll->addElement(mc);

// indexed access
auto* p = dynamic_cast<MCParticle*>(coll->getElementAt(0));

// looping
LCIterator<MCParticle> iter(coll);
while(auto* p = iter.next()) { /**/ }
```

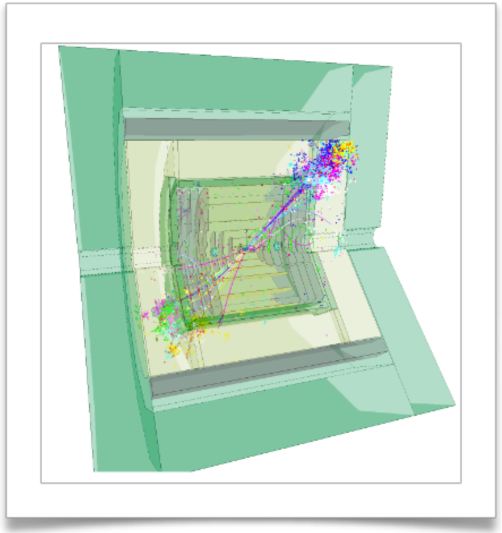
```
auto mc = MCParticle();
auto mc2 = MCParticle();

// setting properties and relations
mc.setMass(3.096);
mc.addToParents(mc2);

// looping over relations
for (auto p : mc.getParents()) { /**/ }

// collections
auto coll = MCParticleCollection();
coll.push_back(mc);

// indexed access
auto p = coll[0];

// looping

for (auto p : coll) { /**/ }
```

some simple examples for code comparison LCIO - EDM4hep
(more complex examples w/ larger differences exist)

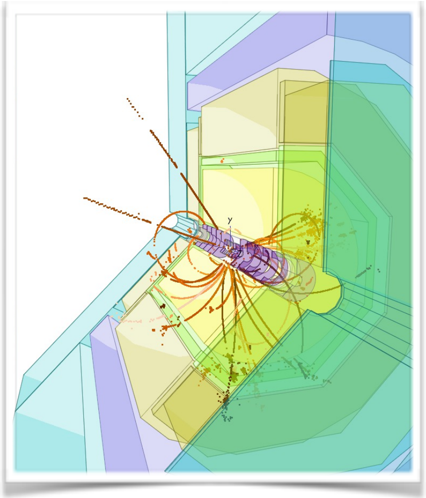
CLICdp and ILD (SiD) migration

ensuring a smooth transition from iLCSoft to Key4hep

- start with using **k4MarlinWrapper** to run the existing digitization and reconstruction chain
  - done as *proof-of-concept* for CLICdp and ILD
  - proper physics validation pending
    - at DESY will dedicate two summer students to this



- gradually port existing algorithms to Gaudi algorithms using EDM4hep (where possible)
  - this requires a reliable and validated in memory **round-trip conversion LCIO2EDM4hep**
  - currently validation ongoing …

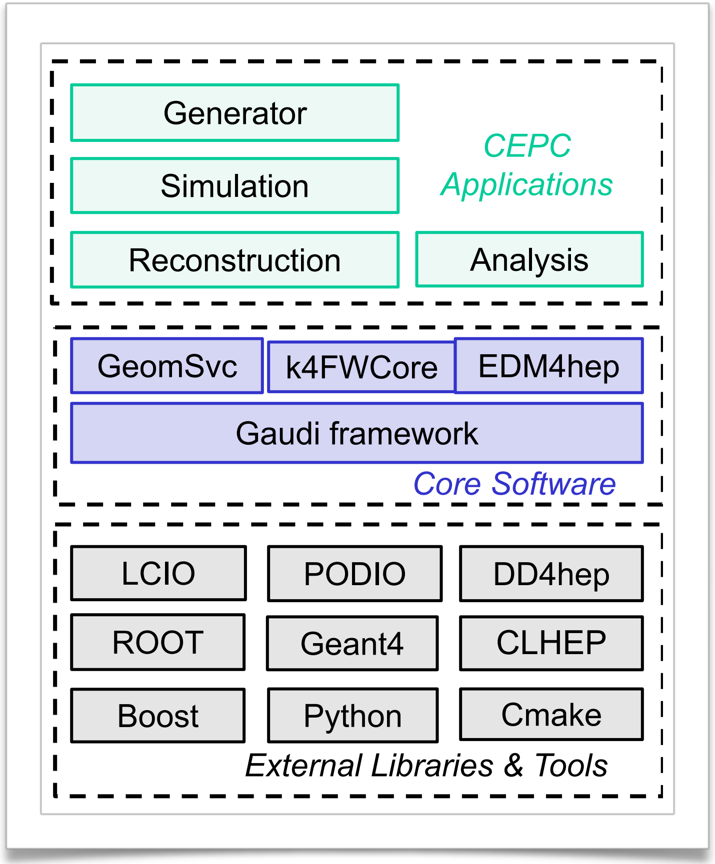


- some algorithms e.g. **PandoraPFA** are well encapsulated and should be easy to port
- others, e.g. **MarlinTrk** potentially much harder (also wrt. thread safety)
- in the long run would like to move to an **ACTS** based pattern recognition and track fitting

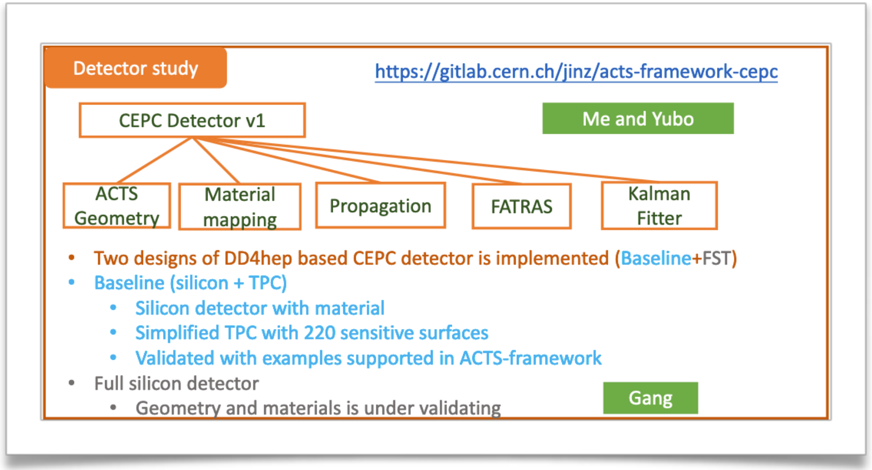
CEPC Migration

started transition from iLCSoft to Key4hep

- CEPC software originally based on iLCSoft
- started the transition to Gaudi/Key4hep earlier than ILC and CLIC
  - not relying on k4MarlinWrapper
  - directly port existing algorithms to Gaudi/EDM4hep
- also started to use **ACTS** for track reconstruction
- IHEP and SDU associated partners in **AIDAinnova**



W.Li, IHEP



J.Zhang, IHEP/DESY

Estimation of manpower cost

Tasks	Subtasks	Manpower cost
Framework	Development of LCIO converter	2 person months
Simulation	Generator interface, Simulation framework, Digitization algorithms	2 person months
	Detector description, Detector response	1 person month
	Performance validation	2 person months
Reconstruction	Package migration	0.5 person month/algorithm
	Performance validation	1.5 person month/algorithm
Analysis	Package migration	1 week/algorithm

Status and Plans for Key4hep

- **core components of Key4hep** already - or close to - a **first production quality release -** some work still ongoing:
- **EDM4hep/PODIO**
  - improving PODIO (parallel I/O, schema evolution, user data,…)
- **Full Simulation**
  - different approaches chosen:
    - FCC, CEPC use **DD4hep/Gaudi**
    - CLIC/ILC use **DD4hep/DDG4** and **ddsim**
  - need to see if these can be unified or if we support both approaches in Key4hep

- all groups have started to think about/work on **migrating their software to Key4hep**
  - FCC-SW with a head start, as based on DD4hep, Gaudi from the start
    - working w/ Delphes or CLIC based full sim/rec
  - ILC and CLIC planning migration based on k4MarlinWrapper
    - to ensure smooth transition of existing large code base

- regular weekly phone meetings
  - if you want to contribute join:

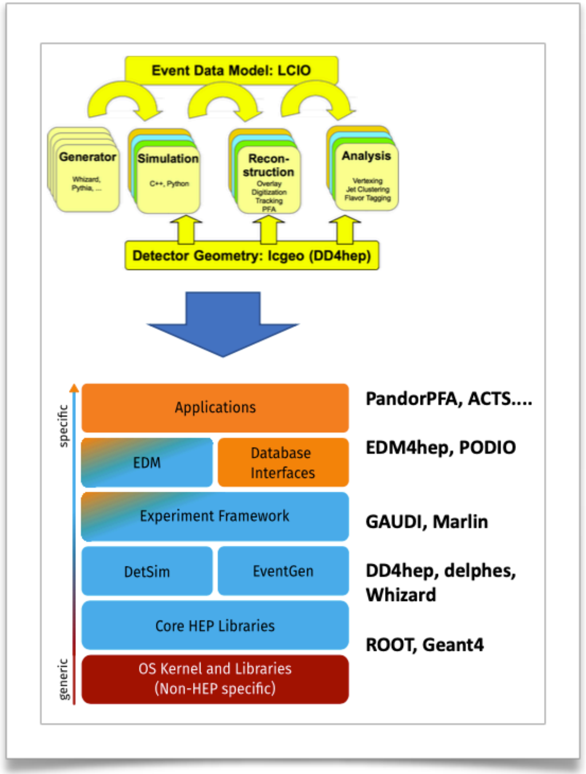
key4hep-sw@cern.ch
hsf-edm4hep-wg@cern.ch

Summary and Outlook

- iLCSoft provides the **framework**, **simulation** and **reconstruction** tools for detector and physics studies for ILC, CLIC and beyond
    - developed as a LC community effort for more than 15 years
    - now is a good time to **modernise the software for the future**

- **Key4hep** started as a new future collider community wide effort in 2020 to put together a modern turnkey software stack

- **this offers an excellent opportunity for the HEP community to develop for the first time a common software ecosystem**
    - **using synergies to develop and maintain common core tools**
    - **focus on the physics questions and algorithms of your experiment**



pointers to documentation

entry points to Key4hep

- Key4hep GitHub Project
  - https://github.com/key4hep
- Key4hep main documentation page
  - https://key4hep.github.io/key4hep-doc/
- Doxygen available., e.g. for EDM4hep
  - https://edm4hep.web.cern.ch/
- iLCSoft Github Project
  - https://github.com/ilcsoft