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
Outline

• Introduction
  • Key4hep - the vision
• the main ingredients of Key4hep
  • DD4hep, Gaudi, PODIO/EDM4hep
• simulation, reconstruction and analysis tools for LC community in iLCSof
• 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 **optimal 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

- 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, ...

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

<table>
<thead>
<tr>
<th></th>
<th>Marlin</th>
<th>Gaudi</th>
</tr>
</thead>
<tbody>
<tr>
<td>language</td>
<td>c++</td>
<td>c++</td>
</tr>
<tr>
<td>working unit</td>
<td>Processor</td>
<td>Algorithm</td>
</tr>
<tr>
<td>config language</td>
<td>XML</td>
<td>Python</td>
</tr>
<tr>
<td>transient data format</td>
<td>LCIO</td>
<td>anything</td>
</tr>
<tr>
<td>set up function</td>
<td>init</td>
<td>initialize</td>
</tr>
<tr>
<td>work function</td>
<td>processEvent</td>
<td>execute</td>
</tr>
<tr>
<td>wrap up function</td>
<td>end</td>
<td>finalize</td>
</tr>
</tbody>
</table>

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)

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

**schematic view of the data layout and user classes**
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

Detector models in DD4hep
realistic description of active and passive materials

- all LC detector concepts have realistic and detailed detector models in DD4hep with
  - gaps and imperfections, dead material, ...
  - crucial for realistic simulation and reconstruction
  - surfaces w/ material properties for energy loss and multiple scattering in track reconstruction
  - full simulation w/ Geant4 from any detector model that is described in DD4hep

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

Detector models in DD4hep
realistic description of active and passive materials

Tracking in iLCSoft
pattern recognition and Kalman-Filter

- generic tracking API MarlinTrk based on DDRec material surfaces
- many pattern recognition algorithms exist, e.g.
  - ConformalTracking:
    - generic algorithm that works for all Si-Trackers
    - used by CLICdet and SiD (also works for ILD inner)

achieve excellent tracking efficiencies and resolution w/ realistic tracking codes

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

---

**Detector models in DD4hep**

realistic description of active and passive materials

---

**Tracking in iLCSoft**

pattern recognition and Kalman-Filter

---

**Particle Flow Algorithms**

highly granular calorimeter reconstruction

- all current detector concepts for LC are based on highly granular calorimeters
  - optimised for the Particle Flow Algorithm
- PandoraPFA is the de facto standard used by ILD, SiD and CLICdp
  - alternative PFA algorithms exist and provide possibility to cross check
  - Arbor (CEPC), April (SDHCal prototype)

---

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,…

Tracking in iLCSOft
pattern recognition and Kalman-Filter

Particle Flow Algorithms

High Level Reconstruction
analysing the Particle Flow Objects

Detector models in DD4hep
realistic description of active and passive material

- gaps and imperfections, dead material,…
  - crucial for realistic simulation and reconstruction
- surfaces with material properties
  - for energy loss and multiple scattering in track reconstruction
- full simulation w/ Geant4 from any detector model that is described in DD4hep

Very active field of development
- already good set of tools available
- further improvement in HLR tools often directly impacts the final physics performance

\[ \delta \lambda_{\text{HHH}} \text{ improves by } 40\% \text{ w/ perfect jet clustering} \]

Jet finding, flavor tagging, PID, TOF,…

High-Level reconstruction algorithms are crucial to achieve the ultimate physics reach of detectors
- vertex finding and flavor tagging: LCFIPlus
- PID tools: dE/dx, TOF, shower shapes,…
- Jet clustering: Durham, Valencia,…

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 and Arbor, AprilPFA
• 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

very active field of development
• already good set of tools available
• further improvement in HLR tools often directly impacts the final physics performance

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

---

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

```cpp
auto mc = new MCBParticleImpl();
auto mc2 = new MCBParticleImpl();
mc->setMass(0.099);  // setting properties and relations
mc->addParent(mc2);   

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

// collections
auto coll = new LCObjectColl(LCIO::MCPARTICLE);
coll->addElement(no);

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

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

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 iLCSof 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
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

---

Frank Gaede, FCC-Week 2021, 30.06.21