The Key4hep project

An introduction

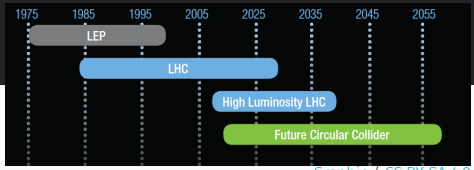
Thomas Madlener
with help from many others

LUXE Analysis Meeting

Mar 06, 2023

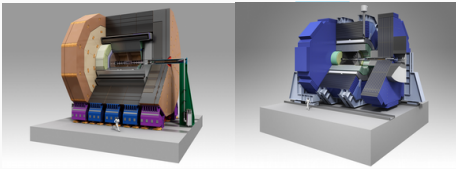
This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

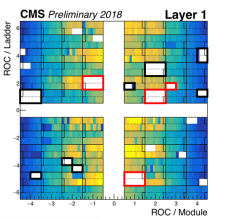
Software Challenges in HEP

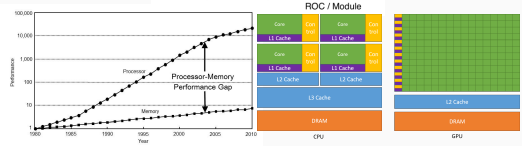


Graphic / CC BY-SA 4.0

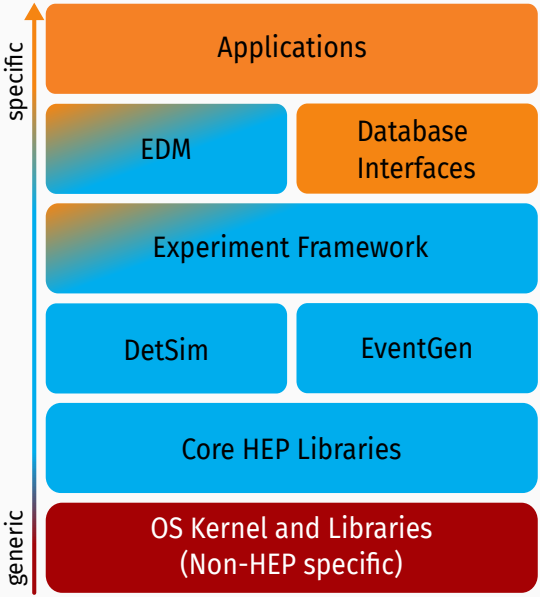
- Long lifetimes of experiments
- Shift of priorities throughout the evolution of an experiment
  - Conceptual and design work
  - Production and the real world
  - Continuous upgrades
  - **Avoid amassing “technical debt”**
- New technological developments potentially lead to new paradigms
  - Optimize resource usage
- Data preservation and ability to look at data in the future







HEP Software Stack



PandoraPFA, FastJet, ACTS, ...

EDM4hep, LCIO, GitCondDB, ...

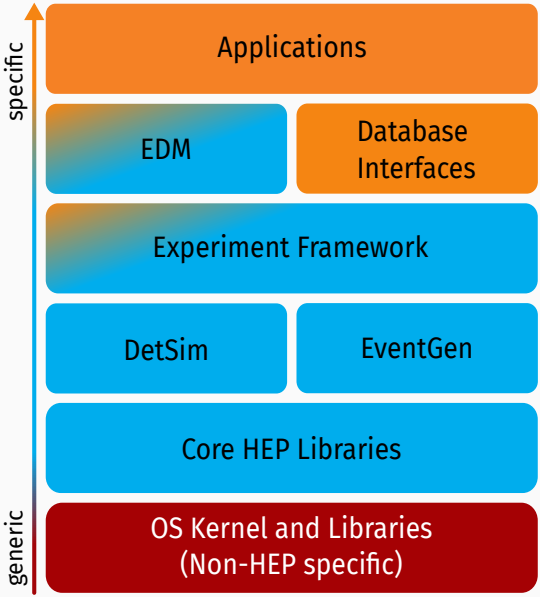
Marlin, Gaudi, CMSSW, ...

DD4hep, Delphes, Pythia, ...

ROOT, Geant4, CLHEP, ...

Python, Compilers, CMake, boost, ...

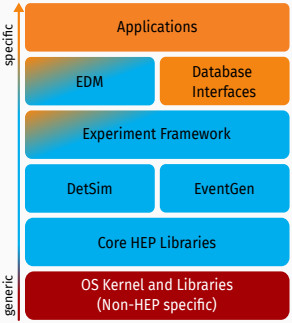
HEP Software Stack



- 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 - A (very) brief introduction

- Future detector studies rely on well maintained software for studying their potential
- Maintenance of a consistent HEP SW stack is non-trivial
- Sharing the burden allows everybody to reap the benefits
  - Make best use of scarce (human) resources
- **Regular contributions from ILC, CLIC, FCC, CEPC, (EIC,) LUXE(?), ...**
  - "Conceived" at 2019 Bologna Future Collider Software Workshop
- Support from major R&D initatives
  - CERN R&D for Future Experiments, AIDAinnova WP12, ECFA



Key4hep goals

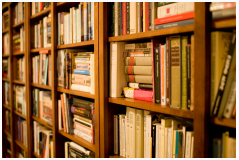
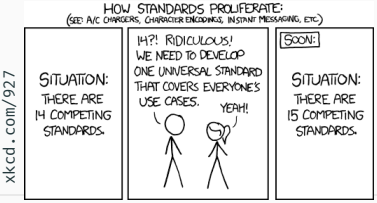


Photo by Stewart B. / CC-BY

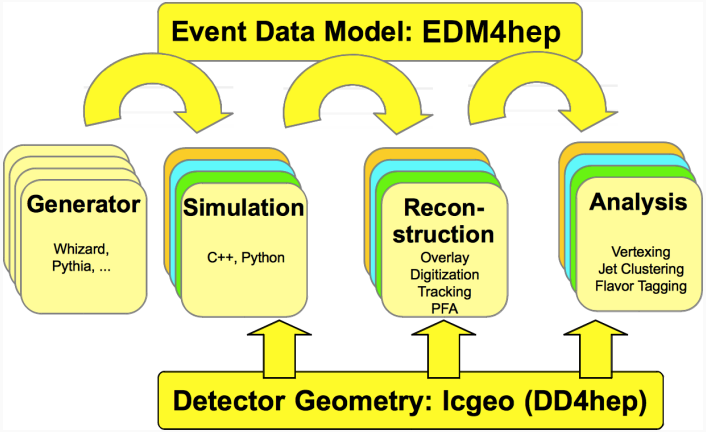
- Provide and maintain a consistent SW stack that allows to do physics studies for **all projects**
- Ensure interoperability of the necessary building blocks
- Reuse existing solutions where possible
  - A lot of experience from LHC experiments and LC communities
- Focus new developments on EW/Higgs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

Non-goal

- Develop and maintain project specific software and workflows

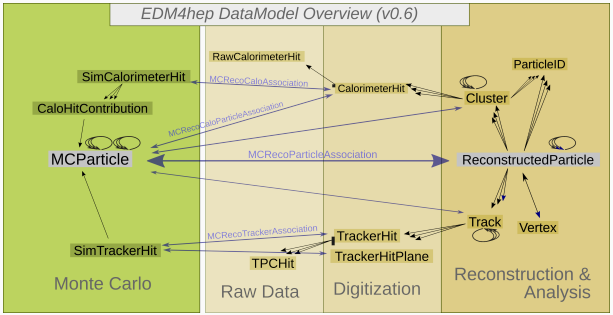


From generation to analysis - the general workflow



- Many steps involved from generating events to analyzing them
- Three main components
  - Event Data Model (EDM)
  - Detector geometry description
  - Processing framework

EDM4hep - The common EDM for Key4hep



key4hep/EDM4hep

edm4hep.web.cern.ch

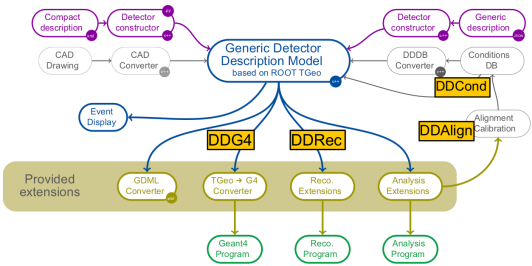
AIDASoft/podio

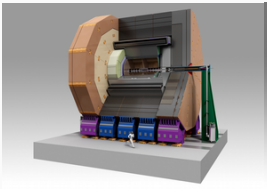
- Interoperatbility of different compontents requires a “lingua franca”
- Based on `LCIO` and `FCC-edm`
  - Focus on usability in analysis
- Generated via `podio` (AIDA 2020)
  - Schema evolution available soon
  - Supports **prototyping of new datatypes**
- Currently finalizing v1
  - Backwards compatible from then

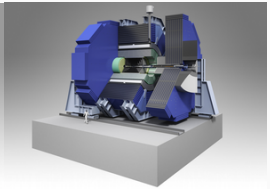
DD4hep - Detector description

- Originally for LC but targetting all of HEP from the start (AIDA 2020)
- 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
  - ILC, CLIC, FCC, CEPC, EIC, LHCb, CMS, ...
- key4hep/k4geo with many detector models

dd4hep.web.cern.ch



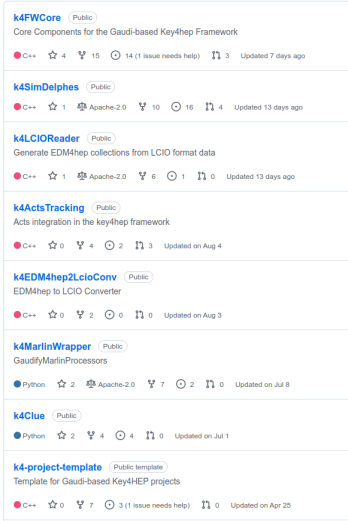




Experiment framework - Conducting all the different pieces

- Key4hep has adopted *Gaudi* as its experiment framework
  - Originally developed by LHCb, used by ATLAS, FCCSW
  - “Battle-proven” from LHC data taking
  - Several (legacy) “flavors”
- `k4FWCore` - core functionality
  - Data service for EDM4hep (podio generated EDMs)
- LC world (iLCSoft) uses *Marlin*
  - **k4MarlinWrapper** for seamless integration
- Dedicated packages for different tasks
- Main guideline: **Use EDM4hep for event data and DD4hep for detector description**

key4hep github org



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 $\rightarrow$ **spack recipe**
  - Build instructions, dependencies, versions and location of source code
  - $\sim$ 6650 packages currently available from spack
  - Key4hep maintains repository with additional packages
- The whole Key4hep software stack can be built from scratch using spack

```
spack install key4hep-stack
```

Key4hep resources and releases

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

```
/cvmfs/sw.hsf.org/key4hep/setup.sh
/cvmfs/ilc.desy.de/key4hep/setup.sh
```

  - Built via spack ($\sim$ 400 SW packages)
- 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
- Main **documentation at** key4hep.web.cern.ch

Key4hep - What's in it for LUXE?

- Profit from already existing solutions and experience
  - Many tools for different tasks already exist
  - Managing a consistent software stack and releases
- Common solutions reduce overhead
  - Keeping things running
  - Integrating new developments
  - Onboarding new people
- Common solutions live longer
  - Analysis and data preservation

Why does Key4hep care about LUXE?

- Integration of novel (and different) reconstruction and analysis techniques
- Another community to add to the list
  - Each community has unique experiences to contribute
  - *The more the merrier*
- Gaining experience from an actually running experiment
  - A big step on a slightly smaller scale

Open questions in adapting Key4hep for LUXE

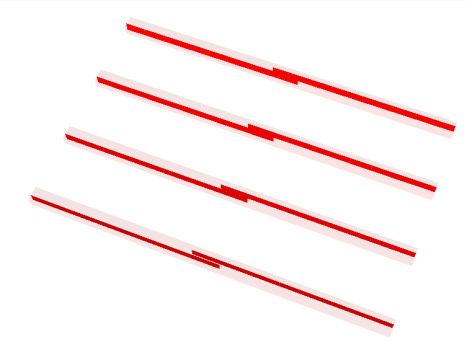
- Key4hep (currently) with strong focus on future collider experiments / Higgs factories
- Some of LUXEs detectors are quite different to the ones in HEP collider experiments
- Existing algorithms and tools (implicitly) assume some sort of 4π detector layout
- (Very) different experimental conditions, require different reconstruction algorithms, tools and data representations
- **Need to decide which parts are useful for LUXE**

What has already happened

- Yee has implemented a first version of the positron tracker geometry
  - LUXESoftware/luxegeo
  - Plan to implement the full LUXE geometry with DD4hep
- Can run simulation, digitization with existing Key4hep software
- Work on reconstruction ongoing
- Dedicated Key4hep installation for LUXE at

`/cvmfs/ilc.desy.de/key4hep/luxe_setup.sh`

  - Some packages with different versions than Key4hep default
  - (Almost) trivial for a first shot

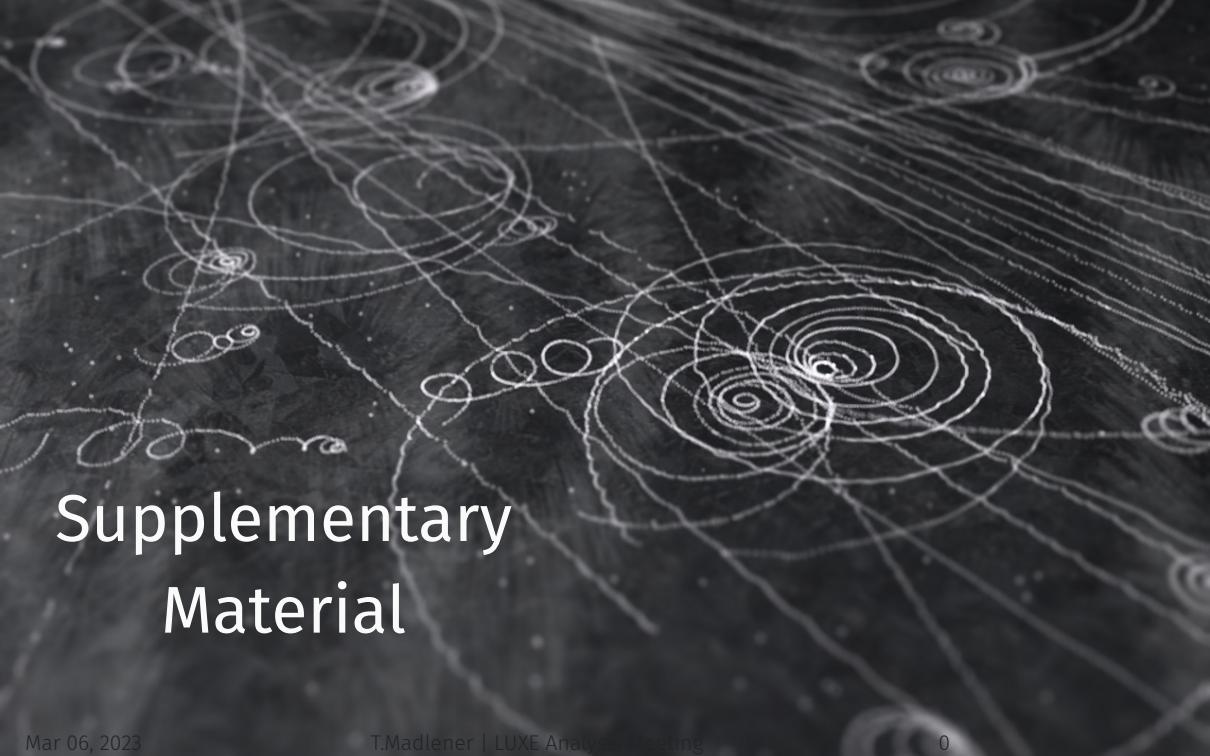


Summary

- Key4hep provides a common software stack for **all future (collider) projects**
- Very successful in **bringing together communities** and **focusing on common approaches**
  - Common **EDM4hep** for data exchange
  - **DD4hep** for detector description
  - Shared tools for building, developing and deploying software stack
- LUXE and Key4hep will both profit from a collaboration
  - Some work has already started to adapt Key4hep for LUXE
- Need to discuss and decide what and how we want to adapt Key4hep for LUXE

(Opinionated) discussion points (for this week and beyond)

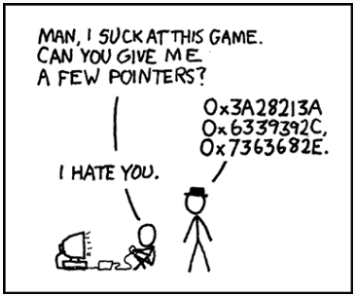
- Can we agree on / come up with a datamodel for representing LUXE measurements?
- Full detector description in DD4hep?
- Which parts of Key4hep do we want to use?
  - Which are already ready and easily usable for LUXE?
  - Which require work from LUXE, but would be nice to have in the long run?
- Who wants to (and can) do what?
  - We are happy to help get you started
- Details, details, details, ...



Supplementary Material

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
- iLCSoft
  - iLCSoft - github organisation
  - ilcsoft.desy.de



xkcd.com/138

Key4hep 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` Marlin proc. wrapper key4hep/k4MarlinWrapper
- Many packages migrated from FCCSW to Key4hep
  - `k4SimGeant4` for Geant4 simulation integration HEP-FCC/k4SimGeant4
  - `k4Gen` for generic generator interface HEP-FCC/k4Gen
  - ...
- Ongoing work to integrate more components
  - ACTS tracking framework acts-project/acts | key4hep/k4ActsTracking
  - CLUE fast clustering algorithms .cern.ch/kalos/CLUE | key4hep/k4CLUE

Ongoing work (selection)

ACTS integration

- ACTS can now digest DD4hep detectors (with annotations)
- Minimal EDM4hep I/O support
  - More general solution under discussion
- Major effort with significant personpower requirements

Gaudi modernization

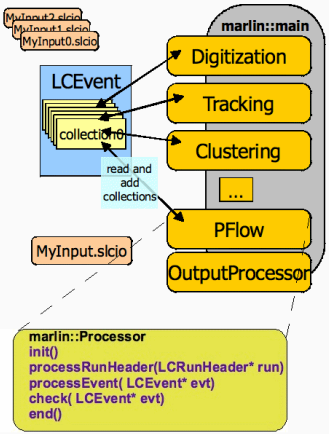
- Switch towards more modern Gaudi approach (*Gaudi Functional*)
  - “Thread safe by default”
- Missing documentation is a major hurdle

“Framework independent” algorithms

- EIC chose Jana2 over Gaudi
- Can “the hard part” still be shared?

Reconstruction and Analysis with Marlin

- `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



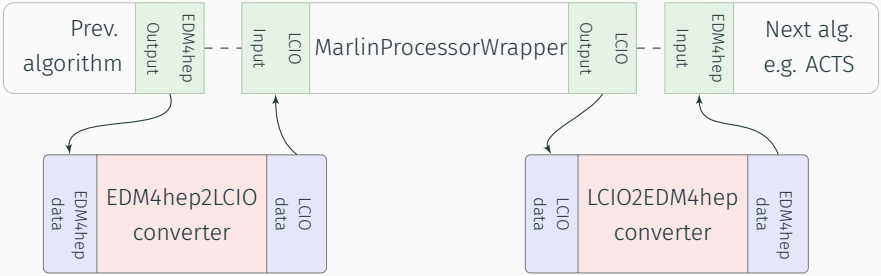
Marlin vs Gaudi

- Conceptually the two frameworks are very similar
  - Schedule different working units
  - Marshall data
- 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`

k4MarlinWrapper

- Wraps **Marlin processor** in a Gaudi algorithm and allows to **run them unchanged**
- Automatic, on-the-fly conversion between LCIO and EDM4hep
- **Allows to "mix and match" existing reconstruction algorithms with new developments**



Spack recipe

```python
class Evtgen(CMakePackage):
    """EvtGen is a Monte Carlo event generator that simulates
    the decays of heavy flavour particles, primarily B and D mesons."""

    homepage = "https://evtgen.hepforge.org/"
    url = "https://evtgen.hepforge.org/downloads?f=EvtGen-02.00.00.tar.gz"

    tags = ["hep"]

    maintainers = ["vvolkl"]

    version("02.00.00", sha256="02372308e1261b8369d10538a3aa65fe60728ab343fcb64b224dac7313deb719")
    # switched to cmake in 02.00.00
    version(
        "01.07.00",
        sha256="2648f1e2be5f11568d589d2079f22f589c283a2960390bbdb8d9d7f71bc9c014",
        deprecated=True,
    )

    variant("pythia8", default=True, description="Build with pythia8")
    variant("tauola", default=False, description="Build with tauola")
    variant("photos", default=False, description="Build with photos")
    variant("hepmc3", default=False, description="Link with hepmc3 (instead of hepmc)")

    patch("g2c.patch", when="@01.07.00")
    patch("evtgen-2.0.0.patch", when="@02.00.00 ^pythia8@8.304:")

    depends_on("hepmc", when="~hepmc3")
    depends_on("hepmc3", when="+hepmc3")
    depends_on("pythia8", when="+pythia8")
```

Build system

Where to find source code

Available versions

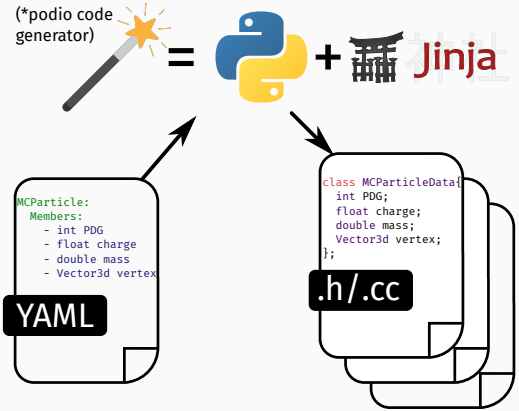
Variants / build options

On-the-fly patches

Dependencies

podio as generator for EDM4hep

- Traditionally HEP c++ EDMs are heavily Object Oriented
- Use **podio** to generate thread safe code starting from a high level description
- Provide an easy to use interface to the users



AIDASoft/podio

podio supports different I/O backends

- Default **ROOT** backend
  - POD buffers are stored as branches in a `TTree`
  - Files can be interpreted **without EDM library**(!)
  - Can be used in `RDataFrame` or with `uproot`
- Alternative **SIO** backend
  - Persistency library used in `LCIO`
  - Complete events are stored as binary records
- Adding more I/O backends is possible