

A common software for future colliders: The Key4hep turnkey software stack

Lepton-Photon 2021

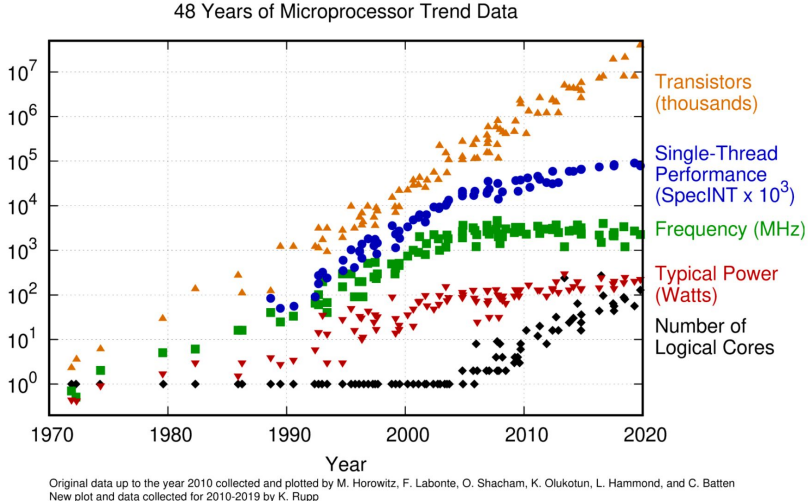
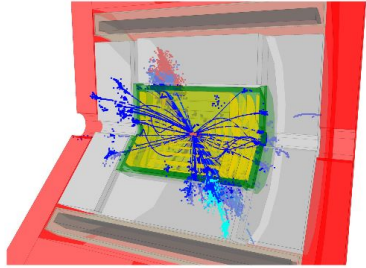
Jan 12, 2022

Valentin Volk, for the Key4hep SW group
CERN

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (<https://cds.cern.ch/record/2649646/>, CERN-OPEN-2018-006).

Software Challenges for (Future) Detector Developments

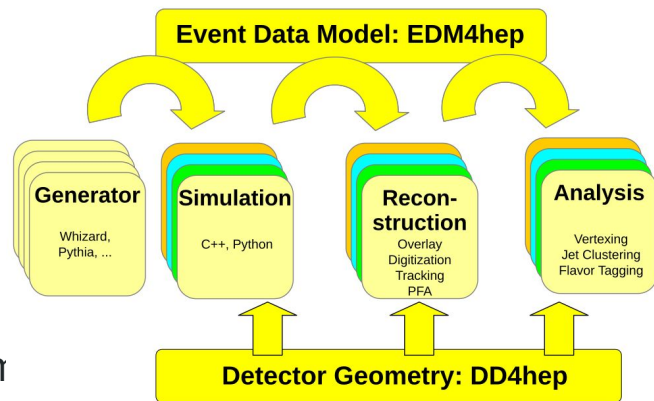
- Complex workflows
 - MC Simulations: Event Generation, Particle Propagation; Backgrounds, Digitization, Reconstruction, Analyses ...
- Performance
 - Need distributed computing infrastructure
 - and parallel programming to use evolving hardware efficiently
- Advantage: Not all features needed right away
 - ... gives time for proper design!



Goals of the Key4hep project

Software stack that connects and extends packages to provide a complete data processing framework, comprising fast and full simulation, reconstruction, and analysis.

- Contributions from different Future Collider communities
 - FCC, CLIC, ILC, CEPC, ...
- Consistent choice of technologies for interoperability
 - EDM4hep: datamodel
 - Gaudi: framework
 - DD4hep: geometry information
 - Spack: package manager
- Ease of use for librarians, developers and users
- Provide examples, documentation, templates and common practices



Status of Experiment Software

key4hep-stack/2021-10-29 comprises

...in a consistent stack!

CEPCSW

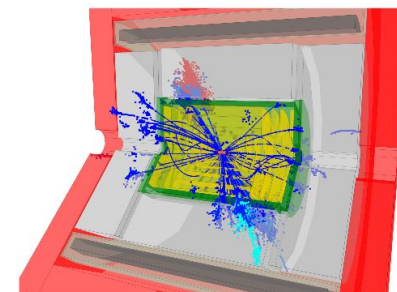
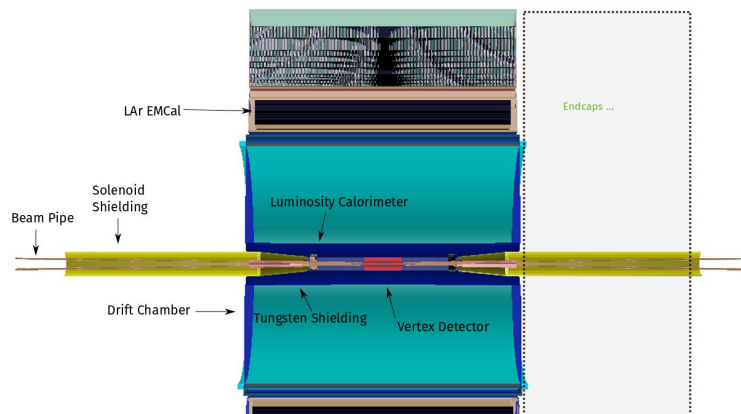
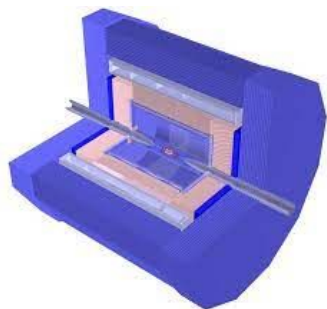
v0.2.2

FCCSW

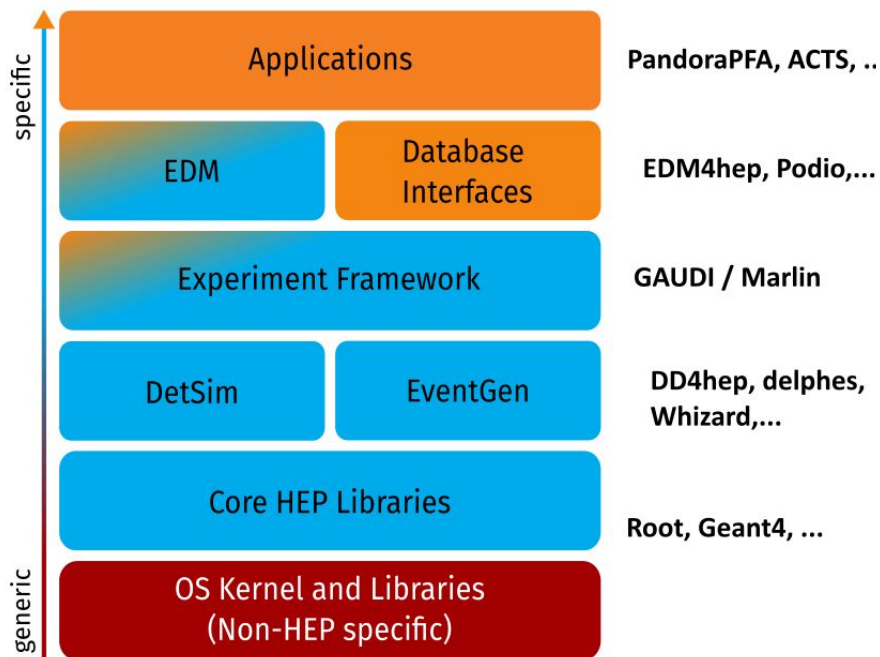
v1.0pre06

CLIC/ILCSofT

v02-02-03



HEP Software stack



- Pieces of software are not living in isolation: interaction between 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

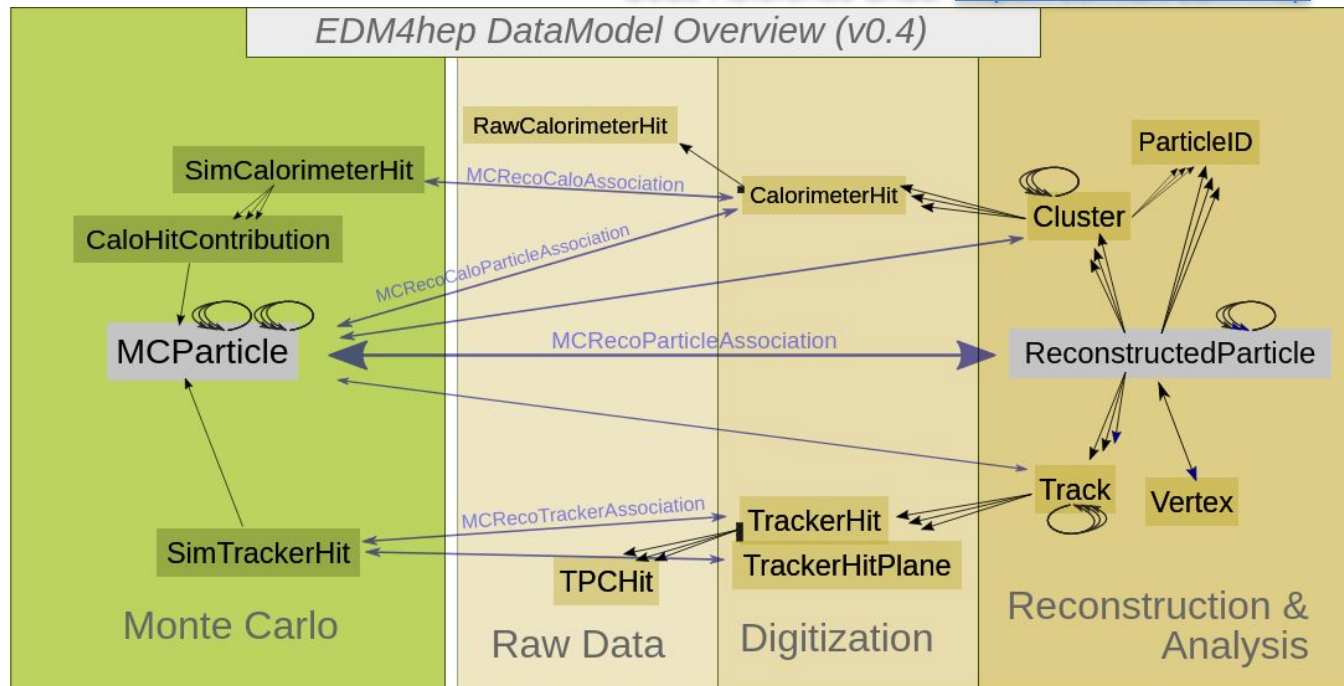
EDM4hep

To facilitate interoperability, different components should talk the same language

In HEP, this is the Event Data Model: describes structure of HEP data

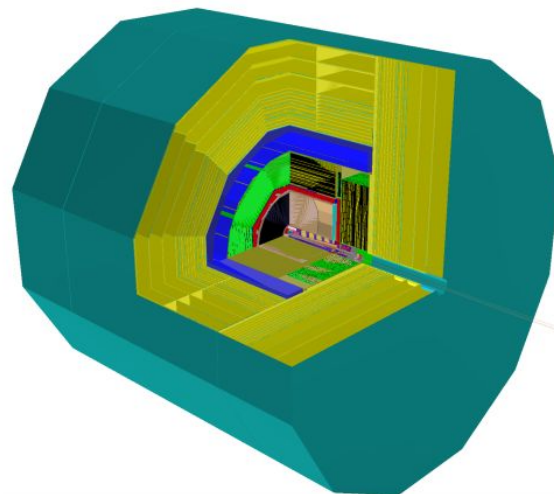
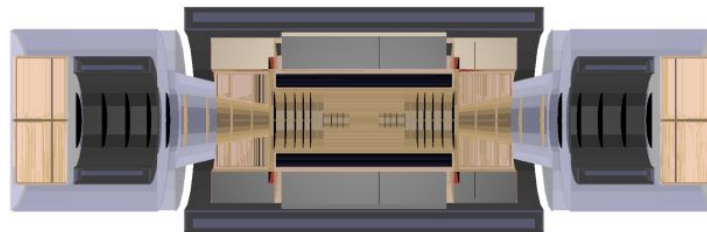
Code Reference under <https://cern.ch/edm4hep>

- Based on LCIO and FCC-edm
- Focus on: fast, efficient, multithreaded and transparent I/O system
- Implemented with PODIO - [vCHEP 2021: EDM4hep and podio](#)



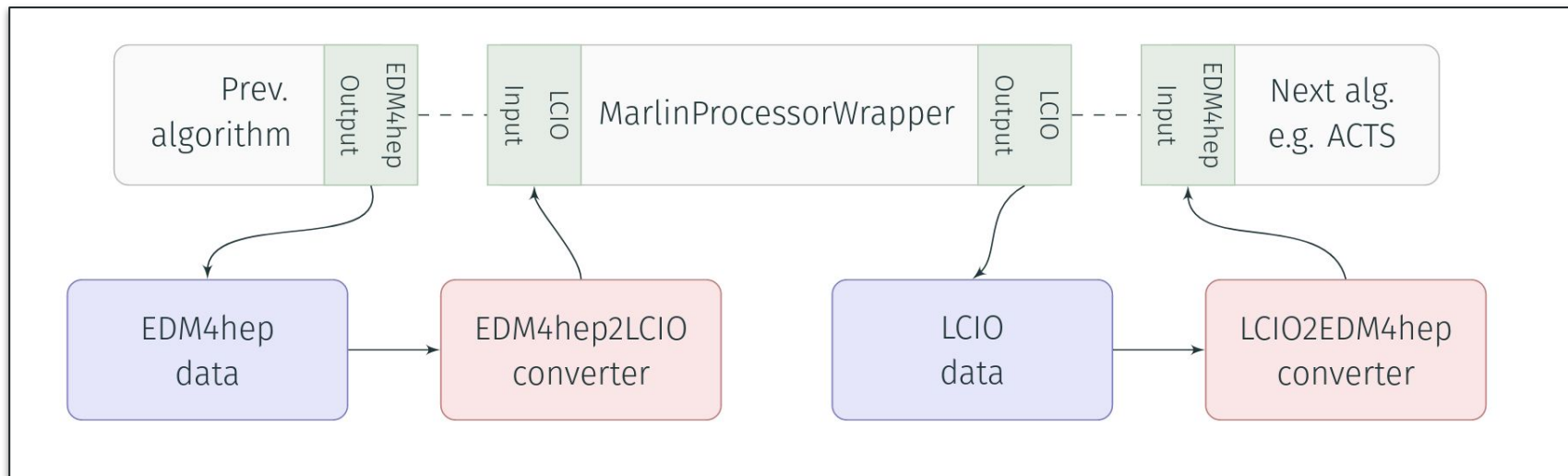
DD4hep - Detector Description Toolkit for HEP

- 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” by now
 - ILC, CLIC, FCC, CEPC, EIC, ..
 - LHCb and CMS use DD4hep for Run3
- `ddsims` already directly outputs EDM4hep
 - When writing `...edm4hep.root` files



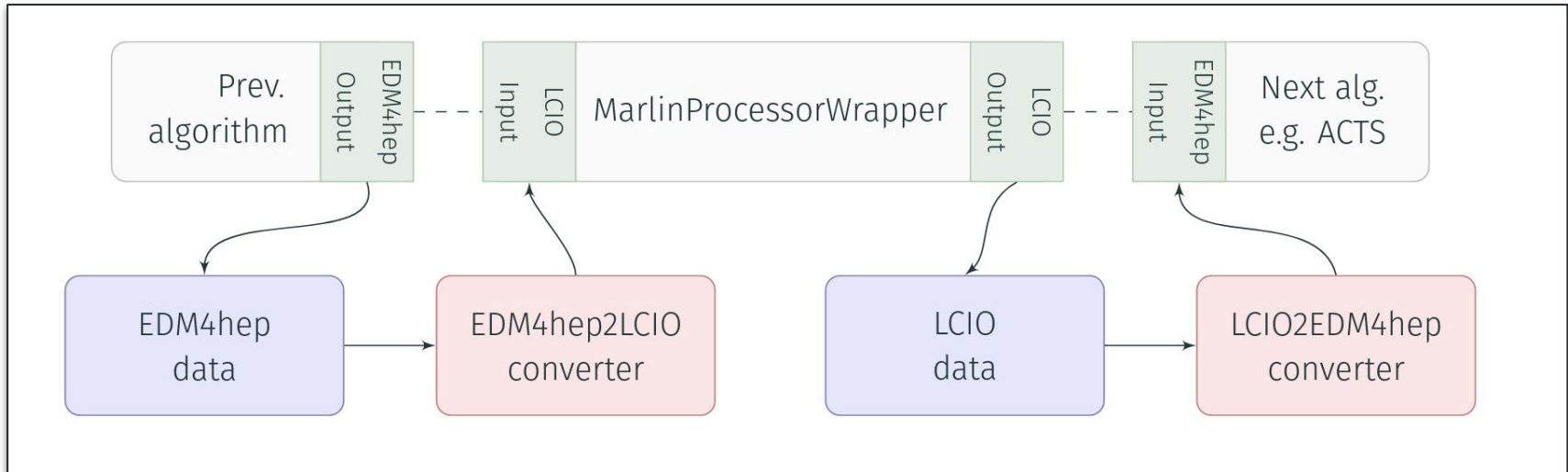
Reconstruction

- Ongoing integration of new experiment-independent reconstruction packages
 - ACTS: k4ActsTracking
 - CLUE: k4Clue
- iLCSoft Reconstruction chain usable through **k4MarlinWrapper**
 - Allows running all existing *Marlin* processors from iLCSoft in the Key4hep Gaudi framework



k4MarlinWrapper: Run iLCSoft through Gaudi

- Run *Marlin* Processors through *Gaudi* Algorithms, without changes to *Marlin*
- Enhanced *Marlin* XML to *Gaudi* Python steering files converter
 - Optional execution markers
 - *Constants* parsing: propagated through the steering file
- Input and output support for LCIO and EDM4hep

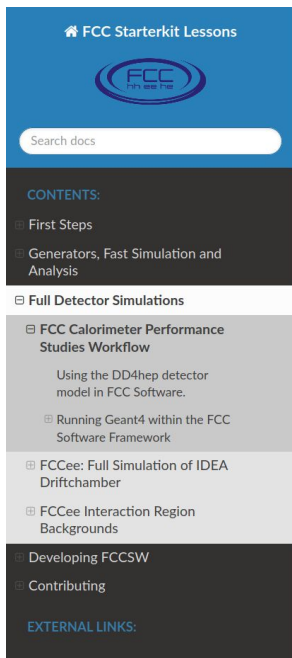


Software Infrastructure

- Documentation
 - cern.ch/key4hep (main documentation)
 - cern.ch/edm4hep (doxygen code reference)
- Spack package manager to build whole stack from source
 - Allows dealing with multiple package configurations and system architectures
 - Several versions, compilers, external library versions, ...
- Automated builds and continuous integration (CI) where possible
 - Regular nightly builds of the complete stack
- Distribution via CVMFS
 - Latest release can be found at `/cvmfs/sw.hsf.org/key4hep`
- Release early and often
 - Make fixes available early
 - Discover problems and collect feedback as early as possible

Tutorials: cern.ch/key4hep

- Work in progress to have automatically tested examples of software use
- Ongoing effort to run notebooks on SWAN (swan.cern.ch)



The screenshot shows the navigation menu for 'FCC Starterkit Lessons'. At the top is the FCC logo and a search bar. Below is a 'CONTENTS:' section with a list of topics: 'First Steps', 'Generators, Fast Simulation and Analysis', 'Full Detector Simulations', 'FCC Calorimeter Performance Studies Workflow', 'Running Geant4 within the FCC Software Framework', 'FCCee: Full Simulation of IDEA Driftchamber', 'FCCee Interaction Region Backgrounds', 'Developing FCCSW', and 'Contributing'. At the bottom is an 'EXTERNAL LINKS:' section.

Docs » [Full Detector Simulations](#) » FCC Calorimeter Performance Studies Workflow

[Edit on GitHub](#)

FCC Calorimeter Performance Studies Workflow

📖 Learning Objectives

This tutorial will teach you how to:

- **simulate** the single particle response of the calorimeter detector system
- **reconstruct** physics object from raw signals
- produce **plots** of energy resolutions and other quantities.

First, make sure your setup of the FCC software is working. You can check that the command to run jobs in the Gaudi framework is available on the command line:

```
which fccrun
```

If you don't see a valid path like `/usr/local/bin/fccrun` you should consult [the documentation page on FCCSW setup](#)

Using the DD4hep detector model in FCC Software.

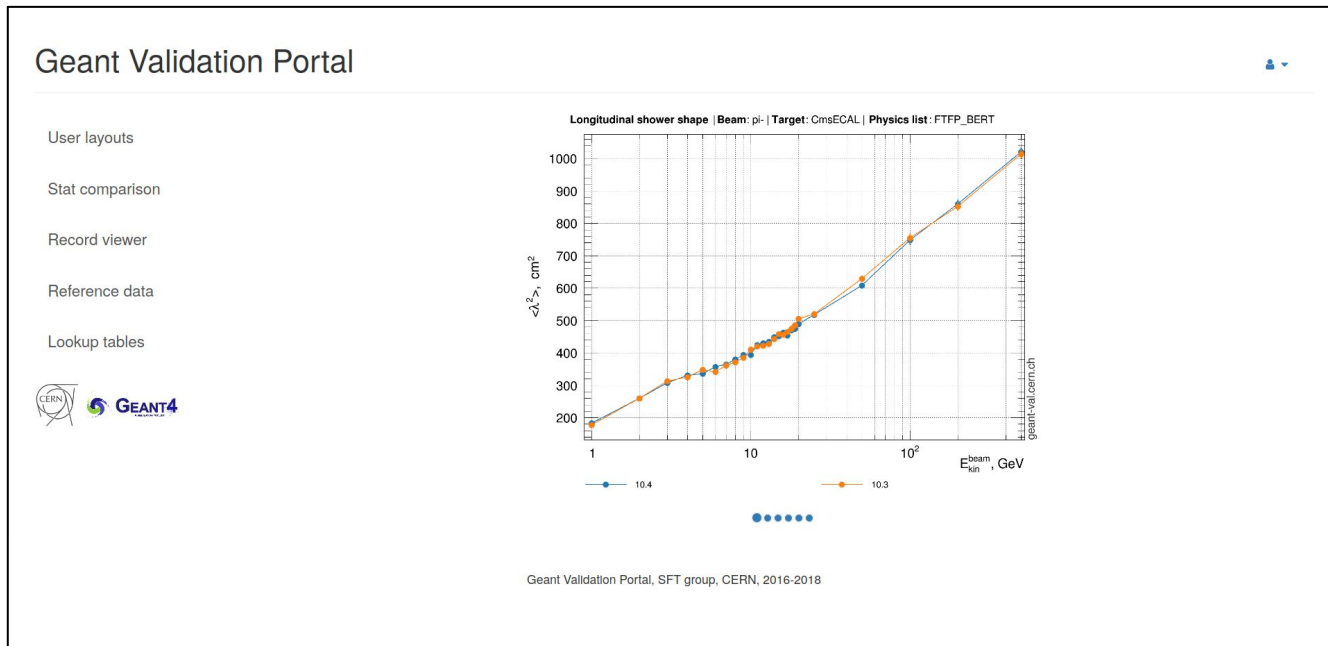
The Geant4 geometry that is used for the full simulation of the detector is not written directly, but generated using the DD4hep library. The detector description in this library consists of two parts: A compiled C++ library that constructs the geometry, and a set of xml files that contain parameters

Performance Plots and Comparison

Need for common package to evaluate detector performance

[iLCSoft/ILDPerformance: Package to evaluate the Performance of the ILD detector simulation](#)

Investigate tooling of
geant-val.cern.ch

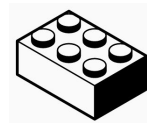
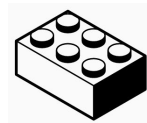
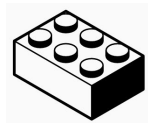


Summary and outlook

- Opportunity for HEP community to develop a common experiment software ecosystem, for the first time
 - Leverage synergies between communities and existing experiments to develop and maintain common core tools
- Projects already using EDM4hep/Key4hep in their software:
 - FCCSW, ILC, CLIC & CEPC
- Further consolidation and R&D, use of new common reconstruction packages ongoing

Backup

Modular approach



Updated style of “job option files” allows for easier re-use of parts of a job

```
# Geant4 algorithm
# Translates EDM to G4Event, passes the event to G4, writes out outputs via tools
from Configurables import SimG4Alg
geantsim = SimG4Alg("SimG4Alg")
from Configurables import SimG4PrimariesFromEdmTool
geantsim.eventProvider = SimG4PrimariesFromEdmTool("EdmConverter")
geantsim.eventProvider.GenParticles.Path = "GenParticles"
ApplicationMgr().TopAlg += [geantsim]
```

... even python-style import of configuration blocks!

```
from k4_workflow_blocks.fccsw.detector_fcc_hh_main import *
```