The Turnkey Software Stack: Where Are We and Where We Want to Go

IAS HEP 2020
Experiment / Detector / Software Mini-Workshop

Hong Kong, 17 Jan 2020
G Ganis, CERN

Where did we start

- Kick-off workshop on common software for future experiments
  - Bologna workshop, June 2019
- Present: LHC, ILC, CLIC, FCC, CEPC, SCTF, HSF
- Agreed to:
  - Investigate the possibility to have a common event data model
    - EDM4hep, gererated using PODIO

    Contribute to the development of a Common Turnkey Software Stack
    - Key4hep
  - One framework (Gaudi best candidate),

    DD4hep, EDM4hep, Geant4, ROOT, ...

What happened since Bologna

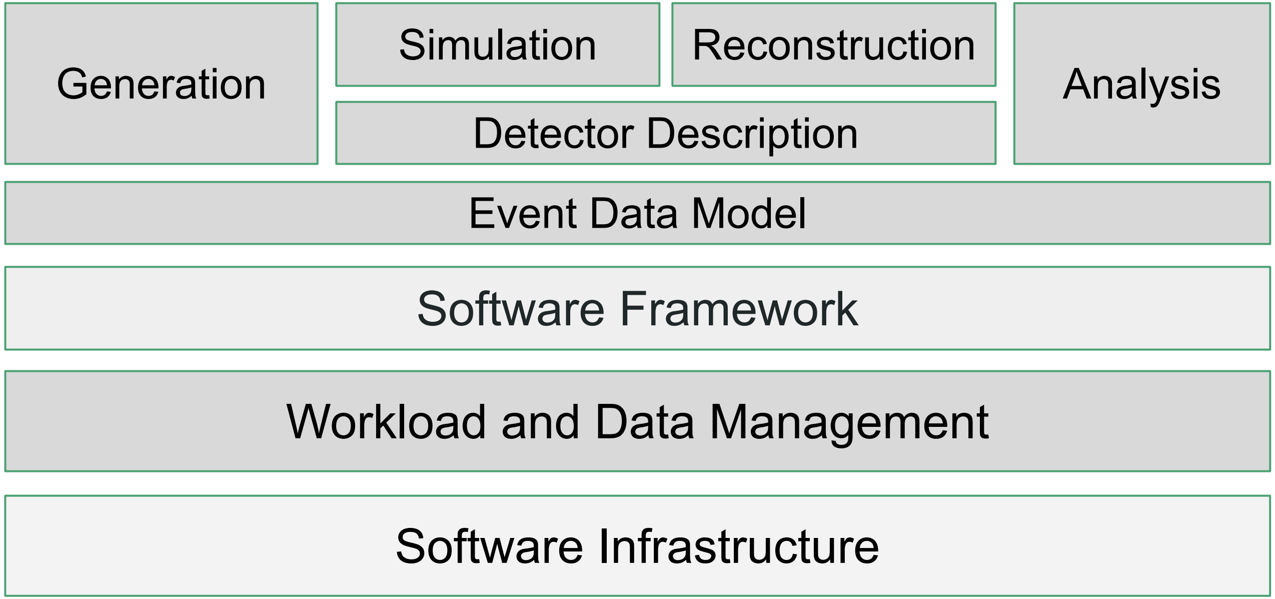
- Manpower *quest*
  - AIDA++ software submissions included Key4hep as R&D line
  - EP software R&D working package got concrete
    - A fellow hired, started Jan 2020 (V Volkl)
  - Second CERN fellow hired on CREMLIN PLUS funds
    - starting in March 2020
- Dissemination of the idea
  - Collaboration meetings: FCC, CLIC, ...
  - Talk at CHEP 2019
- Start discussions / work on EDM4hep
  - July 2019

Disclaimer

- The talk reflects personal ideas of what should key4hep provide
- Concrete examples are from FCCSW, which I know better
    - They are meant to illustrate how a framework based on Gaudi, <podio-edm>, DD4hep, Geant4, ROOT, … could look like

- Everything of course is up for discussion

Overview of a Software Stack Components

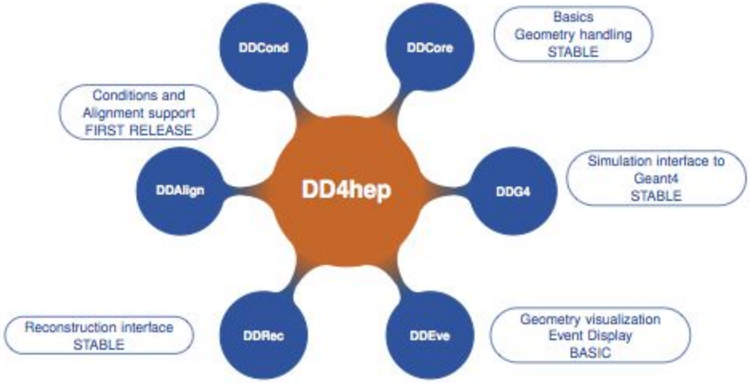


EDM4hep status

- Task force meeting every 2-4 weeks since July 2019
    - Representatives from all future collider projects and LHC
- Started from comparing/merging FCC-EDM and (p)LCIO
    - Git repository available at EDM4hep
- Based on PODIO high-level EDM generator
    - As FCC-EDM, pLCIO
- Details in the next talk (F Gaede)

Detector geometry description: DD4hep

- Complete detector description
- Single source of information
- Support for the full-experiment life cycle
- Details in the next-to-next talk (A Sailer)



- For key4hep
  - Default palette of usable sub-detector solutions

Core Framework: Gaudi

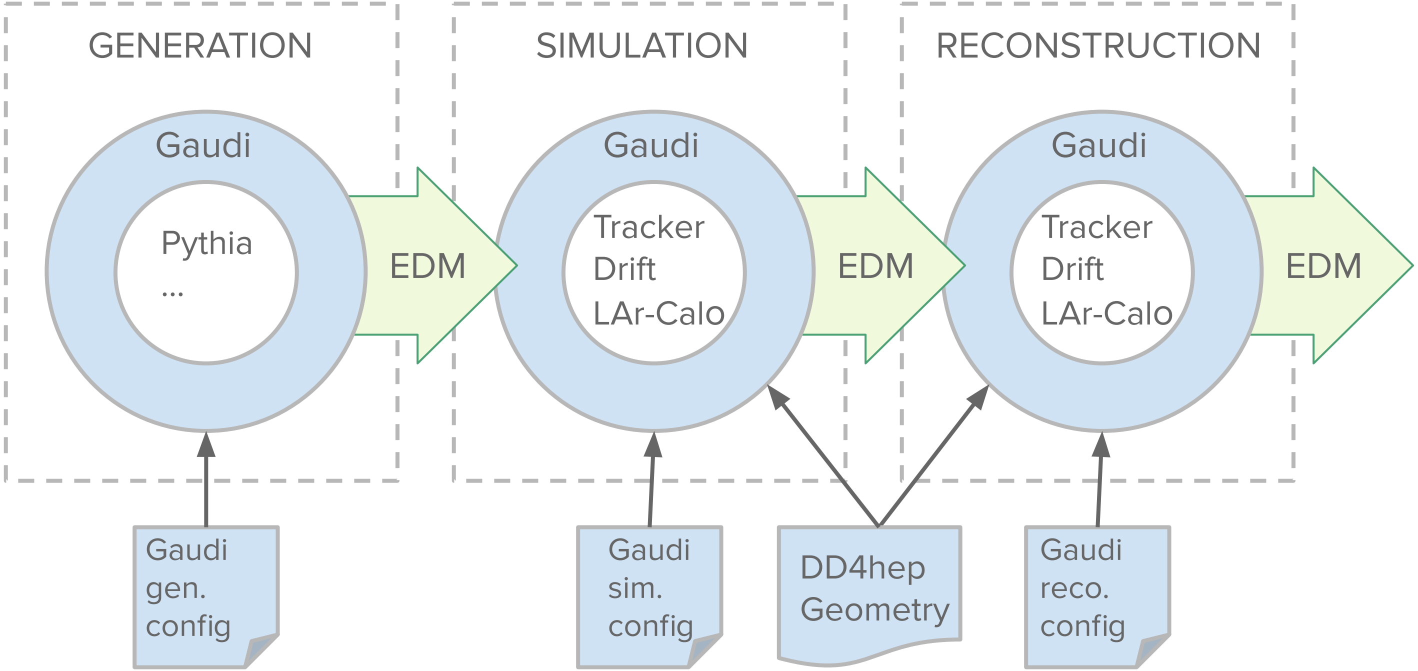
- Framework toolkit to provide required interfaces and services to build HEP experiment frameworks
  - Opensource project and experiment independent

- Data processing framework designed to manage experiment workflows
  - Separate data and algorithms; well defined interfaces
  - User’s code encapsulated in Algorithm’s, Tool’s / Interface’s, Service’s
  - Different persistent and transient views of data
  - C++, with Python configuration

- Originating from LHCb, Gaudi is adopted also by ATLAS
  - Actively developed to face LHC Run 3 and Run 4 challenges (high PU)

- Latest version: v33r0, fully licensed (Apache), support Python 3

Gaudi @ FCCSW


GENERATION
SIMULATION
RECONSTRUCTION
Gaudi
Pythia
...
EDM
Gaudi
Tracker
Drift
LAr-Calo
EDM
Gaudi
Tracker
Drift
LAr-Calo
EDM
Gaudi gen. config
Gaudi sim. config
DD4hep Geometry
Gaudi reco. config

Structure of FCCSW repository

FWCore	Data service
Generation	Handling of process generation
Sim	Full/Fast/Parametrized simulation
Detector	Digitization / Detector response
Reconstruction	Reconstruction algorithms
Examples	Example of python configuration files
…	

Each module defines Gaudi algorithms and tools which can be used to configure the application

fccrun command

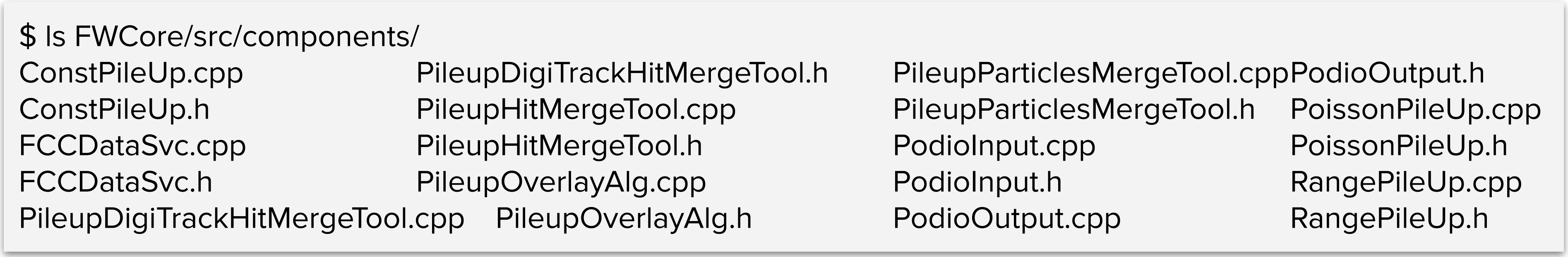
- Python script starting gaudimain
- High-level control of the job to be run

```
fccrun [-h] [--dry-run] [-v] [-n NUM_EVENTS] [-l] [--gdb] [--ncpus NCPUS] [config_files [config_files ...]]

  -h, --help          Show this help message and exit
  --dry-run           Do not actually run the job, just parse the config files
  -v, --verbose       Run job with verbose output
  -n NUM_EVENTS, --num-events NUM_EVENTS        Number of events to run
  -l, --list          Print all the configurable components available in the framework and exit
  --gdb               Attach gdb debugger
  --ncpus NCPUS       Start Gaudi in parallel mode using NCPUS processes.
                      0 => serial mode (default), -1 => use all CPUs
```

FWCore component

- Provides the data service FCCDataSvc handling input data
- Depends on fcc-edm, podio, ROOT
- Provides also services to overlay data files, used for the pileup which could be of more general use



```
$ ls FWCore/src/components/
ConstPileUp.cpp                  PileupDigiTrackHitMergeTool.h      PileupParticlesMergeTool.cppPodioOutput.h
ConstPileUp.h                    PileupHitMergeTool.cpp             PileupParticlesMergeTool.h   PoissonPileUp.cpp
FCCDataSvc.cpp                   PileupHitMergeTool.h               PodioInput.cpp               PoissonPileUp.h
FCCDataSvc.h                     PileupOverlayAlg.cpp               PodioInput.h                 RangePileUp.cpp
PileupDigiTrackHitMergeTool.cpp    PileupOverlayAlg.h               PodioOutput.cpp              RangePileUp.h
```

- Similar component in CEPCSW

Monte Carlo Generation in FCCSW

- The Generation repository provides
    - Pythia interface
    - Particle guns
    - Data format Converter tools
    - Utility tools to filter and smear MC particles
- MC generators are typically standalone codes
    - Noticeable exception is Pythia8, which provides a callable interface
- FCCSW interoperates MC generators mostly through common data formats
    - HepMC, LHEF
    - Pythia8 used to read LHEF files

Monte Carlo Generators and FCCSW

- Generators repository: GenSer @ LCG software stacks
  - Generator Service hosted by EP-SFT

    Collaboration with the authors and with the LHC experiments to prepare validated code for communities at the LHC
  - Actively used by ATLAS, LHCb, SWAN and some SME experiments
  - Deployed now via CernVM-FS and RPMs
- GenSer generators palette biased towards LHC
  - Good for FCC-hh, incomplete for lepton colliders
- General purpose generators such as Pythia8, Whizard, MadGraph5 available
  - But not much experience on how to use them effectively for lepton colliders

MC Generators for precision physics

- Existing generators not enough for the precision expected at the FCC-ee/CEPC at Z, WW, HZ
  - General purpose generators are not enough
  - LEP generators, or their immediate evolution KKMC, OKish in the short term but need improvement (or rewriting)
- GenSer integration of KKMC and BHLUMI on going
  - Wrappers to produce HepMC and/or LHEF output required
  - Similar work will be needed for MCSANC, BabaYaga, …
- This is also an area where joining efforts may help

Machine-Detector-Interface Software Integration

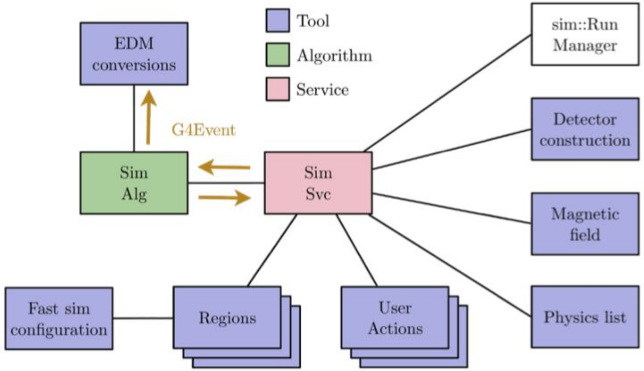
- (Beam- and) MDI- related backgrounds are source of systematics
    - Critical aspect is detector occupancy, possibly also radiation damage
    - Crucial during machine design phase
- Non-exhaustive list of programs to calculate these backgrounds
    - MDISim: *Synchtron Radiation, Single beam induced backgrounds*
    - SYNC_BKG, SYNRAD+: *Synchtron Radiation*
    - GuineaPig++: *IP backgrounds, (In)coherent Pairs Creation, γγ to hadrons*
    - Pythia8: *γγ to hadrons*
    - BBBrem+SAD: *Radiative Bhabhas*
- The programs are the same but the result depends on the interaction region, which is different for each project
- The technology and the code repositories, may be in common

MDI possible workflow

- Integrating the MDI calculations through shared data formats
- MDI code provides sets of events with the 4-vectors and vertex of the relevant particles
  - γ‘s for SR; e+e- pairs, hadrons for IP processes, ...
  - May include the interaction in the beam-pipe (as in MDISim)
- Evaluation of detector occupancy in key4hep as for other signals/bkgs
  - Through interaction of MDI particles in the detector
  - Through overlay of MDI events to “signal” events for a more detailed background simulation
  - This may also be done with a weighted mixture of MDI processes

Simulation and FCCSW

- Geant4
  - Gaudi components exists to create
    - User Actions
    - Regions
    - Sensitive detectors
    - Selective output options
  - Mixing fast and full G4 simulation possible
    - SimG4Full / SimG4Fast
- Delphes
  - Gaudi interface
    - FCC EDM output

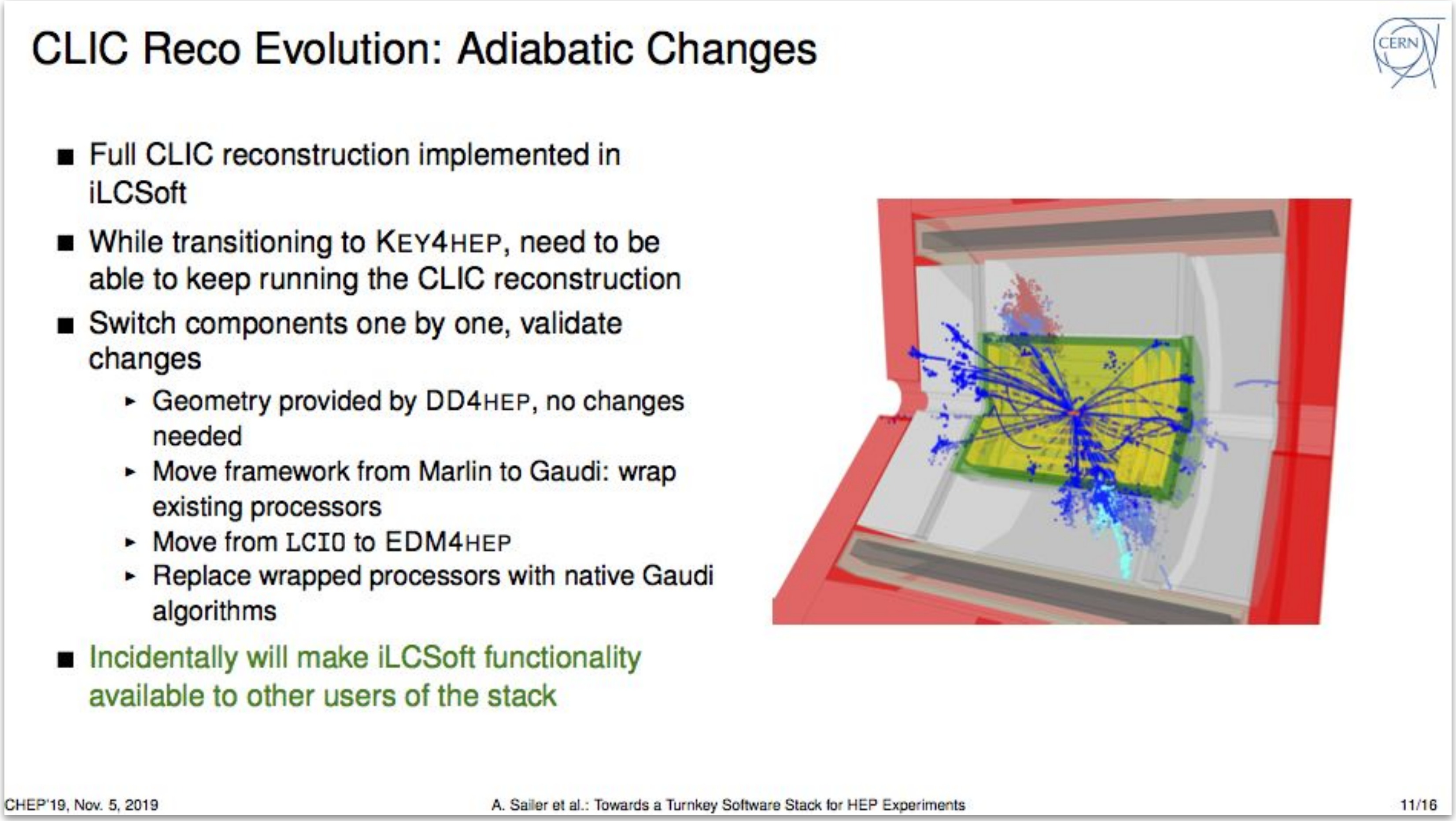


Reconstruction

- Reco algorithms are Gaudi algorithms tools interchangeable, in principle, at need
- Should aim at having a palette of algorithms for different purposes
- Available in FCCSW:
    - Tracking
        - Track seeding (TrickTrack) for silicon tracker
        - Hough Transform for drift chambers (not yet in the master)
        - Under implementation / investigation: ACTS integration, Conformal tracking
    - Calorimeters
        - Sliding window (rectangular/ellipse)
        - Topo-clustering
        - Under investigation: deep learning

Integration with existing frameworks - examples

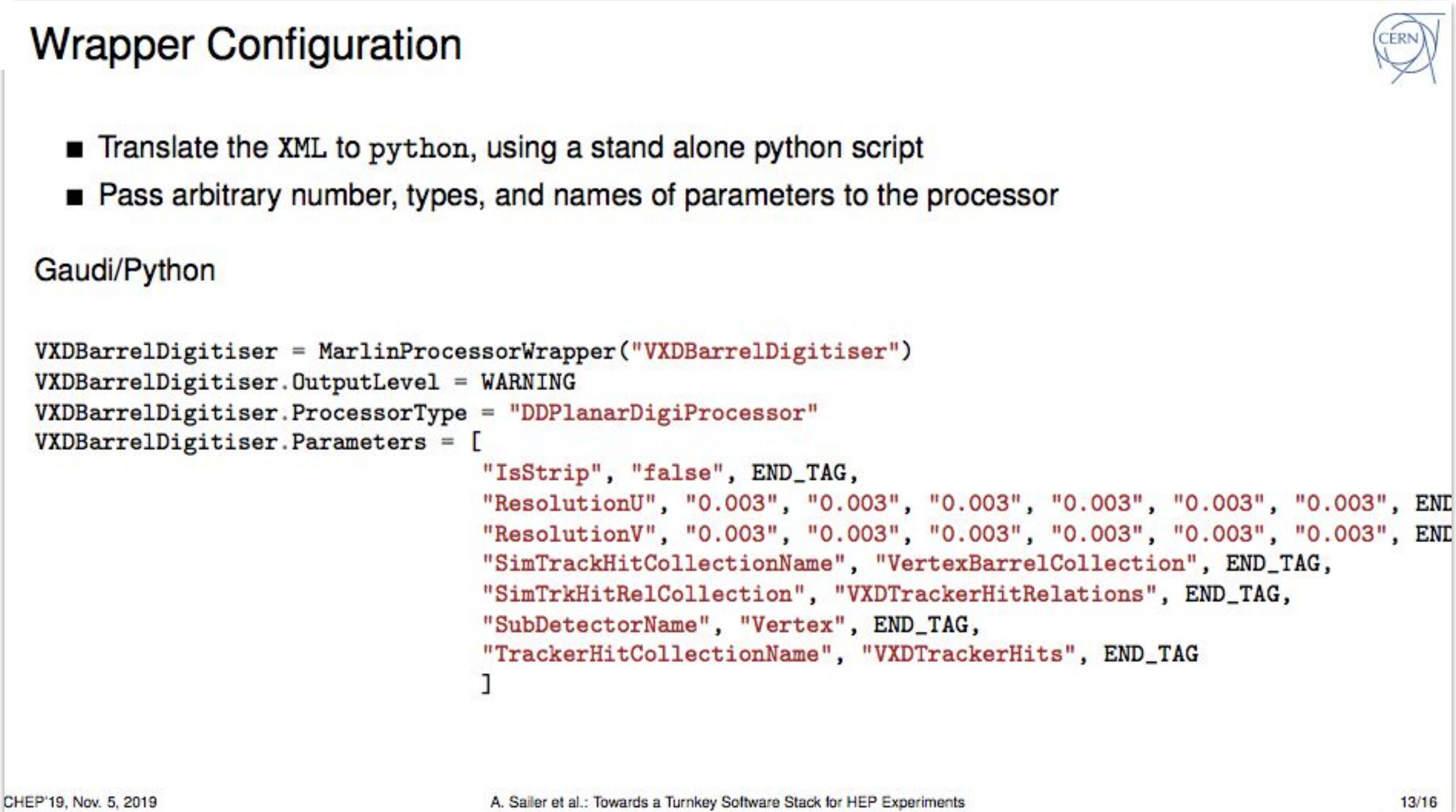
- CLIC
- FCC
- CEPC could possibly be very similar to FCC, given that they are based on Gaudi



CLIC Reco Evolution: Adiabatic Changes

- Full CLIC reconstruction implemented in iLCSoft
- While transitioning to KEY4HEP, need to be able to keep running the CLIC reconstruction
- Switch components one by one, validate changes
  - Geometry provided by DD4HEP, no changes needed
  - Move framework from Marlin to Gaudi: wrap existing processors
  - Move from `LCIO` to EDM4HEP
  - Replace wrapped processors with native Gaudi algorithms
- Incidentally will make iLCSoft functionality available to other users of the stack

CHEP'19, Nov. 5, 2019 A. Sailer et al.: Towards a Turnkey Software Stack for HEP Experiments 11/16



Wrapper Configuration

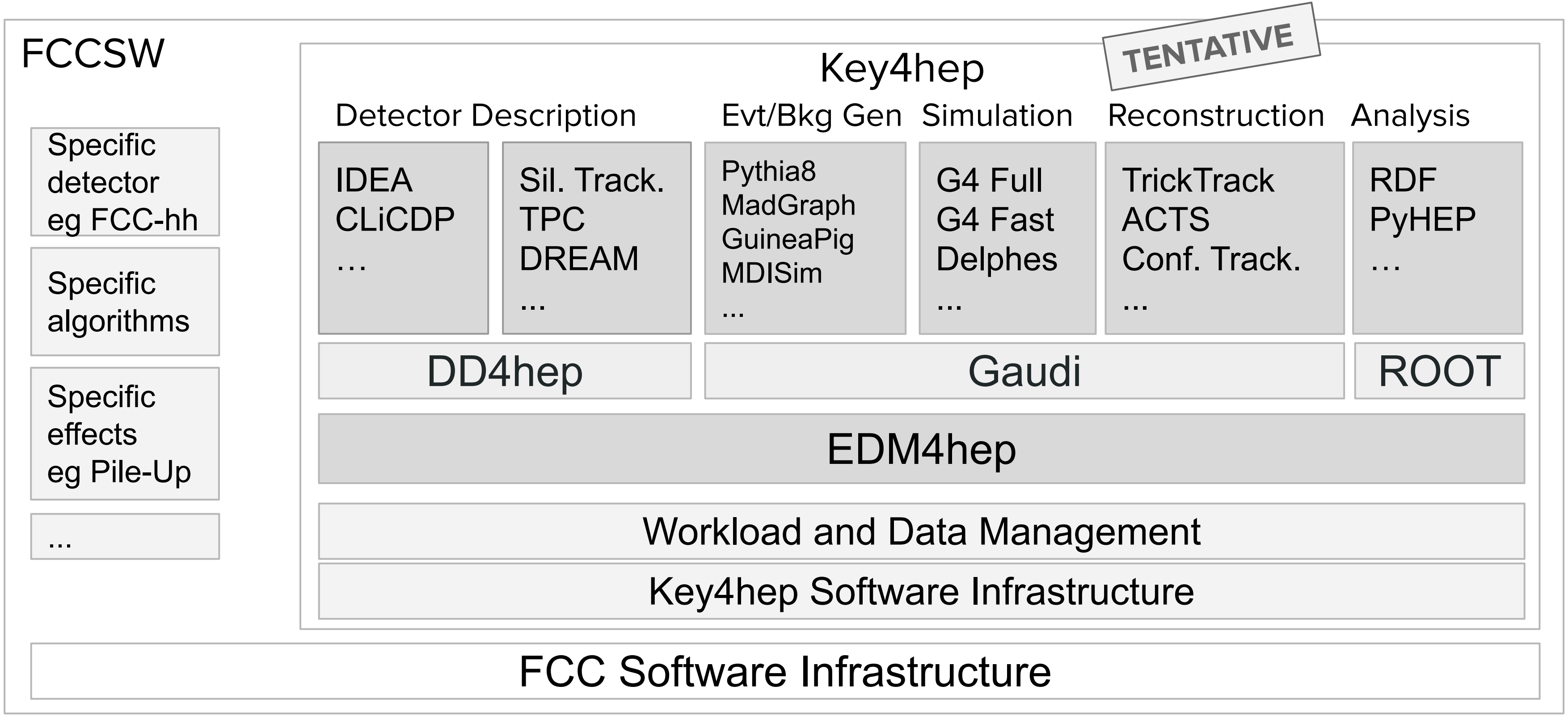
- Translate the `XML` to `python`, using a stand alone python script
- Pass arbitrary number, types, and names of parameters to the processor

Gaudi/Python

```
VXDBarrelDigitiser = MarlinProcessorWrapper("VXDBarrelDigitiser")
VXDBarrelDigitiser.OutputLevel = WARNING
VXDBarrelDigitiser.ProcessorType = "DDPlanarDigiProcessor"
VXDBarrelDigitiser.Parameters = [
                                  "IsStrip", "false", END_TAG,
                                  "ResolutionU", "0.003", "0.003", "0.003", "0.003", "0.003", "0.003", END
                                  "ResolutionV", "0.003", "0.003", "0.003", "0.003", "0.003", "0.003", END
                                  "SimTrackHitCollectionName", "VertexBarrelCollection", END_TAG,
                                  "SimTrkHitRelCollection", "VXDTrackerHitRelations", END_TAG,
                                  "SubDetectorName", "Vertex", END_TAG,
                                  "TrackerHitCollectionName", "VXDTrackerHits", END_TAG
                                  ]
```

CHEP'19, Nov. 5, 2019   A. Sailer et al.: Towards a Turnkey Software Stack for HEP Experiments   13/16

FCC - Connection with Key4HEP



FCCSW

Specific detector eg FCC-hh

Specific algorithms

Specific effects eg Pile-Up

...

Key4hep

TENTATIVE

Detector Description

IDEA
CLiCDP
…

Sil. Track.
TPC
DREAM
...

Evt/Bkg Gen

Pythia8
MadGraph
GuineaPig
MDISim
...

Simulation

G4 Full
G4 Fast
Delphes
...

Reconstruction

TrickTrack
ACTS
Conf. Track.
...

Analysis

RDF
PyHEP
…

DD4hep

Gaudi

ROOT

EDM4hep

Workload and Data Management

Key4hep Software Infrastructure

FCC Software Infrastructure

Other possible items

- Common Analysis tools
- Common Condition Database solutions
- Resource management
  - Job submission (DIRAC, …)
  - (Distributed) storage management (RUCIO, DOMA …)
  - File catalog
- Build/test Infrastructure
  - Packaging (exercise with Spack)
  - Deployement
  - Continuous Integration

Final considerations

- Steps forward depend crucially on EDM4HEP availability
- Once EDM4HEP is available, possible rapid development with
  - Key4HEP core ≈ FCCSW/CEPCSW core + EDM4HEP
  - Algorithms (FCCSW, CEPCSW, other) adapted in turn to EDM4HEP as the need comes
- Deliver early and often approach

Thank you!