Status of the simulation software in CEPCSW

Tao Lin

(on behalf of the software team)

lintao@ihep.ac.cn

IHEP

CEPC workshop

27 Oct 2023, Nanjing

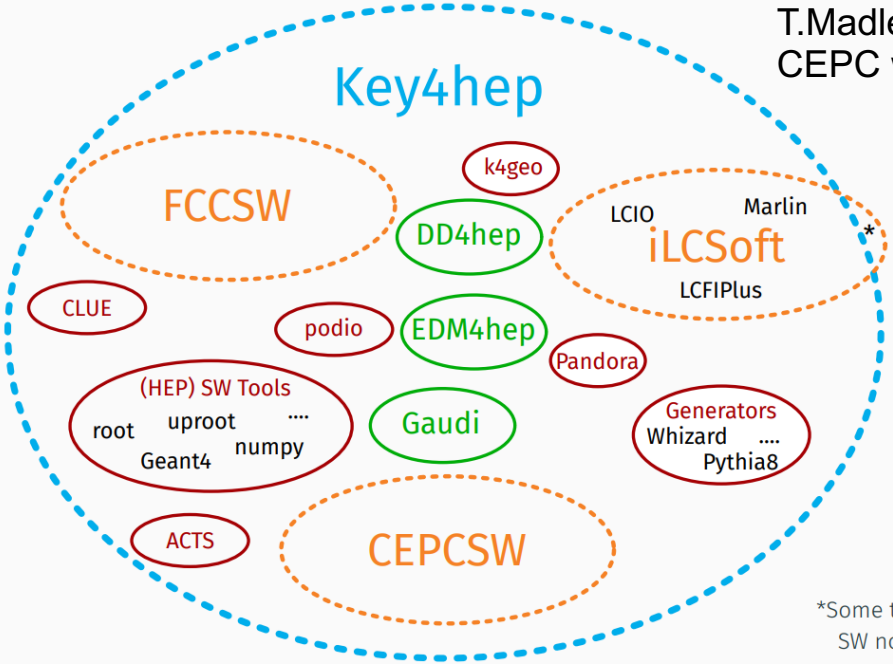
Outline

- Introduction
- Simulation framework in CEPCSW
- Gaussino-based simulation framework
- Parameterized simulation: k4SimDelphes
- Summary

Introduction (1)

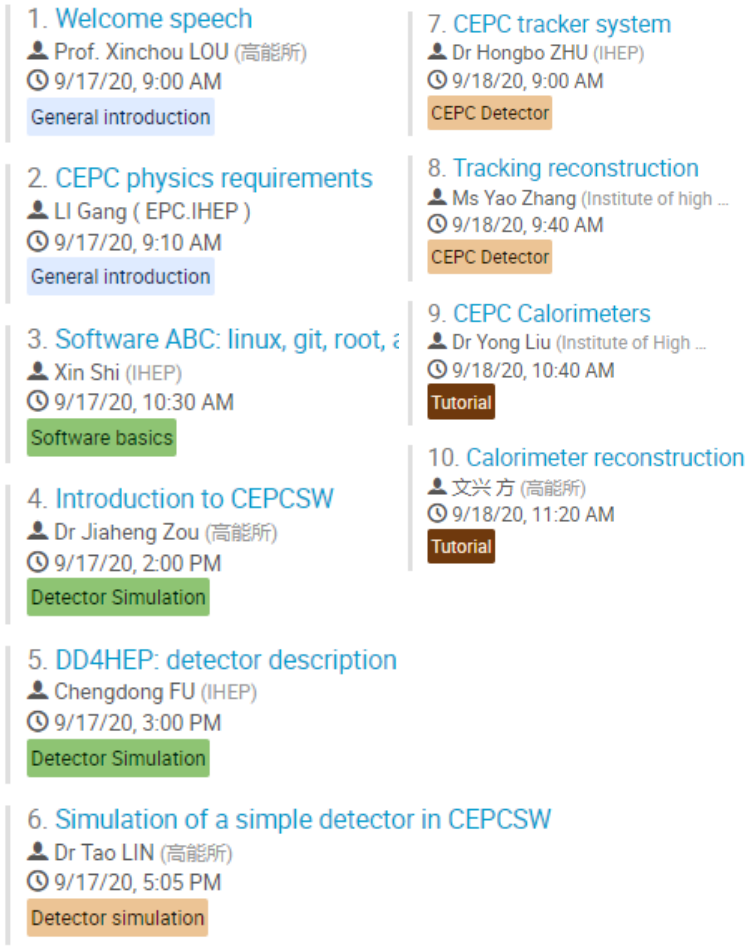
- New CEPC software (CEPCSW) prototype was proposed at the Oxford workshop in April 2019.
- The consensus among CEPC, CLIC, FCC, ILC and other future experiments was reached at the Bologna workshop in June, 2019.
  - Develop a Common Turnkey Software Stack (Key4hep) for future collider experiments

T.Madlener | Key4hep & EDM4hep
CEPC workshop, Edinburgh



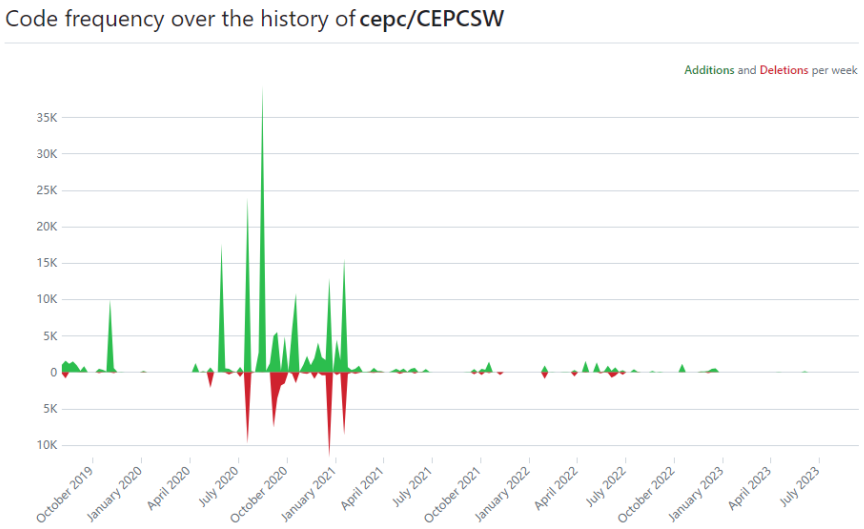
Introduction (2)

- As the first application based on Key4hep, following development activities had been initiated.
  - Development of k4LCIOReader: LCIO to EDM4hep converter
  - Migration of reconstruction algorithms
  - Development of simulation framework in CEPCSW
  - Validation between iLCSoft and CEPCSW.
- CEPCSW Tutorial and detector study, IHEP, 17-18 Sept 2020
  - Covering underlying framework, DD4hep, detector simulation and reconstruction within CEPCSW
  - https://indico.ihep.ac.cn/event/12341/



Introduction (3)

- Status
  - For detector design for CDR: code is already included in CEPCSW.
  - For 4th concept design: some code is not merged into official CEPCSW yet.



Code frequency in official CEPCSW

ECAL software

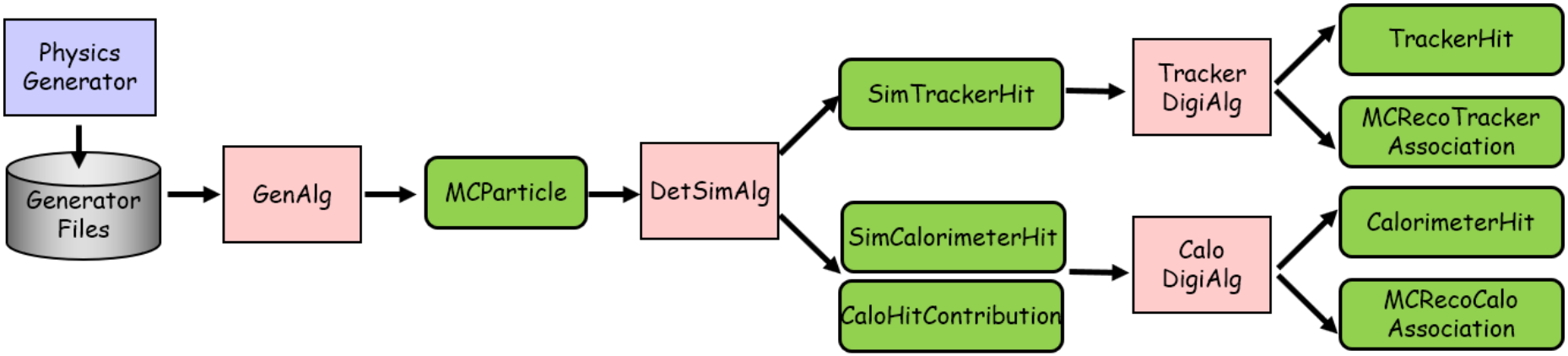
- Official: https://github.com/cepc/CEPCSW
- ECAL: https://github.com/qwert2333/CEPCSW
- Drift Chamber: https://github.com/myliu-hub/CEPC_Track_Finding
- VXD Test beam: https://github.com/ihep-sft-group/CEPCSW-VXDTestbeam

Outline

- Introduction
- Simulation framework in CEPCSW
- Gaussino-based simulation framework
- Parameterized simulation: k4SimDelphes
- Summary

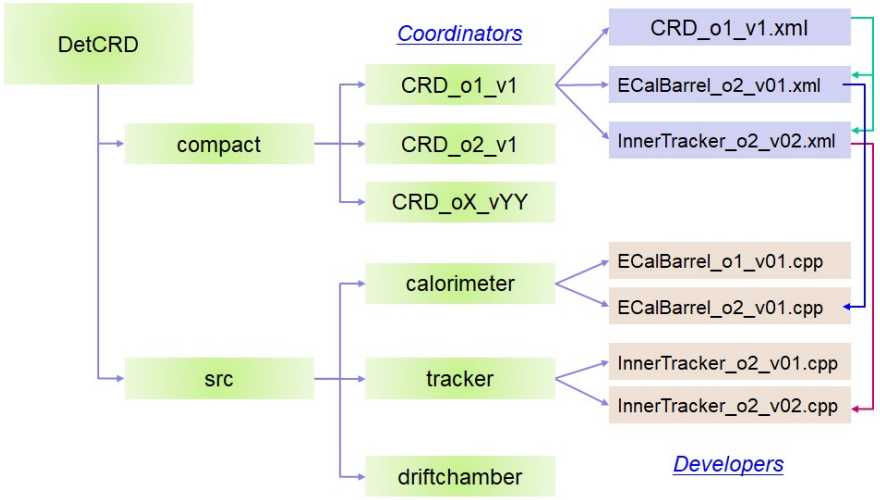
Simulation framework in CEPCSW (1)

- Complete simulation chain with EDM4hep
  - Physics generator
    - MCParticle
  - Detector Simulation based on Geant4
    - MCParticle (with secondaries), SimTrackerHit, SimCalorimeterHits
  - Digitization
    - TrackerHit, CalorimeterHit



Simulation framework in CEPCSW (2)

- Geometry management with DD4hep
  - Consists of C++ constructors and XML based compact files
  - https://github.com/cepc/CEPCSW/tree/master/Detector



For the CRD detector models, see README by Chengdong:
https://github.com/cepc/CEPCSW/tree/master/Detector/DetCRD/compact

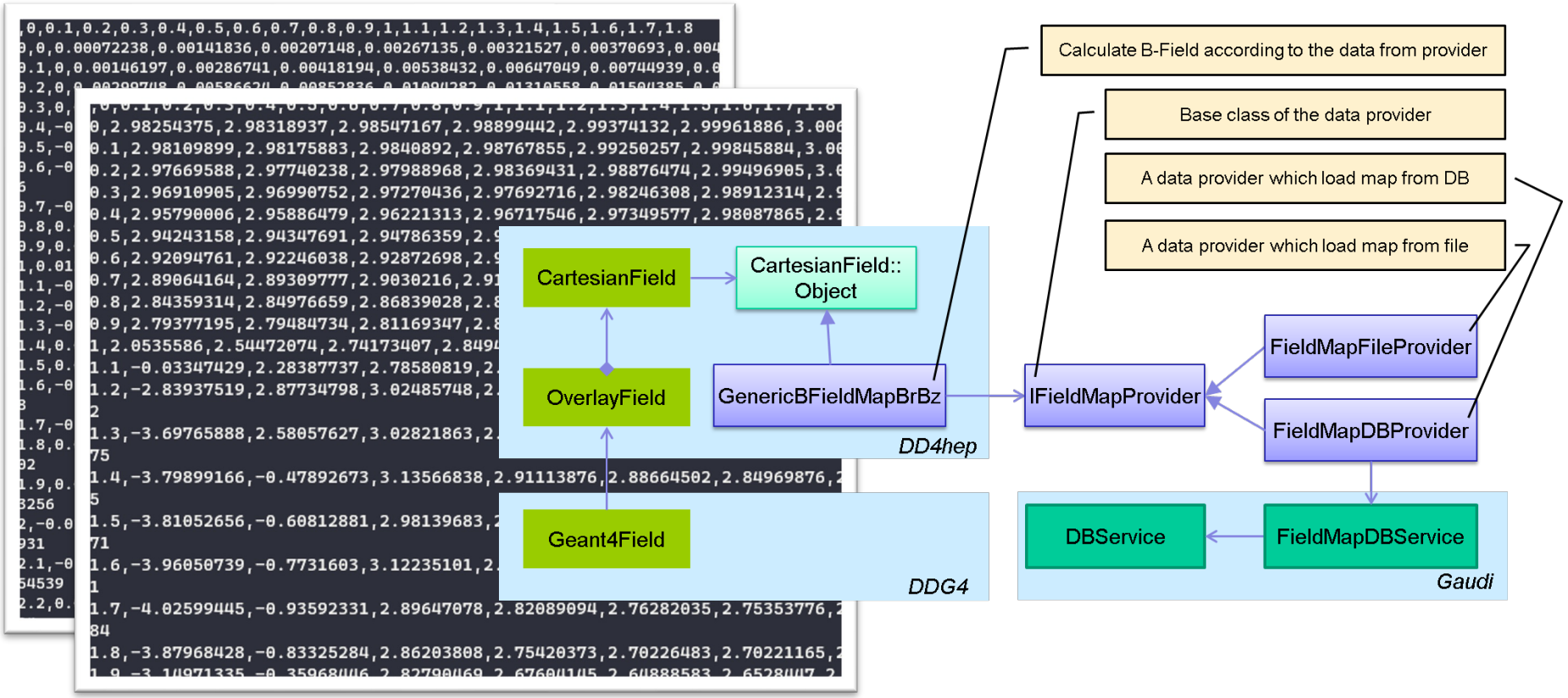
CRD detector models - Overview

The following CRD detector models are available in CEPCSW

Model	Description	MainTracker	Ecal	Hcal	Status
CRD_o1_v01	coil inside simulation model	SIT+DC+SET	crystal	RPC	developing
CRD_o1_v02	strip SET	SIT+DC+SET	crystal	RPC	developing
CRD_o1_v03	MOST2 vertex	SIT+DC+SET	crystal	RPC	developing
CRD_o1_v04	smaller center beam pipe	SIT+DC+SET	crystal	RPC	developing
---	---	---	---	---	---

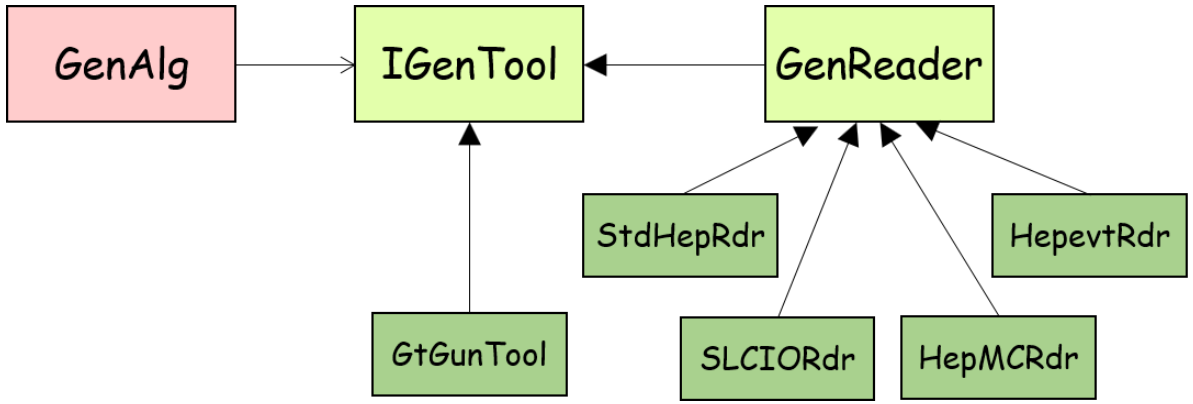
Simulation framework in CEPCSW (3)

- Non-uniform magnetic fields
  - The Br/Bz csv files are provided by magnetic group.



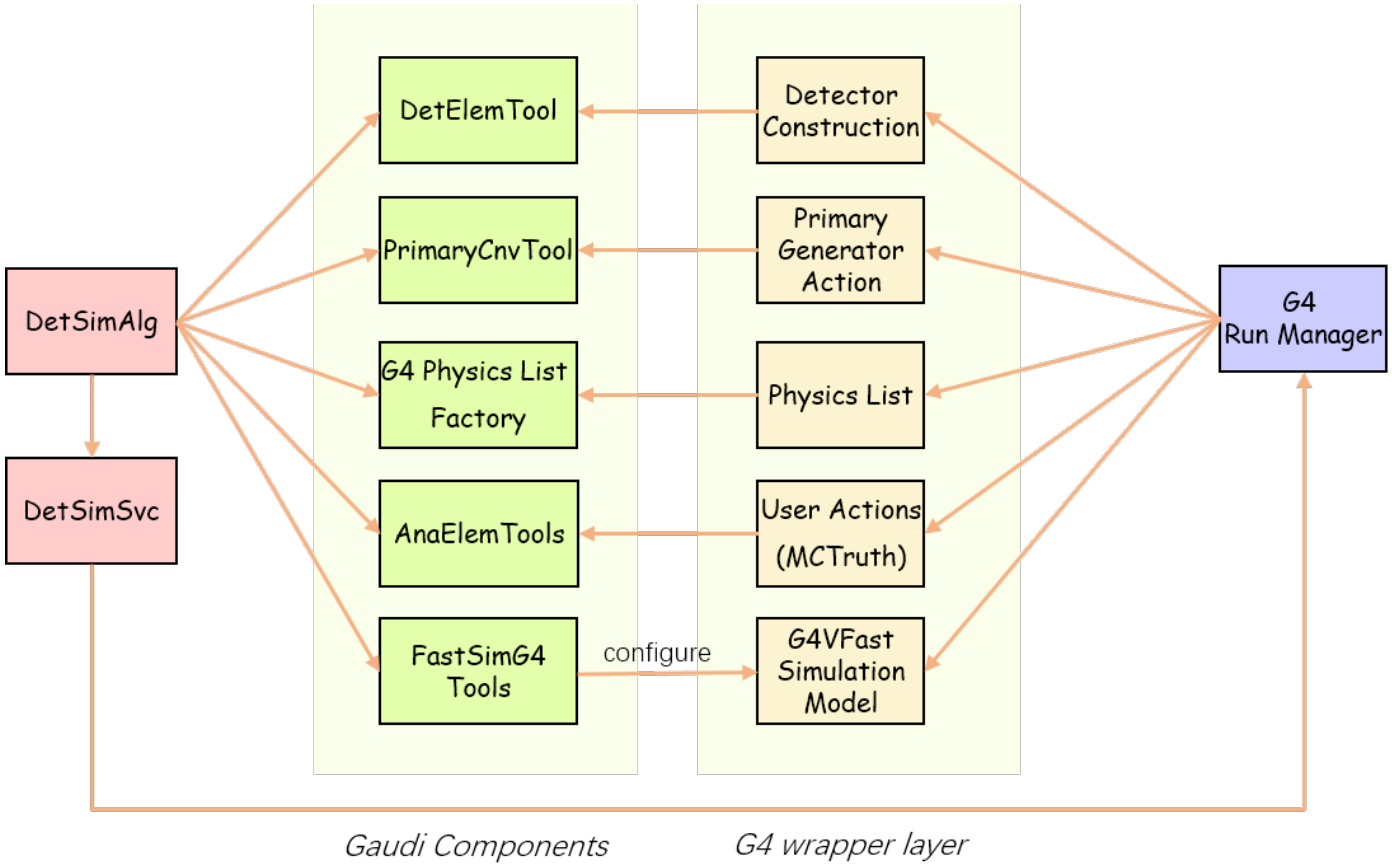
Simulation framework in CEPCSW (4)

- Physics generator interface
  - Physics generators with different formats are integrated, including StdHep, HepEvt, LCIO, HepMC formats.
  - Particle gun is supported.
  - Beam background generators, such as Guinea Pig.



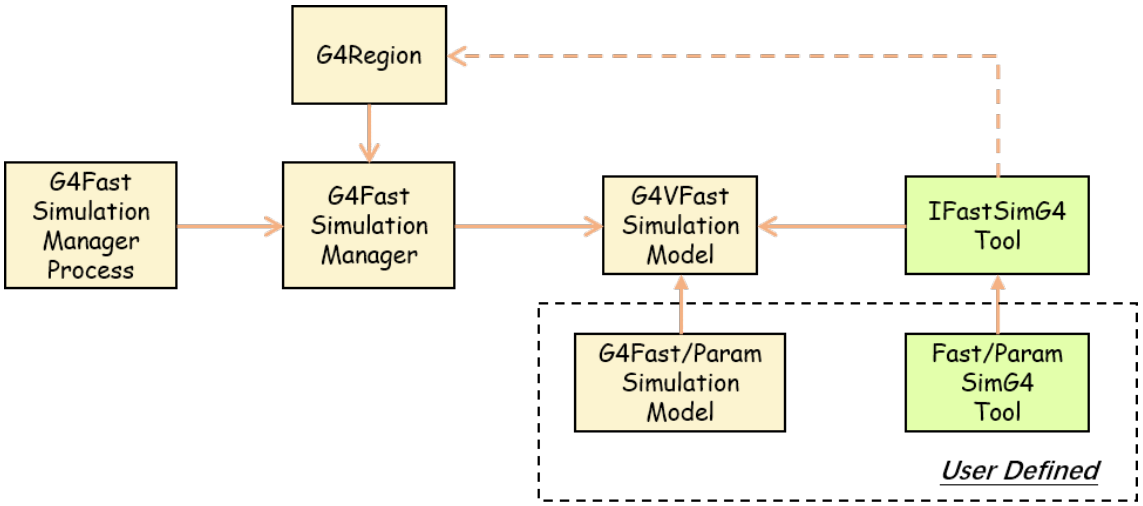
Simulation framework in CEPCSW (5)

- Integration with Geant4 and Gaudi
  - A thin layer is developed to manage corresponding Geant4 objects.

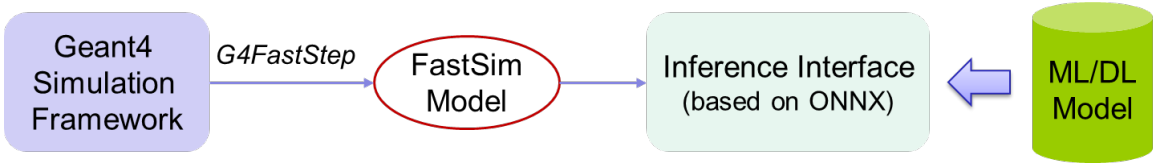


Simulation framework in CEPCSW (6)

- Integration with Fast Simulation
  - Region based: when a particle enter a region, fast simulation will be triggered by Geant4.



  - Support ML methods via ONNX inference interface.
    - Example: Fast pulse simulation (MLP) in drift chamber done by Wenxing

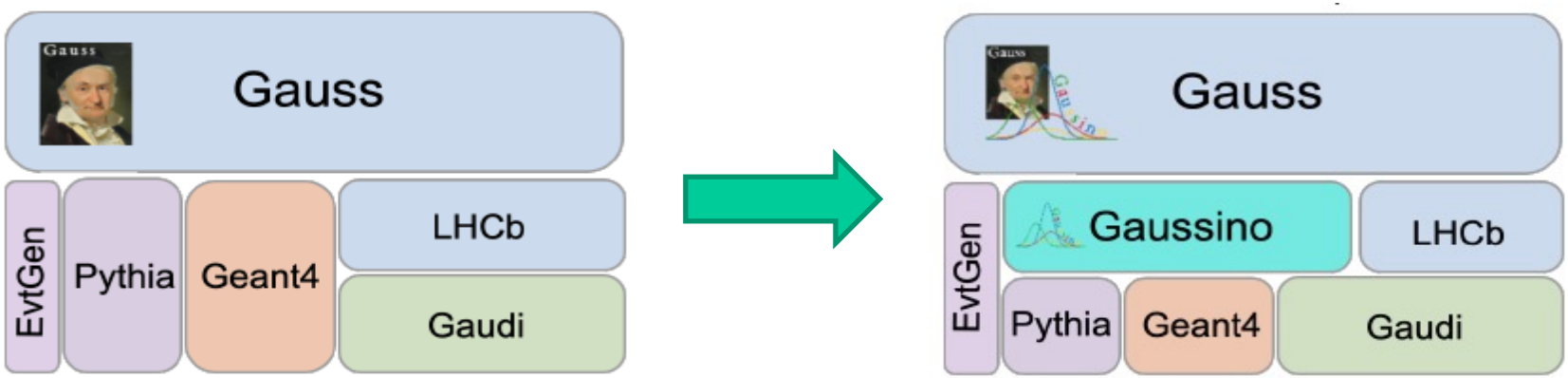


Outline

- Introduction
- Simulation framework in CEPCSW
- Gaussino-based simulation framework
- Parameterized simulation: k4SimDelphes
- Summary

Gaussino-based simulation (1)

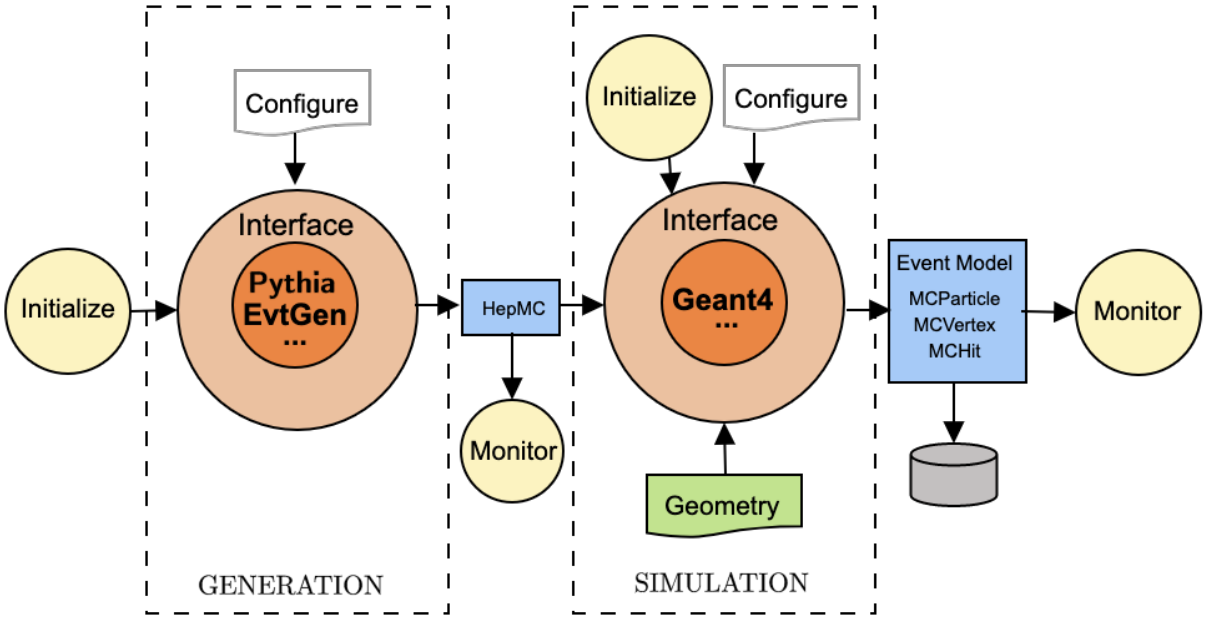
- CEPC also works together with Key4hep project members and is re-implementing CEPC detector simulation with Gaussino
- Evolution of the simulation framework from LHCb
  - The underlying framework is moving to Gaudi Functional and Gaudi Hive
  - Better support for multi-threading, machine learning, fast simulation methods
  - Gauss-on-Gaussino is a new version of LHCb simulation framework



- Gaussino is being added to Key4hep by extracting experiment-independent parts from Gauss

Gaussino-based simulation (2)

- Gaussino is a thread-safe simulation framework based on Gaudi Functional and provides interfaces to Pythia and Geant4.
- Modular design
  - Gaudi Functional Algorithms
  - Gaudi Tools
- Four components
  - Generation of events
  - The detector simulation
  - Geometry service
  - Monitoring & output



- Easy to configure by customizing the algorithms, services and tools

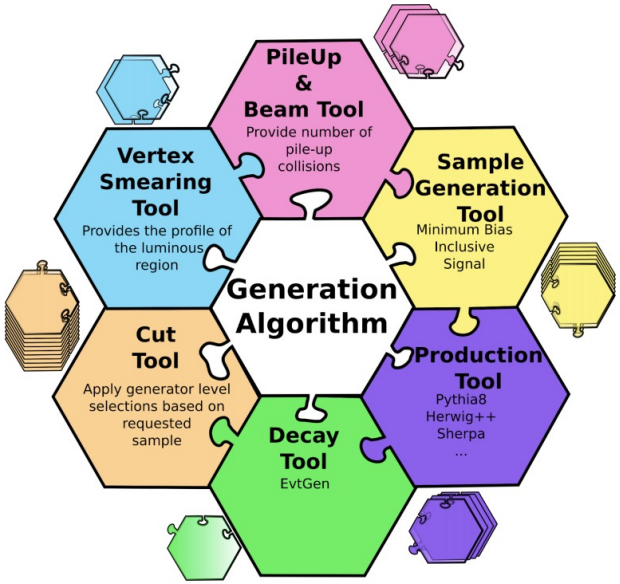
Gaussino-based simulation (3)

- Generation algorithms: “Generation” and “ParticleGun”
  - The input is LHCb GenHeader
  - The output is HepMC GenEvent

```
class Generation
    : public Gaudi::Functional::MultiTransformer<
        std::tuple<std::vector<HepMC3::GenEventPtr>, LHCb::GenCollisions, LHCb::GenHeader>( const LHCb::GenHeader& ),
        Gaudi::Functional::Traits::BaseClass_t<RndAlgSeeder>>
```

```
class ParticleGun
    : public Gaudi::Functional::MultiTransformer<
        std::tuple<HepMC3::GenEventPtrs, LHCb::GenCollisions, LHCb::GenHeader>( const LHCb::GenHeader& ),
        Gaudi::Functional::Traits::BaseClass_t<RndAlgSeeder>>
```

- The generation algorithm
  - It consists of different Gen tools
  - Thread safety of generators
- The particle gun algorithm
  - N Particles, Particle type, momentum, etc.



Gaussino-based simulation (4)

- Detector simulation algorithm: GiGaAlg
  - The input is HepMC GenEvent
  - The output is G4Event and MC truths

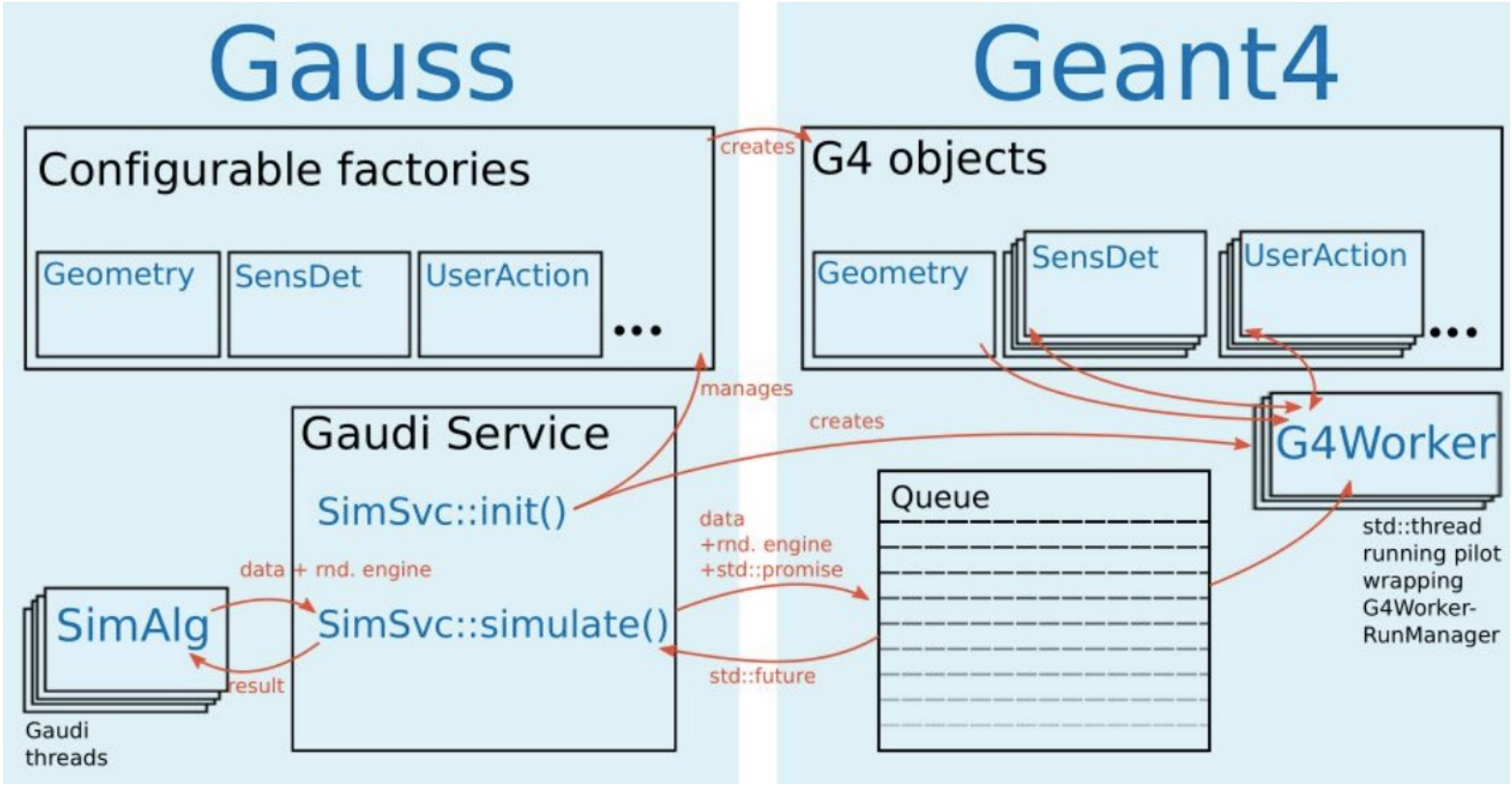
```
class GiGaAlg : public Gaudi::Functional::MultiTransformer<std::tuple<G4EventProxies, Gaussino::MCTruthPtrs>(
                    const HepMC3::GenEventPtrs& ),
                Gaudi::Functional::Traits::BaseClass_t<RndAlgSeeder>>
```

- GiGaMT (Service) is used by GiGaAlg to simulate events with Geant4.
  - The major interface to Geant4, including Detector Construction, Physics List and User Actions.

```
class GiGaMT : public Service, virtual public IGiGaMTSvc, virtual public IGiGaMTSetUpSvc
{
  // TODO: GiGaActionInitializer is very modular. No idea if any other option might be used here.
  // Gaudi::Property<std::string> m_MTRunMgrFactoryName{this, "MTRunManagerFactory", "GiGaMTRunManagerFAC"};
  ToolHandle<GiGaFactoryBase<GiGaMTRunManager>> m_mTRunManagerFactory{this, "MTRunManagerFactory",
                                                                      "GiGaMTRunManagerFAC"};
  ToolHandle<GiGaFactoryBase<G4VUserPhysicsList>> m_physListFactory{this, "PhysicsListFactory", "GiGaMT_FTFP_BERT"};
  ToolHandle<GiGaFactoryBase<GiGaWorkerPilot>> m_workerPilotFactory{this, "WorkerPilotFactory", "GiGaWorkerPilotFAC"};
  ToolHandle<GiGaFactoryBase<G4VUserDetectorConstruction>> m_detConstFactory{this, "DetectorConstruction",
                                                                             "GiGaMTDetectorConstructionFAC"};
  ToolHandle<GiGaFactoryBase<G4VUserActionInitialization>> m_ActionInitializerFactory{this, "ActionInitializer",
                                                                                      "GiGaActionInitializer"};
  ToolHandle<IHepMC3ToMCTruthConverter> m_converterTool{this, "HepMCConverter", "HepMC3ToMCTruthConverter"};
  ToolHandleArray<IG4MonitoringTool> m_MoniTools{this};
  Gaudi::Property<std::vector<std::string>> m_MoniToolNames{
      this, "MonitorTools", {}, tool_array_setter( m_MoniTools, m_MoniToolNames )};
```

Gaussino-based simulation (5)

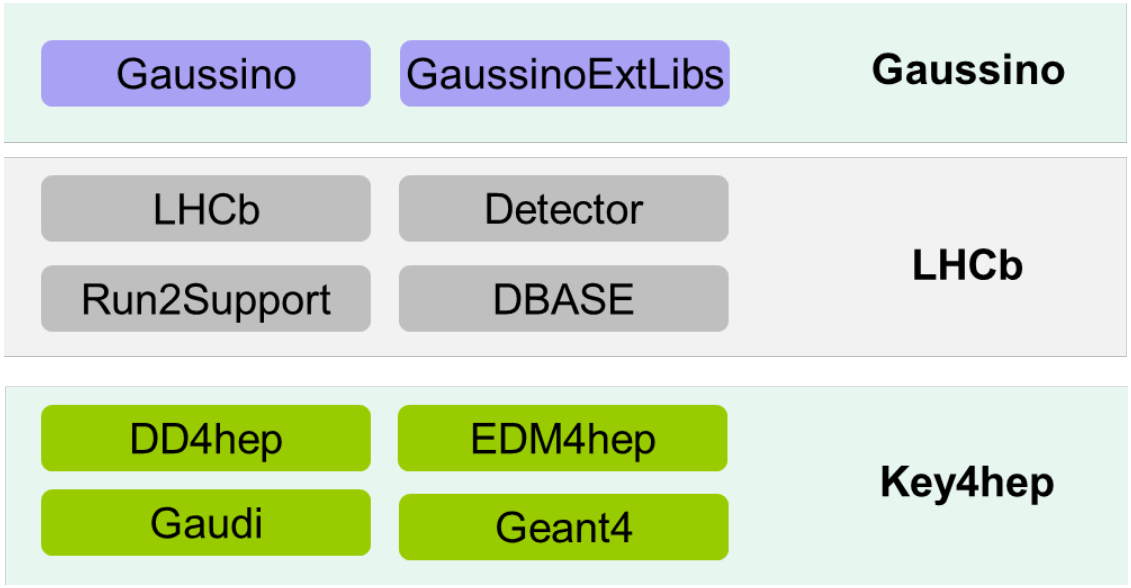
- Multithreading with Geant4
  - A queue is used to communicate between Gaussino and Geant4



Sim/GiGaMTCore/include/GiGaMTCoreRun/GiGaWorkerPilot.h

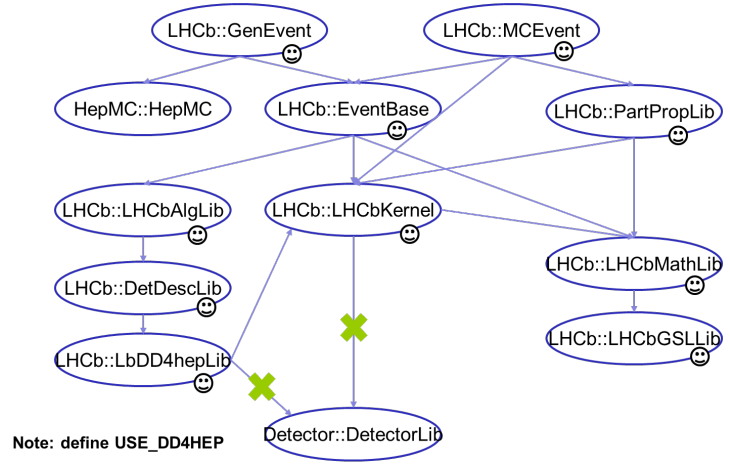
Gaussino-based simulation (6)

- Gaussino still depends on LHCb software and can not be used by other experiments directly
- Development of CEPC-on-Gaussino was planned with the following three steps
  - Using the original version having the dependency on the LHCb software
  - Creating the modified version in which the LHCb dependency is removed
  - Directly using the Key4hep version (not available at the moment)



Gaussino-based simulation (7)

- CEPC-on-Gaussino :
  - https://gitlab.cern.ch/talin/build-cepc-on-gaussino
  - https://gitlab.cern.ch/talin/Gaussino/-/tree/cepc-on-gaussino
- Reuse GenEvent and MCEvent from the LHCb project
  - A minimum number of packages are selected
  - Non-required dependencies were removed
- Implement the simulation of CEPC vertex detector
  - Use DD4hepCnvSvc as Geometry Service.
  - Implement G4 Sensitive Detector and Hit object for tracker detector.
  - Implement a monitor tool to save output.




```
CEPCTest/
├── CMakeLists.txt
├── include
│   └── CEPCTest
│       ├── DDG4SensitiveDetector.h
│       ├── Geant4Hits.h
│       ├── GenericTrackerSensDetTool.h
│       └── GenericTrackerSensitiveDetector.h
├── options
│   └── cepc_gaussino.py
└── src
    ├── Components
    │   ├── GenericTrackerMonTool.cpp
    │   ├── GenericTrackerMonTool.hh
    │   └── GenericTrackerSensDetToolComponent.cpp
    └── Lib
        ├── DDG4SensitiveDetector.cpp
        ├── Geant4Hits.cpp
        ├── GenericTrackerSensDetTool.cpp
        └── GenericTrackerSensitiveDetector.cpp
```

Gaussino-based simulation (8)

- Need to take care of the name of the sensitive detector factory.
  - DD4hep instance needs to know the name to retrieve the DD4hep detector component.

```
#include "CEPCTest/GenericTrackerSensDetTool.h"

DECLARE_COMPONENT_WITH_ID(GenericTrackerSensDetTool, "GenericTrackerSensDet")
DECLARE_COMPONENT_WITH_ID(GenericTrackerSensDetTool, "VXD")
```

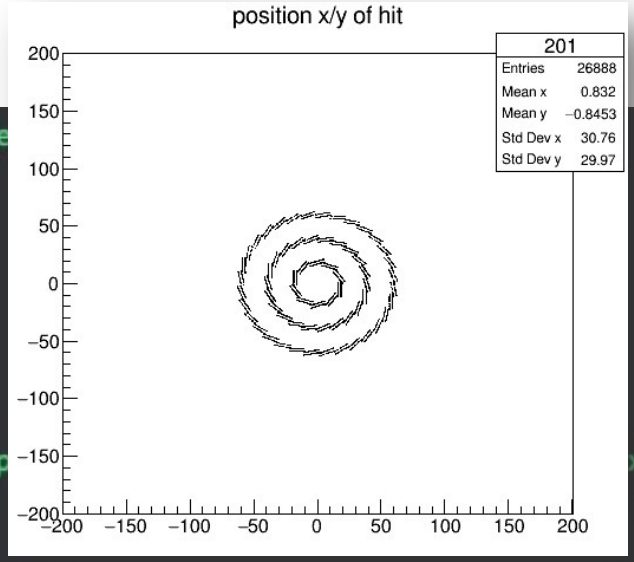
- CEPC-on-Gaussino is easy to use
  - Register the sensitive detector factory and associate it with DD4hep detector name
  - Register the monitor tool

```
montool = giga.addTool(GenericTrackerMonTool, name="GenericTracke
montool.OutputLevel = DEBUG
montool.CollectionName = "VXDCollection"

giga.MonitorTools = ["GenericTrackerMonTool"]

GaussinoGeometry().GeometryService = "DD4hepCnvSvc"

DD4hepCnvSvc().DescriptionLocation = "CEPCSW/Detector/DetCRD/comp                xml"
DD4hepCnvSvc().SensDetMappings = {"VXD": "VXD"}
# DD4hepCnvSvc().OutputLevel = DEBUG
```



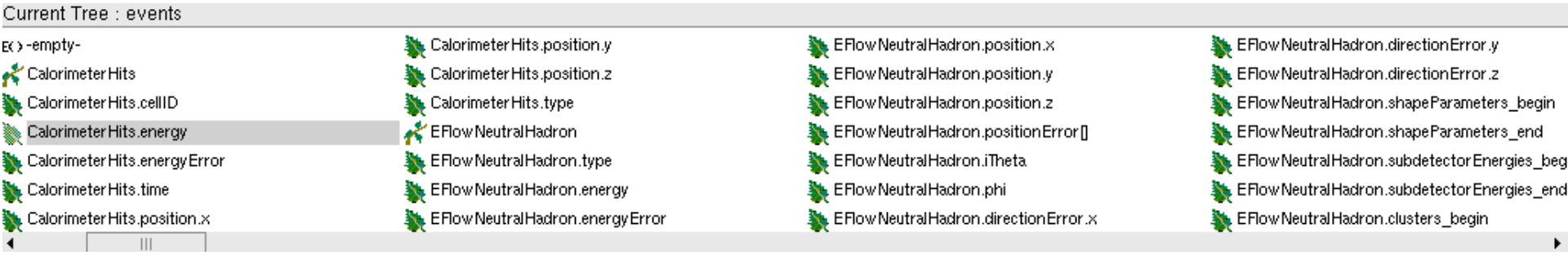
Outline

- Introduction
- Simulation framework in CEPCSW
- Gaussino-based simulation framework
- Parameterized simulation: k4SimDelphes
- Summary

k4SimDelphes

- Delphes is also integrated into Key4hep.
  - EDM4hep is one of the supported output formats.
  - k4SimDelphes offers both standalone executables and the integration with framework.
    - https://github.com/key4hep/k4SimDelphes

```
$ DelphesSTDHEP_EDM4HEP \
    delphes/cards/delphes_card_CEPC.tcl \
    k4SimDelphes/edm4hep_output_config.tcl \
    delphes_output_edm4hep.root \
    z_ee.hep
```



Also see Key4hep-doc:
https://key4hep.github.io/key4hep-doc/k4simdelphes/doc/starterkit/k4SimDelphes/Readme.html

Outline

- Introduction
- Simulation framework in CEPCSW
- Gaussino-based simulation framework
- Parameterized simulation: k4SimDelphes
- Summary

Summary

- CEPC simulation software is integrated with Key4hep.
  - Adopting a common software makes it easy to share between different experiments, such as k4SimDelphes could be used.
- A complete simulation chain is available in CEPCSW.
  - It is already used for detector software development.
- Currently migrating to Gaussino-based simulation.
  - Reuse both GenEvent and MCEvent from LHCb and remove the dependencies.
  - Implement the CEPC VXD detector simulation.
  - Next step:
    - Integrate with the Gaudi functional version of k4FWCore and save the output in EDM4hep.