Status and Plan: Offline Software Development

Weidong Li

CEPC Weekly Meeting

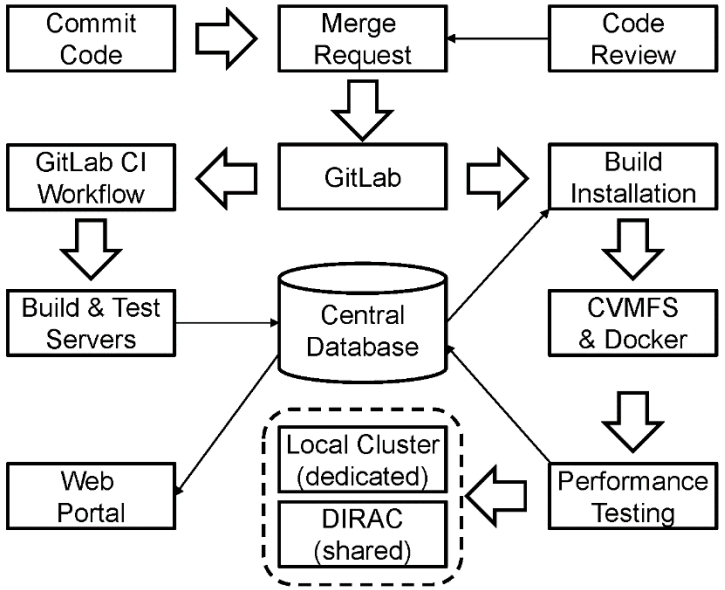
June 3, 2026

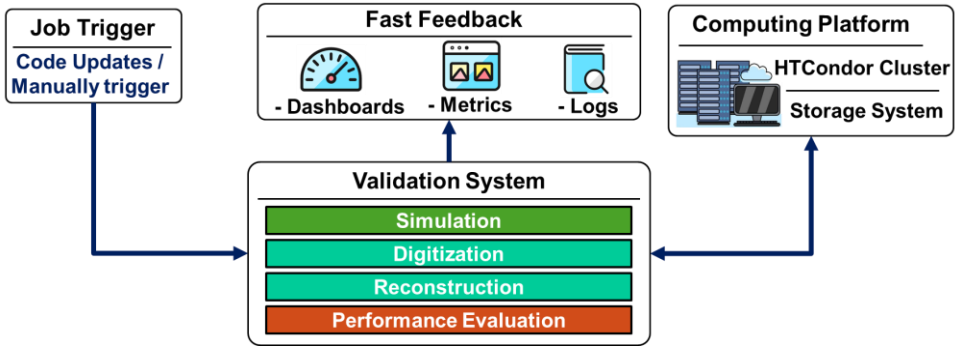
2026 Plan (reported on 18th December 2025)

#	Task	Duration	Status
1	Develop a validation system that integrates reconstruction and analysis algorithms to produce comparative efficiency and resolution plots	Q1	85%
2	Implement automation pipelines with Kafka and database integration to streamline validation workflows	Q2	100%
3	Optimize CyberPFA algorithms and validate their performance against baseline results	Q2	50%
4	Deploy new Gaussino simulation framework	Q2	20%
5	Enhance Phoenix event display	Q3	50%
6	Integrate DAQ system with CEPCSW workflows	Q3	0%
7	Upgrade core software stack: migrate to the latest version of EDM4hep integration and provide Gaudi multithreading support	Q4	60%
8	Apply ML to calorimeter simulation	Q4	50%

Automated validation system (1)

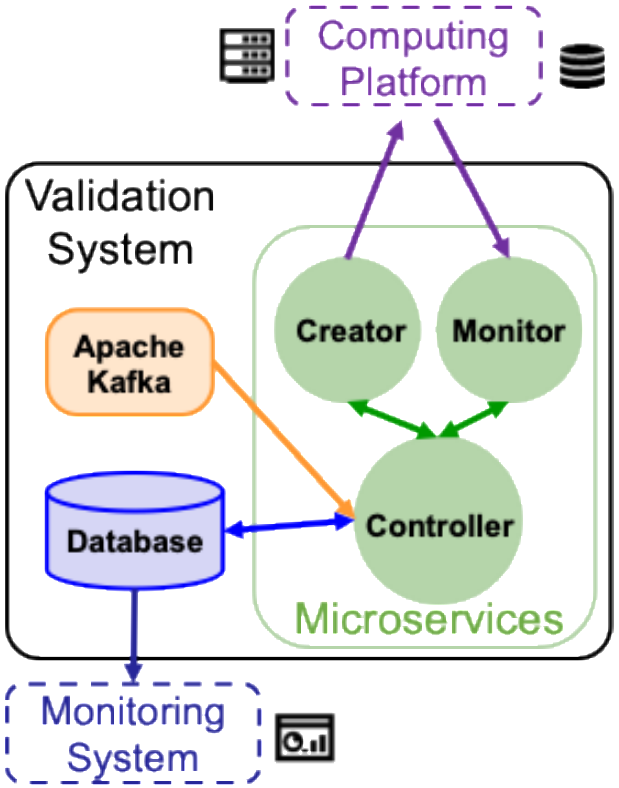
- Integrated with GitHub/GitLab CI/CD
- Development tests: Fast unit/integration tests on each commit or MR; physics validation runs daily/weekly; results stored centrally with web monitoring.
- Release validation: Large scale production launched manually; key physics quantities compared with reference samples.
- Automated releases: Daily CEPCSW builds, docs, and containers published to CVMFS via CI/CD.
- Central database: Stores all test results with multi platform test matrix and real time web status.





Automated validation system (2)

- Development of Kafka-based automated release validation
  - Kafka functions as the primary coordination layer for managing multi-stage computational workflows.
  - An event-driven workflow was set up: modules listen to Kafka topics for new triggers, and when a message arrives, the workflow starts.
  - The execution flow of a processing step involves the Creator, Controller, and Monitor. Together, these components form a fully integrated workflow suitable for deployment as a standalone microservice.
  - The Performance Validation module provides performance metrics and comparisons against standard distributions at each processing stage, all accessible to users through the web.



External Libraries: migration to latest version

- **Strategy**: **Core software** -> Simulation -> Reconstruction -> Analysis
- After the new release, there will be validation.

	LCG 105	LCG 109	Impact packages
Compiler	GCC 11.3.0	GCC 13.1.0	All
CMake	3.26.2	3.30.6	All
Python	3.9.12	3.13.11	All
Gaudi	v37r2	v40r2	All
ROOT	6.30.2	6.38.0	All
Geant4	11.2.0	11.4.0	Simulation
DD4hep	01.27.02	01.35	Simulation, Reconstruction
EDM4hep	00.10	1.0	Simulation, Reconstruction
podio	00.17.01	01.07	Core software
k4FWCore	8th Oct 2023	01.06	Core software

CEPC TDR version

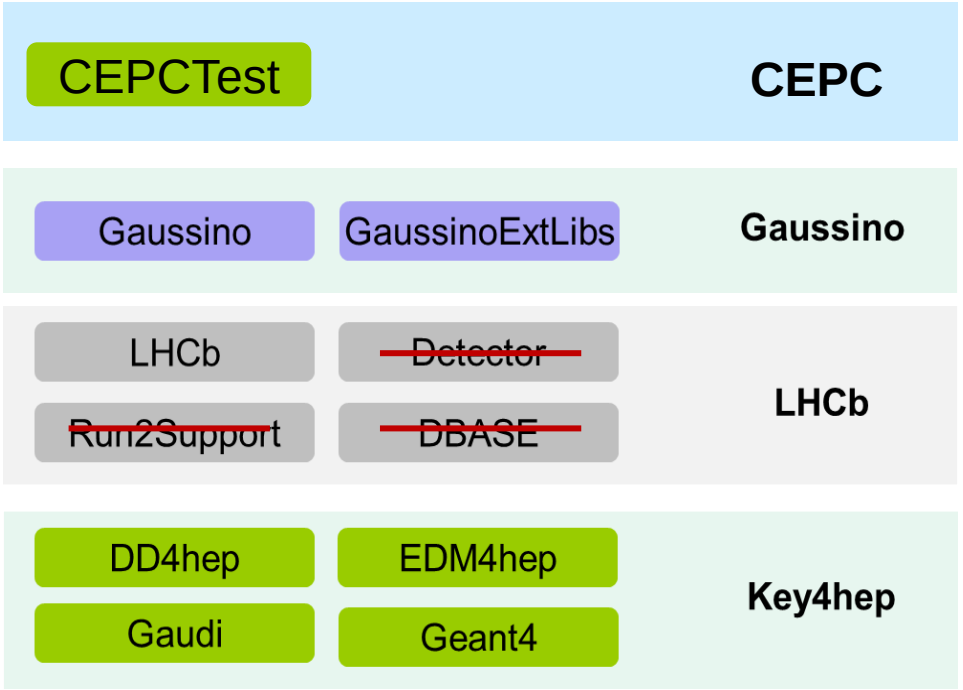
Migrate from GaudiAlg to legacy Algorithm.

Ongoing.

Still use legacy k4DataSvc. Will migrate to IOSvc.

Deployment of the new Gaussino simulation framework

- Development of CEPC-on-Gaussino was planned with the following three steps
  - ✓ Using the original version having the dependency on the LHCb software
  - ✓ Creating a modified version with less LHCb dependency
  - ☐ Directly using the Key4hep version without LHCb dependency (not available at the moment)




```
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
```

Modified Project	Git repo
LHCb	https://gitlab.cern.ch/talin/LHCb/-/tree/cepc-on-gaussino?ref_type=heads
gaussinoextlibs	https://gitlab.cern.ch/talin/gaussinoextlibs/-/tree/cepc-on-gaussino?ref_type=heads
Gaussino	https://gitlab.cern.ch/talin/Gaussino/-/tree/cepc-on-gaussino?ref_type=heads
CEPCSW	https://gitlab.cern.ch/talin/CEPCSW/-/tree/cepc-on-gaussino?ref_type=heads
Installation script	https://gitlab.cern.ch/talin/build-cepc-on-gaussino

Support simulation of tracker detectors.

MT Implementation Option A: AthenaMT-style

- Concept
  - Use GaudiHive dataflow scheduler + slot-based concurrency
  - Algorithms remain classical Gaudi algorithms, but must be reentrant
  - Parallelism comes from running many event slots and independent algorithms concurrently
- Strengths
  - Minimal disruption: CEPCSW algorithms can be adapted with ReadHandle/WriteHandle
  - Supports multi-event and intra-event parallelism
  - Well-tested in ATLAS AthenaMT at HL-LHC scale
  - Works naturally with conditions data, services, and multiple stores
- Challenges
  - Requires careful thread-safety audit of services and tools
  - More complex runtime behavior than functional model
  - Dependency graph must be explicitly declared

Four-Year Plan (1)

- 2026 plan
  - completed Key4hep upgrades by updating the underlying software stack
  - developed a new simulation software framework based on Gaussino
  - providing full support for multi-threaded simulation and reconstruction workflows
  - Optimize the new calorimeter PFA reconstruction methods to further exploit detector performance
- 2027 plan
  - Using machine-learning–based simulation to achieve fine-grained modeling of gas detectors
  - Developing fast calorimeter simulation models to improve overall simulation efficiency and physics accuracy
  - Supporting multiple detector configurations for flexible modeling and large-scale experiment design studies

Four-Year Plan (2)

- 2028 plan
  - Develop machine-learning–based TPC reconstruction by building a TPC track-reconstruction software framework and integrating it with the ACTS-based silicon-tracker tracking chain, enabling efficient and robust track reconstruction in high-background environments.
  - Develop dN/dx reconstruction for gas detectors to provide reliable particle-identification capability for physics analysis.
- 2029 plan
  - Develop an integrated analysis framework that unifies common algorithms and tools—including particle identification, jet reconstruction, jet tagging, kinematics, and vertex fitting—into a coherent workflow.
  - Enable efficient parallel computing both within and across compute nodes, fully leveraging multi-core CPU architectures to achieve high-performance data analysis.
  - Support AI-assisted analysis to simplify the physics-analysis process and improve productivity through intelligent automation.

Backup

Other ongoing activities

- Integrating AI Agents/Agentic AI into the CEPCSW development workflow
  - Enable autonomous development assistance including code generation, refactoring, and patch proposals to accelerate CEPCSW evolution
  - Enhance documentation and debugging to raise overall software quality and developer efficiency
  - Apply agents to development operations such as pull-request review, CI log summarization, and workflow-template generation
  - Deploy agents safely by starting with routine, low-risk tasks while maintaining strict human oversight for all physics-critical decisions

Integration：CEPCSW + DAQ + High Level Trigger

- Integration goals
  - Enable real-time event selection, fast feedback, and high-quality offline reconstruction
  - Ensure consistent data models, metadata, and conditions across DAQ, HLT, and CEPCSW
- High Level Trigger (HLT)
  - Runs CEPCSW-based fast reconstruction
  - Applies software trigger algorithms
  - Produces HLT-accepted raw streams + online monitoring outputs
- Data Flow
  - DAQ event building — DAQ assembles full events, publishes trigger primitives, and streams raw data to HLT
  - HLT fast-reconstruction — HLT consumes the DAQ stream, runs CEPCSW-based fast tracking/calorimetry, applies physics + detector-quality selections, and outputs HLT-accepted raw events
  - Data transfer + registration — Data Transfer Layer moves accepted events to the offline buffer and metadata service registers run/file and conditions tags