



Key4hep, a framework for future HEP experiments and its use in FCC

Gerardo Ganis^a , Clément Helsens, Valentin Völkl

CERN, Geneva, Switzerland

Received: 10 June 2021 / Accepted: 20 November 2021
© The Author(s) 2022, corrected publication 2022

Abstract The road map to the FCC Feasibility Study Report, for submission to the next Update of the European Strategy for Particle Physics, will require detailed simulation and advanced reconstruction algorithms to explore and maximise the physics reach of proposed detector solutions. The optimisation process will require maximal flexibility in changing detector geometries, materials and sensitive areas, and efficient tools to quantify the overall performance. To synergise such developments, the CEPC, CLIC, FCC, ILC and SCT communities have engaged in the commissioning of a ‘Turnkey Software Stack’ (Key4hep), which would provide all the necessary ingredients, from simulation to analysis, for future experiments. This approach is based on the positive experience of the linear collider projects ILC and CLIC, which have developed and used a common software stack (iLCSOFT) over the last decade. Key4hep aims to cover most, if not all, future linear and circular machines colliding leptons (electrons, muons) and hadrons. The common software ecosystem will facilitate writing specific components for experiments ensuring coherency and maximising the re-use of established solutions. Project-specific software frameworks will require adaptation to fully profit from the common software base. In this essay, we present the status and plans for re-framing the FCC software framework, FCCSW, around Key4hep and discuss the challenges associated with the transition.

1 Introduction

The road map to deepen the understanding of the physics potential of the Future Circular Collider (FCC), due for the next Update of the European Strategy for Particle Physics (UESPP), will require detailed simulation and advanced reconstruction algorithms to explore and maximise the physics reach of proposed detector solutions. The results submitted to the 2019 UESPP were obtained using FCCSW, an experiment software based on the LHC experience integrated with the results of developing common software projects, such as DD4hep, which served the initial needs of the FCC community [1] well. The next phase of the studies will require further developments of the software, in particular concerning the flexibility in sub-detector choices, including the simulation and reconstruction of their response. The interplay between reconstruction algorithms and detector geometry, for example in particle flow clustering, means that the detector hardware cannot be developed and designed

^a e-mail: gerardo.ganis@cern.ch (corresponding author)

independently from the software. At the same time, developing and validating sophisticated algorithms, including accounting for a large number of edge cases, require a significant amount of resources. Thus, the communities for future experiments—CEPC [2], CLIC [3], FCC [4], ILC [5], SCT [6], STC [7]—came to the agreement that the development of a common software solution would benefit everyone [8,9]. Software development is a collaborative effort, and one way **Key4hep** facilitates writing-specific components for experiments is by maximising the re-use of established solutions and packages to benefit from existing community developments, for example, **ROOT** [10], **Geant4** [11], **DD4hep** [12], **Gaudi** [13] and **PODIO** [14]. By using and contributing to other experiment-independent software packages, an ‘ecosystem’ of HEP software is fostered.

In this essay, we describe the proposal of the turnkey software stack; we outline the requirements and planned ingredients and showcase the evolution of **FCCSW** towards this common software solution. We focus on the functionality provided by the ecosystem because we believe that matching the needs of the FCC project is the real challenge ahead. We expect other aspects, such as the one of computing performance, which will certainly become very important for the FCC operations, to be addressed, when relevant, at the level of the components that **Key4hep** brings together.

2 **Key4hep**: the Turnkey Software Stack for Future Colliders

The turnkey software stack, nicknamed **Key4hep**, aims to encompass all the libraries needed for generation, simulation, reconstruction, and analysis of events at the participating future colliders. The project effectively started in February 2020 after two kick-off events, which took place in Bologna [8] and Hong-Kong [9]. A more detailed description of the vision and goals of the project can be found in Ref. [15]. In this section, we will recall the main ingredients.

2.1 Ingredients

A software ecosystem for HEP data processing comprises several components that need to be chosen carefully. The most important are the following:

1. A data processing framework, which provides the structure for all dependent components. Despite the fact that the linear collider community has used **Marlin** [16] for many years very successfully, the choice has fallen on **Gaudi** because it is adopted by LHC experiments and it has a large user and developer community, offering—or planning to offer—support for access to heterogeneous resources, different architectures, and task-oriented concurrency.¹
2. A geometry description tool; for this purpose all of the communities concerned already use **DD4hep**, which offers a complete detector description for simulation, reconstruction and analysis.²

¹ A selected number of features provided by **Marlin** and currently not available in **Gaudi**, e.g. capability to automatically generate steering file templates, will be contributed back to **Gaudi** as part of the **Key4hep** project.

² **DD4hep** is not only of interest for future experiments community but also for LHC’s: indeed it will be used in production, as of run 3, by CMS and LHCb. This will entail a consolidation and development process

Footnote 2 continued

highly beneficial for the future experiments, in particular with respect to aspects more relevant for running experiments, e.g. alignment and calibration. In addition, support for common digitisation modules is planned

3. A common event data model (see Section 2.2).
4. A common build infrastructure. For the ease of use by librarians and developers, one needs to be able to build any and all pieces of the stack with minimum effort. Most packages have a similar build system based on CMake and follow common good practices, which already reduces the maintenance burden, but the large number of packages requires additional tooling to automate the installation and ensure a consistently linked set of packages with correct dependency resolution. The investigations of the HEP Software Foundation packaging working group have identified the scientific package manager Spack [18] as a suitable solution, and it has since been successfully used for prototype builds and established as the main build tool for Key4hep software [19]. In addition, not only does Spack allow sharing of the build results but also the build recipes with a wider community.

2.2 EDM4hep: the Event Data Model

The event data model defines the structures needed to describe the required event information in the persistent store. While, in principle, this can be different from what is used in the transient store seen by the data processing algorithms, and also from algorithm to algorithm, adopting the same event model everywhere reduces the need for conversions, enhances generality and improves inter-operability.

Since the beginning of the FCC physics potential studies, the choice has been for an event data model managed by PODIO [14], a toolkit that generates the data model implementation from templates. This allows for a separation of the high-level description in YAML files using plain-old-data simple types, and the low level persistency layer that can be optimised according to the back-end. Once the required classes are defined in the YAML file, the PODIO tool creates the source code automatically. For Key4hep, the same technology has been chosen, with an event data model, referred to as EDM4hep, initially based on the LCIO and FCC-edm classes, and improved from there as needed.

An underlying assumption or challenge is that the same event data model can be used for all types of HEP experiments, in particular for hadron and lepton colliders, a case which is particularly applicable to the integrated FCC programme, which has a lepton collision phase followed by a hadron collision phase, similar to what happened with LEP and LHC; this seems to be the case with EDM4hep, and it will be further discussed in Sect. 3.

2.3 Development strategy

During the kick-off meetings [8,9], it became clear that to maximise the chances of an early adoption a *deliver-early-deliver-often* approach needed to be adopted. Furthermore, early identification of key components, to be made quickly available in usable form, was required in order to plan and make progress with the evaluation process.

The activities started from the definition of the common event data model EDM4hep, which defines a sort of common language underlying the inter-operation of the various components. Once a usable version of EDM4hep³ was available, the core component of the framework, k4FWCore,⁴ which provides the interfaces and the data service, needed to be made available; k4FWCore is largely based on the corresponding components of

as part of the research and development program AIDAInnova [17] with the aim of facilitating the translation of the energy deposits produced by the full simulation into something, which can be used to reconstruct physics objects.

³ Currently version 0.3.2, <https://github.com/key4hep/EDM4hep>.

⁴ Currently version 1.0pre06, <https://github.com/key4hep/k4FWCore>.

FCCSW. The operative availability of **EDM4hep** and **k4FWCore** enabled the development and integration of data processing components, in the form of **Gaudi** algorithms and tasks.

It was soon realised that having a full workflow producing ‘usable’ events in **EDM4hep** format would allow acceleration of the process. For this purpose, the choice has been to integrate **Delphes** [20], a popular tool for super-fast parameterised detector response. The component **k4SimDelphes**, derived from a similar one existing in **FCCSW**, provides a **Key4hep** interface to **Delphes** both in terms of standalone applications, running on top of any of the input formats supported by **Delphes** and a **Gaudi** algorithm.

In parallel, to facilitate the transition and interplay with **iLCSOft**, a **Gaudi**-based wrapper around **Marlin** processes, which are the equivalent of the **Gaudi** algorithms, has been developed, enabling the complete **iLCSOft** reconstruction via **Gaudi**⁵ to be run. The picture here is completed by the provision of an on-the-fly converter **LCIO–EDM4hep–LCIO**⁶ enabling the possibility of running an algorithm developed for **Marlin** and **LCIO**, for example the flavour tagging code **LCFIplus**, inside a full **Gaudi** application. By design, **EDM4hep** allows a one-to-one conversion to **LCIO** and back without loss of information. In addition, the **PODIO**-generated interfaces allow simple ad hoc user data extensions of the data model should that be required.

The basic components described above enable the development of specific new elements or the adoption of existing ones—a process which is now ongoing for generation, full simulation and reconstruction capabilities.

3 FCCSW and Key4hep

The FCC software framework, **FCCSW**, has a lot of commonalities with **Key4hep**, stemming from the fact that they are both based on **Gaudi**. The main differences are the event data model and the structure governing the components, which in **Key4hep** is flat while **FCCSW** features a hierarchical structure with sub-components organised in categories.

For **FCCSW** moving to **Key4hep** conceptually means using common components available from **Key4hep** as much as possible, but FCC-specific components will be kept under FCC responsibility.

From the technical point of view, the following chronologically ordered steps were identified:

1. Replacement of **FCC-edm** with **EDM4hep**;
2. Identification and migration of non-FCC-specific **FCCSW** components to **Key4hep**;
3. Completion of the ecosystem, either in **Key4hep** or in the components remaining in **FCCSW**
 - (a) Addition of missing sub-components, sub-detectors; optimisation/replacement of full/fast simulation techniques, reconstruction algorithms, etc.

⁵ Currently version 0.3.2, <https://github.com/key4hep/k4LCIORReader>.

⁶ Currently version 0.3.1, <https://github.com/key4hep/k4MarlinWrapper>.

Table 1 Original FCCSW components and their fate in the new structures, as well as additional information on the current progress in transitioning them to use Key4hep interfaces (“status”) and whether they will be ultimately hosted by Key4hep (“migration”). See text for more details about dual-readout

Old FCCSW (version ≤ 0.16)	Key4hep	New FCCSW	Status	Migration
FWCore	k4FWCore		Done	Yes
Sim/SimDelphesInterface	k4SimDelphes		Done	Yes
Generation		k4Gen	Done	Under evaluation
Sim		k4SimGeant4	Done	Under evaluation
Reconstruction/Rec[...]Calorimeter		k4RecCalorimeter	Done	Under evaluation
Reconstruction/RecDriftChamber		To be determined		
Detector		FCCDetectors	Done	No, FCC specific
	To be determined	Dual-readout		Under evaluation

3.1 Current status

3.1.1 Migration to EDM4hep

The replacement of FCC-edm with EDM4hep has been completed without any major problem for all components, i.e. including those specific to FCC-hh. This means that all the data structures needed to describe hadronic collisions could be handled with the structures provided by EDM4hep.

3.1.2 General and FCC-specific FCCSW components

As mentioned above, Key4hep and FCCSW are conceptually very similar. The FCCSW components are therefore natural candidates to be generalised and become parts of upstream Key4hep. Table 1 summarises the current status of the main components.

As can be seen from the table, in preparing the migration to Key4hep, a general restructuring of the FCCSW repository has taken place, resulting in a flat structure similar to that of Key4hep, with sub-repositories promoted to separate repositories. Some of the components, FWCore and SimDelphesInterface, have been polished, renamed and migrated, and are already extensively used. For some others, the preparation work has been completed and they are being analysed for migration. The detector descriptions have moved to a separate repository, FCCDetectors, which will not be migrated, as it is FCC-specific. The dual-readout module includes Geant4 simulation and reconstruction of the IDEA dual readout calorimeter; it has always been hosted in the FCC project as a separate component, i.e. never part of FCCSW. Finally, the RecDriftChamber module implemented an approximate reconstruction of the IDEA drift chamber and will not be migrated, since a detailed simulation and reconstruction for such a detector are being developed in Key4hep by the IDEA working groups [21].

3.1.3 Completion of the ecosystem

The completion of the ecosystem consists firstly of identifying the missing pieces in the findings of the other FCC working groups, in particular the Physics Performance and Detector

Concepts groups and secondly in creating the appropriate conditions for the provision of the required software module.

Although this step will only take place once, the existing code is properly re-framed within **Key4hep** or **FCCSW** following the new paradigm. Some developments have already been enabled and can serve as an example of these activities. The first example is the ongoing process of integration of the simulation and reconstruction code for the **IDEA** drift chamber and dual-readout calorimeter. For both of these important and innovative detectors, the **IDEA** collaboration developed standalone programs, which are now being reviewed for inclusion in **Key4hep**. The process is well advanced and required some modification of **EDM4hep**. The second example relates to track-related reconstruction algorithms, such as vertex finders, jet clustering and alike. The **Delphes** tool has recently been improved with a detailed simulation of the reconstructed parameters of electrically charged particles, with an output similar to that of full simulation and reconstruction. This allows algorithms to be developed, which can serve as an advanced starting point when the tool is available.

The completion of the ecosystem also includes a set of interfaces with external packages, which are typically being developed as part of general purpose research and development programs. Examples of these packages, for which the investigation work has already started, are the advanced tracking reconstruction system **ACTS** [22], the particle flow package **PandoraPFA** [23] and the clustering algorithm **CLUE** [24]. One of the main challenges affecting the integration process is ensuring that the event data model contains all of the information required.

3.2 The plug-and-play or LEGO approach to detector composition and simulation

The optimisation of detector design to support the full physics potential of **FCC-ee** adds the challenge of finding the best combination of detection technologies. The fact that at least two—but possibly four—interaction regions are planned opens the way to more solutions, potentially fine-tuned on specific physics aspects.

An infrastructure supporting the possibility to easily interchange sub-detector components is required for a broad investigation of all solutions. The **DD4hep** and **Gaudi** frameworks chosen for detector description and workflow execution support this through a plugin system, which allows switching of both sub-detectors and workflow components at runtime. The process has already been applied when adapting the liquid Argon calorimeter solution developed for **FCC-hh** to the **FCC-ee** case. Modular simulation workflows that allow the use of different fast and full simulation components in the same job are being implemented. The extension and subsequent validation of this approach, which for obvious reasons has been nicknamed **LEGO**, is the goal and challenge of this exercise.

4 Conclusions and outlook

The turnkey software stack, **Key4hep**, aims to create a complete data processing ecosystem for the benefit of HEP experiments, with immediate application to future collider designs and physics potential studies. The ecosystem will be built on established solutions such as **ROOT**, **Geant4**, **DD4hep** and **Gaudi**, and features a new event data model **EDM4hep** based on **PODIO**. **Key4hep** already provides enough components and modules to enable the migration of existing frameworks based on **Gaudi**. In this essay, we described the status of the migration of **FCCSW** which, at the time of writing, is close to completion. The **Key4hep** projects seem to be on the right path to find the equilibrium to serve the needs

of the several participating projects well, with the appropriate balance between common infrastructure and project-specific developments, and a clear path to promote the latter to the common repositories.

The challenge ahead is to maintain the current momentum and to create the conditions to attract and support people who will develop and implement their tools in the Key4hep context, thereby creating a critical mass of the best tools to cover most of the needs of HEP experiments. This task includes the provision of sufficiently extensive documentation, training and adequate advertising. Ultimately this will depend on the capability to guarantee an adequate workforce and funding.

Acknowledgements 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).

Funding Open access funding provided by CERN (European Organization for Nuclear Research). This project is co-funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No 95175.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. J. Cervantes et al., "A software framework for FCC studies: status and plans," CHEP 2019. EPJ Web of Conferences **245**, 05018 (2020). <https://doi.org/10.1051/epjconf/202024505018> (See for example)
2. CEPC Study Group, "CEPC conceptual design report: Volume I-accelerator," [arXiv: 1809.00285 [hep-ex]]
3. P. N. Burrows et al. "The Compact Linear Collider (CLIC) - 2018 Summary Report," <https://doi.org/10.23731/CYRM-2018-002>[arXiv: 1812.06018 [physics.acc-ph]]
4. A. Abada et al. "FCC Physics Opportunities: Future Circular Collider Conceptual Design Report Volume 1," Eur. Phys. J. C79 (2019) no. 6, 474 <https://doi.org/10.1140/epjc/s10052-019-6904-3>
5. P. Bambade et al., "The International Linear Collider: A Global Project," [arXiv: 1903.01629 [hep-ex]]
6. Super Charm-Tau Factory <https://ctd.inp.nsk.su/c-tau/>
7. Q. Luo et al., "Progress of Conceptual Study for the Accelerators of a 2-7GeV Super Tau Charm Facility at China," proceedings of 10th International Particle Accelerator Conference, Melbourne, Australia, 19-24 May 2019, JACoW Publishing, <https://doi.org/10.18429/JACoW-IPAC2019-MOPRB031>
8. F. Bedeschi, M. Caccia, P. Giacomelli, "Future Collider Software Workshop", 12-13 June 2019, Bologna, <https://agenda.infn.it/event/19047/>
9. P. Giacomelli, J. Guimaraes Da Costa, "Mini-workshop: Experiment/Detector - Software and Physics Requirements for e+e- Colliders," 16-17 January 2020, Hong Kong, <http://iasprogram.ust.hk/hep/2020/>
10. I. Antcheva et al., ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization. Comput. Phys. Commun. **180**, 2499-2512 (2009). <https://doi.org/10.1016/j.cpc.2009.08.005>
11. J. Allison et al., Geant4 developments and applications. IEEE Trans. Nucl. Sci. **53**, 270 (2006). <https://doi.org/10.1109/TNS.2006.869826>
12. F. Gaede, M. Frank, M. Petric, A. Sailer, DD4hep a community driven detector description for HEP. EPJ Web Conf. **245**, 02004 (2020). <https://doi.org/10.1051/epjconf/202024502004>
13. M. Clemencic et al., Gaudi evolution for future challenges. J. Phys. Conf. Ser. **898**(4), 042044 (2017). <https://doi.org/10.1088/1742-6596/898/4/042044>
14. F. Gaede et al., PODIO: recent developments in the Plain Old Data EDM toolkit. EPJ Web Conf. **245**, 05024 (2020). <https://doi.org/10.1051/epjconf/202024505024>

15. A. Sailer et al., Towards a Turnkey Software Stack for HEP Experiments, CHEP 2019. EPJ Web Conf. **245**, 10002 (2020). <https://doi.org/10.1051/epjconf/202024510002>
16. F. Gaede, Marlin and LCCD: Software tools for the ILC. Nucl. Instrum. Meth. A **559**, 177–180 (2006). <https://doi.org/10.1016/j.nima.2005.11.138>
17. AIDAInnova, European Union’s Horizon 2020 (Integrating Activity), WP12, task 12.2, <https://aidainnova.web.cern.ch/wp12>
18. T. Gamblin et al., “The Spack Package Manager: Bringing Order to HPC Software Chaos,” proceedings of Supercomputing 2015 (SC’15), Austin, Texas, November 15–20 2015, doi 10.1145/2807591.2807623
19. V. Volkl et al., “Building HEP Software with Spack: Experiences from Pilot Builds for Key4hep and Outlook for LCG Releases,” proceedings of vCHEP 2021. EPJ Web of Conferences **251**, 03056 (2021). <https://doi.org/10.1051/epjconf/202125103056>
20. J. de Favereau et al., DELPHES 3, A modular framework for fast simulation of a generic collider experiment. JHEP **02**, 057 (2014). [https://doi.org/10.1007/JHEP02\(2014\)057](https://doi.org/10.1007/JHEP02(2014)057) [arXiv: 1307.6346 [hep-ex]]
21. See for example, N. De Filippis, “Status of the IDEA Drift Chamber Software,” presented at FCC Software Meeting of 11 December 2020, <https://indico.cern.ch/event/979157/>
22. P. Gessinger et al., The Acts project: track reconstruction software for HL-LHC and beyond. EPJ Web Conf. **245**, 10003 (2020). <https://doi.org/10.1051/epjconf/202024510003>
23. J. S. Marshall, M. A. Thomson, “The Pandora Software Development Kit for Pattern Recognition,” Eur. Phys. J. C **75** (2015) no.9, 439 <https://doi.org/10.1140/epjc/s10052-015-3659-3>[arXiv: 1506.05348 [physics.data-an]]
24. M. Rovere et al., “CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High Energy Physics,” Jan 2020, [arXiv: 2001.09761 [hep-ex]]