Project information

Project full title	European network for developing new horizons for RIs
Project acronym	EURIZON
Grant agreement no.	871072
Instrument	Research and Innovation Action (RIA)
Duration	01/02/2020 – 31/01/2024
Website	https://www.eurizon-project.eu/

Deliverable information

Deliverable no.	D5.6
Deliverable title	Final report on the software developments
Deliverable responsible	CERN
Related Work-Package/Task	WP5, Task 5.3
Type (e.g. Report; other)	Report
Author(s)	Lorenzo Valentini, Andre Sailer
Dissemination level	Public
Document Version	1
Date	25/09/2023
Download page	

Document information

Version no.	Date	Author(s)	Comment
1	19/09/2023	Lorenzo Valentini	First draft
2	25/09/2023	Andre Sailer	Second draft

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

Preamble

In the first deliverable D5.1 of task 5.3, the status of the software for the SCT (Super Charm Tau factory) detector was described. The deliverable described the *Gaudi*-based *Aurora* framework, and the SCT specific details of the software developments. The Aurora framework is based on the DD4hep geometry description, and a podio based event data model (EDM) library. The D5.1 deliverable also alluded to the plans to integrate more closely with the Key4hep software stack, which is under development by different communities performing future collider studies, such as CEPC, CLIC, FCC, and ILC. Following the unprovoked Russian war against Ukraine and the pivot of this project towards EURIZON, contributions to the SCT specific software framework were ended. Subsequently developments focused uniquely on the general Key4hep stack and its use in a distributed computing environment. Therefore, this report describes the Key4hep project, its current development status, and how it was integrated and tested with the iLCDirac distributed computing tool.

1 Introduction

Detector studies for future experiments require advanced software tools to estimate performance and optimize their design and technology choices. These advanced software tools must include the possibility for full or parameterized detector simulation, the reconstruction of tracks and calorimeter clusters, jet clustering, flavour tagging, and analysis. A combined solution for all these issues should also allow experiments to navigate seamlessly between different stages of their life cycle: for example, from parameterized detector studies to find a performance envelope of their experiment to full and detailed simulation and reconstruction studies to confirm the validity and feasibility of the assumptions used in the parameterized simulation. The consistency of the software stack also allows experiments to extract performance parametrisations from detailed simulation studies to create large-scale samples that are otherwise not feasible for small communities with limited access to computing resources.

The Key4hep project offers a solution for these use-cases by providing a structured software stack, which integrates individual packages towards a complete data processing framework for high energy

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

physics experiments. The sharing of common components will reduce the overhead faced by different communities. Moreover, Key4hep aims to provide an easy-to-use product for librarians, who provide software installations, the developers creating or adapting components, and the end-users. Figure 1 schematically shows the three major ingredients for the project: a processing framework which connects all the pieces; a way to describe the geometry of the experiments and use the information for simulation, reconstruction, or analysis; and an event data model to exchange data between the pieces and for persistency. For Key4hep the processing framework is Gaudi, the geometry information is provided via DD4hep, and the event data model is provided by EDM4hep and podio.

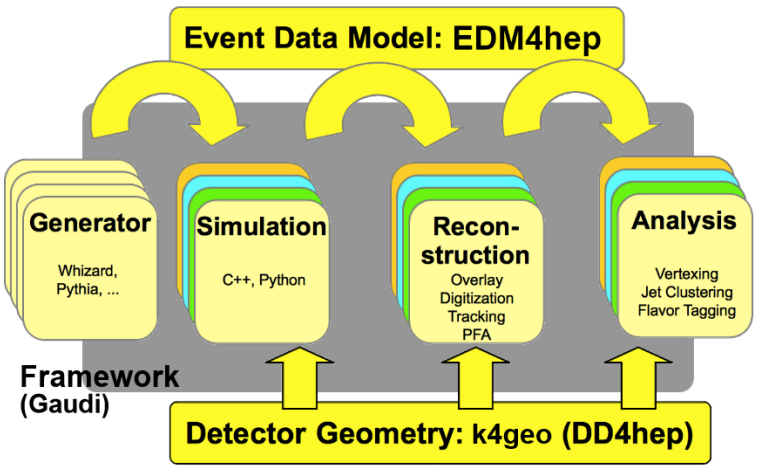

Figure 1: Main ingredients for the Key4hep project: geometry information, event data information, and a processing framework

The Key4hep software is jointly developed by many stakeholders from CEPC, CLIC, FCC, ILC, EIC, or the Muon Collider. Open meetings are held every week. The source code is publicly available on https://github.com/key4hep. Documentation about the project and how to use it can be found under https://key4hep.web.cern.ch.

2 Key4hep Software Stack

The Key4hep software stack is based on the Gaudi data processing framework, the EDM4hep event data model, and the DD4hep detector description toolkit. In addition, the software stack uses additional libraries or programs for Monte Carlo event generation, such as Whizard or Pythia, the Geant4 detector simulation toolkit, and the Delphes library for parameterized simulations. All these libraries, and more basic packages, are built together with the *spack* package manager.

The functionality that Key4hep provides is built on top of these packages. A core library "k4FWCore" provides functionality such as reading and writing the EDM4hep files, and common interfaces for other services. k4gen provides interfaces to Monte Carlo generators and tools for event handling, for example vertex smearing. k4SimGeant4 contains interfaces to perform Geant4 simulation inside Gaudi, while k4SimDelphes offers access to Delphes' parameterized simulations. In addition, there are packages, such as k4Clue for calorimeter clustering, or k4ACTSTracking for track reconstruction, providing access to the reconstruction algorithms of CLUE and ACTS respectively. CLUE is a clustering algorithm implemented by the CMS experiment, and ACTS is "A Common Tracking Software" originally developed for the ATLAS experiment. The main contribution by EURIZON is the k4MarlinWrapper, that

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

allows for the complete integration of functionalities from the iLCSoft environment developed and used by CLIC and ILC (see the next section).

Figure 2 shows the possible dataflows with the Key4hep software stack. Starting with the events created by Monte Carlo Generators, one can either perform full simulation and reconstruction, or parameterized simulation, and use the same analysis in the end in either case.

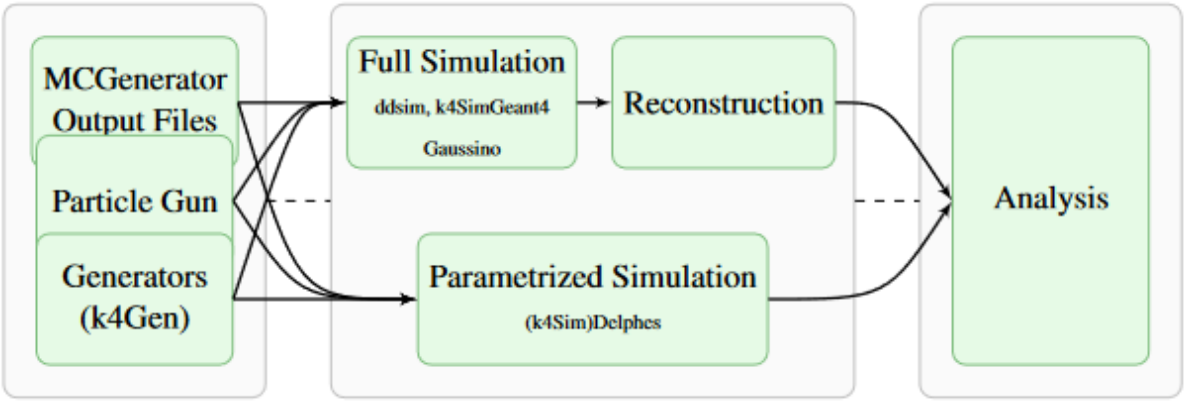

Figure 2: Possible data flows in the Key4hep software stack

For developers and users, nightly builds of the software are created by the CERN team, and freely available on the CVMFS file system under /cvmfs/sw.hsf-nightlies.org/key4hep for three different operating systems: CentOS 7, Ubuntu2204, and Alma9. Releases of the software, which are kept indefinitely to provide a stable set of components, are also made when requested or when major developments have been made. These releases are also found on CVMFS under /cvmfs/sw.hsf.org/key4hep. The software provided on CVMFS is also used for processing on the computing grid (see Section 3).

Other tasks in WP5 also make use of the Key4hep software stack. For example, the detector geometry of the drift chamber under development in task 5.5 is implemented in DD4hep. To support the reconstruction for the drift chamber, the event data model (see Figure 3), was modified to include the "RawTimeSeries" and "TrackerPulse" data types. The RawTimeSeries is used to store the electronic pulses induced in drift chamber wires. In the digitisation stage, a *RawTimeSeries* is converted to a *TrackerPulse*, which can then be used for the track reconstruction.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

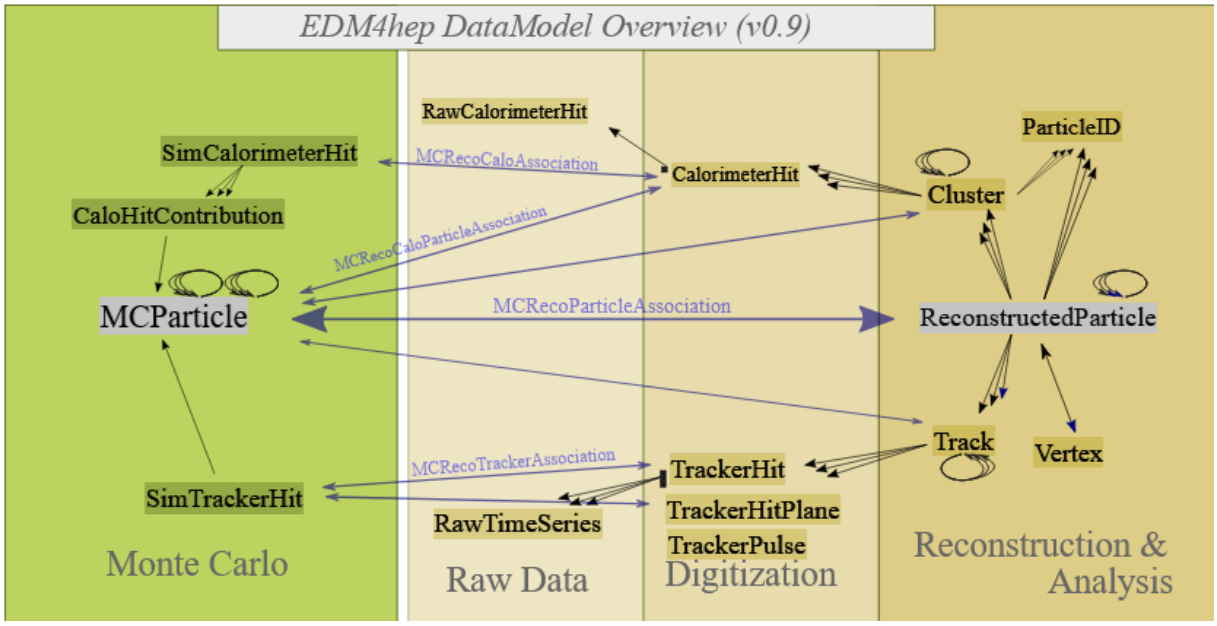

Figure 3: Details of the EDM4hep event data model.

2.1 Integration of iLCSoft

The re-use of existing tools allows one to perform interesting studies more quickly, rather than develop and debug software until it is stable enough to use it. One of the major developments for Key4hep was therefore the *MarlinProcessorWrapper*. The *MarlinProcessorWrapper* encapsulates the processors that are used in the Marlin processing framework (as opposed to Gaudi) of the iLCSoft software stack.

Because the Gaudi configuration is based on the Python programming language, and Marlin is using XML for this purpose, a converter was developed, which converts Marlin's XML files into Python based steering files for Gaudi. Another difference between the iLCSoft and Key4hep software stacks is the Event Data Model, which in the Marlin case is LCIO instead of EDM4hep. To assemble arbitrary processing chains between Gaudi- and EDM4hep-native algorithms and those using the MarlinProcessorWrapper, the event data model must be converted in memory during run-time from LCIO to EDM4hep and back. This idea is schematically shown in Figure 4. This MarlinProcessorWrapper on the one hand allows the ILC and CLIC communities to smoothly transition to the Key4hep software stack, and on the other hand makes the existing tools from iLCSoft available for other communities, e.g. FCC, with limited overhead. The iLCSoft software chain includes functionality for digitisation, track reconstruction, particle flow clustering with PandoraPFA, interfaces to FastJet, the LCFIPlus flavour tagging, and high-level reconstruction tools for Particle ID and kinematic fitting. The MarlinProcessorWrapper also works with any processor that users might have written for their own purposes.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

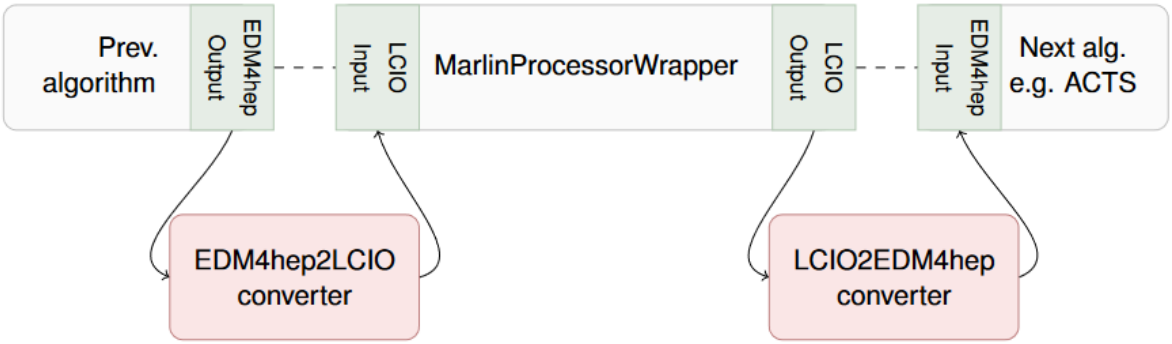

Figure 4: Sketch showing the use of the MarlinProcessorWrapper in a processing chain.

3 Distributed Computing with Key4hep and iLCDirac

The size of the Monte Carlo samples needed for future detector studies makes it necessary to use distributed computing to access resources that might not be available directly for the participants. Sophisticated distributed computing tools, such as DIRAC (Distributed Infrastructure with Remote Agent Control), also offer convenient workload and data management solutions that make the use of distributed resources more efficient for end-users. DIRAC builds a layer between the users and the resources, thus offering a homogeneous interface to a heterogeneous set of computing or storage resources.

For the work that is performed as part of EURIZON WP5, the ILC and FCC Virtual Organisations (VO) of the European Grid Infrastructure offer access to distributed resources. Both VOs are part of the iLCDirac instance, originally deployed only for the ILC VO. As part of the iLCDirac extension to DIRAC, a set of common interfaces is available to all iLCSOFT applications. Recent developments in iLCDirac added code wrappers to facilitate the management of new workflows originating from Key4hep and used by the FCC, such as those employing Gaudi and Delphes. Documentation for the use of iLCDirac can be found under https://ilcdirac-doc.web.cern.ch.

3.1 iLCDirac Jobs and Applications

The guiding principle, at the time of the design of iLCDirac, had been to avoid creating tightly coupled relationships between jobs and applications. In this framework, an application is independently definable, separate from any specific job-related property. Furthermore, all applications within iLCDirac are characterized by a set of elementary properties that uniquely describe them. Each application has a unique definition, which includes a name and potentially a version. Typically, it requires a steering file or an option file and generates a log file and at least one data file as output. All applications within iLCDirac are designed to inherit from a common *Application* class, see Figure 5.

The Job classes were also structured in a generic fashion, see Figure 6. A foundational Job class, inherited from the DIRAC Job class, is implemented to allow for the definition of general job requirements. There are two primary subclasses: *UserJob*, intended for user-oriented tasks rather than production tasks, and *ProductionJob*, designed for submission to the DIRAC *Transformation System*. Regardless of job type (UserJob or ProductionJob), the application interface remains consistent, characterized by a '*job.append*' method that accepts an application instance as an argument.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

The *Transformation System* of DIRAC enables efficient centralised production of large data samples, such as those needed for detector studies. Centralised productions for a given community usually consist of a few well-defined workflows, where only a small number of high-level parameters are ever changed. For example a community will always simulate the same detector, with the same configuration of the software, and only the event type changes.

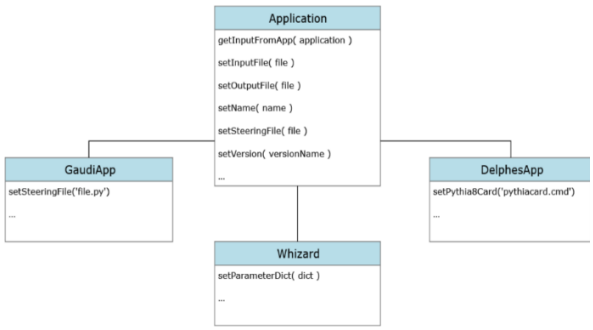

Figure 5: GaudiApp, DelphesApp and Whizard inheriting from the Application base class

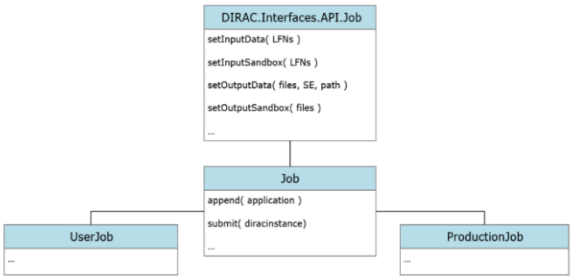

Figure 6: UserJob and ProductionJob inheriting from the Job base class

3.2 Integration of Key4hep Applications in iLCDirac

Thanks to the design of the iLCDirac framework, it was possible to introduce the new Key4hep-based FCC workflows just by creating the new applications, affecting minimally the rest of the software, with only a few exceptions. Now, the Key4hep-based applications and the workflows supported by iLCDirac are those shown in Figure 7 with their output file formats.

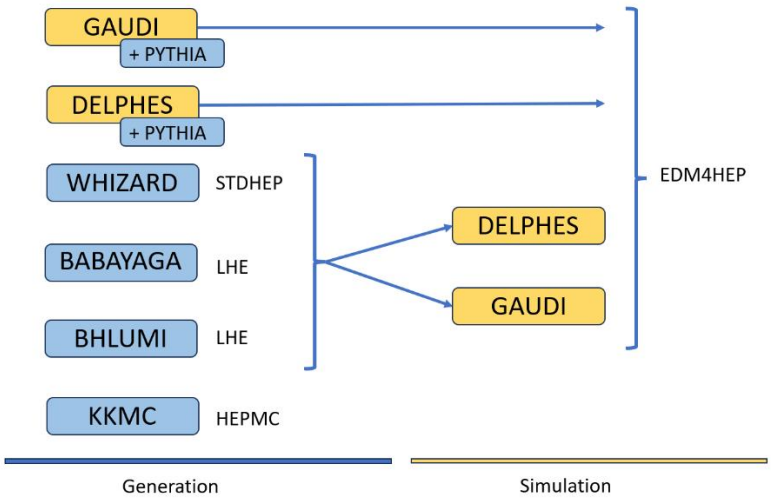

Figure 7: Key4hep based workflows implemented in iLCDirac. STDHEP, LHE, HEPMC and EDM4HEP denote the output file format of the respective applications.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

3.2.1 Generators

Currently, five event generators available in iLCDirac. Of those, *Whizard*[1], KKMC, Bhlumi and Babayaga can be used directly; Pythia8 can be used inside the Delphes or Gaudi applications, before the simulation happens.

No developments were needed for the Whizard event generator, which was already interfaced in iLCDirac. For KKMC, minor adaptations were necessary to use the newer versions that are available in the Key4hep software stack. The interfaces for two applications used for FCC event generation, Bhlumi and Babayaga, were added to iLCDirac. These interfaces follow the scheme used by KKMC: they directly call the generators (available on the Key4hep stack as standalone programs) with a script that is generated automatically, and which contains the input parameters chosen by the user.

3.2.2 Delphes, Gaudi, and Treatment of Pythia Cards

The possibility of using the Delphes parameterized simulations was added to iLCDirac by creating a specific application. In particular, the Delphes application uses the standalone executables available from the k4SimDelphes package: `DelphesSTDHEP_EDM4HEP` for reading STDHEP inputs, `DelphesROOT_EDM4HEP` for reading ROOT files in the Delphes format, `DelphesPythia8_EDM4HEP` for running Pythia8 as part of the simulation. Since Pythia8 has LHEF reader capabilities, the last executable can also be used for running simulations after a generator that produces outputs in the LHEF format. Only the output of KKMC, which is in the HEPMC format, currently cannot be used for a Delphes simulation.

Pythia8 is used for some FCC workflows, based on either the Gaudi or Delphes applications. To generate events, Pythia8 requires, among other inputs, a so-called *Pythia Card*, containing information about the specific generation process. At the submission of a workload, Pythia cards will be read from the local directory of the user running the job, or, if not present, retrieved from CVMFS. To ensure that the generation process is consistent with the metadata of the job, the content of the card is checked and compared to the metadata and with the permitted values of the random seeds.

The Gaudi application already existed at the beginning of the project. In principle, it can support any Gaudi workflow, but this happens according to the available steering files. However, the treatment of Pythia cards was aligned with that used also for the Delphes application.

These developments to support different Monte Carlo generators were necessary for their convenient use in a distributed environment. When a user submits many jobs for a Monte Carlo generator, the iLCDirac system ensures that each job will use a unique seed for the random number generator, and that the output files will have different names. This ensures that the user does not have to worry about accidentally creating the same file with name and content in each job.

3.3 FCC Transformation Interface

Besides the applications used in the FCC transformation workflows, also the convention for the storage of the output files was implemented. It was decided that FCC productions would save their outputs using paths that follow a logic slightly different from that used in the ILC or CLIC transformations. They follow the convention:

[1] Whizard stands for: W, HIggs, Z And Related Decays

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

/fcc/<initial_state>/<campaign>/<energy>/<process_and_final_state>[/<detector>]/<type>/DiracProdID
This required differentiating between FCC and ILC transformations in the *ProductionJob* class, where the output paths are decided.

To configure the transformations for the FCC community, the *dirac-fcc-make-transformations* command was implemented. This command allows the production managers of FCC to define the high-level parameters, such as the event type or detector model, and the chosen workflow to create the large-scale centralised productions. Figure 8 shows an example how the configuration for transformation looks like. Usually the production manager would provide different *generatorSteeringFile* values which contain the event type definition, keeping the rest of the options constant. In this example the detector is defined by the DetectorCard set for the DelphesApp.

Figure 9 shows the speed with which the 5000 requested jobs are run. First jobs are "Waiting" (blue), going in to the "Running" (beige) when resources are available, ending up in "Done" (green) or "Failed" (red). The DIRAC transformation system takes care 3re-submitting any failed job so that exactly 5000 successful jobs, and thus output files, will be provided, for example.

```
[delphesapp]
ExecutableName = DelphesPythia8_EDM4HEP
DetectorCard = card_IDEA.tcl
OutputCard = edm4hep_IDEA.tcl
Version = key4hep_230408

[Production Parameters]
machine = ee
prodGroup = several

softwareVersion = key4hep_230408
generatorApplication = delphesapp
generatorSteeringFile = p8_ee_Zbb_ecm91.cmd

configVersion = key4hep-devel-2
configPackage = fccConfig
```

Figure 8: Partial configuration for an FCC transformation

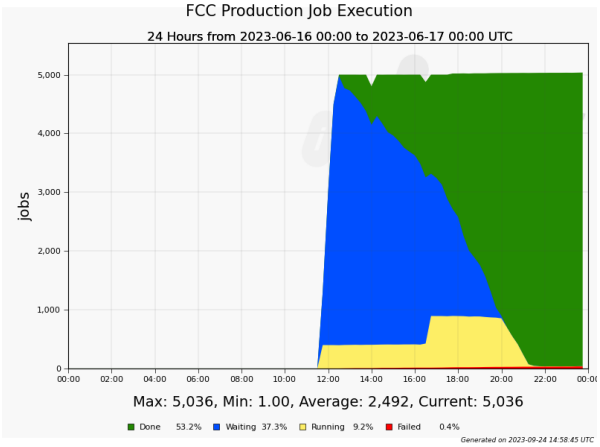

Figure 9: Job execution for an FCC transformation. Percentages are based on area, not fraction at the end.

3.4 Additional iLCDirac Developments

To enable a simpler use of the Key4Hep nightly builds, which change their location every night (due to the date or a *hash* or both in the path of the files), it was necessary to base the path of the detectors on the K4GEO environment variables, instead of storing the complete path to the detector. This allows the user to pick any nightly build, and iLCDirac will automatically obtain the correct detector locations.

4 Conclusions

The contributions of the EURIZON WP5 Task 5.3 to the Key4hep software stack are extremely valuable to preserve proven functionality from iLCSoft for future detector studies. The k4MarlinWrapper is an essential piece of the Key4hep project which allowed its easy adoption by the CLIC community and provides core functionality used by the FCC and other communities. The developments to integrate Key4hep into the iLCDirac distributed computing environment will further allow the wide exploitation of the Key4hep software, and support detector studies through its ease-of-use and wide-ranging

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.

transnational access to computing and storage resources. The development of FCC workflows on iLCDirac has progressed, and iLCDirac has already been used to produce real data for FCC generation and simulation. The natural evolution of this project will be the addition of new workflows and the completion of the workflows that are still unfinished.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 871072.