FCWSW:
Status and (Users) workflows

FCC Week 2021
Valentin Volkl (CERN)
for the FCC software team
Introduction

- **FCCSW: Gaudi-based framework for FCC Detector- and Physics Studies**
  - Kick-off 2016, used for Conceptual Design Report
  - Should cover Generation, Fast- and Full Simulation, Digitization, Reconstruction
- **Uses established and upcoming experiment-independent software libraries**
  - try not to re-invent the wheel
- **Modular approach**
  - “Plugin”- system for runtime configuration

Recent transition to take further advantage of common Key4hep software
Key4hep Transition

Recent update from FCCSW 0.16 to 1.0.preX

Really three separate transitions:

1. Gaudi CMake upgrade
2. Change of Data Model to EDM4hep
3. Restructure Repositories
Transition to Key4hep: Gaudi CMake Overhaul

- With version v35, Gaudi modernized its CMake-based build system
- Required lots of changes in the CMakeLists.txt of FCCSW
  - Decided to do this transition at the same time
- Less opaque macros, more plain, modern cmake code!
Transition to Key4hep: Data Model

- Foundation of common software
  - Provides interface
- Same implementation as old FCC-EDM, but updated type definitions
  - Mostly straightforward replacements
  - Clearer distinction between Rec / Sim types
  - “Vertex” folded into MCParticle

- Completely transitioned!
  - Apart from few FCC-hh specific calo-reconstruction tools
  - Calorimeter Reconstruction transitioned by Juraj Smienko and Brieuc Francois

- … but long tail of validation and testing
Transition to Key4hep: Repository Restructure

- key4hep/k4FwCore: Basic I/O components
- key4hep/k4Gen: Generators and Particle Guns
- key4hep/k4SimDelphes: Delphes Fast Sim
- key4hep/k4SimGeant4: Geant4 Full Sim
- hep-fcc/fccdetectors: DD4hep models of FCC detector geometries for Full Sim
- hep-fcc/k4RecCalorimeter: Calorimeter Reconstruction Code
- hep-fcc/dual-readout: DD4hep model of the DREAM dual readout calorimeter

Separate repositories make it easier:
- to manage dependencies
- keep track of changes (separate commit and release history)
- share code between Key4hep stakeholders

Some added complexity when changing code in several repositories

- Tooling can help! [link to docs]
- Nightly builds from the HEAD of every repository are available:
  - source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
Workflow overview

The FCCSW repository houses no more actual code, but still list of examples

The Starterkit webpage gives a more verbose introduction:
User workflow: Delphes

- k4SimDelphes gives output in edm4hep
  - [link to starterkit](#)

- Part of a coherent approach to Simulation / Reconstruction in Key4hep
  - Same structure in output of Full / Fast approaches
  - results in a “ReconstructedParticles” branch

- Currently run via standalone executables
  - DelphesPythia8_EDM4HEP, etc.
  - Integration in framework finished
  - but needs validation
Full Simulations

- See FCCSW/Examples/options/geant_fullsim_*
- DD4hep has its own plugin systems - ongoing work to make changes to parameters and whole detector subsystems even more “automatic”
- Right now, still need to harmonize between LCGEO / FCCDetectors
  - Issues with different conventions for sensitive detectors
- DD4hep implementations are crucial for common software
ILCSoft is a mature software suite that has been previously used for FCC studies (CLD, hh-jet-tagging)

See the documentation (https://key4hep.github.io/key4hep-doc/examples/clic.html) for details

Allows to run both full workflows and individual Marlin processors as part of FCCSW workflows
Fast Sim in Geant

- Users can attach **parametrizations** to detector regions
- “Geant4 fast and full simulation for Future CircularCollider studies” [link](https://github.com/HEP-FCC/FCCSW/blob/master/Examples/options/geant_fastsim_tklayout.py) to CHEP ‘17 proceedings (Anna Zaborowska)
Modular approach

Updated style of “job option files” allows for easier re-use of parts of a job

```python
# Geant4 algorithm
# Translates EDM to G4Event, passes the event to G4, writes out outputs via tools
from Configurables import SimG4Alg
geantsim = SimG4Alg("SimG4Alg")
from Configurables import SimG4PrimariesFromEdmTool
geantsim.eventProvider = SimG4PrimariesFromEdmTool("EdmConverter")
geantsim.eventProvider.GenParticles.Path = "GenParticles"
ApplicationMgr().TopAlg += [geantsim]
```

... even python-style import of configuration blocks!

```python
from k4_workflow_blocks.fccsw.detector_fcc_hh_main import *
```
Conclusion

- Key4hep transition took effort but opens many possibilities for collaboration
  - Foundation for a full simulation / reconstruction chain
- Modular approach with plug-and-play components
- Feedback and questions always welcome (cern.ch/fccsw-forum for example, or join the meetings!)