Status and Plans for the FCC Software Project

Benedikt Hegner (CERN)
On behalf of the FCC SW Team

FCC Week
15.4.2016

Status of the Software Project

- Aim of the Software Project is to **support all of hh/ee/eh studies**
  - Need to support multiple detectors in simulation and reconstruction
  - Need to support simulation in different levels of details
- Since FCC Week 2015 plenty of work finished
  - Most of the progress up to our highly motivated students!
- Simulation
  - **Delphes** integrated and **ready to use**
  - Technical infrastructure for combined **fast/full simulation with Geant4** in place
- Reconstruction
  - Joint project with ATLAS to apply their track reconstruction software (**ACTS**)
  - **PAPAS** for fast simulation and particle-flow reconstruction
- Analysis
  - Standalone reader for FCC data model
  - **Heppy** as python-based analysis framework
  - Both can be installed on your laptop!
- **For details see other presentations in this session**

FCC Software in the HEP SW Landscape

- We do not have resources to do everything by ourselves
  - Whenever there is something (almost) ready to use ⇒ take advantage of the work others do!
- Our software is based on the following external software
  - **Gaudi** as underlying framework
  - **Delphes** for parameterized simulation
  - **Geant4** for simulation
  - **DD4hep** for detector description
- Collaborating with
  - ATLAS on tracking
  - CMS on analysis interface
  - LHCb on simulation framework and infrastructure
  - CLIC on grid processing (planned)
  - **Surprisingly successful cooperation within HEP SW community**
- We are as well contributing to the HEP Software with our additions
  - **Heppy and PAPAS** as integrated Python-solution
  - **PODIO** for data models

FCC Data Model

- Reviewing the situation in the LHC experiments most of them overdid on inheritance and polymorphism
- Most physicists don't feel productive in current data models
  - Many n-tuple "frameworks" around
- During the years of LHC software understanding of OO evolved a lot!

⇒ We decided to make some heavier investment into data models

Data Model - Current Status

- The outcome of our efforts on the data models is the PODIO (= POD I/O) library project
  - Replacing the “albers” prototype presented at FCC Week 2015
  - http://github.com/HEP-FCC/podio
  - Interfaced to C++ and Python
- PODIO being watched by non-FCC projects
  - Became part of the EU AIDA2020 program
  - Linear Collider Community interested in replacing LCIO with it
  - **LHCb actively evaluating it for their upgrade project**

⇒ In my opinion it was a good choice not to copy existing solutions

Multi-language support

- Our aim is to make the start into our software as easy as possible
- **Python** is easy-to-use
  - Good for analysis, to test ideas and to prototype algorithms
- **C++** is fast and strict
  - For detailed studies and high statistics
- Both languages are powerful
- Decided to **support both languages on almost equal footing**
  - Interface between C++ and Python world is the FCC data model
- Code/features can migrate from Python to C++ once necessary
- More details in presentation by *Clement Helsens*

⇒ Should boost you in productivity

Documentation

Documentation is always the weakest part in a software project

- We however try to develop code/documentation at same pace

We provide two places of documentation

- **Wiki** pages for tutorials
- **Wiki** pages for user level information
- **Doxygen** documentation for technical information

Not always clear where to draw the line between user/developer.

For all SW docs and infrastructure we have a single entry point

http://fccsw.web.cern.ch/fccsw/

Please note: To access the wiki pages you need to get a (ligthweight) CERN account: https://account.cern.ch/account/Externals/

Single entry point for FCC SW

FCCSW

Main page for documentation and resources.

Documentation

Doxygen:

- PODIO: Underlying Data Model library
- FCC Core Software
- FCC Event Data Model
- DD4hep: Detector Description Toolkit

Other links:

- C++ Coding Style
- C++ Guidelines
- FCCSW Readmes
- FCCSW TWiki Pages

Other resources

- FCC JIRA Issue Tracker
- FCC on GitHub
- FCC Continuous Integration

Code quality

experimental

Static checks:

- FCC Core Software (cleaned)
  FCC Core Software (verbose)
- FCC Core Software (w/ flint++)

Runtime performance:

- FCC EDM example (igprof)

last generated on 09 Feb by hegner

http://fccsw.web.cern.ch/fccsw/

Software Quality and Automatization

Spent some effort on automatization of procedures

Jenkins

- Continuous integration
- Software tested every night
- (Doxygen) documentation created every night
- (Core) team can trigger tests by pushing a button

Software Quality addressed by

- Integration tests
- Static code checks
  - Flint++ for quick checks
  - Clang and SAS for code and convention checking

There is a long wish list of features to add,
but the SW team **established a solid baseline.**

Collaborative Development

All of FCC's Software is open-source and available on GitHub

http://github.com/HEP-FCC

Collaboration works via **pull-requests and continuous integration**

- Clone the repository
- Add your great new feature
- Open a "pull request"
- Get your changes automatically tested
- We review it together with you and eventually add it to the official repo

For a hands-on introduction/tutorial see

https://twiki.cern.ch/twiki/bin/view/FCC/FccSoftwareGit

We are open to any idea and addition!

The Team

- Software is often treated as a topic that just has to work without thinking of the hard work and the people contributing to the effort. For FCC the people that have contributed with code so far are:
  - Alice Robson
  - Andi Salzburger
  - Andrea Dell`Acqua
  - Anna Zaborowska
  - Benedikt Hegner
  - Clement Helsens
  - Colin Bernet
  - Joschka Lingemann
  - Julia Hrdinka
  - Michele De Gruttola
  - Valentin Voelkl
  - Zbynek Drasal

You are very welcome to join this list!

Plans

- Geometry
  - Once baseline / strawman detector concepts exist ⇒ integrate and test in simulation
- Simulation
  - Recently started implementing infrastructure for pile-up/beam background handling
  - For other points see presentations by *Anna Zaborowska* and *Clement Helsens*
- Reconstruction
  - For Tracking see talk by *Julia Hdrinka*
  - For Particle Flow:
    - Porting of PAPAS Python code to C++
    - Interfacing to PandoraPF (currently used in CLIC)
- Analysis
  - See presentation by *Clement Helsens*
- We are definitely not short of ideas
  - But if you find a feature missing - let us know!

Summary

- Quite some progress since FCC Week 2015
  - Mostly up to our junior team members!
  - Delphes and PAPAS parameterized simulation are ready to use
- Our work bases a lot on existing software
  - Not trying to reinvent the wheel yet another time
- Picked a few areas where we think existing solutions need to be improved
  - Seems to be a good idea
- Provided documentation and tutorials

- Join our mailing list to get more information and regular updates
  fcc-experiments-sw-dev@cern.ch

Data Model - Basic Idea

- Simple memory model
  - Employ **simple structs (PODs)** instead of fat objects
  - Allow for data access for vectorization
- Simple class hierarchies
  - Wherever possible use **concrete types**
  - Favor composition over inheritance
- Simple I/O setup
  - Keep **transient to persistent layer as thin as possible**
  - Whenever possible store PODs as is
- No global states and hidden call-backs
  - should integrate well w/ multi-threaded frameworks
- Simple model generation
  - Employ **code generation** to make it easy for the user
  - Quick turn-around for improvement on the back-end
- Keep it framework-independent