



# Software for Future Experiments

---

SFT GROUP MEETING

Joschka Lingemann  
EP-SFT CERN

# Working at SFT

---

Started my Fellowship in September 2015

- First few months ramping down previous activity in CMS
- Previously worked on CMS hardware trigger

Project: Work on software to support the FCC design study

- Development: Simulation framework, event processing framework, ...
- Infrastructure: Installations, continuous integration, ...

Very small group of a handful of contributors

Design study timeline: First drafts in the beginning of 2018

# The FCC software goal

---

Support experiments for all colliders: ee, hh & eh

Support physics and detector studies

- Detector concepts: Moving targets
- Both fast and full simulation essential

One software stack: Support all experiments from event generation to physics analysis

Collaborative approach:

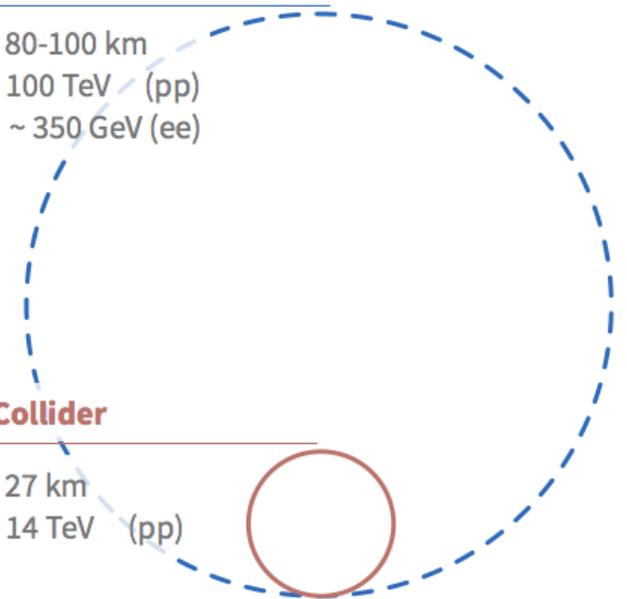
- Extract from the LHC experiments if possible
- Invest into new solutions where necessary
- Benefit from AIDA / HSF efforts

## Future Circular Collider

Circumference: 80-100 km  
Energy: 100 TeV (pp)  
~ 350 GeV (ee)

## Large Hadron Collider

Circumference: 27 km  
Energy: 14 TeV (pp)

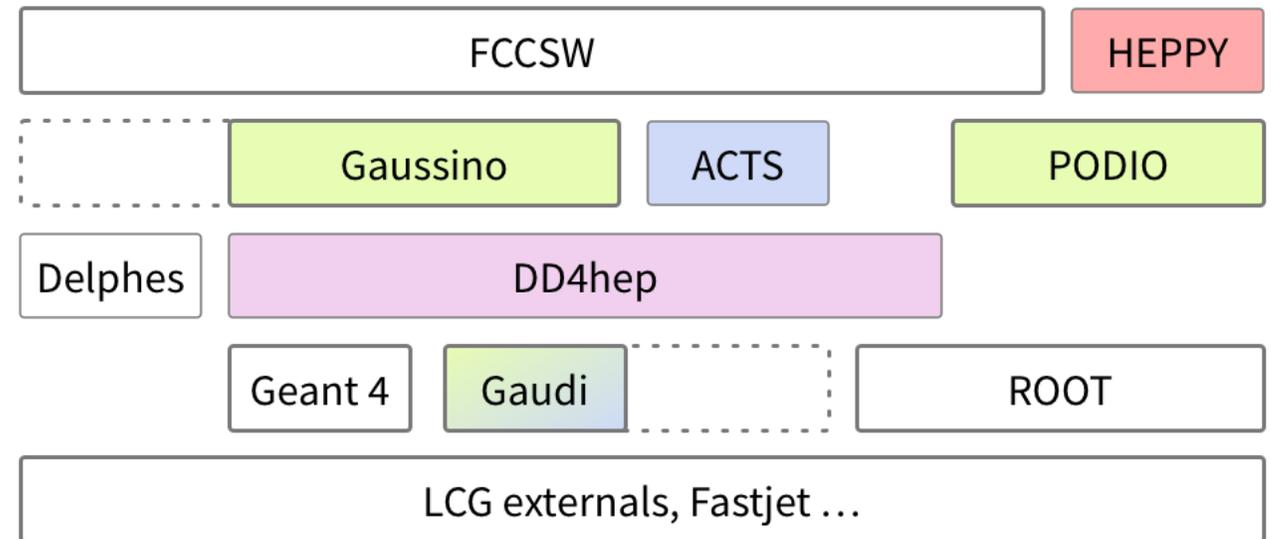


# Collaborating where we can

---

Not re-inventing the wheel for FCC:

- Take existing solutions
- Invest to extract from experiment
- Collaborating with
  - LHCb
  - ATLAS
  - CMS
  - CLIC / ILC



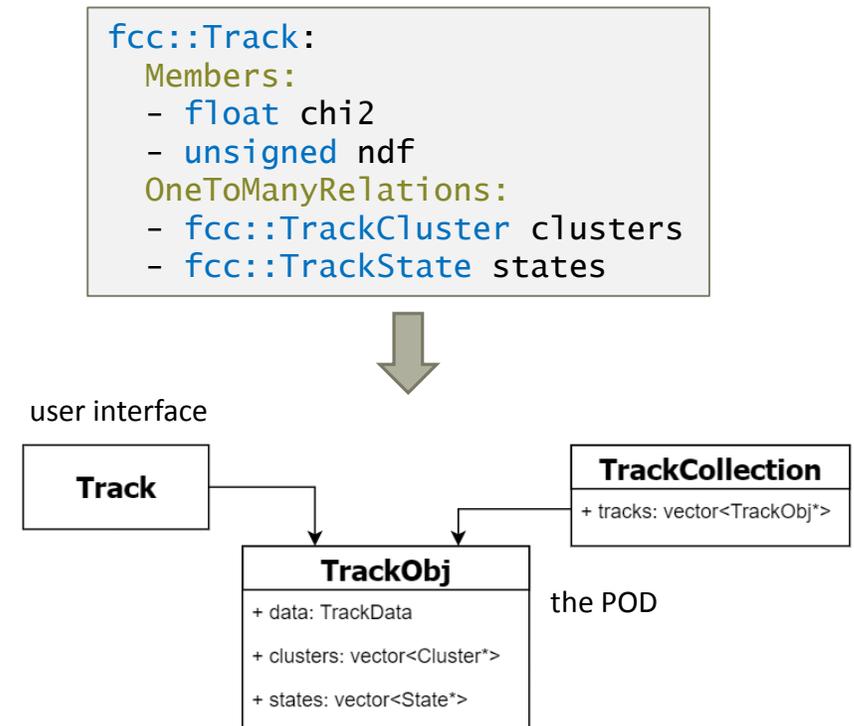
# Event Data Model: PODIO

## Plain Old Data I/O (PODIO)

- Focus on re-usability and flexibility
- Code fully generated from text files
  - Simply describe your data, PODIO does the rest
  - Easily adapt data model to changing requirements
- Python & C++ supported on the same footing
- For more details, see past talks by Benedikt

FCC is the first user of the library, several features added:

- Namespacing of the datatypes
- Support for arrays
- Usability improvements & bug fixes



# Event Data Model: Using PODIO for FCC

---

Event Data Model with PODs => No object inheritance

- Composition instead of inheritance
- Re-usability of client code: Use composing objects
  - E.g.: Jet clustering code runs both on MC particles and reconstructed particles
  - Both have the same “core” particle that can be used for clustering code

Forbid changes to collections after creation: Again composition (e.g. tagged jet = tag + jet)

Reviewed the first iteration of our datamodel

- Fix small inconsistencies
- Reviewed how users approach problems, given our EDM:
  - unclear usage from user point of view
  - Renaming to clarify and remove unnecessary components
- Add some missing components

Important part:  
User support & good examples

# Generic Detector Description: DD4hep

---

DD4hep is a generic tool for detector description

- For details see recent presentation by Marko P.

FCC and Linear Collider Community are the first users:

- Feedback important to try and keep goals aligned
- Bi-weekly developers meeting
- Workshop on conditions (esp. alignment data)
- Keep FCC integration up to date with changes in DD4hep

Move FCC implementations upstream where it makes sense:

- Generic things like eta-phi segmentation for early prototype detectors

User support in FCC:  
How do I implement my  
detector?

# Event Processing with Gaudi for FCC

---

Event processing framework used by LHCb and ATLAS

- Chosen as framework for FCCSW
- Mostly a good experience and positive feedback from users
- A lot of infrastructure out-of-the box:
  - Testing
  - Build tools
    - Although they could be simplified :)

Missing: Documentation and guidelines

- Improved a lot with the software upgrade in LHCb
- Sometimes not on the Gaudi-level but LHCb-specific
- We also started some documentation (should be unified)

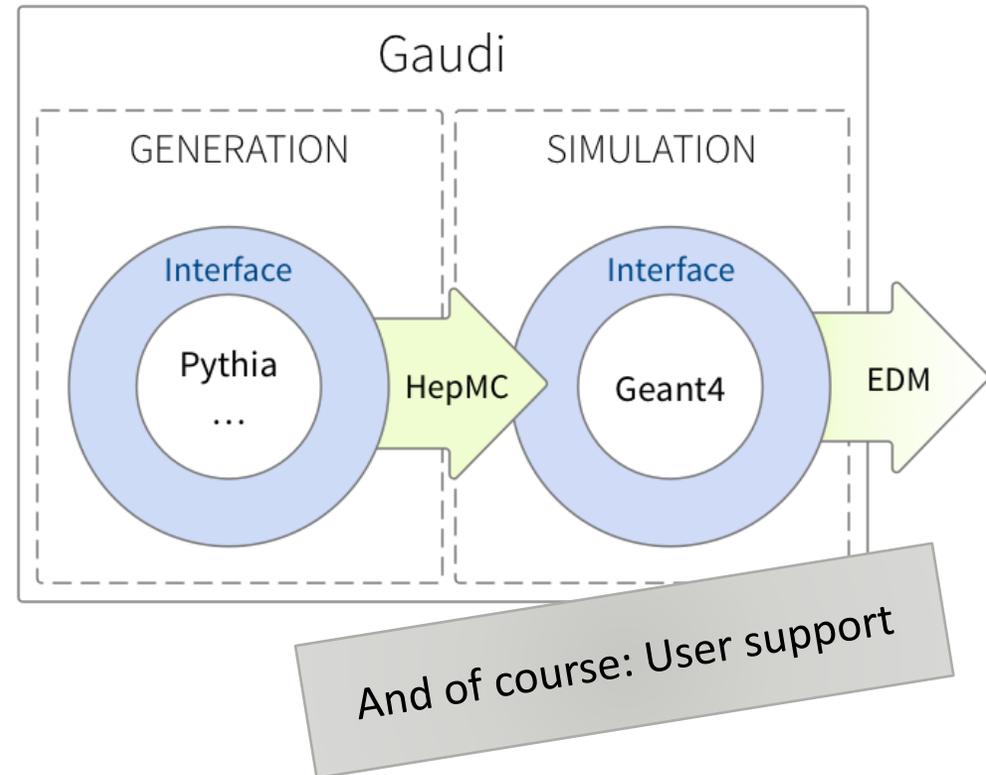
# Event Generation and Geant4 Simulation: Gaussino

## Event Generation

- On the fly: Pythia & Particle gun
- Read LHE files during showering
  - Existing workflow for MadGraph
- Re-use functionality from Gaussino

## Simulation with Geant4

- Integrated full & fast simulation
  - See presentation by Anna
- Simplifying simulation from Gaussino
- **Simulation is feature complete for Design Study and used**
- Re-integration with LHCb Gauss pending



# Parametric Simulation

---

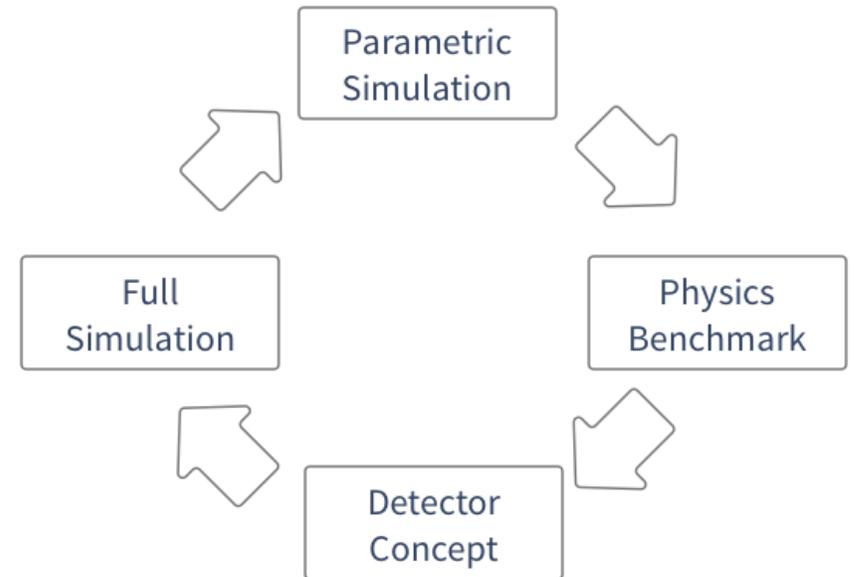
Why parametric simulation?

- Define & study physics benchmarks
- Scan detector parameters
- Validate simulation and analyses

Delphes & PAPAS (**P**arametrized **P**article **S**imulation)

- Both integrated in FCC software (allow cross-checks)
- Use particle flow algorithms from PAPAS

Delphes integration: Re-written to be more modular & configurable



---

## PART II: INFRASTRUCTURE

# Collaborating: The downsides

---

Fragmented code base: Several small dedicated repositories

- Confusing to newcomers
- Unclear where to contribute
- Getting a working installation: 5+ repositories clone & compile + dependencies
- Documentation mostly in the repositories
  - Hard to find what you need to know

**User confusion is guaranteed**

=> Automate & simplify as many of these processes as possible

# Documentation

---

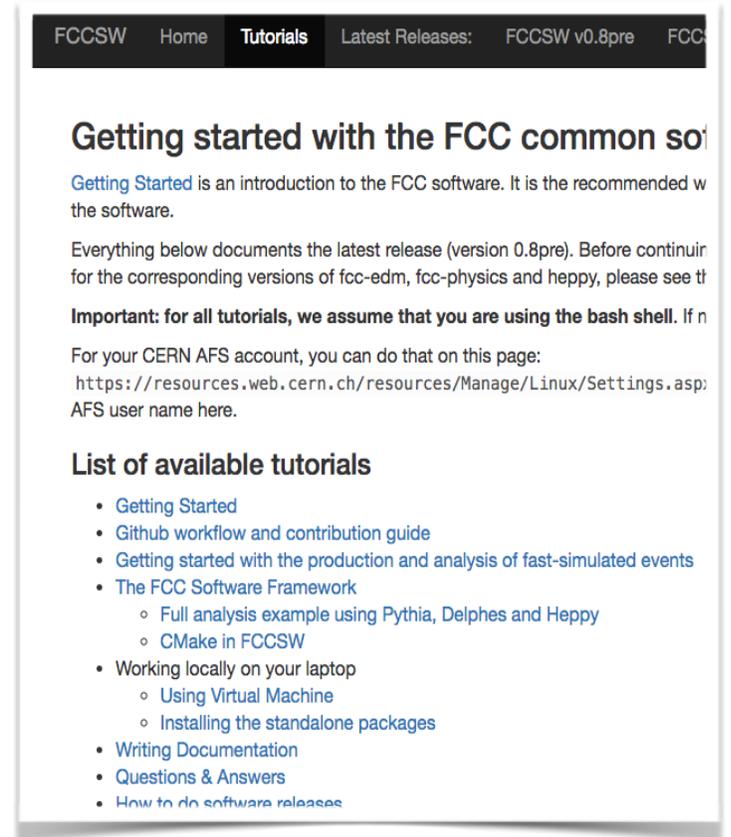
When I started: markdown in several repositories + Twiki

- Hard to maintain & confusing

New FCC-software website with Jekyll:

- Static web-pages, markdown support out of the box
- Bootstrap step with python:
  - Collect all markdown files with python using GitHub API
  - Few adaptations to make the website self-contained
- Yaml file with release information (dependency versions, etc.)
- Moved Twiki to git repository solely for tutorials

One point to start: [fccsw.web.cern.ch](https://fccsw.web.cern.ch)



The screenshot shows a web page with a dark navigation bar at the top containing links for 'FCCSW', 'Home', 'Tutorials', 'Latest Releases:', 'FCCSW v0.8pre', and 'FCC'. The main content area has a heading 'Getting started with the FCC common so' followed by a paragraph: 'Getting Started is an introduction to the FCC software. It is the recommended way to get started with the software.' Below this is another paragraph: 'Everything below documents the latest release (version 0.8pre). Before continuing with the software for the corresponding versions of fcc-edm, fcc-physics and heppy, please see the corresponding documentation.' A bolded section follows: 'Important: for all tutorials, we assume that you are using the bash shell. If not, please see the documentation for the corresponding shell.' Then, a paragraph: 'For your CERN AFS account, you can do that on this page: <https://resources.web.cern.ch/resources/Manage/Linux/Settings.asp>; AFS user name here.' The final section is 'List of available tutorials' with a bulleted list of links: 'Getting Started', 'Github workflow and contribution guide', 'Getting started with the production and analysis of fast-simulated events', 'The FCC Software Framework' (with sub-links for 'Full analysis example using Pythia, Delphes and Heppy' and 'CMake in FCCSW'), 'Working locally on your laptop' (with sub-links for 'Using Virtual Machine' and 'Installing the standalone packages'), 'Writing Documentation', 'Questions & Answers', and 'How to do software releases'.

# Supporting contributors

---

Introduced an off-week “technical meeting”:

- Room for discussion of questions and problems with the software
- Meant to also discuss design ideas / questions about implementation
- Feedback: Quite useful to resolve questions easily
- Sometimes meeting very short (O(10 mins)) but good to touch base

Mattermost: Chat client supported by CERN

- Created for users to give feedback / ask questions
- Fast communication
- Unfortunately, not widely used after first week(s)

# Deploying the software: Spack

---

Package management / distribution system:

- Developed in HPC community, favoured by HSF community
- Candidate to replace LCG-CMake (see Javier's presentation)
- Some generic HEP software packages maintained by HSF

The FCC-specific needs:

- Our own packages (obviously)
- Build against LCG software stack on cvmfs
  - Use LCG text files describing package specs in the release
  - Create yaml configuration file to allow spack to build against that stack
- Nicely synchronises information used for website creation

Nice test-bed for the tool: Our software stack is not very extensive

# Virtual Machines

Virtual machine for use on laptop, CernVM

- CVMFS installation + CernVM = Easy to use virtual machine
- Image can be minimal
- Very convenient to maintain
  - No overhead for the maintainers (uses CVMFS)
  - Standard setup for users with favoured Hypervisor
- Caching of CVMFS problematic for some users
  - Offline work only partly possible

Investigate an alternative to use additionally

A screenshot of a terminal window titled 'Terminal - guest@localhost' within a 'CernVM 3.1 [Running]' environment. The terminal shows a series of messages from Pythia indicating the number of events generated, from 100 to 900. At the end, a user enters the command 'history', and the terminal displays the command history: 1 source /cvmfs/fcc.cern.ch/sw/0.8pre/setup.sh, 2 fcc-pythia8-generate /cvmfs/fcc.cern.ch/sw/0.8pre/fcc-physics/snapshot/x86\_64-slc6-gcc49-opt/share/ee\_Zmumu\_Hbb.txt, and 3 history.

```
Pythia::next(): 100 events have been generated
Pythia::next(): 200 events have been generated
Pythia::next(): 300 events have been generated
Pythia::next(): 400 events have been generated
Pythia::next(): 500 events have been generated
Pythia::next(): 600 events have been generated
Pythia::next(): 700 events have been generated
Pythia::next(): 800 events have been generated
Pythia::next(): 900 events have been generated
[guest@localhost ~]$ history
 1 source /cvmfs/fcc.cern.ch/sw/0.8pre/setup.sh
 2 fcc-pythia8-generate /cvmfs/fcc.cern.ch/sw/0.8pre/fcc-physics/snapshot
/x86_64-slc6-gcc49-opt/share/ee_Zmumu_Hbb.txt
 3 history
[guest@localhost ~]$
```



# Continuous integration

---

Uses SFT Jenkins instance

- Check pull-requests of all our repositories
- Nightlies:
  - Build of all repositories
  - Integration test, checking if masters are synchronised
  - Nightly install if integration test works (WIP)

Usefulness heavily relies on users to implement proper tests

- We have many examples and some documentation
- When we started this: Added first tests
- Usually in the first PR have to remind ppl but after that self-reliant

Also in the nightlies: Documentation, website creation, etc



# Code quality and CI

Important part to keep a code base maintainable:

- Coding conventions for uniformity
- Defined when I started, based on LHCb / Google

Help people to follow these conventions

- Clang-format for formatting
- SAS for naming conventions (and all tests it brings)
  - Some contributions + supervision of summer student

Further static tests done with Flint++

- very light weight and fast

Made sure to follow up and fix warnings, etc.

All comes together in the webpage...

Code quality checks:

#### Static checks

- FCC Core Software (SAS cleaned) (SAS verbose) (flint++)
- Gaussino (cleaned) (verbose) (flint++)

#### Runtime performance

- FCC EDM example (igprof)

#### Information

The snapshots correspond to the most recent version of the code. **Expect that you may encounter problems**

FCCSW

FCCSW snapshot [github](#) | [doxygen](#)

**No central installation**, see Twiki pages / repository README.md on how to install and setup.

```
erty("gdml", m_gdmlFileName="GeantDetector.gdml");
[Error] ./Detector/DetComponents/src/GeoConstruction.h:29: Single - arg
[Warning] ./Detector/DetComponents/src/GeoSvc.cpp:64: Consider using 'make
[Error] ./Detector/DetComponents/src/MaterialScan.h:1: Missing include pToGdmlDumpSvc::initialize() {
[Warning] ./Detector/DetInterface/DetInterface/ITrackingGeoSvc.h:16: Class::initialize().isFailure(){
[Error] ./Detector/DetSegmentation/DetSegmentation/GridEta.h:22: Single
[Error] ./Detector/DetSegmentation/DetSegmentation/GridEta.h:24: Single"Unable to initialize Service()"<<endl;
[Error] ./Detector/DetSegmentation/DetSegmentation/GridEtaHandle.h:58: StatusCode::FAILURE;
[Error] ./Detector/DetSegmentation/DetSegmentation/GridEtaHandle.h:60:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridEtaHandle.h:63:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEta.h:20: Single
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEta.h:22: Single
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:5:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:6:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:6:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEta.h:21: Single
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEta.h:23: Single
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:
[Error] ./Detector/DetSegmentation/DetSegmentation/GridPhiEtaHandle.h:
[Warning] ./Detector/DetSensitive/DetSensitive/FlashCalorimeterSD.h:24:
[Warning] ./Generation/Generation/IHepMCFileReaderTool.h:16: Classes with
[Warning] ./Generation/Generation/IHepMCMergeTool.h:14: Classes with virt
[Warning] ./Generation/Generation/IHepMCProviderTool.h:16: Classes with v
[Warning] ./Generation/Generation/IParticleGunTool.h:16: Classes with virt
[Warning] ./Generation/Generation/IPileUpTool.h:19: Don't use static at global or namespace scopes in headers.
[Warning] ./Generation/Generation/IPileUpTool.h:21: Classes with virtual functions should not have a public no
```

# What I take away from here

---

Working on FCC software is nice because:

- A lot of freedom to experiment
- Quick turn around times (small code bases)

=> Very well suited to learn from your own mistakes

Concretely I learned a lot:

- About automation and continuous X
- Maintaining infrastructure that scales
- User support is time consuming and important ;)

# What's next?

---

DevOps at

**TARSIER**  
STUDIOS

Automation + infrastructure  
for development



# Thank you

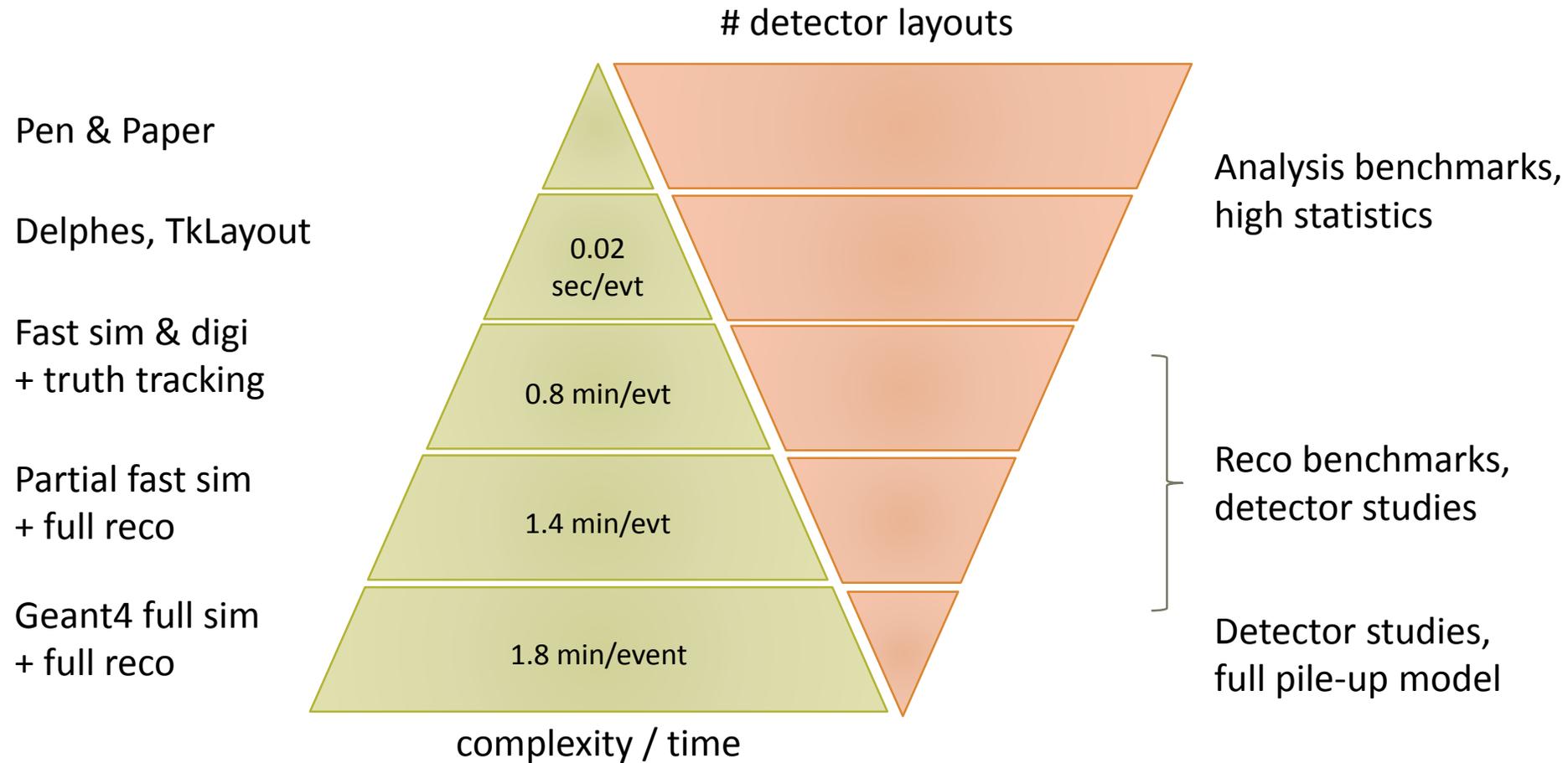
---

It has been a very interesting time at SFT. One of the nicest groups @ CERN.

**Special thanks to Anna, Valentin and Benedikt: It was a pleasure!**

BACKUP

# The right tools for the right job



# Status: Detector Description

Underlying framework: DD4hep

- Collaborative effort with Linear colliders and LHCb

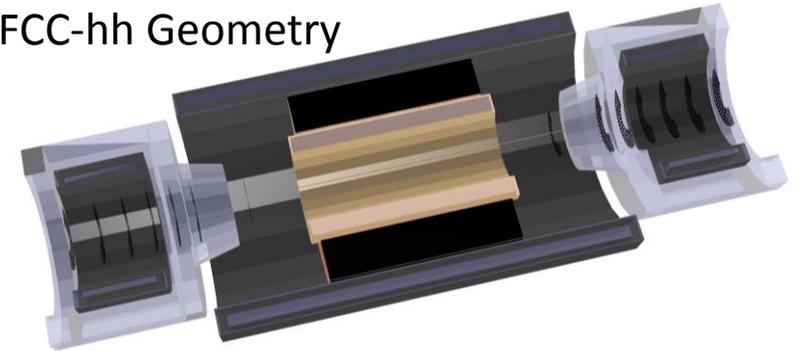
FCC-hh: Baseline concept exists and is now fixed

- All sub-detectors being mainly developed in FCCSW
- So far concentrating on non-forward detectors
- First simulation + reconstruction results shown at FCC week

FCC-ee: Starting with geometry based on CLIC concept

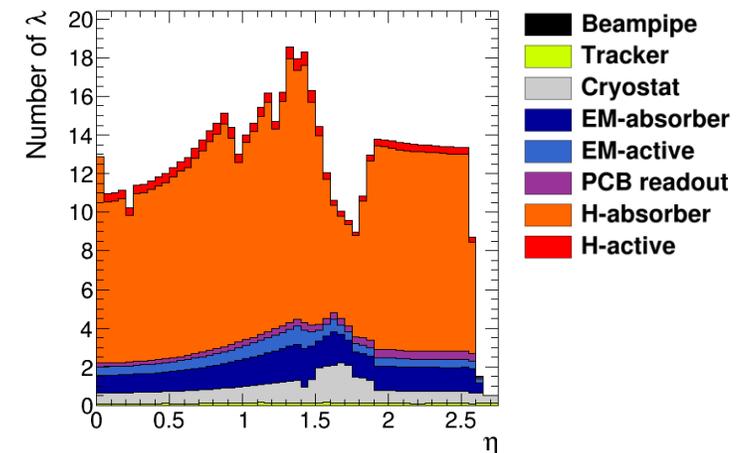
- First working implementation integrated in FCCSW
- Material scans + first simulation results

FCC-hh Geometry



Material scan

(barrel only)



# Integrated Fast Simulation

---

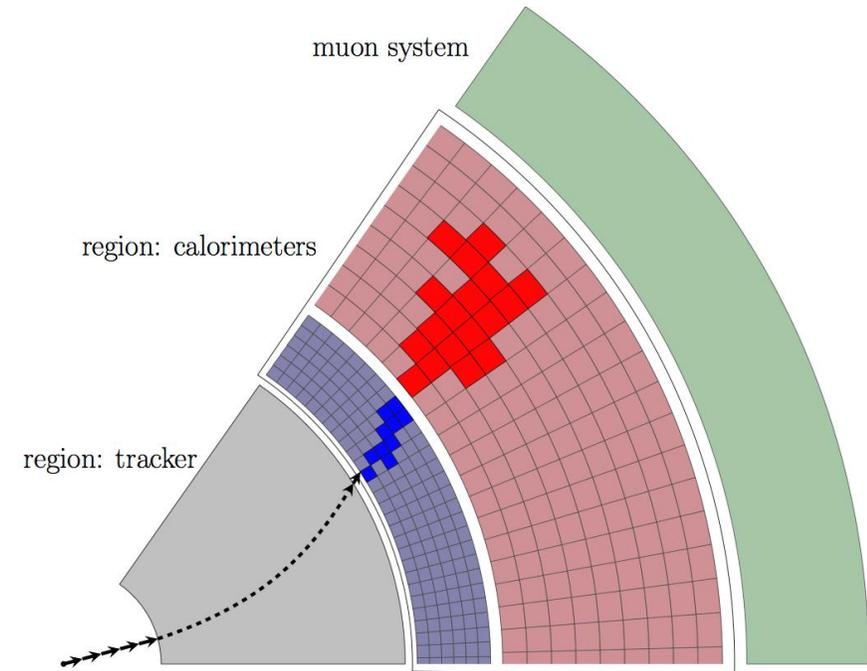
Integrated fast and full simulation:

- Use pre-defined hooks within Geant4
- Mix fast and full simulation in the same event
- Allow to switch based on particle properties, detector region, ...

Existing implementations:

- Parametric electromagnetic showers
- Particle momentum smearing

Plans to extend these models



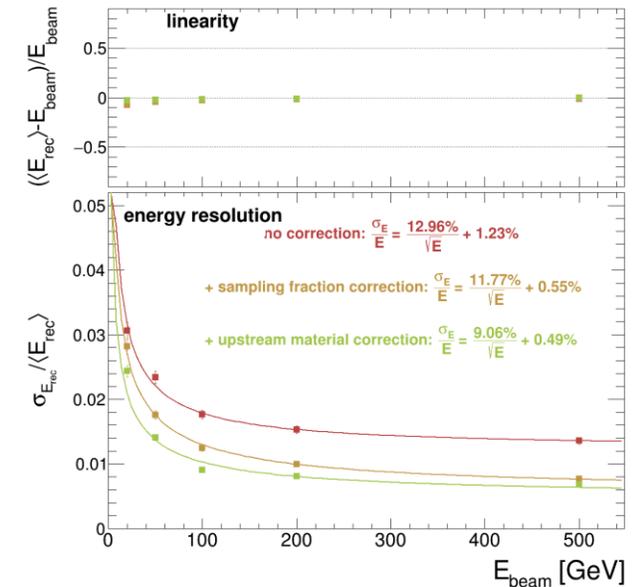
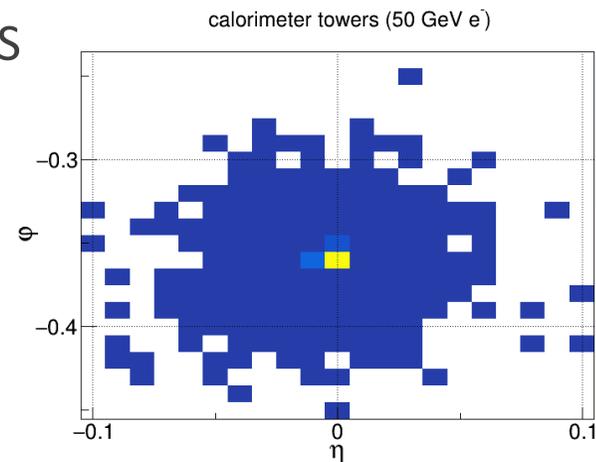
# Reconstruction status (for completeness, no hands-on contribution)

## Tracking

- Extraction of the ATLAS tracking code into standalone package (ACTS)
- Geometry converted to optimized geometry via ACTS plugin using DD4hep
- Valentin Völkl & Julia Hrdinka

## Calorimetry

- Developed within FCCSW, inspired by ATLAS
- Started with dedicated reco for e/gamma
- First results on combined calorimetry
- Anna Zaborowska, Jana Faltova, Coralie Neubüser & Tony Price



# Analysis Front-End

(for completeness, no hands-on contribution)

Python based package: HEPPY

- Originally developed in the CERN-CMS group
- Highly configurable, easy to set up
- Includes PAPAS simulation

Long term: Python performance issues?

- Combine strengths of C++ and Python
- Python to allow testing ideas and prototype
- Invest to port performance critical code to C++
  - Use ported Functionality from Python

Contacts: Alice Robson and Colin Bernet

