Software developments for FCC physics and experiments

B. Hegner, CERN

for the FCC Experiment Software Team

FCC IEEE Workshop 2015

Washington

Driving Considerations

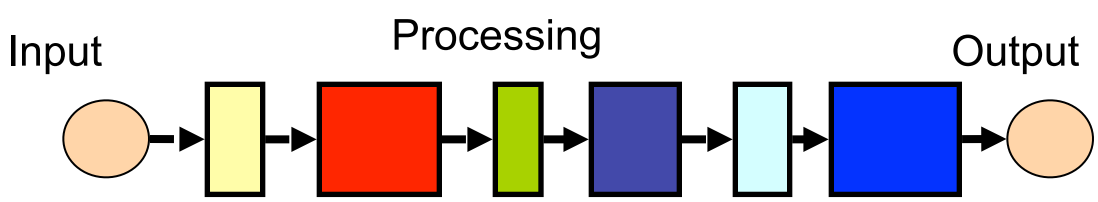
- Provide robust software to allow physics studies for CDR in 2018
- Support all **FCC-ee, -eh, and -hh** communities at the same time
  - Requires flexibility for Geometry and Simulation
- Start pragmatically
- As studies progress move to more sophisticated solutions
  - Allow components to be replaced later on
- FCC software effort relies on effort of other people
  - There is a give and take
  - Aim for, but don't blindly force, synergy with other communities

Early Decisions

- Adapt existing solutions from LHC
  - Gaudi as underlying framework
  - ROOT for I/O
  - Geant4 for simulation
  - C++ and Python for user analysis
- Adapt software developments from ILC/CLIC
  - DD4Hep for detector description
- Invest in **better fast vs. full sim integration**
  - Geant4 fastsim, Atlfast (ATLAS)
- Invest in **proper data model**
  - The LHC experiments' ones are over-engineered
  - The ILC/CLIC implementation (LCIO) isn't state of the art

Gaudi Event Processing Framework

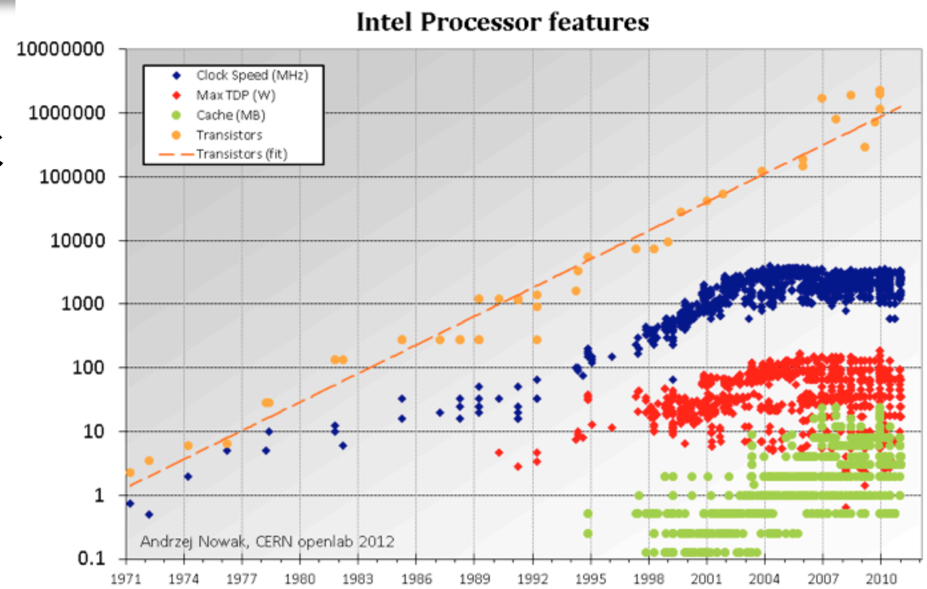
- Gaudi is an event-independent data processing framework
  - Used by LHCb, ATLAS, and a few smaller experiments
- Based on the concept of a software bus
- Work is split up in interdependent "algorithms"



- Parallelization effort with "GaudiHive" to take advantage of ever increasing hardware parallelization

The Power Wall - Importance of Parallelism

- In the past speed increase happened automatically - just wait for your next PC
- That is over now!
- New CPU improvements go into parallelization
- Clock speed will not increase because of Power consumption:



$$Power \propto Frequency^3$$

- **Need to adapt our software to this parallel environment right from the start**
  – Otherwise we waste the computing resources we urgently need

FCC Simulation

- FCC Software needs to support the studies of multiple detectors
- At different stages different level of detail required
  - Smearing vs. fast sim vs. full sim
- FCC choices are
  - Delphes (*)
  - Fast simulation
  - Full simulation with Geant4
- Should all be accessible from within the same framework

(*) http://delphes.hepforge.org

Detector Description

Detector Description in LHC experiments is a not-well organized environment

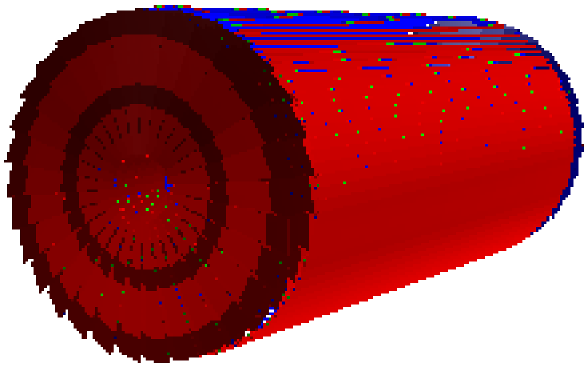
- Detectors modeled long ago and expertise largely gone
- Struggling themselves for the upgrade
- Heterogeneous setups even within experiments

ILC/CLIC efforts triggered the project DD4hep (*)

- Covering simulation, display, alignment in a consistent way

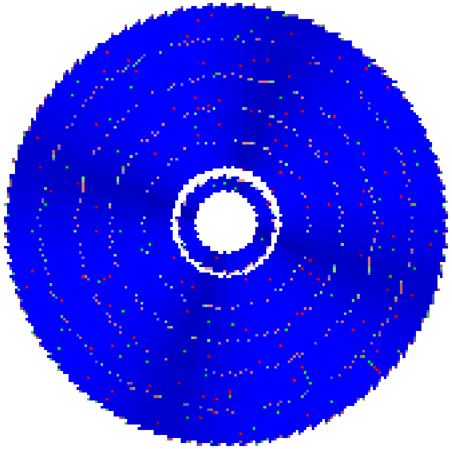
FCC joined these efforts of DD4hep

- Good support by developers!
- Working on first test-detector



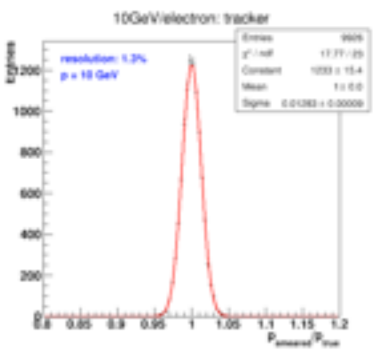
Fast Simulation

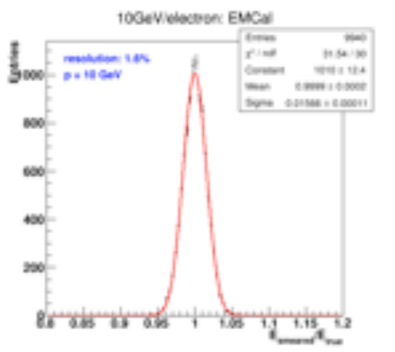
- Goal is to have a **combined fast and full simulation**
  - Decide at the config level where to do what
- (Semi-) automatic extraction of fast simulation parameters from full simulation
  - To be able to do fast-sim for any detector design
- Though not re-inventing the wheel, we are heavily re-designing it

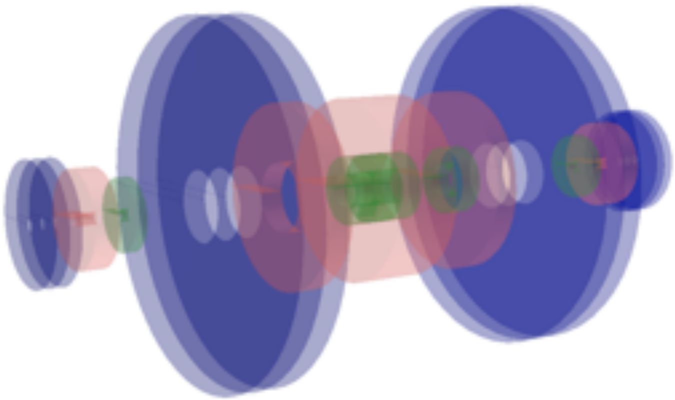


Simulation Status

- First development phase was focussed on producing a demonstrator
  - Using expertise from ATLAS and Geant4 developers
  - Chosen approach worked out nicely
  - Results now being integrated into Geant4 and Gaudi







Data Model I

The FCC requirements for a good data model are not special at all:

- Simplicity
- Flexibility
- Completeness
- Usable in C++ and Python

Data Models of LHC experiments are proven to work

- Fairly complex, and very detector specific beasts

The ILC community has a simple, but complete data model (LCIO)

- Needs adaption to allow direct ROOT access outside FWK
- Parallelism not part of the design
- Developers interested in extension and one should take advantage of it

The proper data model is **essential** **for allowing good results**

Thus it is worth investing here with a new project!

Data Model II

- ROOT as first choice for I/O
- No deep object hierarchies
  - Wherever possible concrete types
- Simple memory layout
  - Employ simple structs instead of fat objects
  - Helps with parallelization
- Allow access from Python and C++
  - Only loose coupling with event processing framework
- Quick turnaround for improvements
  - Employ code generation
- Wrote a demonstrator data model
  - Used throughout all developments now

- Needs morework still

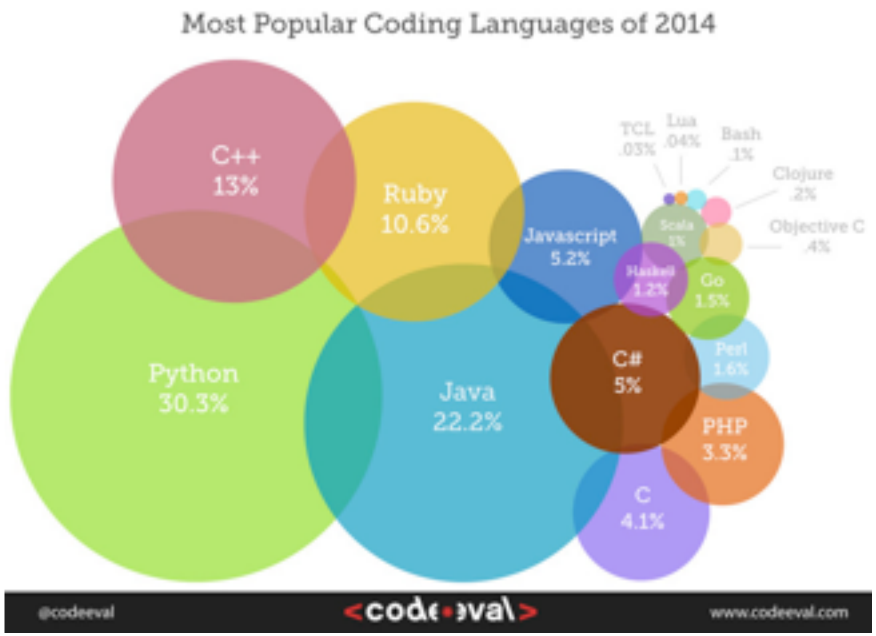
```
ALBERS Data Model
=================

Used
  description.yaml
to create
  5 classes

Read instructions in
the HOWTO.TXT to run
your first example!
```

Hommage to the Square - Josef Albers

Analysis

- Analysis should be easy and powerful
- Lesson from LHC experiments and ILC/CLIC
  - If data model too complex, physicists stop using common software and create their own mini-frameworks
- Need to allow multiple paradigms to do analysis
  - C++ and Python
- Physicists will join from different experiments and will bring along their existing code

Analysis in Python



- Very large user base
- Super easy to learn
- Light & short code
- Good performance
  - usually wraps C or C++ modules
- « Batteries included »
  - massive and easy-to-use standard library
- Dynamic typing
  - good for multichannel analyses
  - code highly reusable
- Dynamic object modification
  - Can attach new attributes (or methods) to an existing object
- Productivity x 5-10 w/r C++
- A lot of fun!

- Supporting this with the **heppy** package originating from CMS

Where are we now?

- Common FCC experiment software project started
- First phase of pick & chose is finished
- Base software environment in place
- Integrated fast/full sim design validated
- Data Model demonstrator finished
- C++ and Python based analysis environment provided
- Soon to do start more efforts on Reconstruction
  - Common ILC/CLIC + FCC reconstruction workshop planned for April/May