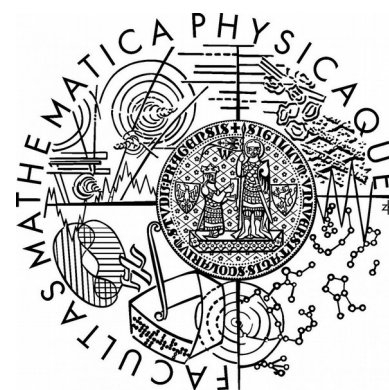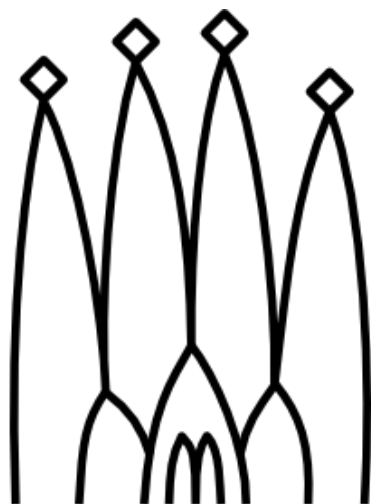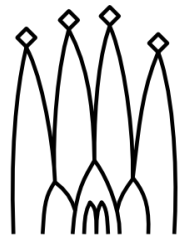# Gaudi and FCCSW

Jana Faltova

FCCSW workshop, 2/10/2019

Gaudi

- "The Gaudi project is a **open project for providing the necessary interfaces and services** for building HEP experiment frameworks in the domain of event data processing applications. The Gaudi framework is experiment independent."
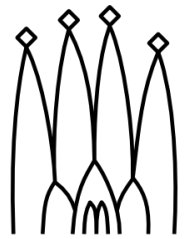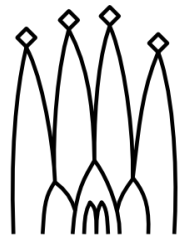
Gaudi

- "The Gaudi project is a **open project for providing the necessary interfaces and services** for building HEP experiment frameworks in the domain of event data processing applications. The Gaudi framework is experiment independent."

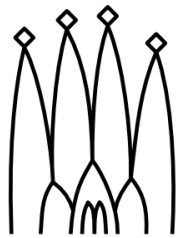- Used by ATLAS, LHCb, Daya Bay, GLAST (Fermi Gamma-ray Space Telescope)

Gaudi

- "The Gaudi project is a **open project for providing the necessary interfaces and services** for building HEP experiment frameworks in the domain of event data processing applications. The Gaudi framework is experiment independent."

- Used by ATLAS, LHCb, Daya Bay, GLAST (Fermi Gamma-ray Space Telescope)

… and

Gaudi

http://gaudi.web.cern.ch/gaudi/

THE GAUDI PROJECT

HOME | RELEASES | LATEST RELEASE | GITLAB PROJECT WEB PORTAL

Welcome to the Gaudi framework project website. The Gaudi project is an open project for providing the necessary interfaces and services for building HEP experiment frameworks in the domain of event data processing applications. The Gaudi framework is experiment independent.

Documentation

Design and User Guide

- Scenarios and Requirements Document (version 1.0) (pdf) [last modified: 1998-11-20]
- Architecture Design Document (version 1.0) (pdf) [last modified: 1998-11-20]
- User Guide (pdf, 2up pdf, html, html as zip) [last modified: 2001-12-19 (version 9)]
- Tips and Tricks
- The Gaudi Testing Infrastructure on the Gaudi Twiki site
- Organization of the Gaudi SVN Repository.
- How-To Migrate to Gaudi v21.

Announcements

- **Latest release is v31r0** (2019-02-11). You can find a summary of the latest modifications in the release notes and detailed description of the classes in the doxygen pages.

To have access to all the available versions, please follow this link.

You can also peek at the CHANGELOG in

Links

Framework Users

- LHCb Computing
- ATLAS Athena framework
- HARP Gaudino framework
- Fermi (previously GLAST)
- MINERvA
- BESIII BOSS framework
- LBNE (Long Baseline Neutrino Detector, WCD group), see also GARPI project

Related LCG Projects

ROOT	JIRA	Data analysis framework
CORAL and COOL	JIRA	COmmon Relational Abstraction Layer and Conditions database project
RELAX	JIRA	RELAX provides Reflex C++ dictionaries for common used libraries and C++ classes installed in the LCG external libraries area

Gaudi architecture

- Goals of Gaudi framework
    - **Easy to use**
    - Insulate users from irrelevant details such as what software libraries we use for data I/O, or for graphics

Gaudi architecture

- Goals of Gaudi framework
  - **Easy to use**
  - Insulate users from irrelevant details such as what software libraries we use for data I/O, or for graphics
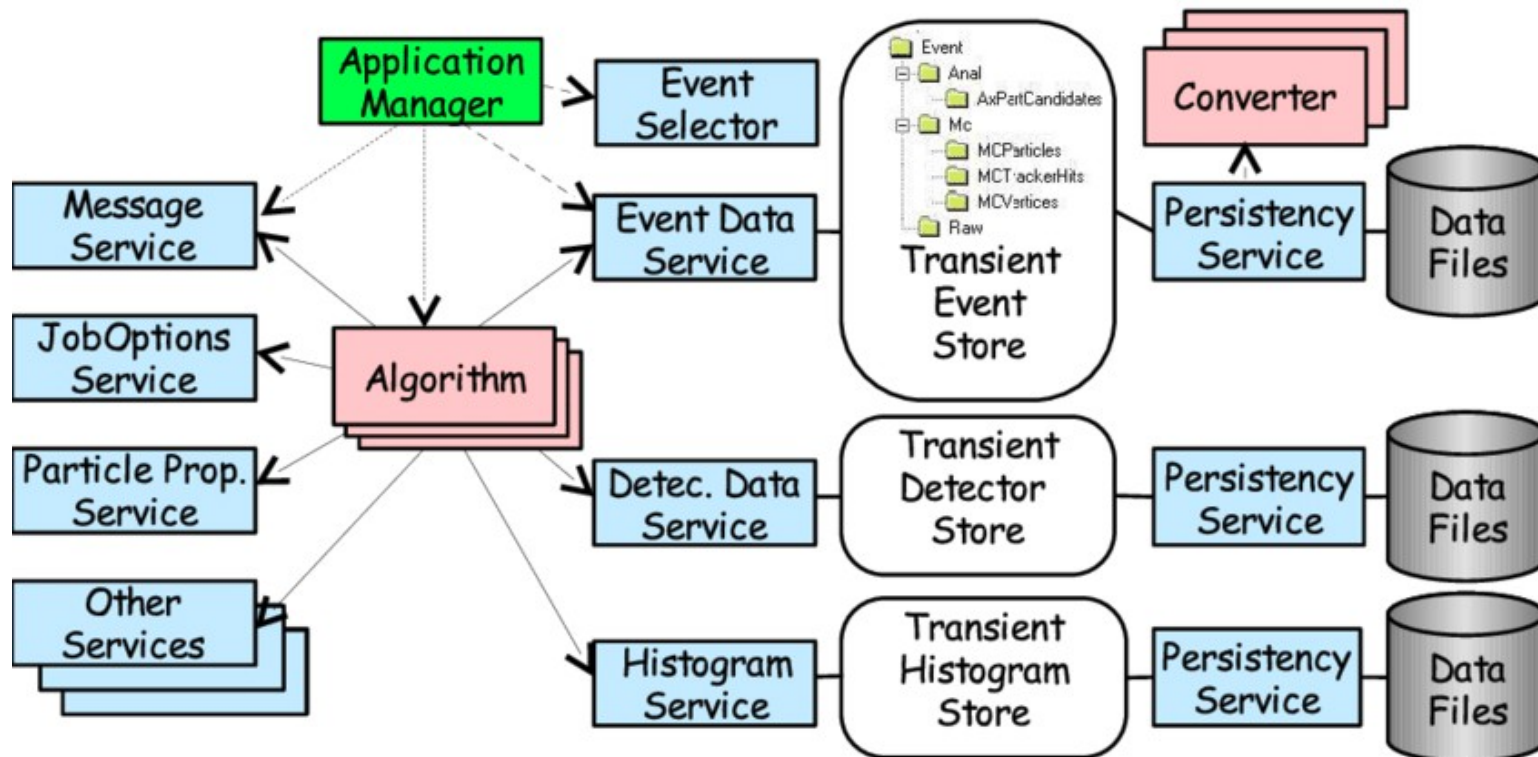
Gaudi Project

- **C++**
  - Note: **Configuration in python**

Gaudi Project

- **C++**
  - – Note: **Configuration in python**

- If you want to write your own FCCSW package, use
  - – **Algorithms**
  - – **Tools**
  - – **Services**

  … explained on the next slides

Algorithm

- Goal: take input data, manipulate it and produce new output data

Algorithm

- Goal: take input data, manipulate it and produce new output data

- Main features
  - **Configurable**
  - Initialize() – called once at the start of the job
  - Execute() – called once per event
  - Finalize() – called once at the end of the job
  - **Uses Services and Tools**

Tools

- **Smaller pieces of code doing one particular thing** (e.g. energy calibration, emulation of electronic noise)

Tools

- **Smaller pieces of code doing one particular thing** (e.g. energy calibration, emulation of electronic noise)

- Main features
  - Can be **called many times per event**
  - **Configurable**
  - Private or public
    - Private: owned and accessible only by the component creating it
    - Public: owned by the framework and accessible globally

Interfaces

- An abstract interface is a class where all the methods are pure virtual

Interfaces

- An abstract interface is a class where all the methods are pure virtual

- When to use it
    - Different tools can implement the same functionality (e.g. vertex creation)
    - **If tools have common properties or methods → implement** a base class (= **interface**)
    - The choice of which tool to use can be done via the job options at run time

Services

- Created by framework to **provide global functionality**
  - Prepared for you to use, you don't have to care!

- e.g. Job Options Svc, Message Svc, Event Data Svc, Histogram Svc, Ntuple Svc, Detector Data Svc

Services

- Created by framework to **provide global functionality**
  - Prepared for you to use, you don't have to care!
- e.g. Job Options Svc, Message Svc, Event Data Svc, Histogram Svc, Ntuple Svc, Detector Data Svc
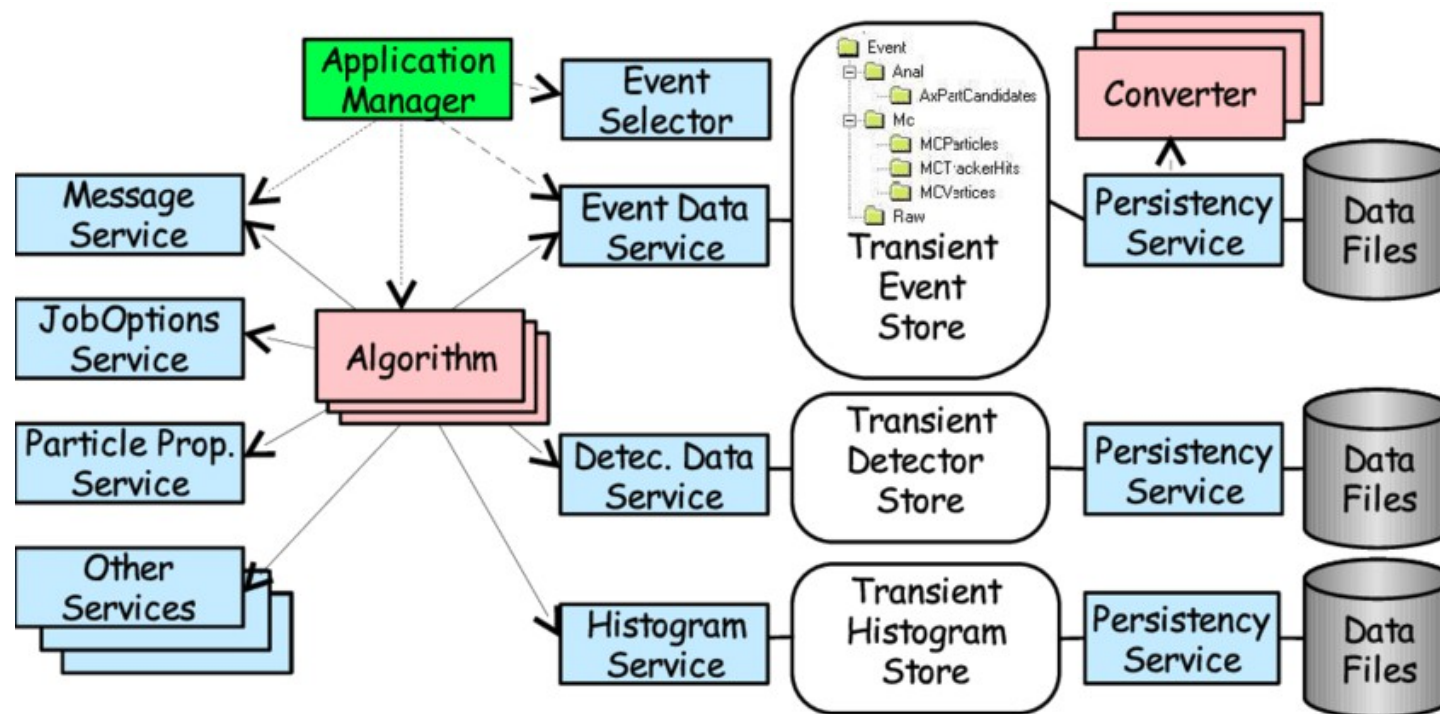
How to run the FCCSW code?

- Use python scripts where you
  - **configure** tools and algorithms you wish to use
  - process algorithms in a required order

```
./run fccrun.py Examples/options/example_options.py
```

you can also specify some options like number of events and output file

```
./run fccrun.py Examples/options/example_options.py --outputfile=my_own_output.root --nevents=1000
```

Code structure in FCCSW

- FCCSW folders
  - Detector, Generation, Sim, Reconstruction, FWCore

Code structure in FCCSW

- FCCSW folders
  - Detector, Generation, Sim, Reconstruction, FWCore

- Basic folder structure of your reconstruction package (Reconstruction/RecYourPackage)
  - CMakeLists.txt
  - **src/components**
    - code (*.h, *.cpp)
  - **options**
    - jobOption files (*.py)
  - **doc**
    - **documentation!**

Code structure in FCCSW

- FCCSW folders
  - Detector, Generation, Sim, Reconstruction, FWCore

- Basic folder structure of your reconstruction package (Reconstruction/RecYourPackage)
  - CMakeLists.txt
  - **src/components**
    - code (*.h, *.cpp)
  - **options**
    - jobOption files (*.py)
  - **doc**
    - **documentation!**

More details in
Code style guidelines

FCCSW code guideline

- Code style guidelines

- Variable names
  - **Meaningful names**
  - Member variables starts with *m_*, constants with *k*

- Function names
  - Functions (both member or non-member) **start with lower case** and capital letter for each new word

- Rules for class & functions declaration, header files

- **Automatic checking script**

FCCSW Guidelines for C++ Code Style

Clashing coding styles are avoided by broadly following the LHCb / Gaudi style guides.

One particular exception is that we extended the 80 characters per line requirement to 120 characters per line.

Goal

Give guidelines on naming conventions and how to structure code.

General tips on how to write good code can be found here.

Contents

Example: Reconstruction

FCCSW/Reconstruction

Example: Reconstruction

FCCSW/Reconstruction

Example: Reconstruction

FCCSW/Reconstruction

Example: Reconstruction

FCCSW/Reconstruction

doc

FCCSW/Reconstruction/doc/RecCalorimeter.md

RecCalorimeter package

Information about calorimeter reconstruction software within FCCSW. The software is being tested using ECAL, but should be general enough to be used for other calorimeters. Let us know if you have any problems or questions (Jana Faltova, Anna Zaborowska).

Detector description

ECAL calorimeter description in `Detector/DetFCChhECalSimple`:

- Tube geometry with alternating layers of active and passive material
- Using phi-eta segmentation with offset (Note: negative eta/phi identifiers not allowed!)
- Calorimeter cells defined by a layer in R + phi-eta segment

Digitisation

Digitisation creates cells out of simulated energy deposits. From the EDM point of view, both input and output of the digitisation uses `fcc::CaloHit`. The input (simulated deposits) contains raw information about the energy deposited in the cells of the sensitive volumes. The output (cells) may contain energy (corrected for the losses in the passive layers) and the noise. The calibration and noise tools could be switched on/off by setting the appropriate flags in your script. The cells may correspond to the active volumes or to the segmentation cells. In particular, different segmentation may be used than the original cells of the sensitive volumes used in the simulation.

RecInterfaces

FCCSW/Reconstruction/RecInterfaces/RecInterfaces/
INoiseCaloCellsTool.h

```
24 lines (18 sloc) | 603 Bytes                                    Raw   Blame   History

1    #ifndef RECINTERFACE_INOISECALOCELLSTOOL_H
2    #define RECINTERFACE_INOISECALOCELLSTOOL_H
3
4    // from Gaudi
5    #include "GaudiKernel/IAlgTool.h"
6
7    /** @class INoiseCaloCellsTool
8     *
9     *  Abstract interface to calorimeter noise tool
10    *
11    *  @author Jana Faltova
12    *  @date   2016-09
13    */
14
15   class INoiseCaloCellsTool : virtual public IAlgTool {
16   public:
17     DeclareInterfaceID(INoiseCaloCellsTool, 1, 0);
18
19     virtual void addRandomCellNoise(std::unordered_map<uint64_t, double>& aCells) = 0;
20     virtual void filterCellNoise(std::unordered_map<uint64_t, double>& aCells) = 0;
21   };
22
23   #endif /* RECINTERFACE_INOISECALOCELLSTOOL_H */
```

RecCalorimeter

FCCSW/Reconstruction/RecCalorimeter

RecCalorimeter

FCCSW/Reconstruction/RecCalorimeter/src/components

– algorithms, tools

RecCalorimeter

FCCSW/Reconstruction/RecCalorimeter/src/components/
NoiseCaloCellsFromFileTool.h

```cpp
32  class NoiseCaloCellsFromFileTool : public GaudiTool, virtual public INoiseCaloCellsTool {
33  public:
34    NoiseCaloCellsFromFileTool(const std::string& type, const std::string& name, const IInterface* parent);
35    virtual ~NoiseCaloCellsFromFileTool() = default;
36    virtual StatusCode initialize() final;
37    virtual StatusCode finalize() final;
38
39    /** @brief Create random CaloHits (gaussian distribution) for the vector of cells (aCells).
40     * Vector of cells must contain all cells in the calorimeter with their cellIDs.
41     */
42    virtual void addRandomCellNoise(std::unordered_map<uint64_t, double>& aCells) final;
43    /** @brief Remove cells with energy bellow threshold*sigma from the vector of cells
44     */
45    virtual void filterCellNoise(std::unordered_map<uint64_t, double>& aCells) final;
46
47    /// Open file and read noise histograms in the memory
48    StatusCode initNoiseFromFile();
49    /// Find the appropriate noise constant from the histogram
50    double getNoiseConstantPerCell(int64_t aCellID);
51
52  private:
53    /// Add pileup contribution to the electronics noise? (only if read from file)
54    Gaudi::Property<bool> m_addPileup{this, "addPileup", true,
55                                "Add pileup contribution to the electronics noise? (only if read from file)"};
56    /// Name of the file with noise constants
57    Gaudi::Property<std::string> m_noiseFileName{this, "noiseFileName", "", "Name of the file with noise constants"};
58    /// Name of the detector readout
59    Gaudi::Property<std::string> m_readoutName{this, "readoutName", "ECalHitsPhiEta", "Name of the detector readout"};
60    /// Name of active layers for sampling calorimeter
61    Gaudi::Property<std::string> m_activeFieldName{this, "activeFieldName", "active_layer",
62                                "Name of active layers for sampling calorimeter"};
```

RecCalorimeter

Reconstruction/RecCalorimeter/src/components/
CreateCaloCells.h

```
41    class CreateCaloCells : public GaudiAlgorithm {
42
43    public:
44      CreateCaloCells(const std::string& name, ISvcLocator* svcLoc);
45
46      StatusCode initialize();
47
48      StatusCode execute();
49
50      StatusCode finalize();
51
52    private:
53      /// Handle for tool to calibrate Geant4 energy to EM scale tool
54      ToolHandle<ICalibrateCaloHitsTool> m_calibTool{"CalibrateCaloHitsTool", this};
55      /// Handle for the calorimeter cells noise tool
56      ToolHandle<INoiseCaloCellsTool> m_noiseTool{"NoiseCaloCellsFlatTool", this};
57      /// Handle for the geometry tool
58      ToolHandle<ICalorimeterTool> m_geoTool{"TubeLayerPhiEtaCaloTool", this};
59
60      /// Calibrate to EM scale?
61      Gaudi::Property<bool> m_doCellCalibration{this, "doCellCalibration", true, "Calibrate to EM scale?"};
62      /// Add noise to cells?
63      Gaudi::Property<bool> m_addCellNoise{this, "addCellNoise", true, "Add noise to cells?"};
64      /// Save only cells with energy above threshold?
65      Gaudi::Property<bool> m_filterCellNoise{this, "filterCellNoise", false,
66                                       "Save only cells with energy above threshold?"};
67      /// Handle for calo hits (input collection)
68      DataHandle<fcc::CaloHitCollection> m_hits{"hits", Gaudi::DataHandle::Reader, this};
69      /// Handle for calo cells (output collection)
```

RecCalorimeter

FCCSW/Reconstruction/RecCalorimeter/tests/options

- jobOption files which you can run
- used in SW tests if defined in CMakeLists.txt
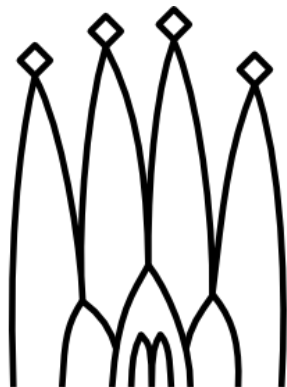
Conclusions

Don't be afraid

- – Gaudi is a nice C++ thing
- – You don't have to be a super-experienced code-writer to start with FCCSW
- – Read documentation
- – Look into examples in the FCCSW
- – **If you are in troubles, just ask**

Backup