OVERVIEW OF MONTE CARLO TOOLS

LAURE MASSACRIER,
SUBATECH NANTES

Subatech

AFTER @ LHC

Probing the Strong Interaction at A Fixed Target ExpeRiment with the LHC beams

12-17 janvier 2014

Ecole de Physique des Houches

Europe/Paris timezone

OUTLINE

- ❑ Introduction
- ❑ Overview of LHC experiment softwares:
  - The ALICE software
  - The CMS software
  - The ATLAS software
  - The LHCb software
- ❑ Monte Carlo Generators for simulation
- ❑ Simulations for AFTER with Pythia8

DISCLAIMERS

- ❑ Software presented from a simple user/analyser point of view
- ❑ Information in the talk based on public information availables on the Collaboration websites, wikipages, PhD thesis and might not always be up-to-date
- ❑ It is hard to present ~200 pages of software documentations in a couple of slides (especially software you never used)

GOALS OF THIS PRESENTATION

- ❑ Try to present the philosophy of the softwares used by LHC experiments, and highlight some main tools of interest
- ❑ Present software problematics on a simple point of view
- ❑ Open the discussion on a software for AFTER

WHICH FUNCTIONALITIES FOR THE AFTER SOFTWARE ?

- ❑ Programming language /supported platforms / compilers?
- ❑ Which tools do you need for simulations (which MC generators?, which transport code?)
- ❑ How do you want to implement the geometry of your detectors (do you need something configurable, modulable?)
- ❑ How many people will be ready to write the core software / maintain it?
- ❑ Do you want to re-use some existing software?
- ❑ What is the expected data flow? Computing resources needed? Do you need a grid environment?

ABOUT WRITING SOFTWARE IN A (LARGE) COLLABORATION ENVIRONNEMENT (NON EXHAUSTIVE)

- ❑ Have a set of coding standards
- ❑ You need a Version Control System
- ❑ Maintain a documentation for users / newcomers (not a so simple task)
- ❑ You need tools in addition to the core of the software such as debuggers, profiling tools…
- ❑ Think to common analysis tools (standardize the analysis as much as possible)

OVERVIEW OF LHC EXPERIMENT SOFTWARES

THE ALICE SOFTWARE (ALIROOT)

Sources:

https://aliceinfo.cern.ch/secure/Offline/sites/aliweb.cern.ch.Offline/files/uploads/OfflineBible.pdf

https://aliceinfo.cern.ch/secure/Offline/sites/aliweb.cern.ch.Offline/files/uploads/OfflineTutorial/ALICE-Tutorial-Part1.pdf

my own experience with ALIROOT

- AliRoot is the ALICE Offline framework
- AliRoot uses the **ROOT** system as a foundation on which the framework for simulations, reconstruction and analysis in built.
- Framework based on Object Oriented programming, essentially **C++ language** (except for large existing libraries such as Pythia6, HIJING, and some remaining legacy code)

THE ALICE SOFTWARE (ALIROOT)

Basic principles that have guided the design of ALIROOT:

- ❑ **Modularity:** allows replacement of part of the sytem with minimal or no impact on the rest (ex: a change of event generator do not affect the user code, code contribution from different detectors are independent)
- ❑ **Reusability:** importance of backward compatibility

ALIROOT is developped inside a **version control system**:

- ❑ Enable a group of people to work together on a set of files
- ❑ It records the history of the files
- ❑ Allows backtracking and file versioning

Concurrent Version System (CVS) → Subversion (SVN) → Git

ROOT is now using Git

THE ALICE SOFTWARE (ALIROOT)

❑ Main features of **SVN**:

- All developers work on their own copy of the project (in one of their directories)
- They often have to synchronize with a global repository both to update with modifications from other people and to commit their own changes
- In case of conflicts, it is the developer's responsibility to resolve the situation (SVN perform a mechanical merge)

❑ What's new with **GIT**:

- Cheap local branching, staging areas, multiple workflow

→ Git allows to have multiple branches that can be entirely independent of each other (you can have a branch for testing, a branch for day to day work…)

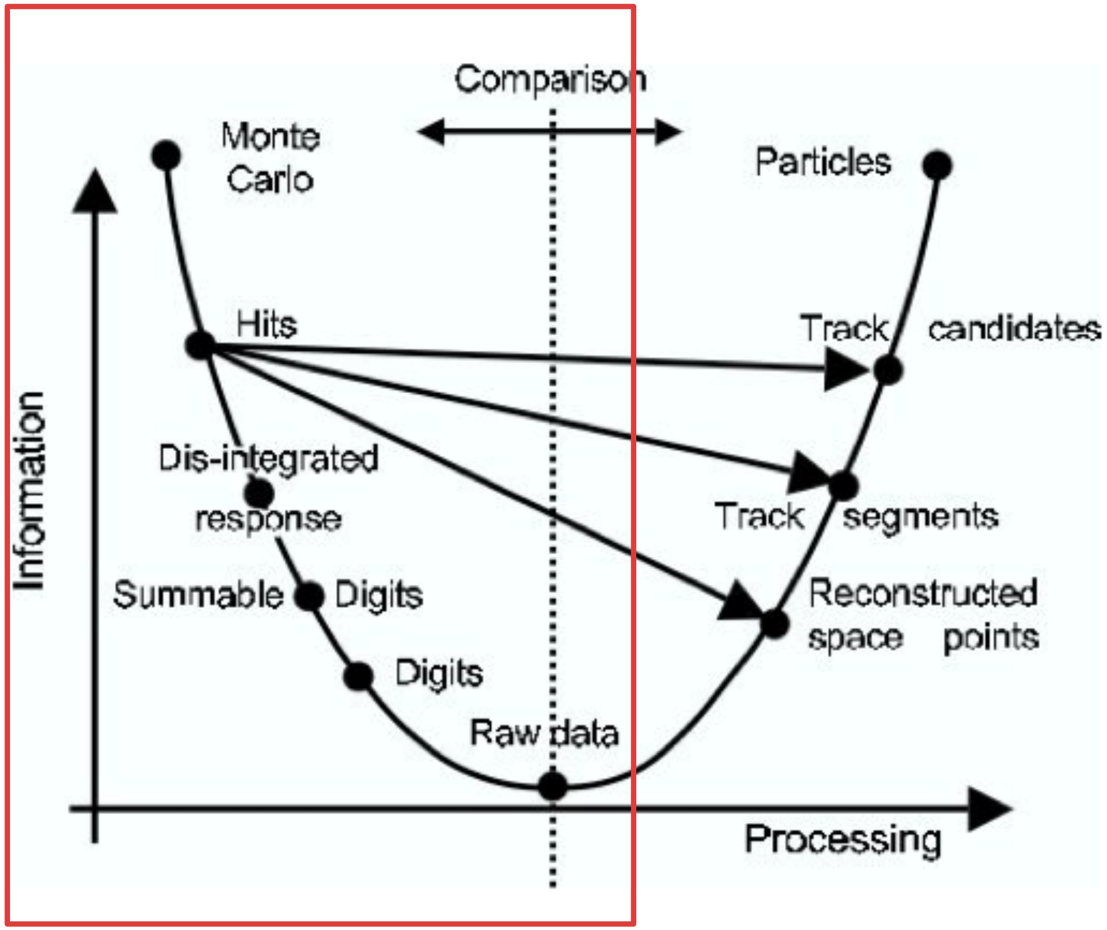
THE ALICE SOFTWARE (ALIROOT)

ALIROOT framework allows several options for simulations:

- ❑ The simulation framework provides an **interface to external generators**, like HIJING, DPMJET…
- ❑ A parametrized underlying event where the **multiplicity** is an **input parameter** can be used
- ❑ Rare signals can be generated using the interface to external generator like Pythia, or **simple parametrization of transverse momentum and rapidity spectra** can be used as input for simulations
- ❑ It provides a tool to assemble events from different signal generators (event **cocktails**)
- ❑ **Afterburners** are used to introduce particle correlations in a controlled way. Afterburners change the momenta of the particles produced by another generator, and thus modify the multi-particle momentum distributions

THE ALICE SOFTWARE (ALIROOT)

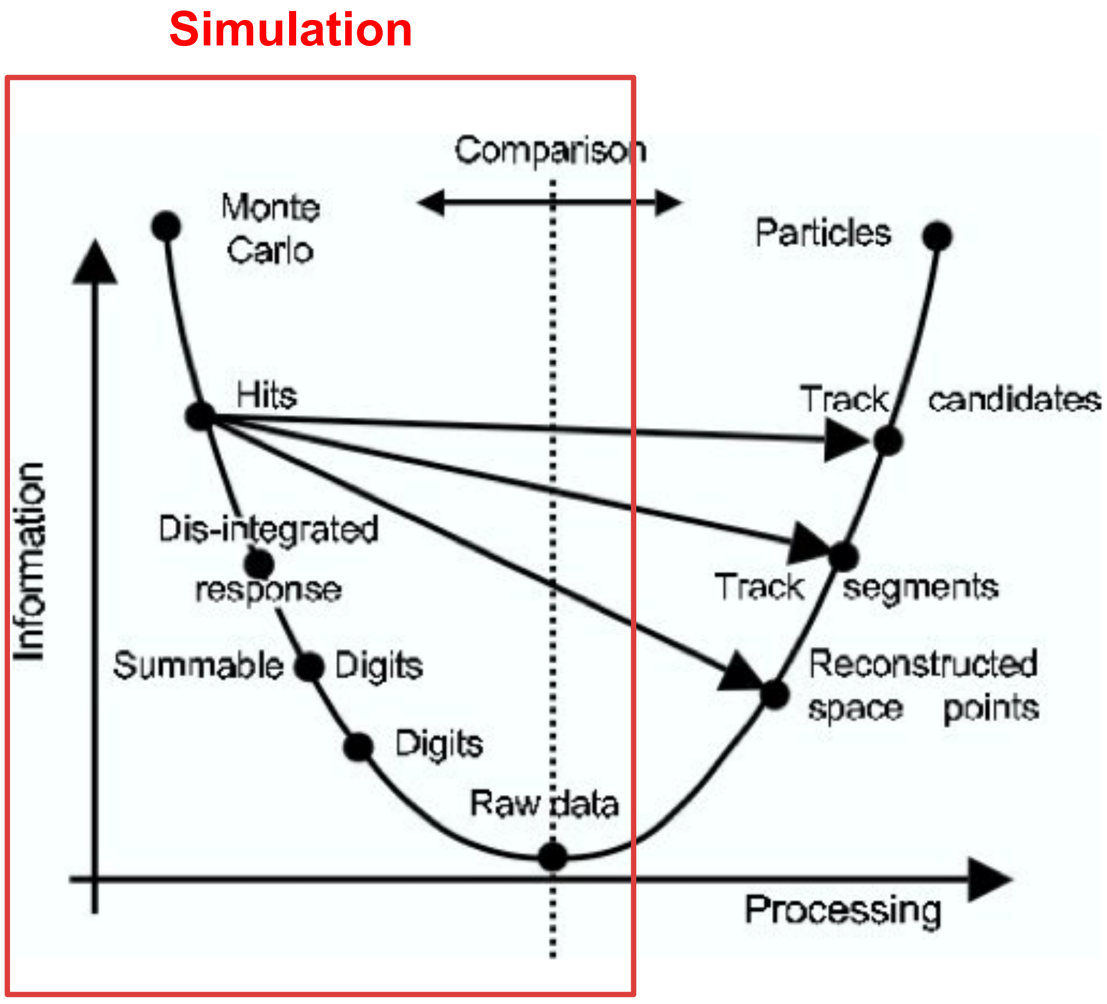
Simulation



Data processing framework

- ❑ Primary interactions simulated with event generators Resulting kinematic tree (mother, daughter particles, production vertex..) is used in transport package.
- ❑ Tranport package transports the particles through the various detectors and produces **HITS** (energy deposition at a given point) HITS keep the label information from the MC particle which generated them.
- ❑ Detector response is then taken into account. **HITS** transformed into **DIGITS**.

THE ALICE SOFTWARE (ALIROOT)



Data processing framework

- ❑ Two types of DIGITS (=digitized signal obtained by a sensitive pad of a detector (ADC counts)):

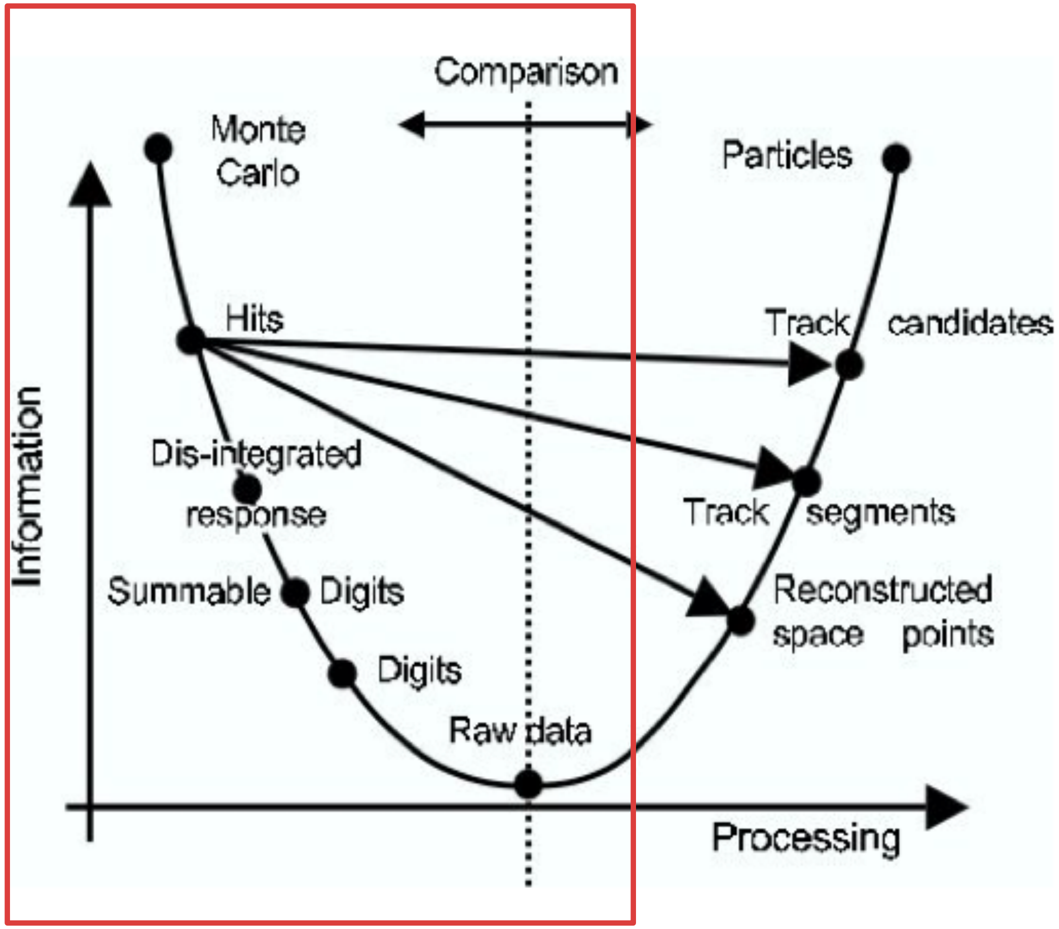
- Summable DIGITS (low thresholds + result additive) Usefull for « event merging » : signal event embedded in a signal-free underlying event

- **DIGITS**: using of real thresholds, similar to what one would get in real data taking. Noise simulations activated when DIGITS are produced

- ❑ DIGITS are stored in ROOT classes

THE ALICE SOFTWARE (ALIROOT)

Simulation



Data processing framework

- ❑ **Raw data** can be saved either in binary format (.raw) or ROOT format

- ❑ Different conversion chains exist:

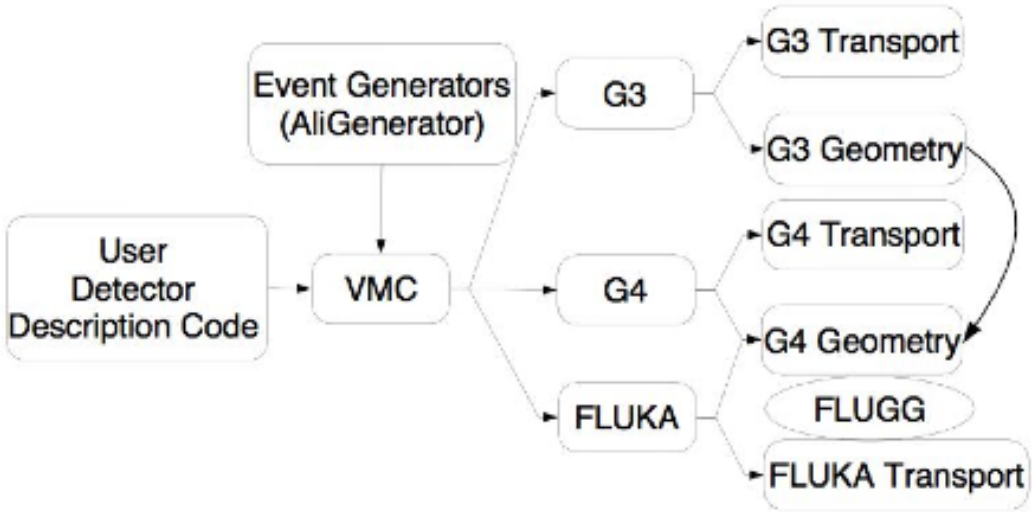
HITS → summable DIGITS → DIGITS
HITS → DIGITS
DIGITS → raw data (optionnal) : to estimate the expected data size , evaluate the HLT algorithm

After creation of the DIGITS the reconstruction and analysis chain can be activated.
Reconstruction takes as input digits or raw data, real or simulated

THE ALICE SOFTWARE (ALIROOT)

The Virtual Monte Carlo tool:

- Provides an interface to particle transport codes (**Geant3** (default one), **Geant4** and **Fluka** (not free of use))
- Provides an interface to construct the geometry of detectors

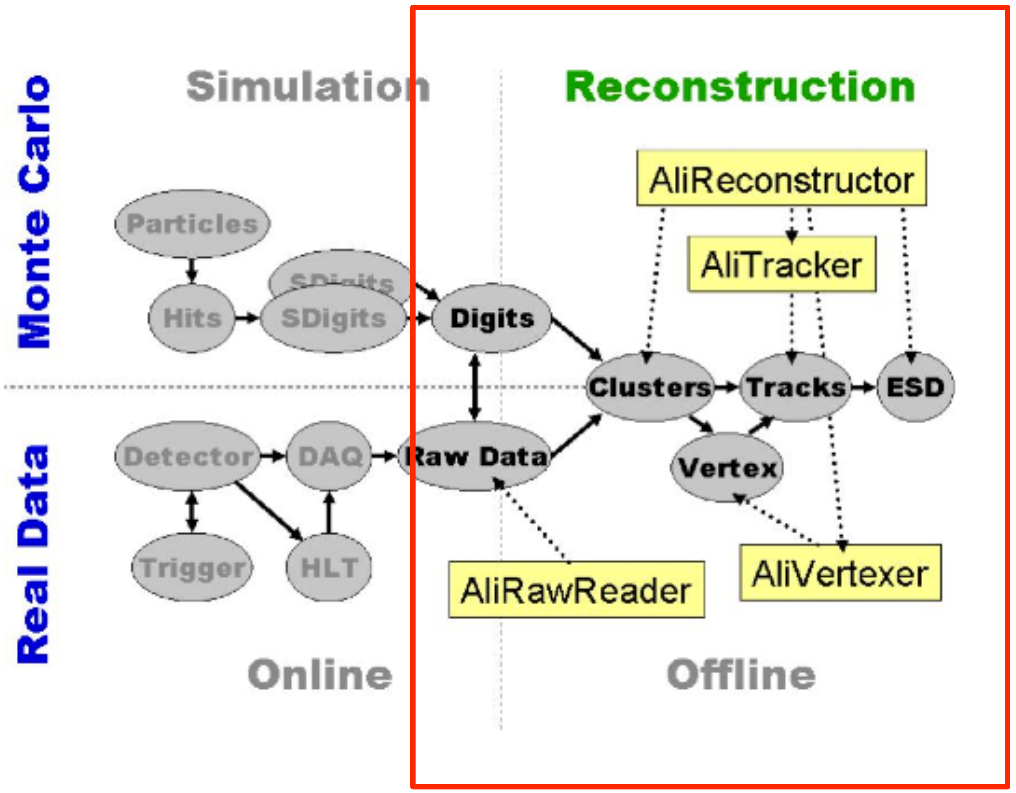


Thanks to virtual Monte Carlo, all FORTRAN user code developed for GEANT3 is converted into C++

Figure 3. Virtual Monte Carlo

THE ALICE SOFTWARE (ALIROOT)

The reconstruction framework



- ❑ Input of the reconstruction framework are **DIGITS** in ROOT tree format, or raw data
- ❑ A local reconstruction of **clusters** is performed in each detector
- ❑ **Vertex** and **tracks** are reconstructed. Particle identification is carried on
- ❑ Output of the reconstruction is the **Event Summary Data (ESD)**

THE ALICE SOFTWARE (ALIROOT)

Main features of the reconstruction software:

- ❑ Easy-to-use interface to extract the information from ESD
- ❑ Flexible reconstruction (reconstruction can be done in one detector even if other detectors are not operationals)
- ❑ Use of common base classes in the different reconstruction modules
- ❑ Exchange of information between the detectors done via a common track class

About the analysis software:

- ❑ The starting point of analyses are ESD (size of the ESD, one order of magnitude lower than raw data)
- ❑ ESD can be filtered to **AOD (Analysis Object Data)**, specific to a given sets of physics objectives (muons,…).
- ❑ Some common analysis tools exist (correction framework, event mixing framework, etc…)

THE CMS SOFTWARE (CMSSW)

Sources:

https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook

https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideSimulation

http://moby.mib.infn.it/~nason/mcws4/chierici.pdf

- ❑ CMSSW is the CMS Offline framework
- ❑ CMSSW uses **ROOT**, an object-oriented data analysis framework
- ❑ CMSSW is written in **C++ language**
- ❑ CMS Software is maintained in a **CVS** software repository
- ❑ The Framework is built around an **Event Data Model (EDM)** and services needed by the simulation, calibration, alignment and reconstruction modules

THE CMS SOFTWARE (CMSSW)

The Event Data Model (EDM):

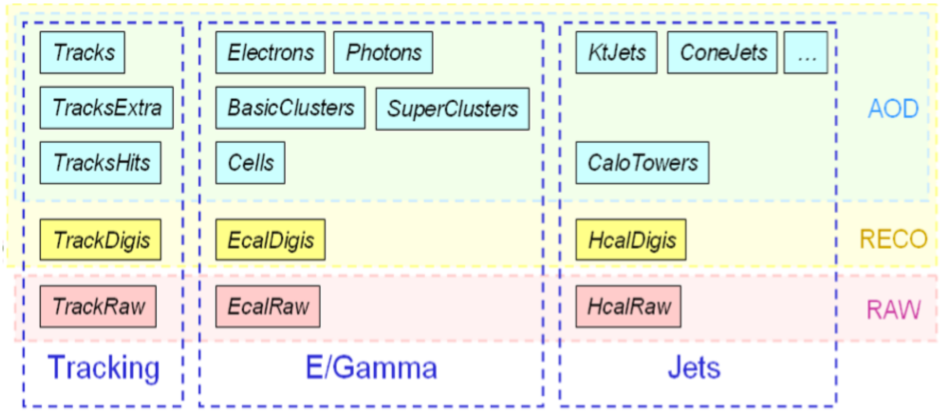
- ❑ EDM is centered aroud the concept of an **Event.** An event is a C++ object container for all raw and reconstructed data related to a particular collision. During processing, data are passed from one module to the next via the Event, and are accessed only through the Event. All objects in the event are stored in ROOT files

- ❑ EDM consists of one executable called cmsRun and many plug-in modules
- ❑ All the code needed in event processing (calibration, reconstruction..) is contained in the modules
- ❑ Same executable is used for detector and MC data
- ❑ cmsRun is configured at run time to choose which data to use, which module to execute, which parameter settings to use for each module,etc…
- ❑ cmsRun is lightweight: **only the required modules are dynamically loaded at the beginning of the job**

THE CMS SOFTWARE (CMSSW)

Events from a software point of view:

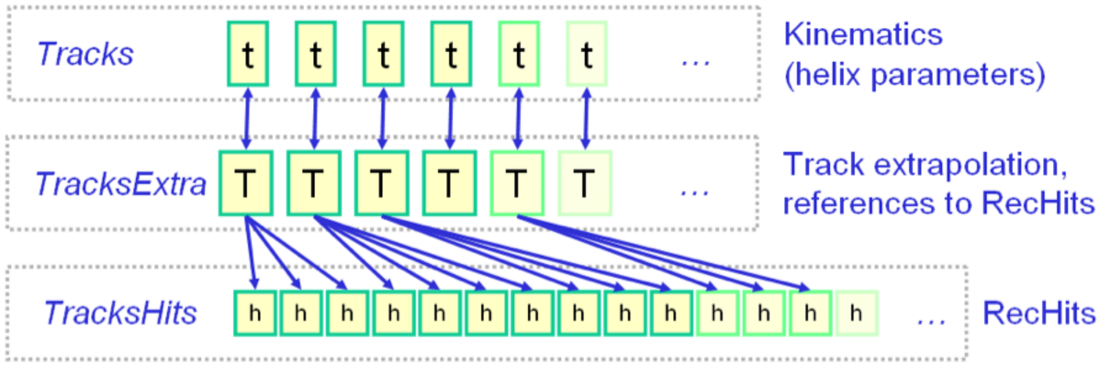
- An Events starts as a collection of the RAW data from a detector or MC event
- As the event is processed, products (or producer modules) are stored in the event as **reconstructed data objects (RECO)**
- Event holds all data taken during the triggered physics event, and data derived from the taken data. Event also contain metadata (software configuration, calibration..)
- The event data is **browsable by ROOT (n-tuples)**
- Products in an event are stored in different containers (particle container, hit container, service container…)
- **The Full Event Data (FEVT)** in an event in the RAW + RECO data
- **Analysis Object Data (AOD)** is a subset of the RECO data in an event.



THE CMS SOFTWARE (CMSSW)

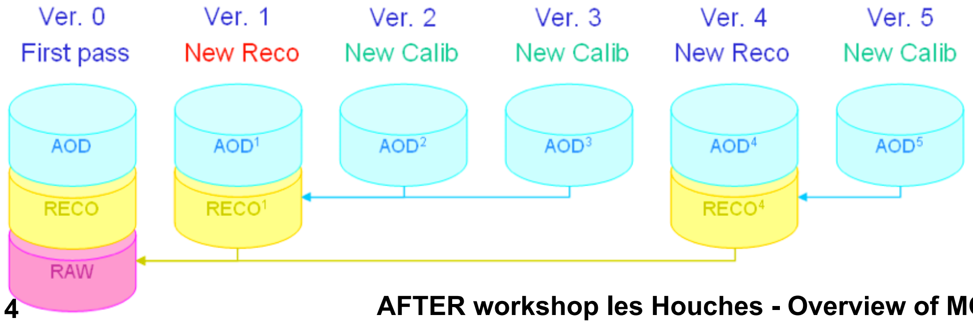
A modular event content:

- The event data architecture is **modular**. Different data layers can be configured, and a given application can use any layer or layers. The branches can be loaded or dropped on demand



Tracks
t t t t t t …
Kinematics (helix parameters)
TracksExtra
T T T T T T …
Track extrapolation, references to RecHits
TracksHits
h h h h h h h h h h h h h h …
RecHits

- Event data can be reprocessed at any stage. For instance, if the available AOD doesn't contain what you want, you can reprocess a reconstruction, to reproduce AOD.

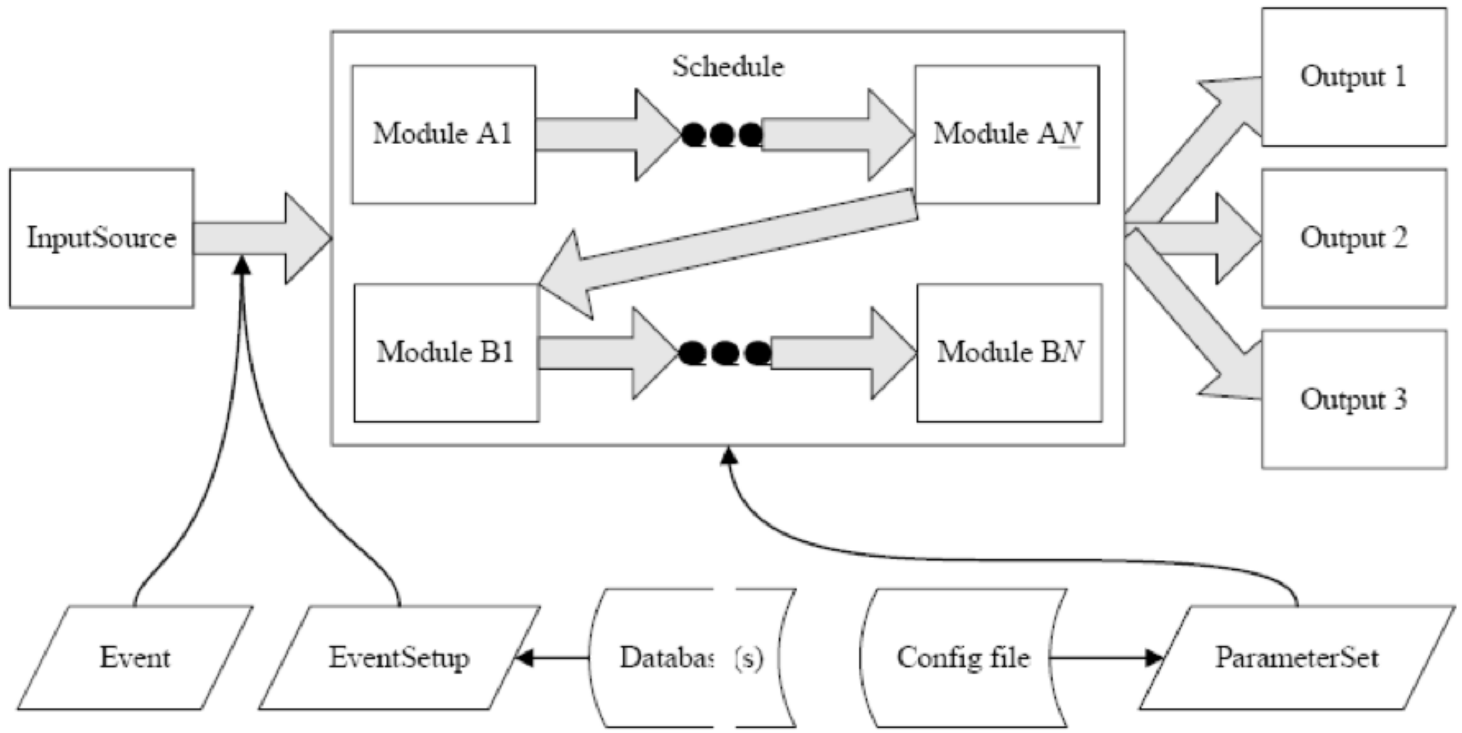


User can remove or insert quantities inside the event

THE CMS SOFTWARE (CMSSW)

The processing model:

- ❑ Events are processed by passing the event through a sequence of modules (specified by the user)
- ❑ A module can get data from the Event and put data back into the event
- ❑ In principle, order of the modules is not important



THE CMS SOFTWARE (CMSSW)

Simulations in CMS:

❑The simulation procedure include the following standard steps:

- Modeling of the interaction region
- Modeling of the particle passage through the hierarchy of volumes that compose CMS detector
- Modeling of the effect of multiple interactions per beam crossing (pileup)
- Modeling of the detector 's electronics response (digitization)

❑ Simulation framework integrated in CMSSW and can be integrated in the event processing chain

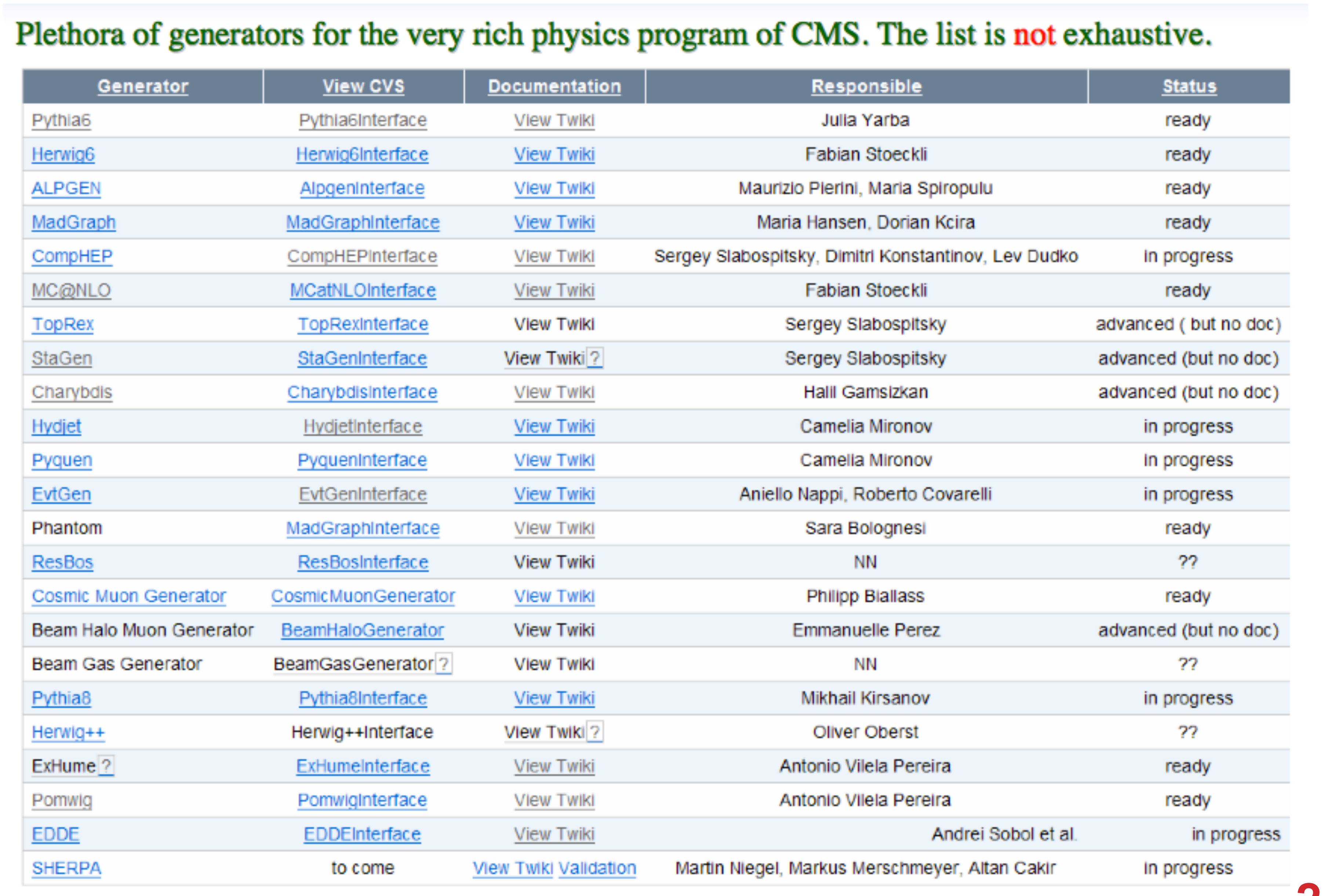
❑ CMSSW offers an **interface with a large number of MC generators** (each generator directly interface with CMSSW is implemented as an individual module that can be plug-in)

❑ **Particle guns** are also available (software testing purposes, beam test…)

❑ **Matrix element generator** also available (ASCII file output)

❑ Full and fast simulations are possible. In full simulation, generated particles are run through a detailed, **Geant4-based simulation** of the CMS detector, and detector electronics response in modeled

THE CMS SOFTWARE (CMSSW)



Plethora of generators for the very rich physics program of CMS. The list is not exhaustive.

Generator	View CVS	Documentation	Responsible	Status
Pythia6	Pythia6Interface	View Twiki	Julia Yarba	ready
Herwig6	Herwig6Interface	View Twiki	Fabian Stoeckli	ready
ALPGEN	AlpgenInterface	View Twiki	Maurizio Pierini, Maria Spiropulu	ready
MadGraph	MadGraphInterface	View Twiki	Maria Hansen, Dorian Kcira	ready
CompHEP	CompHEPInterface	View Twiki	Sergey Slabospitsky, Dimitri Konstantinov, Lev Dudko	in progress
MC@NLO	MCatNLOInterface	View Twiki	Fabian Stoeckli	ready
TopRex	TopRexInterface	View Twiki	Sergey Slabospitsky	advanced (but no doc)
StaGen	StaGenInterface	View Twiki ?	Sergey Slabospitsky	advanced (but no doc)
Charybdis	CharybdisInterface	View Twiki	Halil Gamsizkan	advanced (but no doc)
Hydjet	HydjetInterface	View Twiki	Camelia Mironov	in progress
Pyquen	PyquenInterface	View Twiki	Camelia Mironov	in progress
EvtGen	EvtGenInterface	View Twiki	Aniello Nappi, Roberto Covarelli	in progress
Phantom	MadGraphInterface	View Twiki	Sara Bolognesi	ready
ResBos	ResBosInterface	View Twiki	NN	??
Cosmic Muon Generator	CosmicMuonGenerator	View Twiki	Philipp Biallass	ready
Beam Halo Muon Generator	BeamHaloGenerator	View Twiki	Emmanuelle Perez	advanced (but no doc)
Beam Gas Generator	BeamGasGenerator ?	View Twiki	NN	??
Pythia8	Pythia8Interface	View Twiki	Mikhail Kirsanov	in progress
Herwig++	Herwig++Interface	View Twiki ?	Oliver Oberst	??
ExHume ?	ExHumeInterface	View Twiki	Antonio Vilela Pereira	ready
Pomwig	PomwigInterface	View Twiki	Antonio Vilela Pereira	ready
EDDE	EDDEInterface	View Twiki	Andrei Sobol et al.	in progress
SHERPA	to come	View Twiki Validation	Martin Niegel, Markus Merschmeyer, Altan Cakir	in progress

THE LHCB SOFTWARE (GAUSS)

Sources:

http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/gauss/

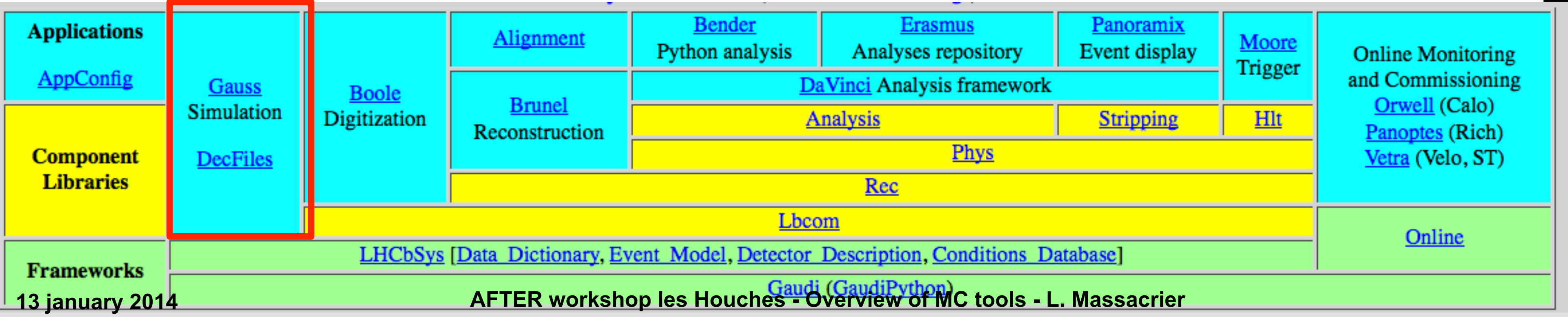
http://lhcb-comp.web.cern.ch/lhcb-comp/Support/CMT/cmt.htm

http://lhcb-comp.web.cern.ch/lhcb-comp/Simulation/Gauss.pdf

http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/Gaudi/Gaudi_v9/GUG/GUG.pdf

http://tel.archives-ouvertes.fr/docs/00/68/33/50/PDF/HDR_P._Robbe.pdf **(2012)**

- ❑ **Gauss** is the simulation program of the LHCb experiment, based on the **Gaudi framework** and on **LHCbSys** classes
- ❑ The programming language is C++ (python also used in scripts)
- ❑ LHCb Software was maintained in a **SVN** repository → moved to Git
- ❑ Gauss is structured as a **Configuration Management Tool (CMT)** project. CMT is used to specify the compilation and execution environment for packages within a project



THE LHCB SOFTWARE (GAUSS)

The Gaudi framework philosphy:

- ❑ The Gaudi project is an open project for providing the necessary interfaces and services for building HEP experiment frameworks in the domain of event processing applications
- ❑ **The Gaudi framework is experiment independent**
- ❑ Gaudi provides an **architecture** (specification of a certain number of components and their interactions between each other)
- ❑ All software written by physicists (generation, reconstruction, analysis) will be in the form of **specialisations** of specific components (standard component + added functionality) but **interface remains the same**
- ❑ In the application framework this is done by deriving new classes from one of the base classes:
  - Data Object (for quantity-like)
  - Algorithm (for procedures)
  - Converter
- ❑ **A clear distinction between « data » objects and « algorithm » objects** (ex mathematical objects such as vectors, points are distincts from the fitting algorithm)

THE LHCB SOFTWARE (GAUSS)

The Gaudi framework provides several services:

- ❑ Event data model/ data storage
- ❑ Detector description database (ex DetectorElement class)
- ❑ Histograms facilities
- ❑ N-tuples / Event collection
- ❑ Standard message services (loggin, error, warnings…)
- ❑ Particle properties service (Pythia ID, Geant3 ID, charge, mass…)
- ❑ Code profiling
- ❑ Random number services
- ❑ GiGa service (usage of Geant4 toolkit as a blackbox)
- ❑ The possibility to develop new services!

THE LHCB SOFTWARE (GAUSS)

- Gauss is interfaced to **specialized packages** available in the Physics community for the **Generation phase** and makes use of the **Geant4** toolkit for the simulation phase

- **The LHCb simulation software consists of two independent phases:**

  1/ The generation of the primary event

  2/ The tracking of the particle produced in the experimental setup (detectors)

  The first step can be further divided into two important main parts, both using external generator packages:

  - production of particles coming out of the primary pp collision of the LHC beam (often **Pythia**)
  - decay and time evolution of the produced particles (often **EvtGen**)

- The framework is very generic, flexible to generate a very large variety of event types (beam gas events, rare decays of B mesons…)

THE LHCB SOFTWARE (GAUSS)

The generation of events:

❑ The generation of the events require three main actions:

1/ **Initialization**: configuration of the algorithm (files in Python), computation of the parameters to be used by the generation software, from user inputs

2/ **Event loop execution**: generation of one physics event, stored in HepMC format. The event is then transferred to the simulation step.

3/ **Finalization**: printing of monitoring counters (get the cross section of generated processes)

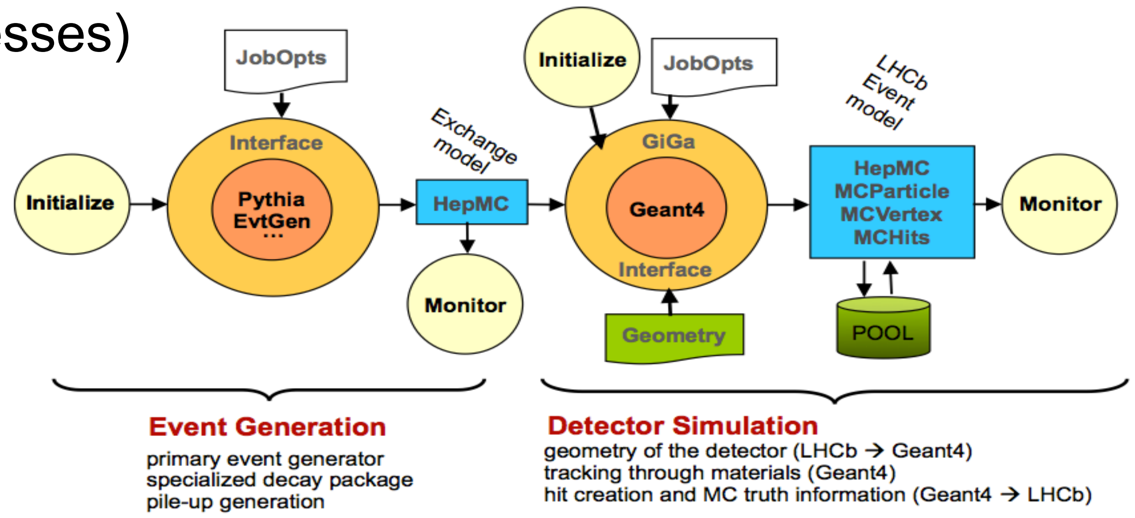


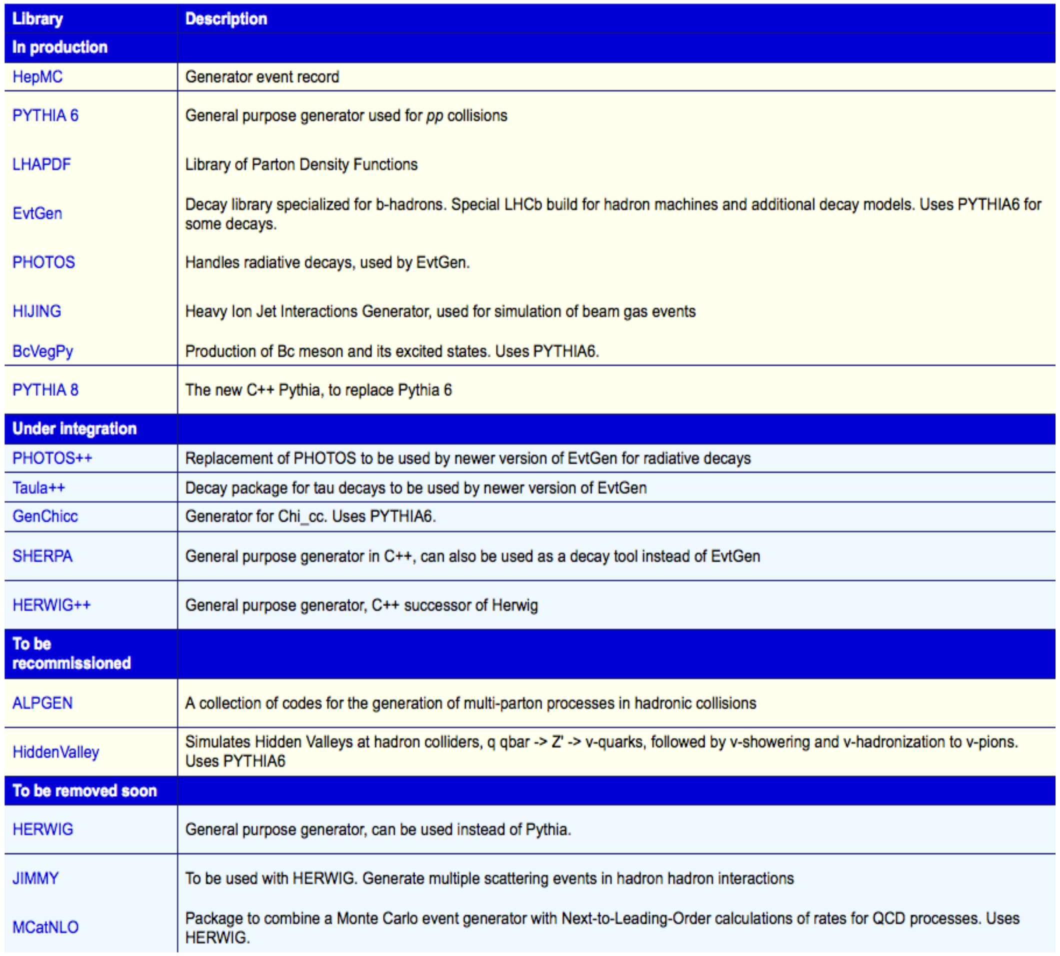
Figure 1.1 – *Structure of the LHCb simulation software, Gauss.*

THE LHCB SOFTWARE (GAUSS)

- The generation algorithm call various tools to realize specific computations:

  - **sample generation tools** (minimum bias, inclusive events, signal events)
  - **production tools** (interface with external MC generators)
  - **decay tools** (force decay chain of Geant4, EvtGen developped at CLEO and BABAR)
  - **pileup tools**
  - **beam tools** (crossing angles)
  - **vertex smearing tool**
  - **cut tools** (can be done at generation level)

There is a sequencing of these tools



Library	Description
In production	
HepMC	Generator event record
PYTHIA 6	General purpose generator used for *pp* collisions
LHAPDF	Library of Parton Density Functions
EvtGen	Decay library specialized for b-hadrons. Special LHCb build for hadron machines and additional decay models. Uses PYTHIA6 for some decays.
PHOTOS	Handles radiative decays, used by EvtGen.
HIJING	Heavy Ion Jet Interactions Generator, used for simulation of beam gas events
BcVegPy	Production of Bc meson and its excited states. Uses PYTHIA6.
PYTHIA 8	The new C++ Pythia, to replace Pythia 6
Under integration	
PHOTOS++	Replacement of PHOTOS to be used by newer version of EvtGen for radiative decays
Taula++	Decay package for tau decays to be used by newer version of EvtGen
GenChicc	Generator for Chi_cc. Uses PYTHIA6.
SHERPA	General purpose generator in C++, can also be used as a decay tool instead of EvtGen
HERWIG++	General purpose generator, C++ successor of Herwig
To be recommissioned	
ALPGEN	A collection of codes for the generation of multi-parton processes in hadronic collisions
HiddenValley	Simulates Hidden Valleys at hadron colliders, q qbar -> Z' -> v-quarks, followed by v-showering and v-hadronization to v-pions. Uses PYTHIA6
To be removed soon	
HERWIG	General purpose generator, can be used instead of Pythia.
JIMMY	To be used with HERWIG. Generate multiple scattering events in hadron hadron interactions
MCatNLO	Package to combine a Monte Carlo event generator with Next-to-Leading-Order calculations of rates for QCD processes. Uses HERWIG.

THE ATLAS SOFTWARE (ATHENA)

Sources:

The ATLAS computing workbook (2009)

http://iopscience.iop.org/1742-6596/219/3/032001/pdf/1742-6596_219_3_032001.pdf

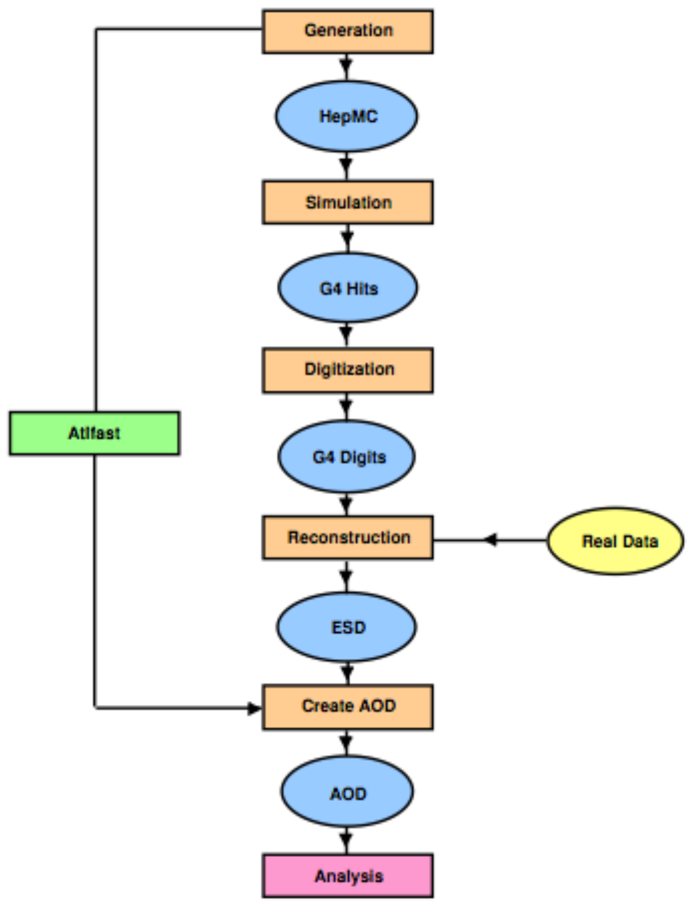
http://tel.archives-ouvertes.fr/docs/00/14/05/24/PDF/thesis_Binet_final.pdf (2006)

http://hep.physics.indiana.edu/~hgevans/p411-p610/material/09_collab/iu110210.pdf (2011)

- ATHENA is the framework of the ATLAS Collaboration
- Most of the information on the software, wiki pages are not public
- Core of ATHENA («packages») is written in C++ language
- ATHENA software is maintained in a **SVN** repository
- **ATHENA (as LHCb) is also developed inside the Gaudi framework!** CMT, processing of events in simulation is similar to LHCb

THE ATLAS SOFTWARE (ATHENA)

Monte Carlo Simulations in ATHENA:



Schematic representation of the Full Chain Monte Carlo production.

ATHENA provides interfaces with several MC generators:
Pythia
Herwig
Sherpa
HIJING
Pythia8
Herwig++
Tauola
Phojet
Photos
…

REQUIRED HUMAN RESOURCES TO WRITE A CORE SOFTWARE

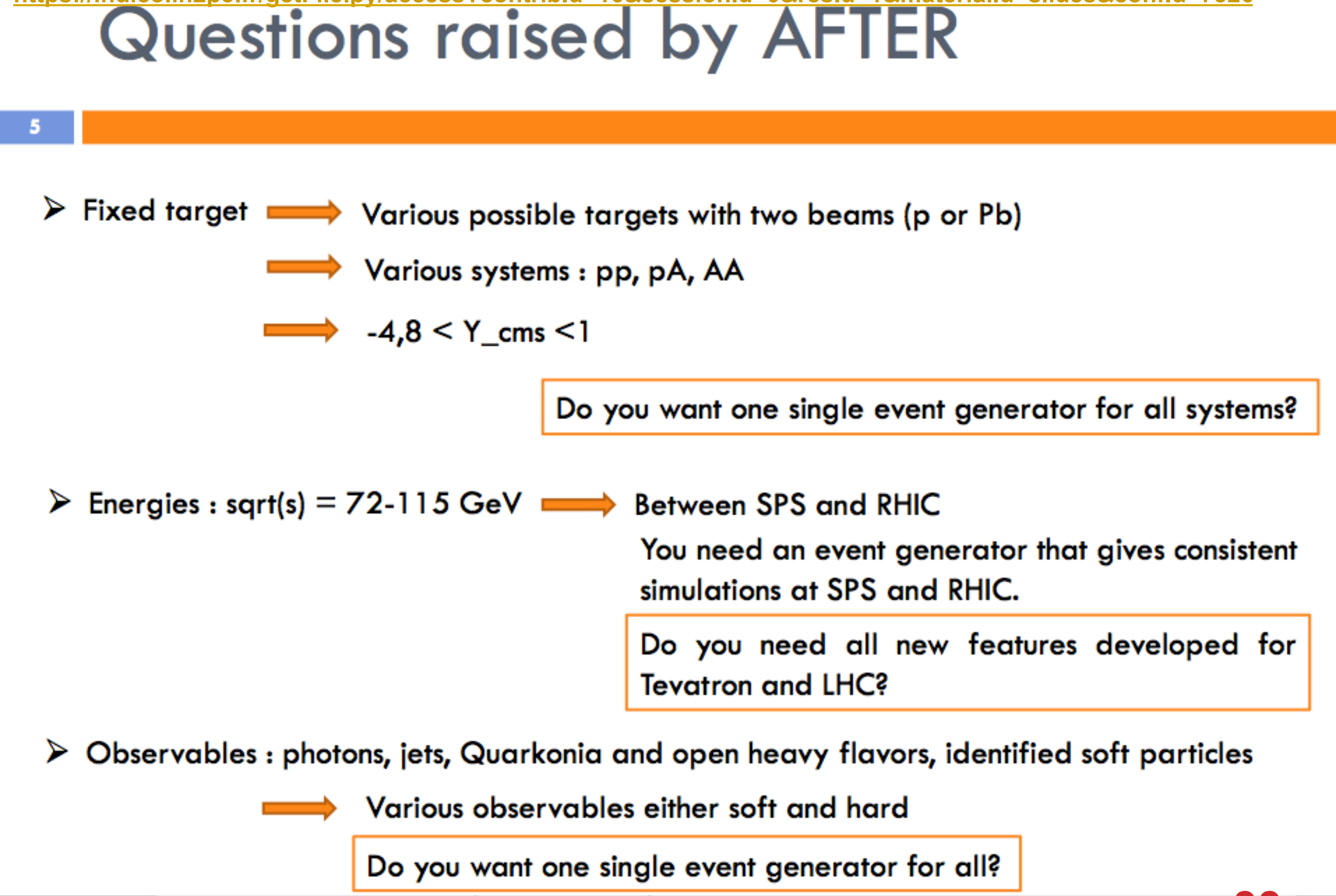
Source: S. Bethke, LHC computing review, LHCC, 21 mars 2001

Year	**2000 have(missing)**	**2001**	**2002**	**2003**	**2004**	**2005**
ALICE	12(5)	17.5	16.5	17.0	17.5	16.5
ATLAS[1]	23(8)	36	35	30	28	29
CMS	15(10)	27	31	33	33	33
LHCb	14(5)	25	24	23	22	21
Totals	**64(28)**	**105.5**	**106.5**	**103**	**100.5**	**99.5**

1 CORE software includes everything except the algorithmic part of the reconstruction software, the simulation and physics analysis. Human resources attributable to the GRID are not included.

MONTE CARLO GENERATORS FOR SIMULATIONS

From S. Porteboeuf presentation at AFTER workshop in Trento
https://indico.in2p3.fr/getFile.py/access?contribId=16&sessionId=0&resId=1&materialId=slides&confId=7326



Questions raised by AFTER

5

- Fixed target $\Rightarrow$ Various possible targets with two beams (p or Pb)
  - $\Rightarrow$ Various systems : pp, pA, AA
  - $\Rightarrow$ -4,8 < Y_cms <1

Do you want one single event generator for all systems?

- Energies : sqrt(s) = 72-115 GeV $\Rightarrow$ Between SPS and RHIC

  You need an event generator that gives consistent simulations at SPS and RHIC.

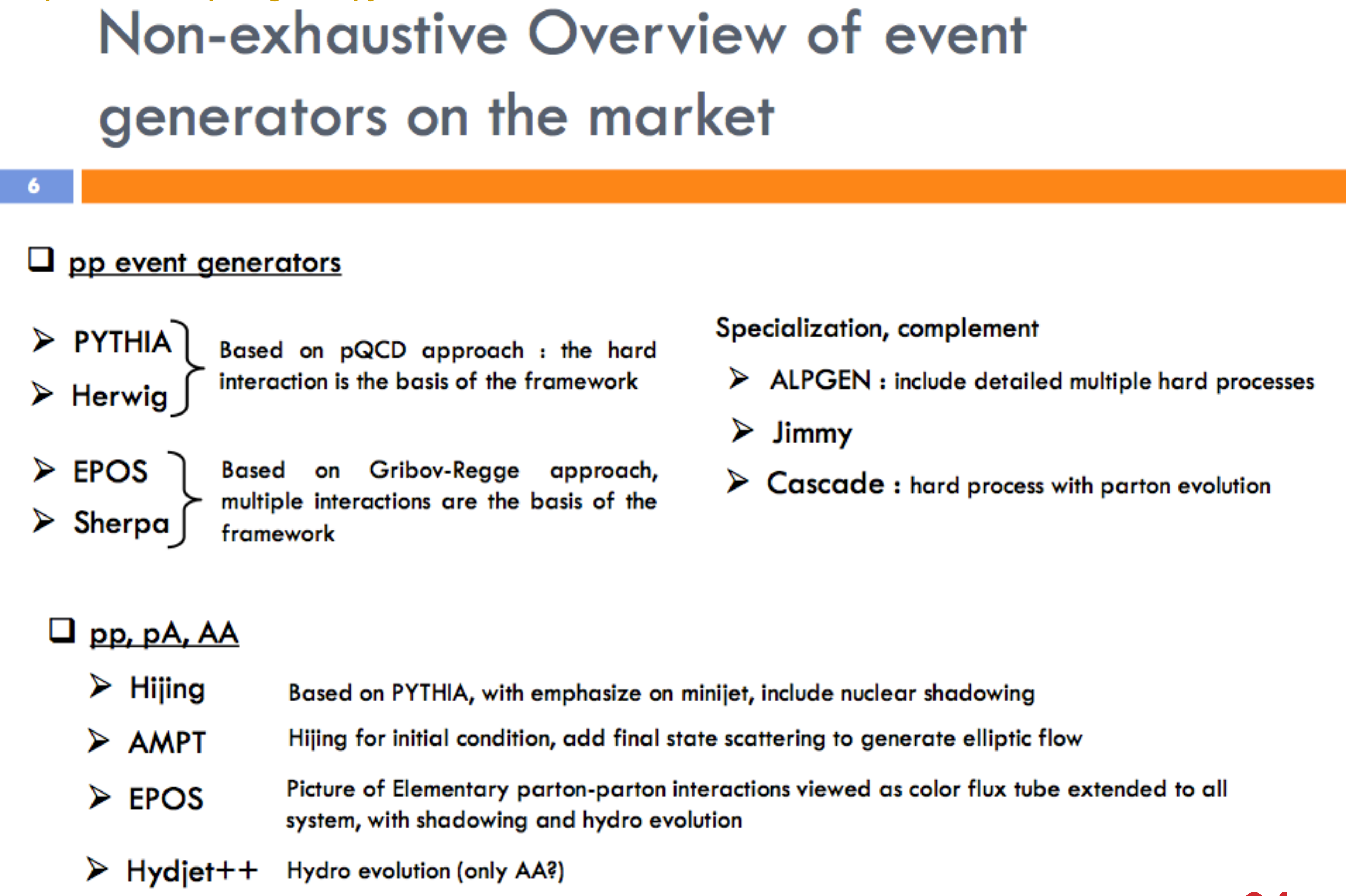
Do you need all new features developed for Tevatron and LHC?

- Observables : photons, jets, Quarkonia and open heavy flavors, identified soft particles
  - $\Rightarrow$ Various observables either soft and hard

Do you want one single event generator for all?

From S. Porteboeuf presentation at AFTER workshop in Trento

https://indico.in2p3.fr/getFile.py/access?contribId=16&sessionId=0&resId=1&materialId=slides&confId=7326



Non-exhaustive Overview of event generators on the market

6

❑ pp event generators

- ➢ PYTHIA
- ➢ Herwig

Based on pQCD approach : the hard interaction is the basis of the framework

- ➢ EPOS
- ➢ Sherpa

Based on Gribov-Regge approach, multiple interactions are the basis of the framework

Specialization, complement

- ➢ ALPGEN : include detailed multiple hard processes
- ➢ Jimmy
- ➢ Cascade : hard process with parton evolution

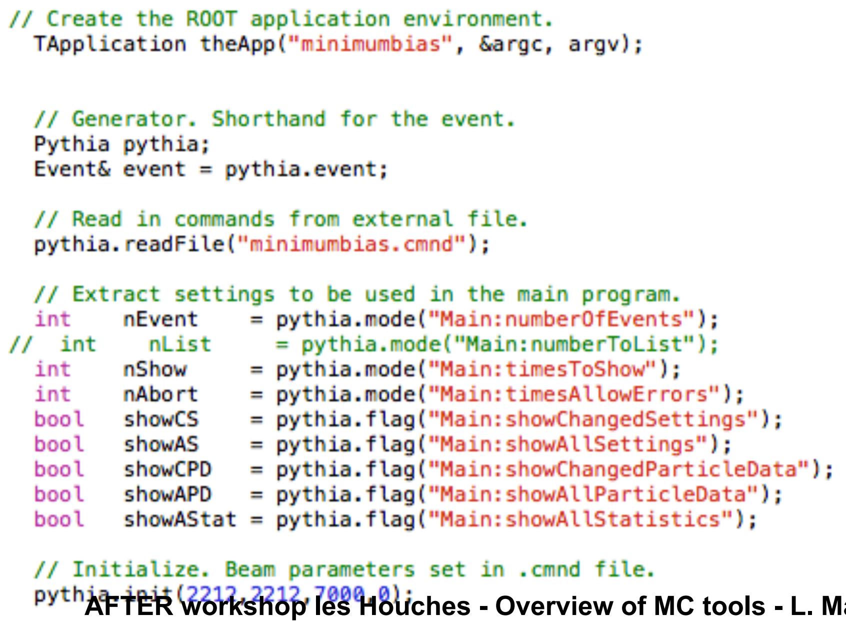
❑ pp, pA, AA

➢ Hijing	Based on PYTHIA, with emphasize on minijet, include nuclear shadowing
➢ AMPT	Hijing for initial condition, add final state scattering to generate elliptic flow
➢ EPOS	Picture of Elementary parton-parton interactions viewed as color flux tube extended to all system, with shadowing and hydro evolution
➢ Hydjet++	Hydro evolution (only AA?)

SIMULATIONS FOR AFTER USING PYTHIA8

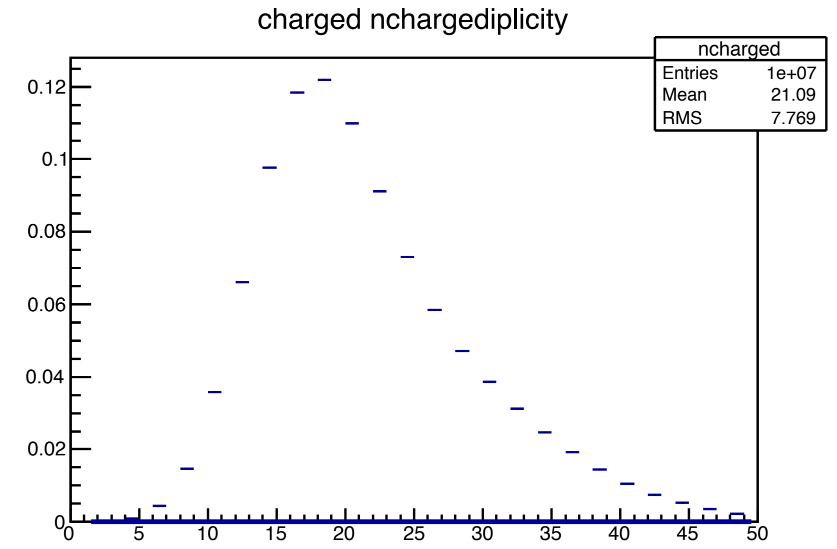
SIMULATIONS FOR AFTER USING PYTHIA8

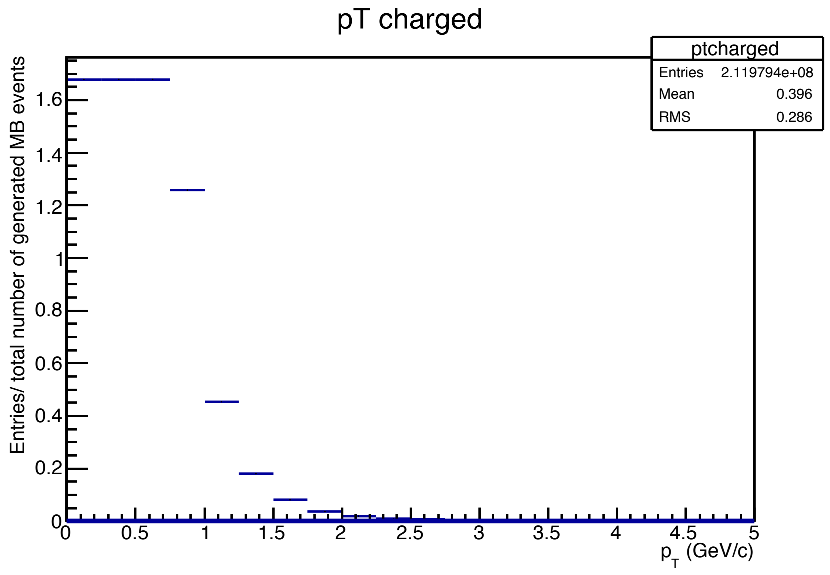
- Some work started with S. Porteboeuf at previous AFTER workshop in Trento
- Pythia8 was ran in standalone mode to make simulations of several particles of interests: charged particles, pions, protons, neutrons, kaons, lambda, J/ψ, χ_c, η_c, D, ϒ, B …
- Simple ROOT histogramming was also implemented (there is lot of room for improvement)
- 10 Millions minimum bias events generated locally (~2h on one computer)

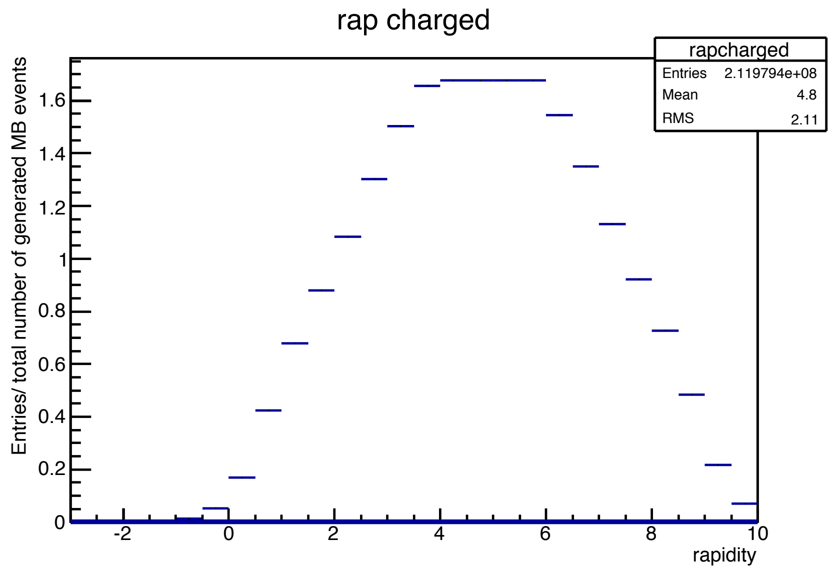


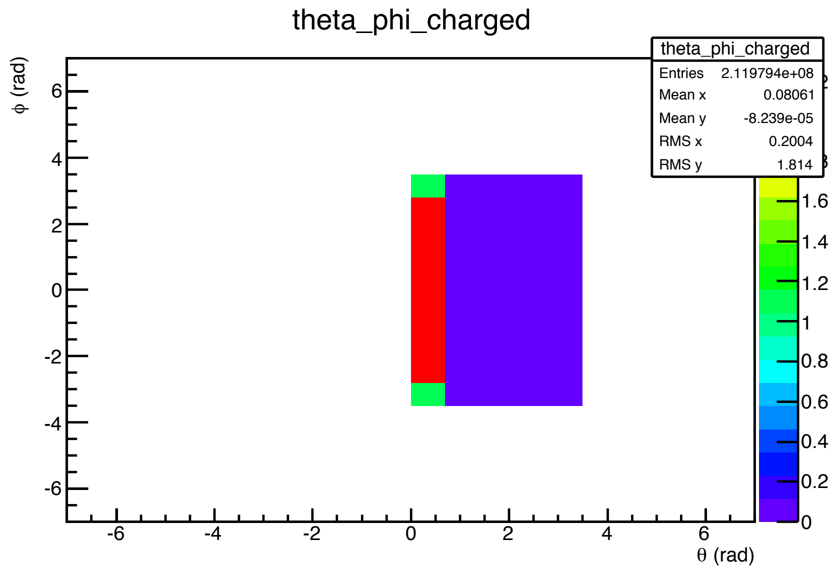
```
// Create the ROOT application environment.
  TApplication theApp("minimumbias", &argc, argv);


  // Generator. Shorthand for the event.
  Pythia pythia;
  Event& event = pythia.event;

  // Read in commands from external file.
  pythia.readFile("minimumbias.cmnd");

  // Extract settings to be used in the main program.
  int     nEvent    = pythia.mode("Main:numberOfEvents");
//  int     nList     = pythia.mode("Main:numberToList");
  int     nShow     = pythia.mode("Main:timesToShow");
  int     nAbort    = pythia.mode("Main:timesAllowErrors");
  bool    showCS    = pythia.flag("Main:showChangedSettings");
  bool    showAS    = pythia.flag("Main:showAllSettings");
  bool    showCPD   = pythia.flag("Main:showChangedParticleData");
  bool    showAPD   = pythia.flag("Main:showAllParticleData");
  bool    showAStat = pythia.flag("Main:showAllStatistics");

  // Initialize. Beam parameters set in .cmnd file.
  pythia.init(2212,2212,7000.0);
```

SOME OUTPUT EXAMPLE: CHARGED PARTICLES

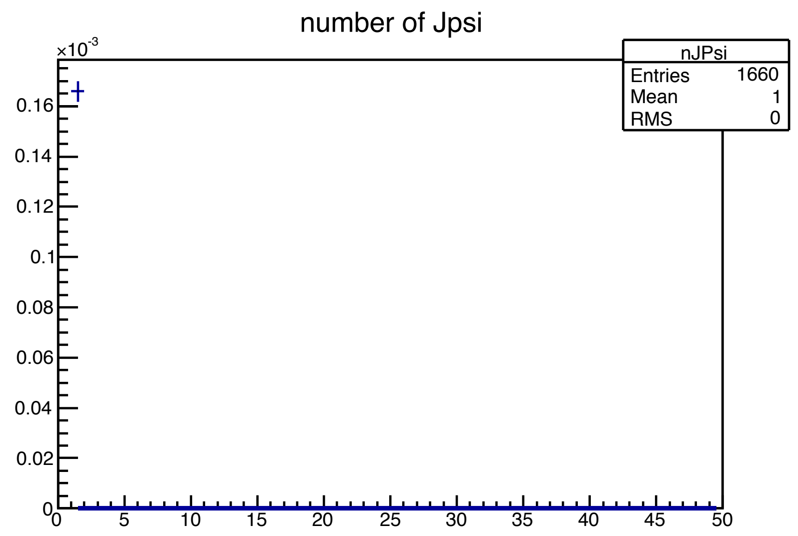


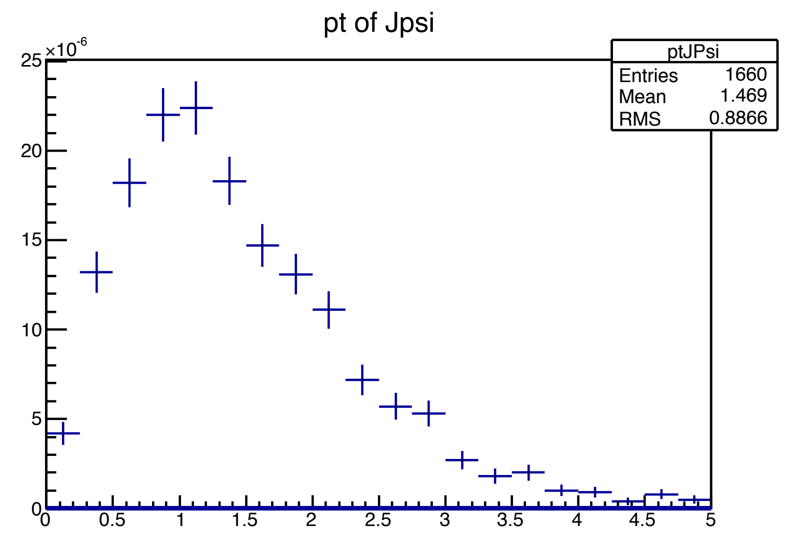


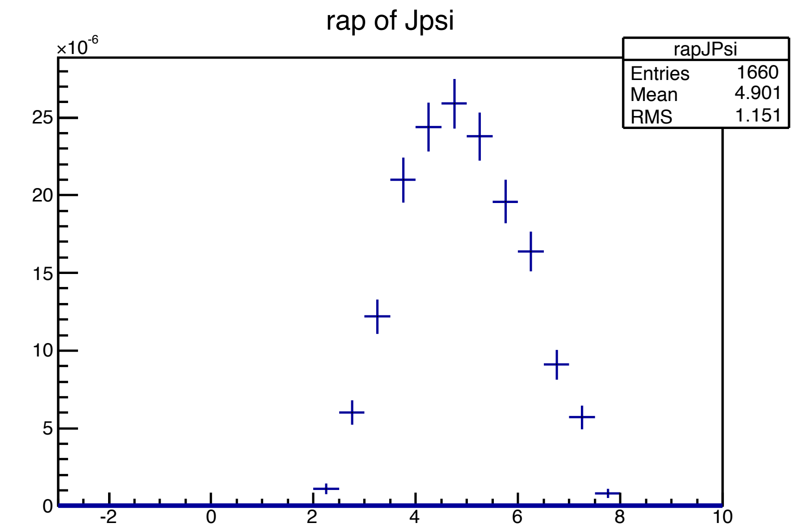


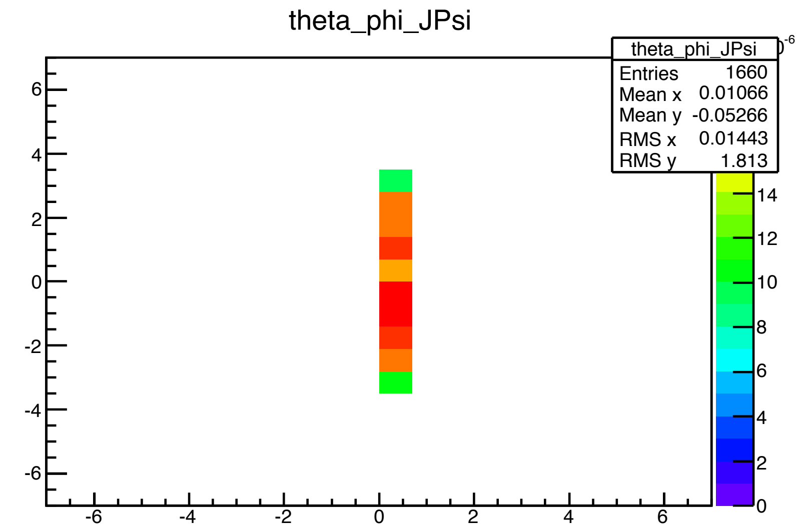


SOME OUTPUT EXAMPLE: J/Ψ









CONCLUSION

In this conclusion, I would like to highlight some of the tools I presented which can be of interest for AFTER:

- **The Virtual MONTE CARLO TOOL of ALICE which provides interfaces with several transport codes (even with Geant 3 written in FORTRAN)**
- **The Event Data Model philosophy adopted in CMS software which permits to have a modular architecture for the event**
- **The Gaudi framework adopted by both LHCB and ATLAS which offers a general architecture experiment independent and benefits from the development coming from both collaborations**

The question is now, what are your needs for AFTER?