EUDET Software Framework

Common Analysis and Simulation Software

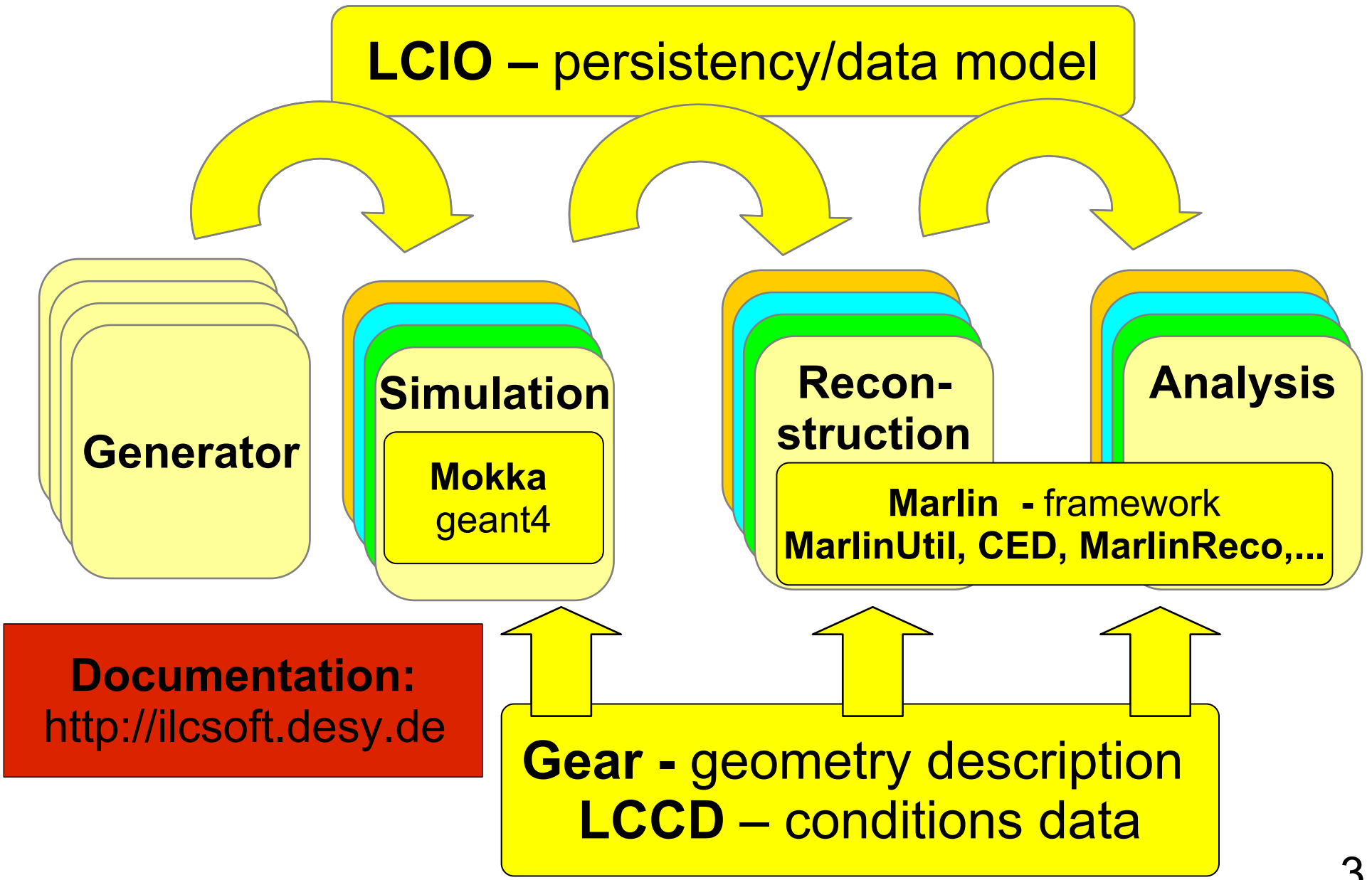
Frank Gaede
DESY

EUDET Annual Meeting
LLR-Paris October 8-10, 2007

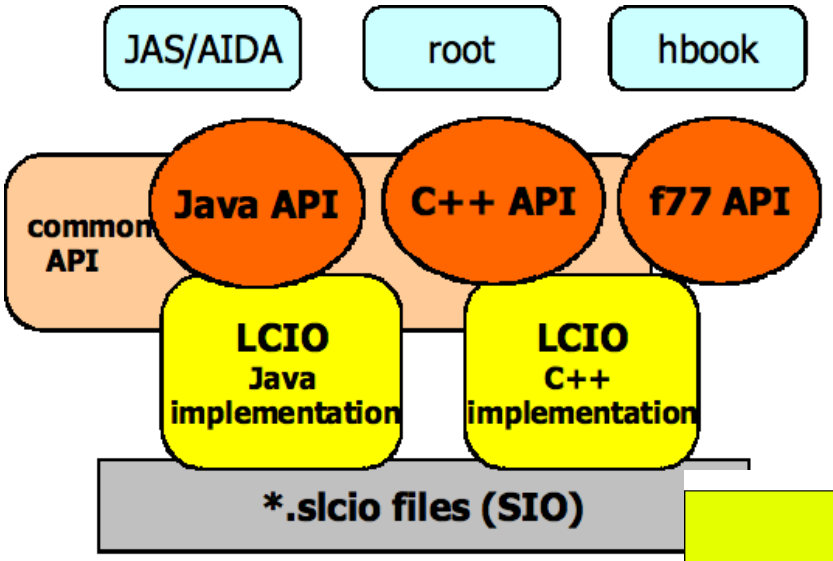
Objectives for task ANALYS

- **development of a common data analysis and simulation infrastructure**
  - development of a software framework for the exchange, analysis and comparison of test beam data
  - development of a software framework for the simulation of test beam experiments
  - creation of a repository for experimental and simulation data
  - embedding into existing GRID infrastructure
- **strategy**
  - the testbeam software effort is tightly integrated with the overall common ILC/LDC software effort !
  - implement tools and functionality specific to testbeams
  - benefit from synergies where possible
  - same for grid tasks: integrate with common ILC grid activities

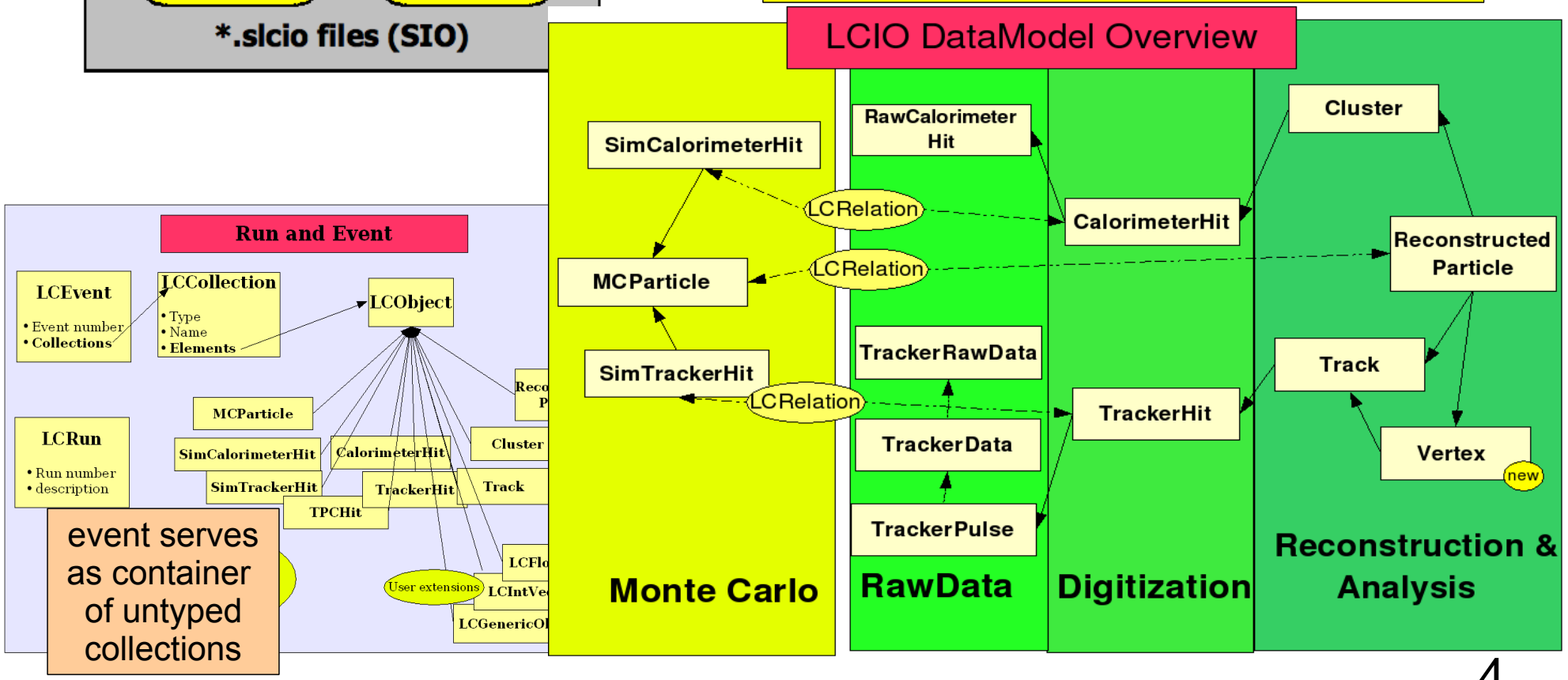
EUDET/LDC SW-framework



LCIO: persistency & event data model



- DESY SLAC joined project
  (first presented at CHEP03)
- Java, C++ and f77 (!) API
- extensible data model
- now standard for
- ILC persistency & datamodel
- -> used in all detector concept
- studies



LCIO recent developments

- extended the event data model
  - introduced dedicated Vertex class
  - introduced raw data classes for tracking detector testbeams: TrackerRawData, TrackerData, TrackerPulse
    - used by JRA1&JRA1
- comand line tool (Java) for checking and manipulating events: dump, merge, split,...
- runtime extensions (next slides)
- under development/discussion
  - improve I/O performance
  - direct access: reading part of the event, split files,...
  - user defined data classes

feedback from DAQ/tbeam groups needed

LCIO runtime extensions (C++)

- long pending user request:
  - attach user objects to LCObjects
  - fast and easy creation of links (relations) between various LCObject subtypes, eg. TrackerHits and Track
- features
  - extension of the object with arbitrary (even non-LCObject) classes
  - extension of single objects or vectors, lists of objects
  - optionally ownership is taken for extension objects (memory management)
  - bidirectional relations between LCObjects
    - one to one
    - one to many
    - many to many

to be used in reconstruction and analysis algorithms - no persistency

LCIO runtime extensions

extensions and relations identified through a tagging **class T**

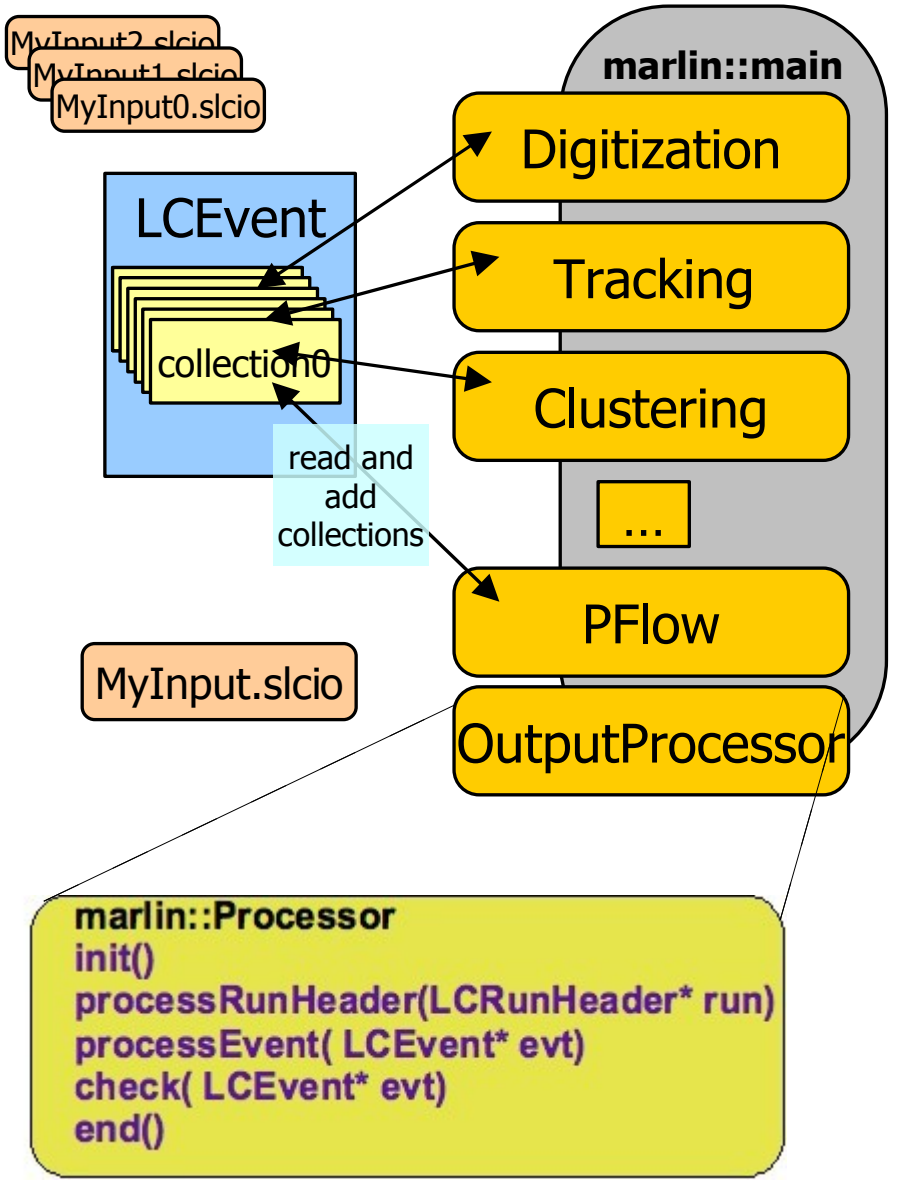
```
// a simple int extension
struct Index : LCIntExtension<Index> {} ;

// a many to many relationship between MCParticles
struct ParentDaughter : LCNToNRelation<ParentDaughter,MCParticle,MCParticle> {} ;
//..
 MCParticle*  mcp = dynamic_cast<MCParticle*>( mcpcol->getElementAt(i) ) ;
//..

 mcp->ext<Index>() = i;    // set an int


 const MCParticleVec& daughters = mcp->getDaughters() ;

 for(unsigned j=0 ; j< daughters.size()   ; j++ ){

   // ---- set biderctional relation
   add_relation<ParentDaughter>( mcp, daughters[j] ) ;
 }

//-------------------------------------------------

 cout << " myindex = " << mcp->ext<Index>  << endl ;

 ParentDaughter::to::rel_type daulist =  mcp->rel<ParentDaughter::to>() ;

 for( ParentDaughter::to::const_iterator idau = daulist->begin();
   idau != daulist->end(); ++idau){

    cout << (*idau)->ext<Index>() << ", " ;
 }
 cout << endl ;
```

for extensions use **ext<T>()**
for relations use **rel<T::to>()** and **rel<T::from>()**

Marlin – core application framework

Modular**A**nalysis & **R**econstruction for the **L I N**ear Collider

- modular C++ application framework for the analysis and reconstruction of ILC data
- **LCIO** as transient data model
- xml steering files:
  - fully configure application
  - order of modules/processors
  - parameters global + processor
- self documenting
  - parameters registered in user code
- consistency check of input/output collection types
- **Plug & Play** of modules

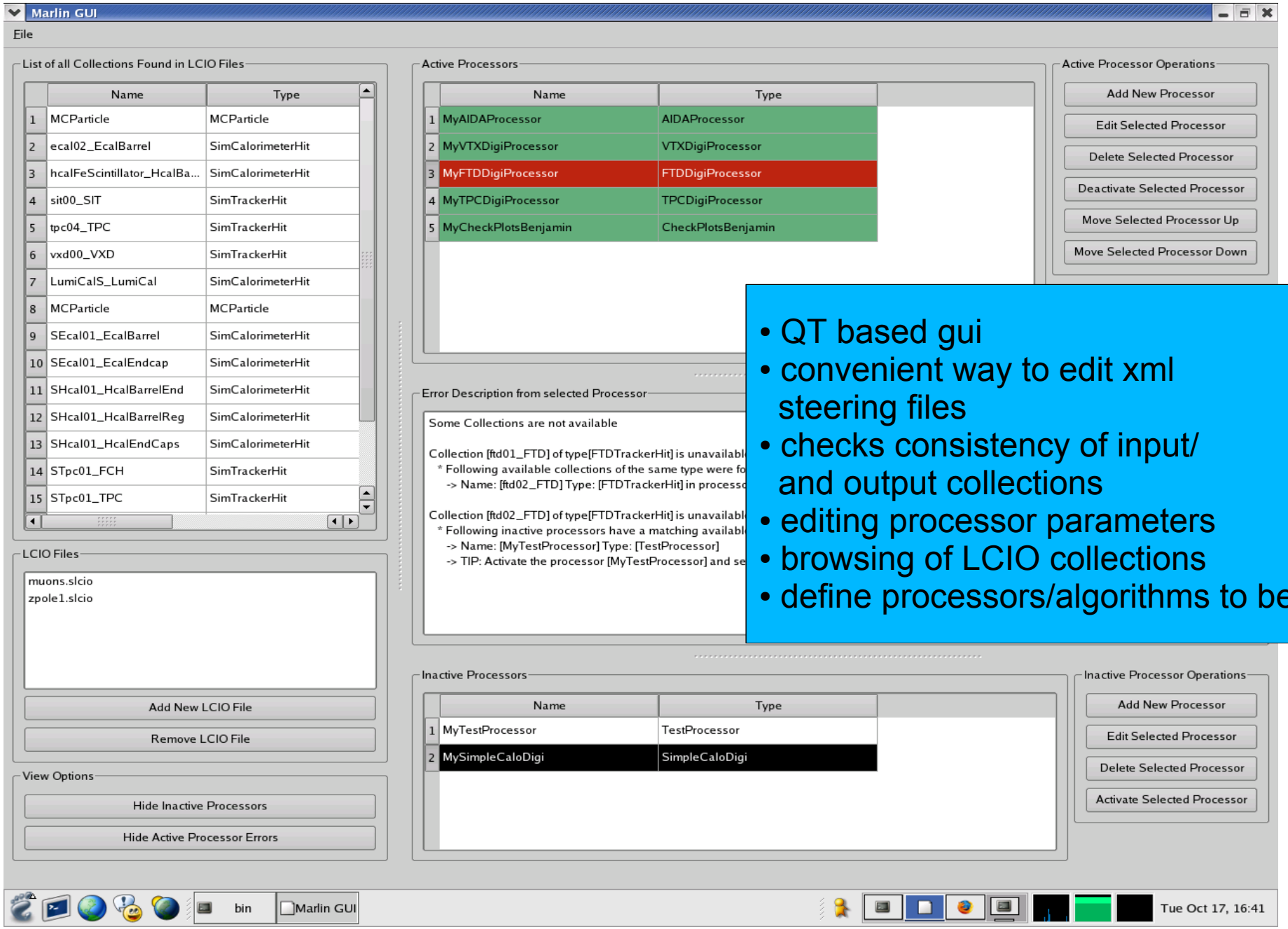


Marlin new developments

- Marlin fully functional since 2005
  - -> focus on increasing user, i.e. developer convenience
- introduced new build system: Cmake (see talk J.Engels)
  - high level scripts for creating makefiles for common platforms Linux, MacOS, Windows
  - 'successor of GNU autotools', e.g. used by KDE
  - allows easy configuration of build process and options
- switched to shared libraries
- provide support for plugins (see talk J.Engels)
  - packages with processors built into shared libraries
  - loaded at program start up
  - no relinking of full application necessary
- MarlinGUI, flow charts, logging mechanism (next slides)

MarlinGUI

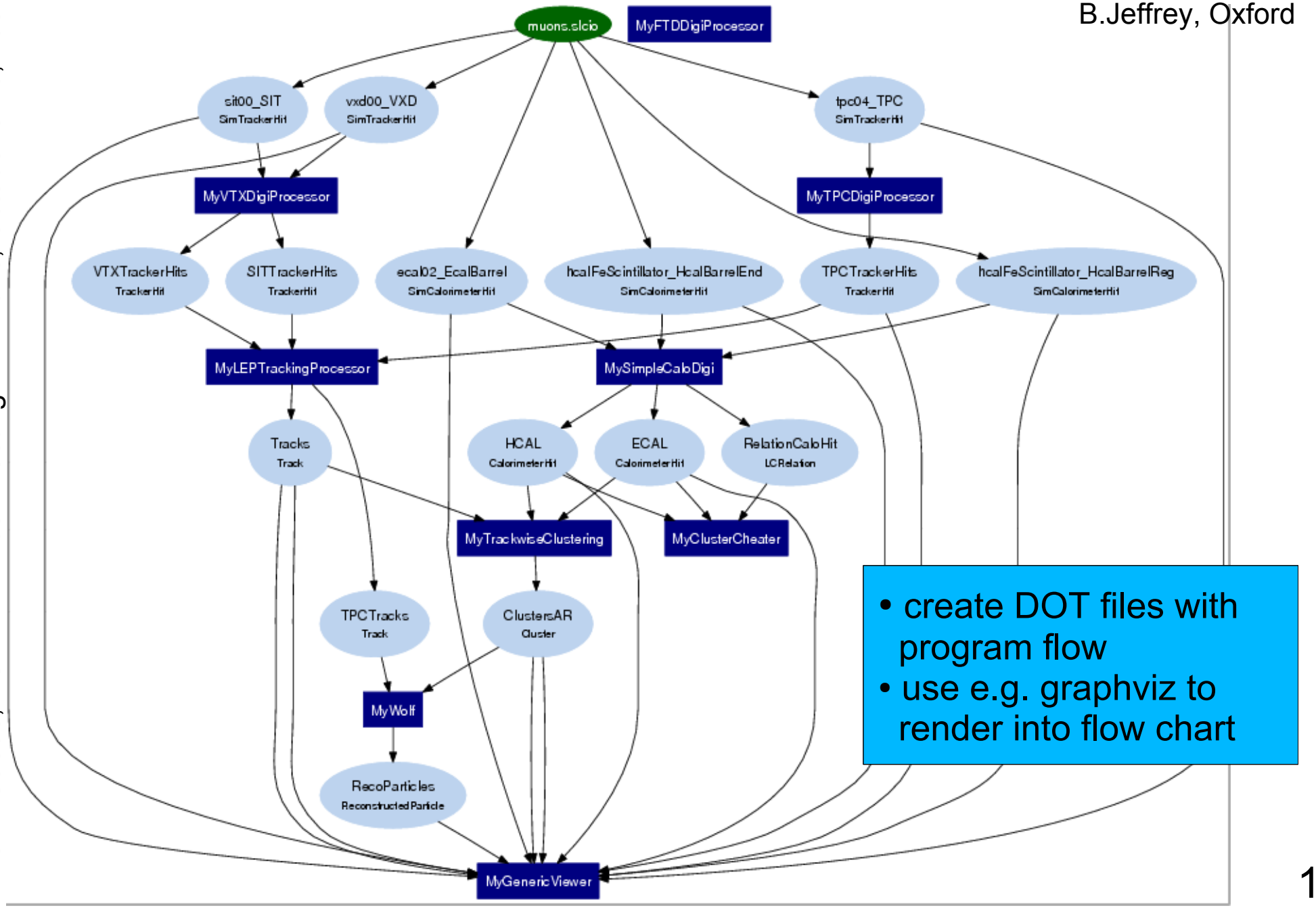
J.Engels, DESY



- QT based gui
- convenient way to edit xml steering files
- checks consistency of input/ and output collections
- editing processor parameters
- browsing of LCIO collections
- define processors/algorithms to be run

Marlin program flow charts

B.Jeffrey, Oxford



- create DOT files with program flow
- use e.g. graphviz to render into flow chart

streamlog – logging library

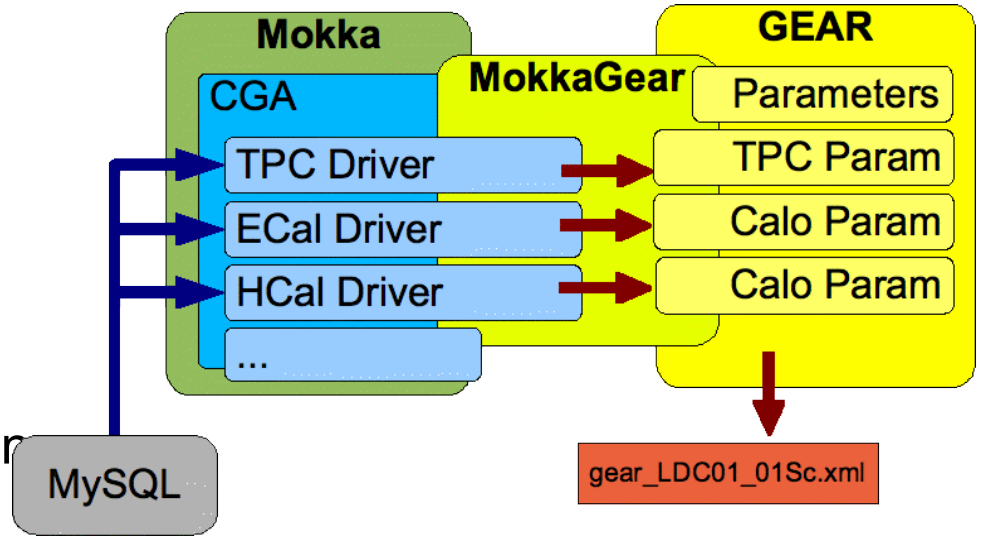
- standalone logging library
  - shipped with Marlin but can also be used in non-Marlin cde
  - verbosity levels: DEBUG0-4, MESSAGE0-4, WARNING0-4, ERROR0-4
  - verbosity level and current namespace can be changed on the scope level (processor name in Marlin)
  - no runtime and space overhead for DEBUG messages when compiled with NDEBUG (production)
    - per compilation unit (plugin)
  - very little overhead (if-statement) for other messages
  - simple macro for replacing std::cout (std::ostream)

```
streamlog_out( DEBUG ) << " digitizing hit : "
                       << hit->getCellID() << std::endl ;
[ DEBUG "TrackDigitizer" ]  digitizing hit : 12345678
```

geometry for reconstruction

GEometry **A**PI for **R**econstruction

- high level abstract interface:
  - per subdetector type (Hcal,TPC,...) parameters/quantities for reco
    - geometry + some navigation
    - implementation uses xml files written from Mokka (simulation)
  - abstract interface for detailed geometry &materials:
    - point properties
    - path properties
    - implementation based on geant4



MokkaGear

- enforce only one source of geometry: the simulation program creates the geometry xml files used in reconstruction

(recently improved by K.Harder et al)

example – GEAR API VXD

Gear: gear::VXDParameters class Reference - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://ilcsoft.desy.de/gear/v00-03/doc/html/classgear_1_1VXDParameters.html

simulation/geant4 LCIO Linux Conferences DESY IT Group LEO English/Ger... Google MyHome Ctime

virtual const **VXDLayerLayout** & **getVXDLayerLayout** () const=0
The layer layout in the Vertex.

virtual int **getVXDType** () const=0
The type of Vertex detector: VXDParameters.CCD, VXDParameters.CMOS or VXDPara

virtual double **getShellHalfLength** () const=0
The half length (z) of the support shell in mm (w/o gap).

virtual double **getShellGap** () const=0
The length of the gap in mm (gap position at z=0).

virtual double **getShellInnerRadius** () const=0
The inner radius of the support shell in mm.

virtual double **getShellOuterRadius** () const=0
The outer radius of the support shell in mm.

virtual double **getShellRadLength** () const=0
The radiation length in the support shell.

virtual bool **isPointInLadder** (**Point3D** p) const=0
returns whether a point is inside a ladder

virtual bool **isPointInSensitive** (**Point3D** p) const=0
returns wheter a point is inside a sensitive volume

virtual **Vector3D** **distanceToNearestLadder** (**Point3D** p) const=0
returns vector from point to nearest ladder

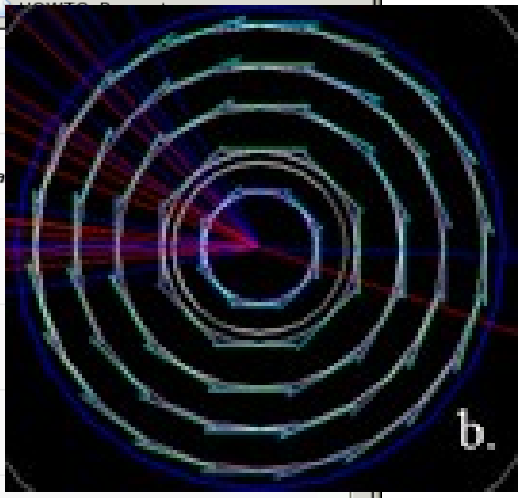
virtual **Vector3D** **distanceToNearestSensitive** (**Point3D** p) const=0
returns vector from point to nearest sensitive volume

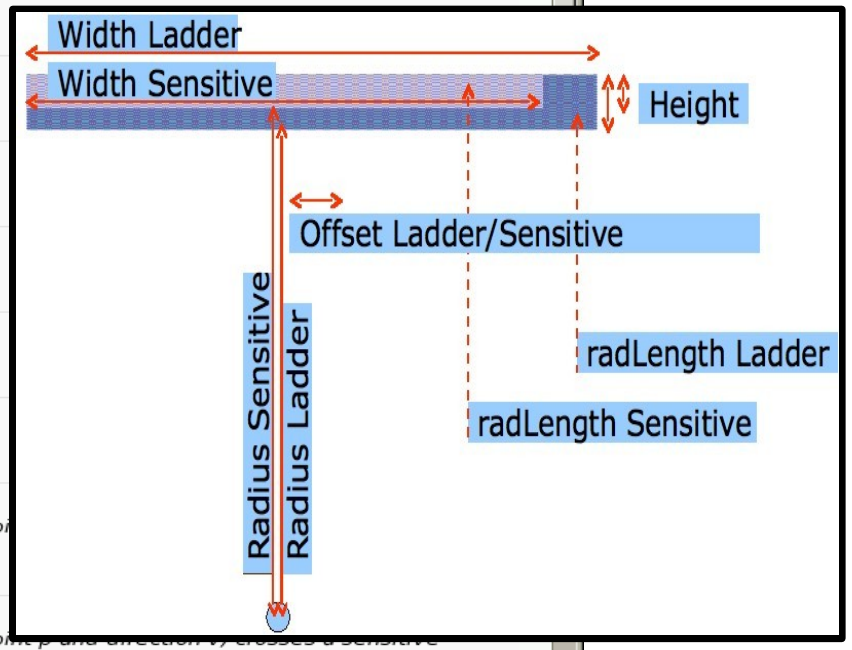
virtual **Vector3D** **intersectionLadder** (**Point3D** p, **Vector3D** v) const=0
returns the first point where a given strainght line (parameters po
is returned if no intersection can be found.

virtual **Vector3D** **intersectionSensitive** (**Point3D** p, **Vector3D** v) const=0
returns the first point where a given strainght line (parameters po
volume (0,0,0) is returned if no intersection can be found.

Find: VXD Find Next Find Previous Highlight Match case

Done

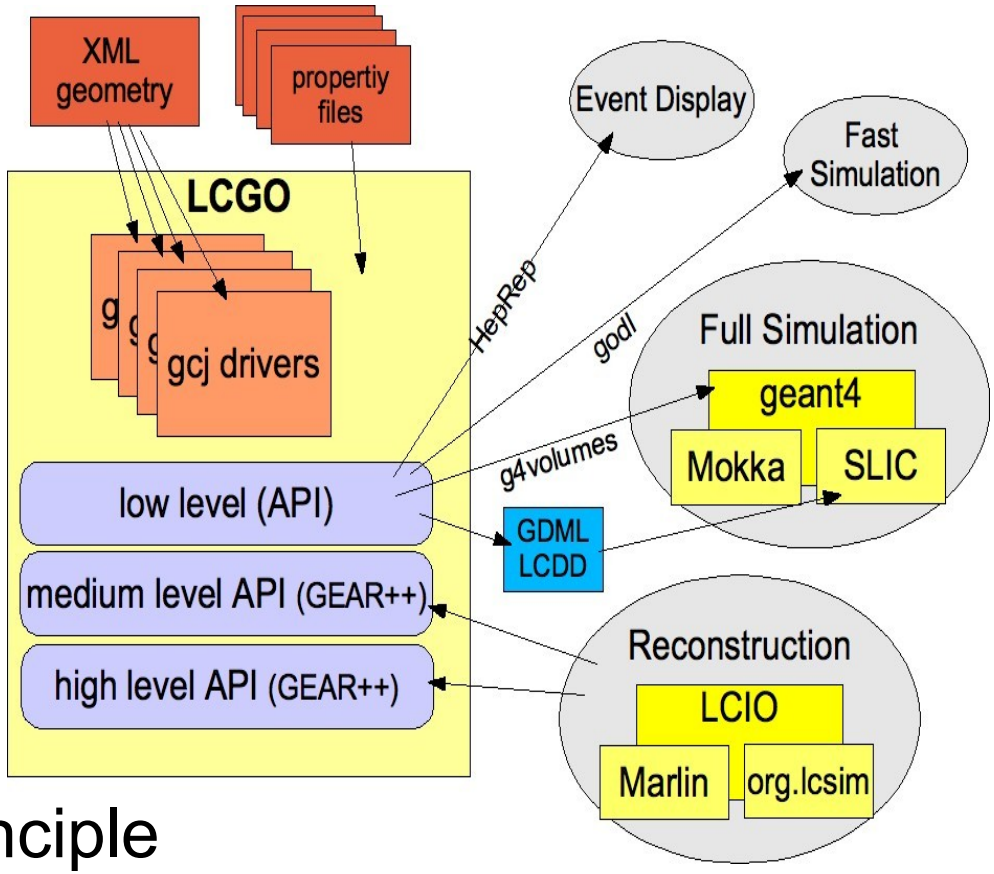




status lcgo

A Common Geometry Toolkit

- **LCGO**: A common geometry toolkit to be used in all(?) ILC frameworks
  - SLAC-DESY project - initially
  - -> of course open for all collaborators, e.g. FNAL
  - work just started – aiming for spring/summer 2007
- requirements/goals for LCGO:
  - be at least as functional as existing systems (org.lcsim, GEAR, Mokka, SLIC,...)
  - enable smooth transition path from existing systems
  - encourage/increase interoperability between systems
  - have no known principle short comings: "everything should be possible"

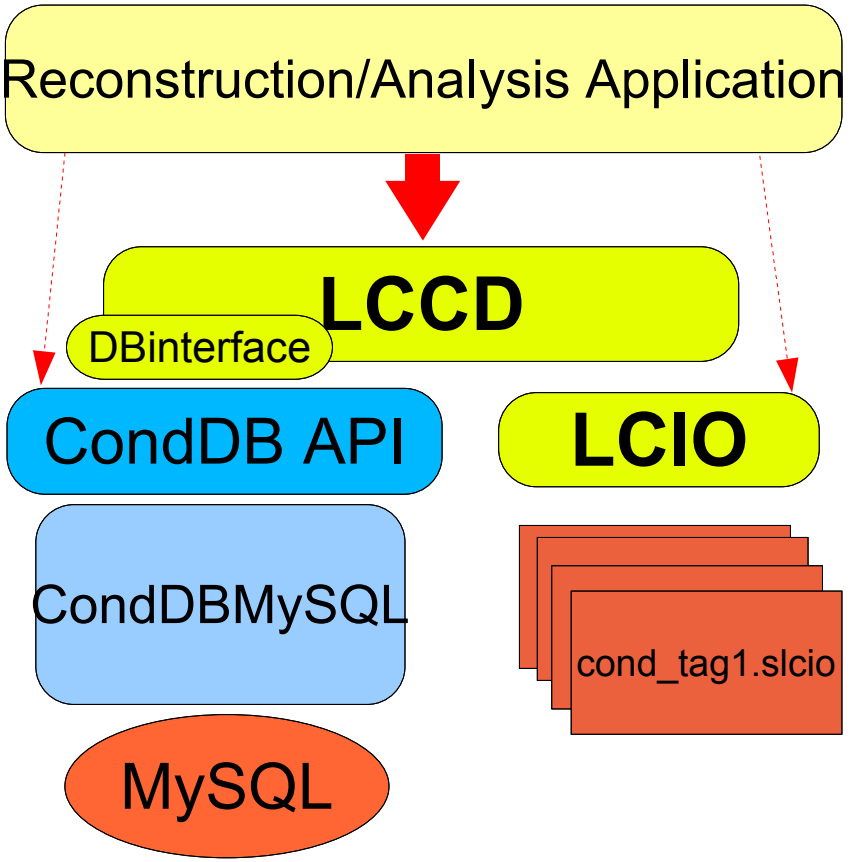


- protoype exists: proof of principle
  - **issues:** gcj code significantly slower than plain C++ or Java
  - a lot of Java code is incompatible with current gcj (Java1.5)
  - is close horizontal collaboration still on the agenda ?
- need to look into other options as well

LCCD

Linear **C**ollider **C**onditions **D**ata Toolkit

- Reading conditions data
  - from conditions database
  - from simple LCIO file
  - from LCIO data stream
  - from dedicated LCIO-DB file
- Writing conditions data
- tag conditions data
- Browse the conditions database
  - through creation of LCIO files
    - vertically (all versions for timestamp)
    - horizontally (all versions for tag)



LCCD is used for the conditions data of the ongoing ILC testbeam studies

Event Overlay

- new Marlin package Overlay (N.Chiapolini)
  - requires Marlin v00-09-09 and LCIO v01-08-04
  - new interface ModifyEvent / modifyEvent(LCEvent* evt){}
- provides LCIO event overlay for simulated data:
  - SimCalorimeterHits
  - SimTrackerHits
  - MCParticle
- read and overlay additional LCIO stream:
  - fixed # bg-events / main event
  - # bg events from poison distribution
  - one run of bg-events per main event
- package can be extended by tbeam groups for other LCIO data types

status core software

- current release of core software: v01-01
  - /afs/desy.de/group/it/ilcsoft/v01-01 (SL3/SL4 only)
  - http://ilcsoft.desy.de/ilcinstall
- fully functional software framework for simulation, reconstruction and analysis of ILC (testbeam) data
- EUDET milestone: “Version 1.0 after 18 month” is reached
- Outlook:
  - improve performance of framework:
    - persistency – more generic data types for DAQ
    - I/O speed
    - conditions data access
    - ... user input needed

usage of root

- policy to not depend on root in core framework
  - only through interfaces, e.g AIDA/RAIDA
  - however a lot of functionality is only available in root directly, e.g. “column-wise n-tuples” - trees
- most users (all?) in Europe use root for their histograming and analysis tasks
- root provides also a lot more functionality that is needed/requested:
  - minuit, event display, geometry,....
  - investigate some 'controlled' i.e. well defined usage of root for special tasks
    - supported by new 'modular' root (boot)
    - stay tuned...

framework usage by EUDET activities

- JRA3 - CALICE tbeam software (talk R.Poeschl)
  - usage of Mokka, LCIO, Marlin, Gear, LCCD
  - started before EUDET
- JRA2 – MarlinTPC (talk M. Killenberg)
  - usage of LCIO, Marlin, Gear, LCCD
  - started with EUDET
- JRA1 – EUTelescope (talk A.Bulgheroni)
  - usage of Mokka, LCIO, Marlin, Gear (LCCD?)
  - ported existing code to common framework early this year
- all groups actively use the grid for data storage and processing within the VOs 'calice' and 'ilc'

Summary

- NA2 task ANALYS: "Provide a software framework for simulation and analysis (of testbeam data)"
- EUDET milestone: "Version 1.0 after 18 month" is reached
- software is fully grid compatible
- grid used for data storage and analysis

All EUDET software activities
within the JRAs are now
carried out in the context of
the existing software framework and
grid installations

EUDET reports/memos for ANALYS and JRA
software are in preparation

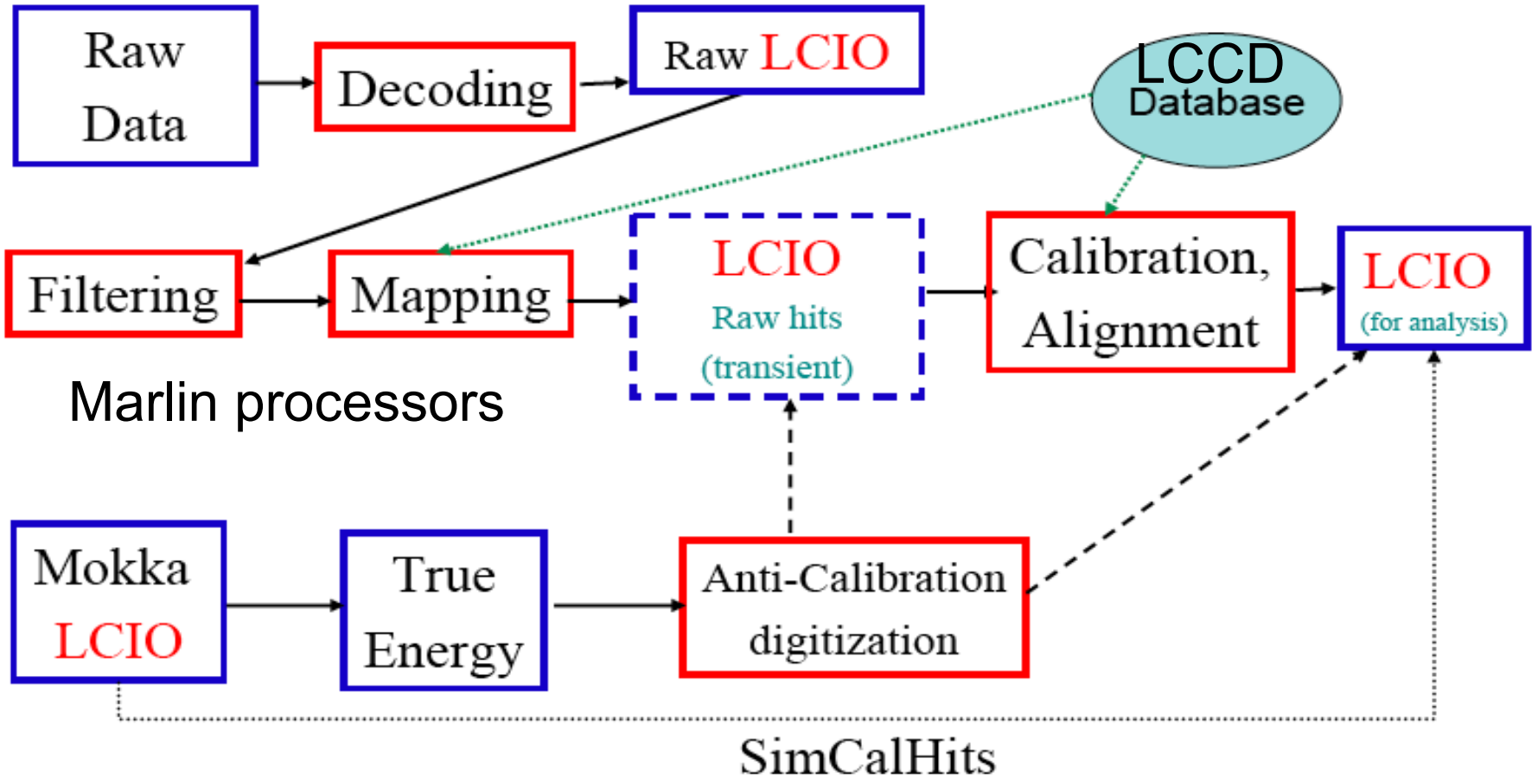
additional material

SW-framework recent developments

- installation script for all ILC software tools
  - download & build tools automatically
  - fully configurable python script
  - defines releases of core software tools
- reference installations/releases in afs (SL3, SL4)
- new build tool: cmake
  - allows for easy configuration and building the tools
  - will make porting the tools to other platforms easier
- Marlin-framework:
  - switched to shared libraries
  - plugin mechanism: decide at runtime(startup script) which modules are needed
  - build process greatly simplified (cmake)
- most of this done within EUDET project

JRA3 – usage of framework example

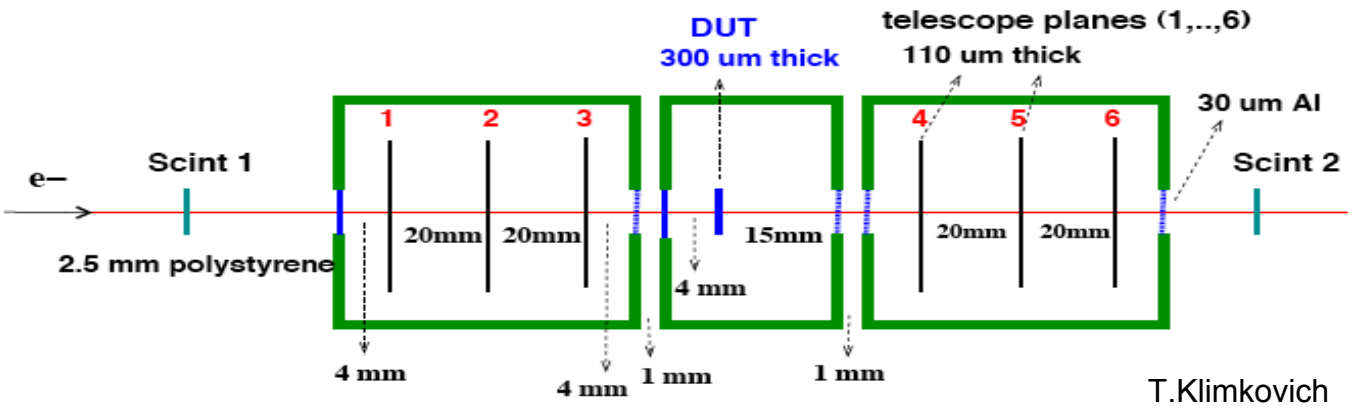
Software – data flow



Deliverables and Requirements

- **requirements**:
  - documentation and its regular update are of utmost importance
  - other EUDET participants should contribute by:
    - properly defining the requirements of the framework
    - providing and interfacing simulation and reconstruction software for the various detector technologies
    - testing the framework.
- **deliverables**:
- we expect to have a **first version** of the common data analysis and simulation framework ready **after 18 month (now)**
  - development however must continue throughout the whole duration of the project to cope with

JRA1 – usage of framework example



T.Klimkovich

- **Simulation**: **Mokka** (based on Geant 4)
  - New geometry driver **EUTelescope** has been created (on the way to be included into official Mokka release)
  - Class **TRKSD00** is used for telescope and DUT sensitive detectors
  - All parameters of the model are stored in **MySQL** database
  - Output: **LCIO** format files
  - Stored information: hit positions, deposited energy, ..
- **Telescope geometry interface** (within **Gear**) is implemented (will be included into next Gear release): detector "SiPlanes" of 2 types: **TelescopeWithDUT** and **TelescopeWithoutDUT**
- **Analysis**: **Marlin, Root, C++**

JRA2 – usage of framework example

Reconstruction

Data Structure	Processor Name	Collection Name
`TrackerRawData`		TPCRawData
	TrackerRawDataToDataConverter	
`TrackerData`		TPCConvertedRawData
	PedestalSubtractor	
`TrackerData`		TPCData
	PulseFinder	
	ChannelMapper	
`TrackerPulse`		TPCPulses
	HitFinder	
`TrackerHit`		TPCHits
	TrackFinder[Method]	
`Track`		TPCSeedTracks
	TrackFitter[Method]	
`Track`		TPCTracks

Correction processors (gain, pad response, linearity, time shift) still missing

Usage of budget - ANALYS

- DESY
  - commitment 12ppm: F.Gaede 25% for full project length
  - 12ppm (scientist) converted to hire a programmer for 18 month
    - started August 2006 – ends December 2007
    - will use funds from COMP to extend contract until end of project (24 month)
- RFWU-Bonn (K.Desch)
  - 8ppm (scientist) combined with funds from JRA2 to hire a postdoc that works on JRA2 and ANALYS (*MarlinTPC* sw project)
    - started early 2007
- IPASCR (J.Cvach)
  - commitment 3ppm: PhD student that works part time on calorimeter simulation with geant

Contributors for task ANALYS

	DESY	ALU-FR	IPASCR	TOTAL
REQUEST				
Perm Staff ppm				
Temp Staff ppm	12.000	8.000		20.000
Perm Staff Cost kEUR				
Temp Staff Cost kEUR	62.500	46.875		109.375
Travels kEUR	1.300	0.867		2.167
Consumables kEUR				
Overheads kEUR	12.760	9.548		22.308
Total Manpower ppm	12.000	8.000		20.000
Total Cost kEUR	76.560	57.290		133.850
COMMITTMENT				
Perm Staff ppm	12.000		3.000	15.000
Temp Staff ppm				
Perm Staff Cost kEUR	62.500		9.000	71.500
Temp Staff Cost kEUR				
Travels kEUR				
Consumables kEUR				
Overheads kEUR	12.500		1.800	14.300
Total Manpower ppm	12.000		3.000	15.000
Total Cost kEUR	75.000		10.800	85.800
TOTAL BUDGET				
Perm Staff ppm	12.000		3.000	15.000
Temp Staff ppm	12.000	8.000		20.000
Perm Staff Cost kEUR	62.500		9.000	71.500
Temp Staff Cost kEUR	62.500	46.875		109.375
Travels kEUR	1.300	0.867		2.167
Consumables kEUR				
Overheads kEUR	25.260	9.548	1.800	36.608
Total Manpower ppm	**24.000**	**8.000**	**3.000**	**35.000**
Total Cost kEUR	**151.560**	**57.290**	**10.800**	**219.650**

ALU-FR now RFWU-Bonn