ILC Software

Overview & News on Core Tools

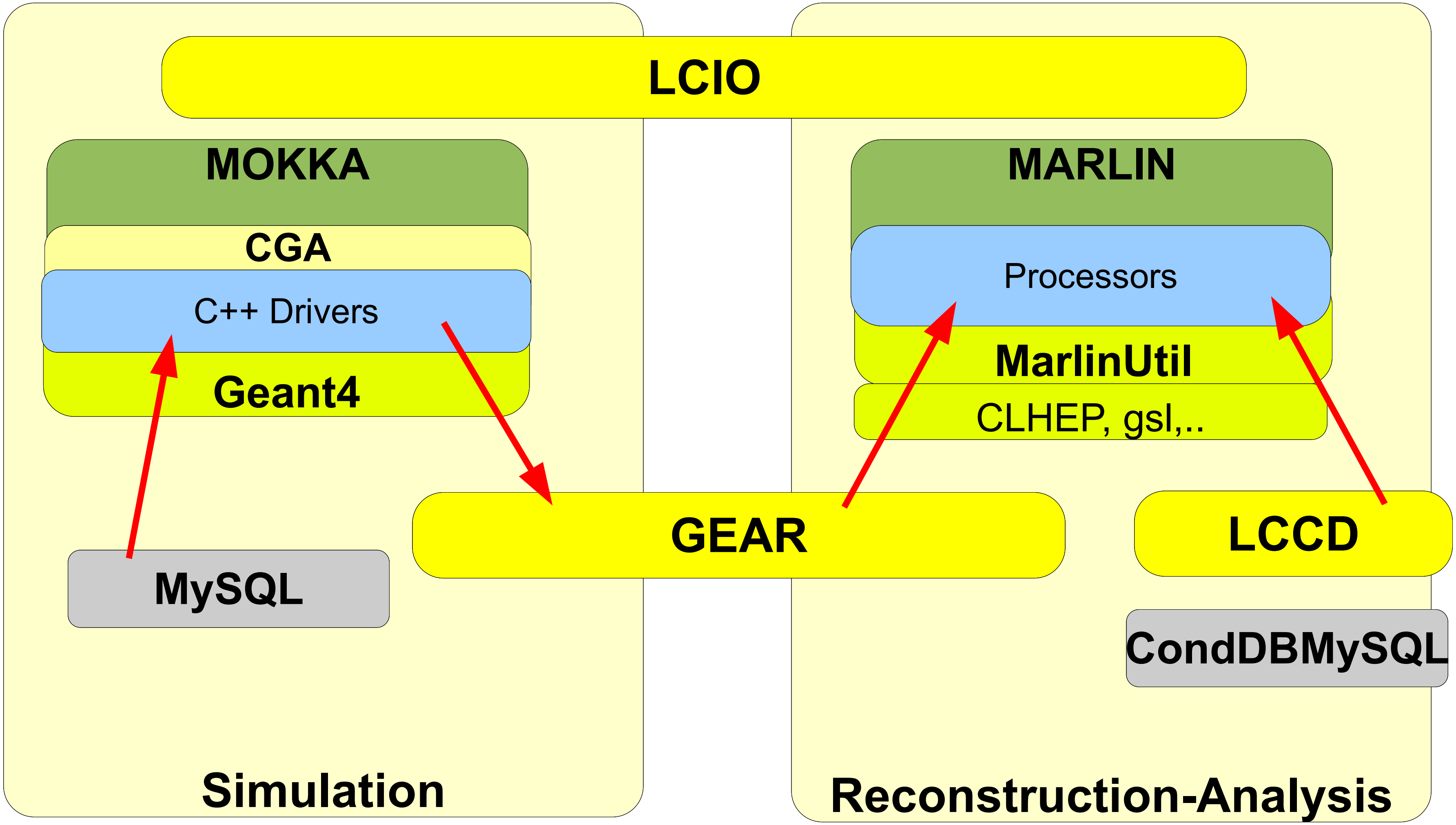
Frank Gaede
DESY

CALICE Meeting – Software session
DESY, February 12, 2007

Outline

- overview core software tools
  - LCIO
  - Marlin
  - LCCD
  - GEAR
- new developments in core tools (since Valencia):
  - Marlin (v00-09-06)
  - LCIO (v01-08)
  - ilcsoft-install

ILC-LDC software framework



all tools are also used in testbeam programs

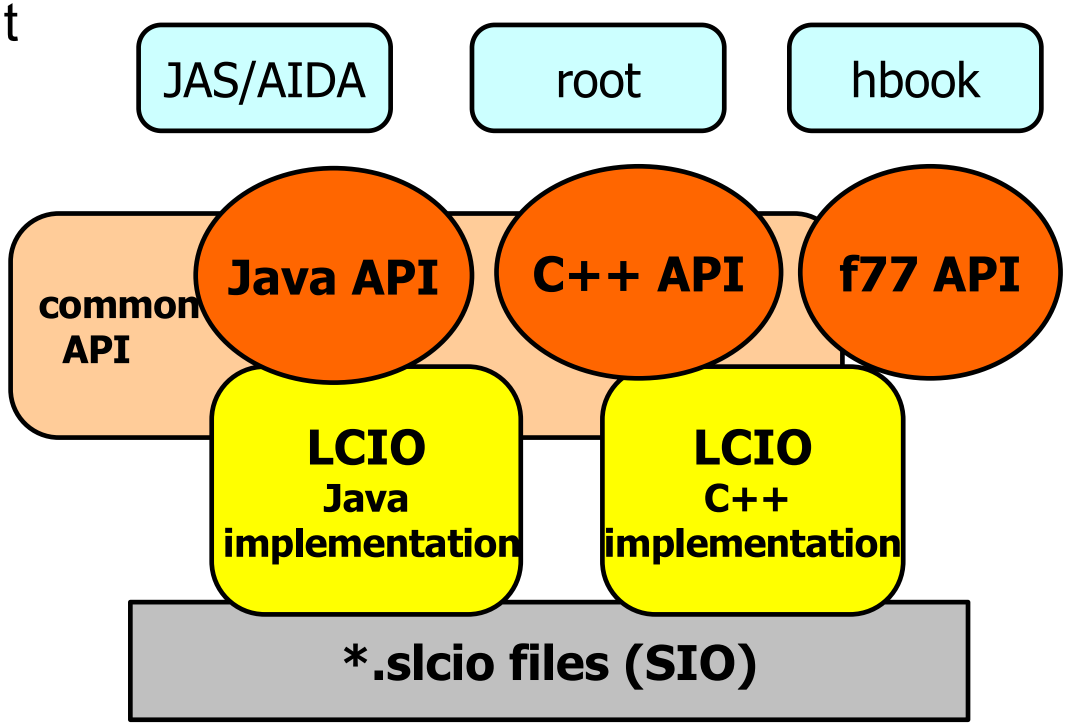
LCIO overview

- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

simple & lightweight

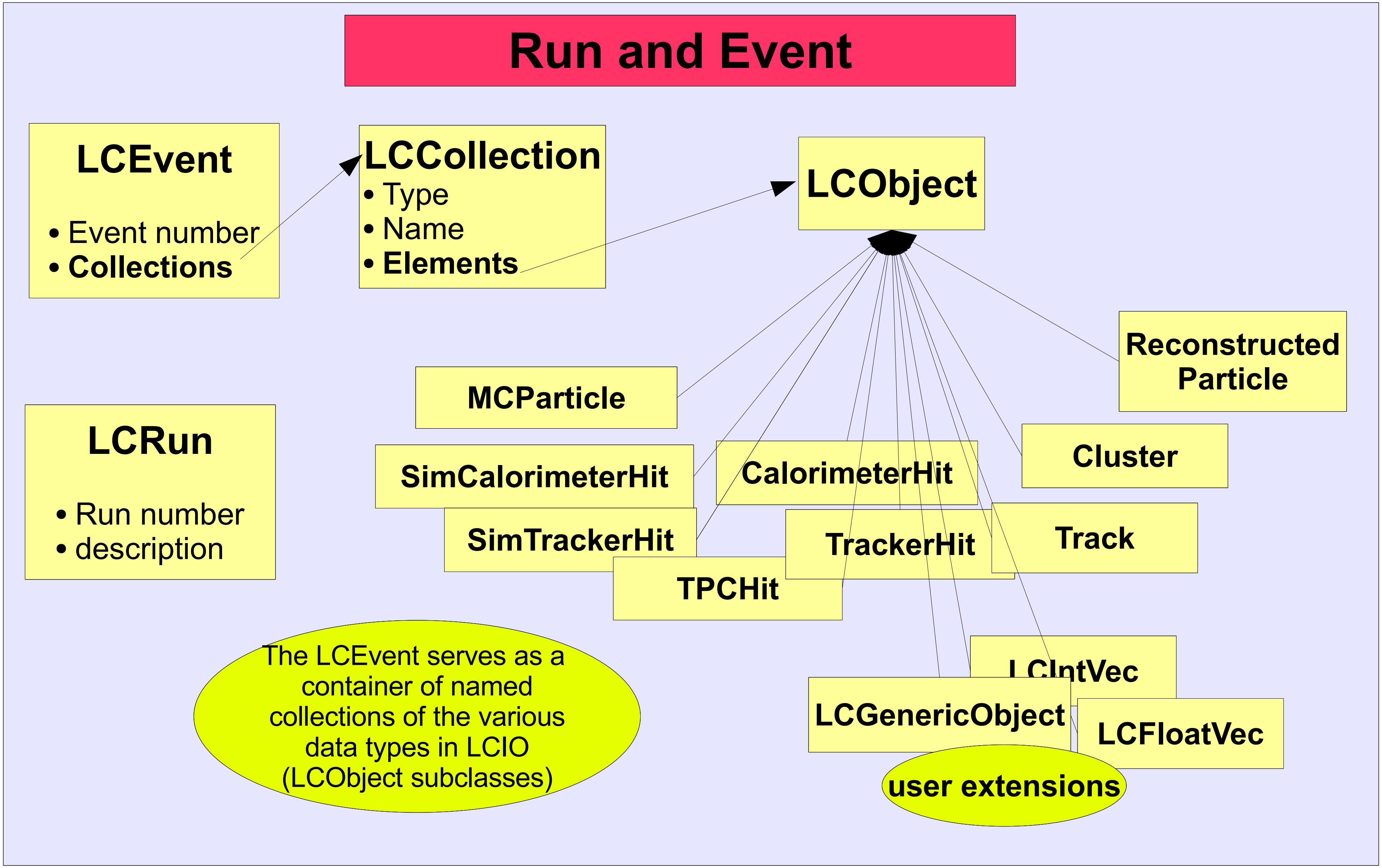
now de facto standard persistency & datamodel for ILC software

SW-Architecture



current release: **v01-08-01**

LCIO Event Data Model I



Run and Event

LCEvent

- Event number
- **Collections**

LCCollection

- Type
- Name
- **Elements**

LCObject

MCParticle

SimCalorimeterHit

SimTrackerHit

TPCHit

CalorimeterHit

TrackerHit

Track

Cluster

Reconstructed Particle

LCIntVec

LCGenericObject

LCFloatVec

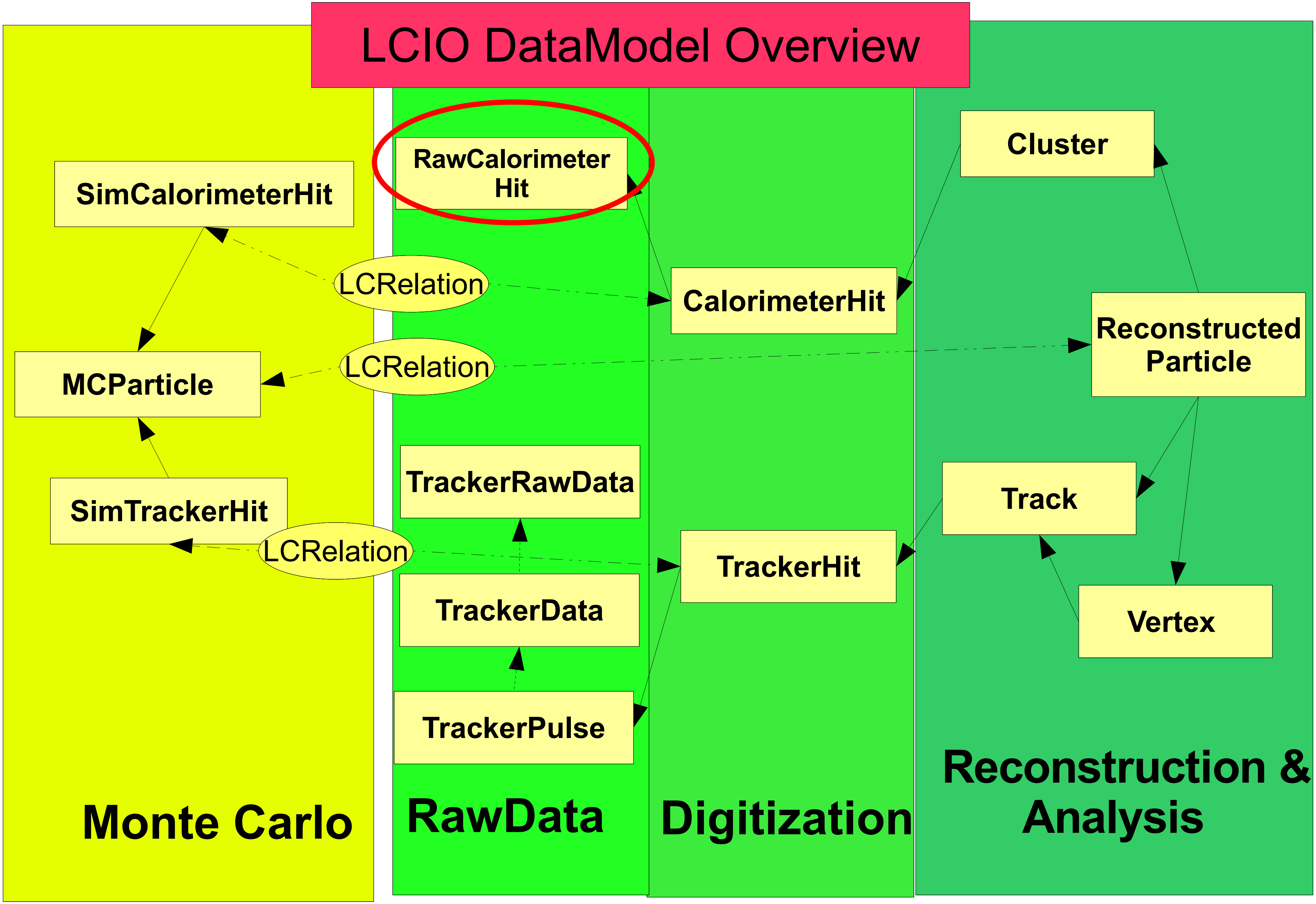
user extensions

LCRun

- Run number
- description

The LCEvent serves as a container of named collections of the various data types in LCIO (LCObject subclasses)

LCIO Event Data Model II



LCIO Event Data Model III

- the LCIO event data model is fairly complete and flexible
- however it is adapted and extended as needed by the community
  - maintaining downward compatibility
  - with international discussion and agreement
- examples:
- introduction of a new **Vertex** class in LCIO
- new raw data classes for prototypes
  - TPC/VXD: TrackerRawData, TrackerData, TrackerPulse
- RawCalorimeterHit for calorimeter prototypes:
- -> possibly need additional class for calibrated hit (float amplitude, error of enrgy...?)

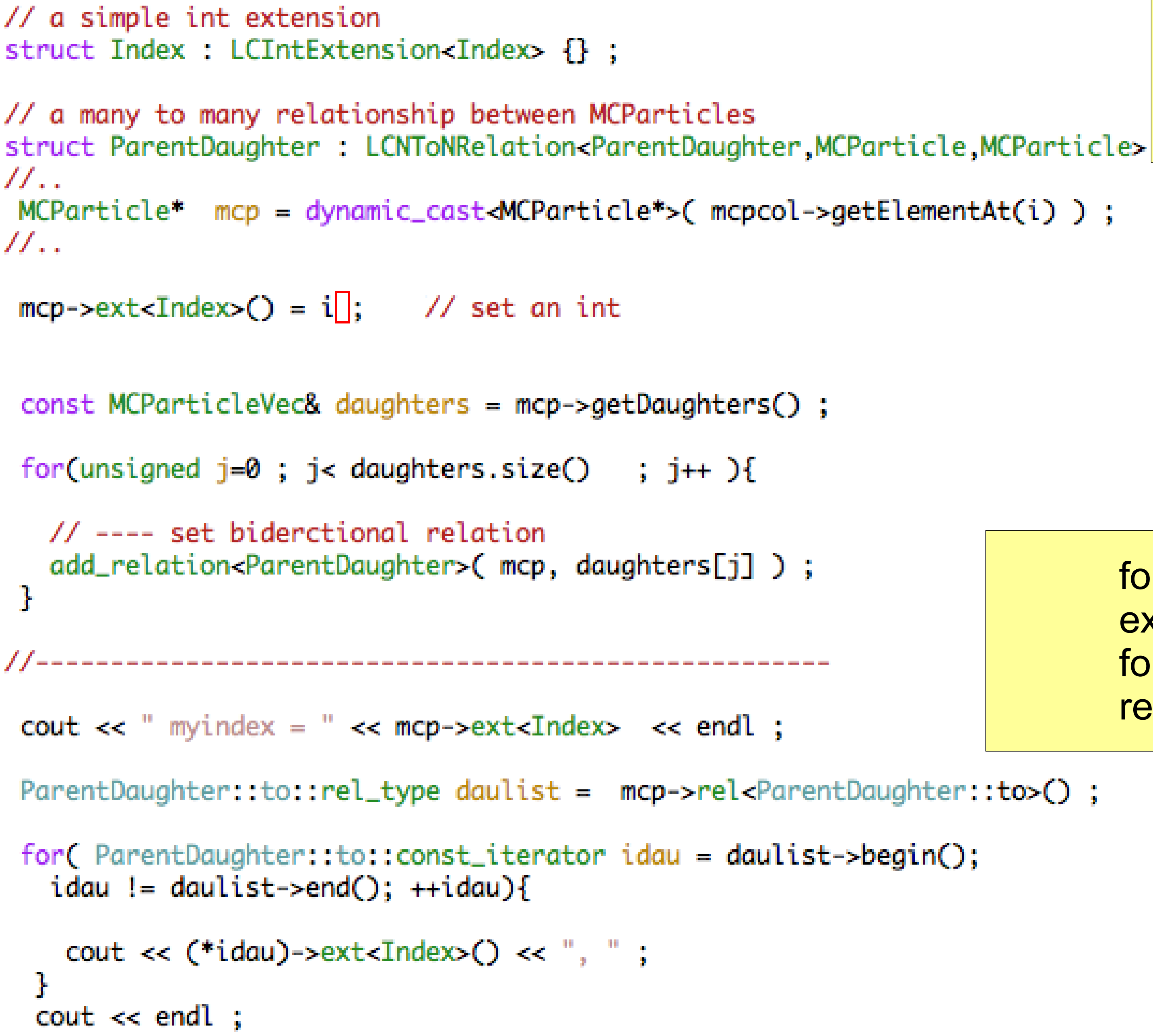
LCIO status

- release v01-08:
  - introduced C++ runtime (user) extensions and relations
    - see LCRTRelation and lcrtrelation.cc for documentation and examples
  - new Vertex class in LCIO
    - see: http://forum.linearcollider.org/index.php?t=getfile&id=32
  - new Java lcio command line tool - for detailed documentation - see:
    - http://confluence.slac.stanford.edu/display/ilc/LCIO+Command+Line+Tool
    - **provides event overlay functionality**
  - modified function UTIL:LCTOOLS:dumpEvent() for a more readable format
    - used e.g. in anajob.cc
  - release v01-08-01 (07.02.07)
    - bug fixes

LCIO runtime extensions

- 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

LCIO runtime extensions



```
// 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 ;
```

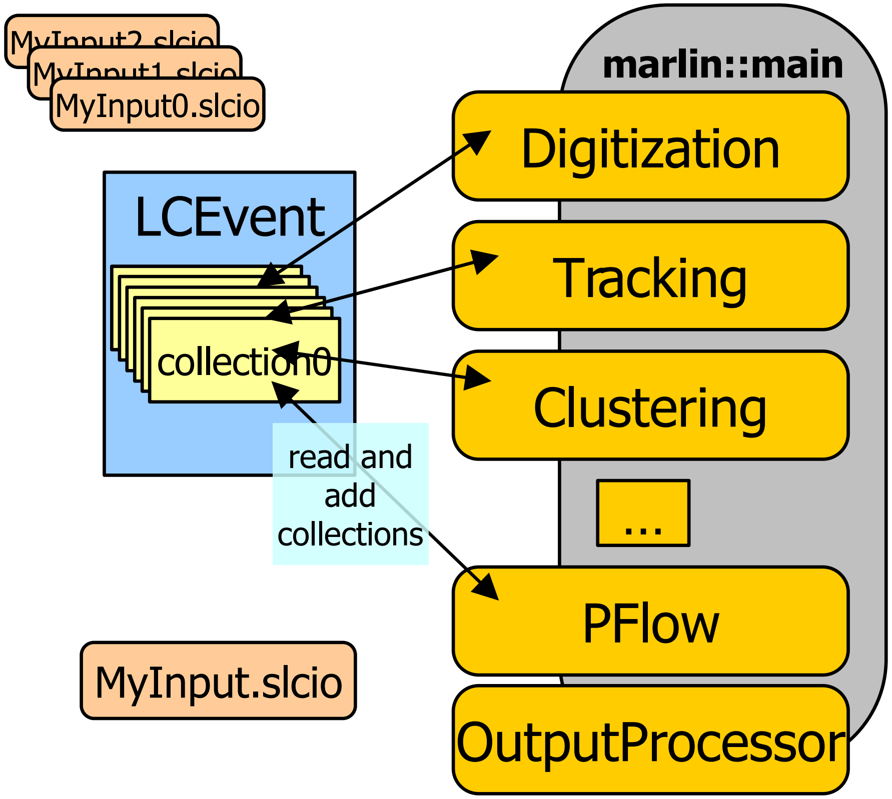
extensions and relations identified through a tagging class T

for extensions use ext<T>()
for relations use rel<T>

Marlin

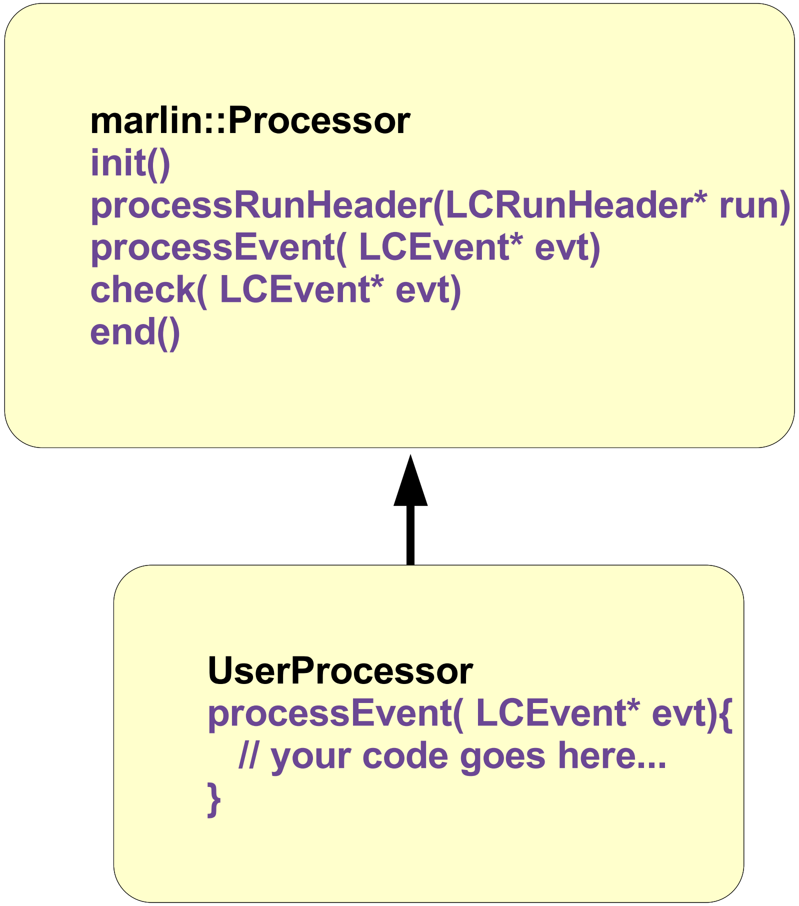
Modular **A**nalysis & **R**econstruction for the **L I N**ear Collider

- modular C++ **application framework** for the analysis and reconstruction of LCIO data
- uses LCIO as transient data model
- software modules called Processors
- provides main program !
- provides simple user steering:
  - program flow (active processors)
  - user defined variables
    - per processor and global
  - input/output files
  - **Plug&Play** of processors



Marlin Processor

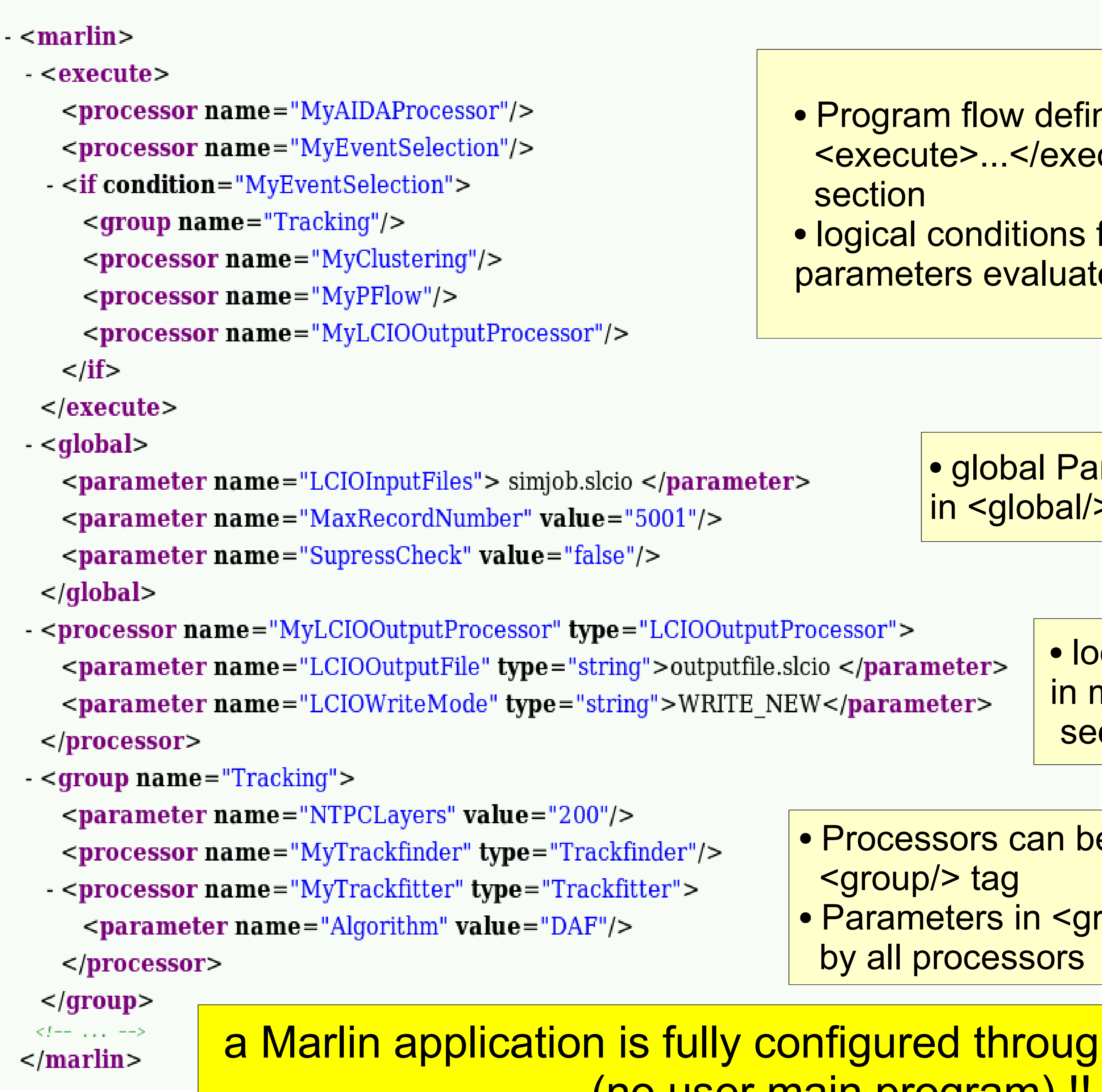
- provides main user callbacks
- has own set of input parameters
  - int, float, string (single and arrays)
  - parameter description
- naturally modularizes the application
- order of processors is defined via steering file:
  - easy to exchange one or several modules w/o recompiling
  - can run the same processor with different parameter set in one job
- processor task can be as simple as creating one histogram or as complex as track finding and fitting in the central tracker



```
marlin::Processor
init()
processRunHeader(LCRunHeader* run)
processEvent( LCEvent* evt)
check( LCEvent* evt)
end()
```

```
UserProcessor
processEvent( LCEvent* evt){
  // your code goes here...
}
```

Marlin – XML steering files



```
- <marlin>
  - <execute>
      <processor name="MyAIDAProcessor"/>
      <processor name="MyEventSelection"/>
    - <if condition="MyEventSelection">
        <group name="Tracking"/>
        <processor name="MyClustering"/>
        <processor name="MyPFlow"/>
        <processor name="MyLCIOOutputProcessor"/>
      </if>
    </execute>
  - <global>
      <parameter name="LCIOInputFiles"> simjob.slcio </parameter>
      <parameter name="MaxRecordNumber" value="5001"/>
      <parameter name="SupressCheck" value="false"/>
    </global>
  - <processor name="MyLCIOOutputProcessor" type="LCIOOutputProcessor">
      <parameter name="LCIOOutputFile" type="string">outputfile.slcio </parameter>
      <parameter name="LCIOWriteMode" type="string">WRITE_NEW</parameter>
    </processor>
  - <group name="Tracking">
      <parameter name="NTPCLayers" value="200"/>
      <processor name="MyTrackfinder" type="Trackfinder"/>
    - <processor name="MyTrackfitter" type="Trackfitter">
        <parameter name="Algorithm" value="DAF"/>
      </processor>
    </group>
    <!-- ... -->
  </marlin>
```

- Program flow defined in <execute>...</execute> section
- logical conditions from parameters evaluated at runtime

- global Parameters defined in <global/> section

- local Parameters defined in mandatory <parameter/> section

- Processors can be enclosed by <group/> tag
- Parameters in <group/> joined by all processors

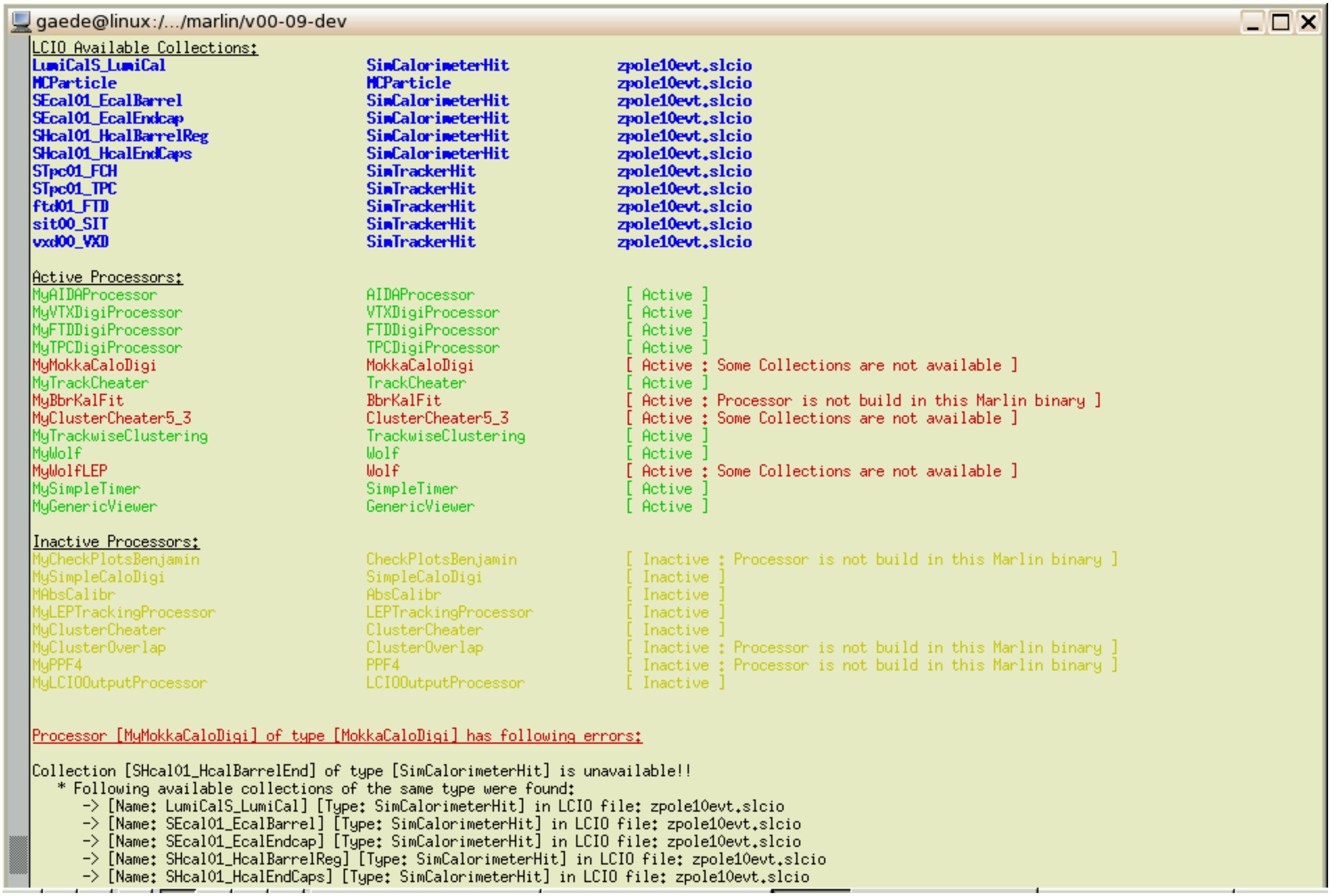
a Marlin application is fully configured through the steering files (no user main program) !!

Marlin Status

- current version: v00-09-06
  - released before Christmas 06
- release notes:
  - first release of MarlinGUI that helps to create/modify xml steering files interactively -> see $MARLIN/gui/README for details
  - new methods Processor::registerInput/OutputCollections() for checking consistency
    - -> user need to uses those in their processors
  - new feature 'Marlin -c steer.xml' checks steering file for consistency (names and types of LCIO collections (author J.Engels)
  - new feature 'Marlin -o old.steer steer.xml '
    - convert old (deprecated!) steering file to xml steering file
  - (you can also use the GUI to read in old steering files)
  - switched to latest version of tinyxml: 2.5.2

consistency check of steering file

- user complaint:
  - marlin steering files are somewhat clumsy to edit
  - -> implement new feature to check consistency of steering files: Marlin -c steer.xml

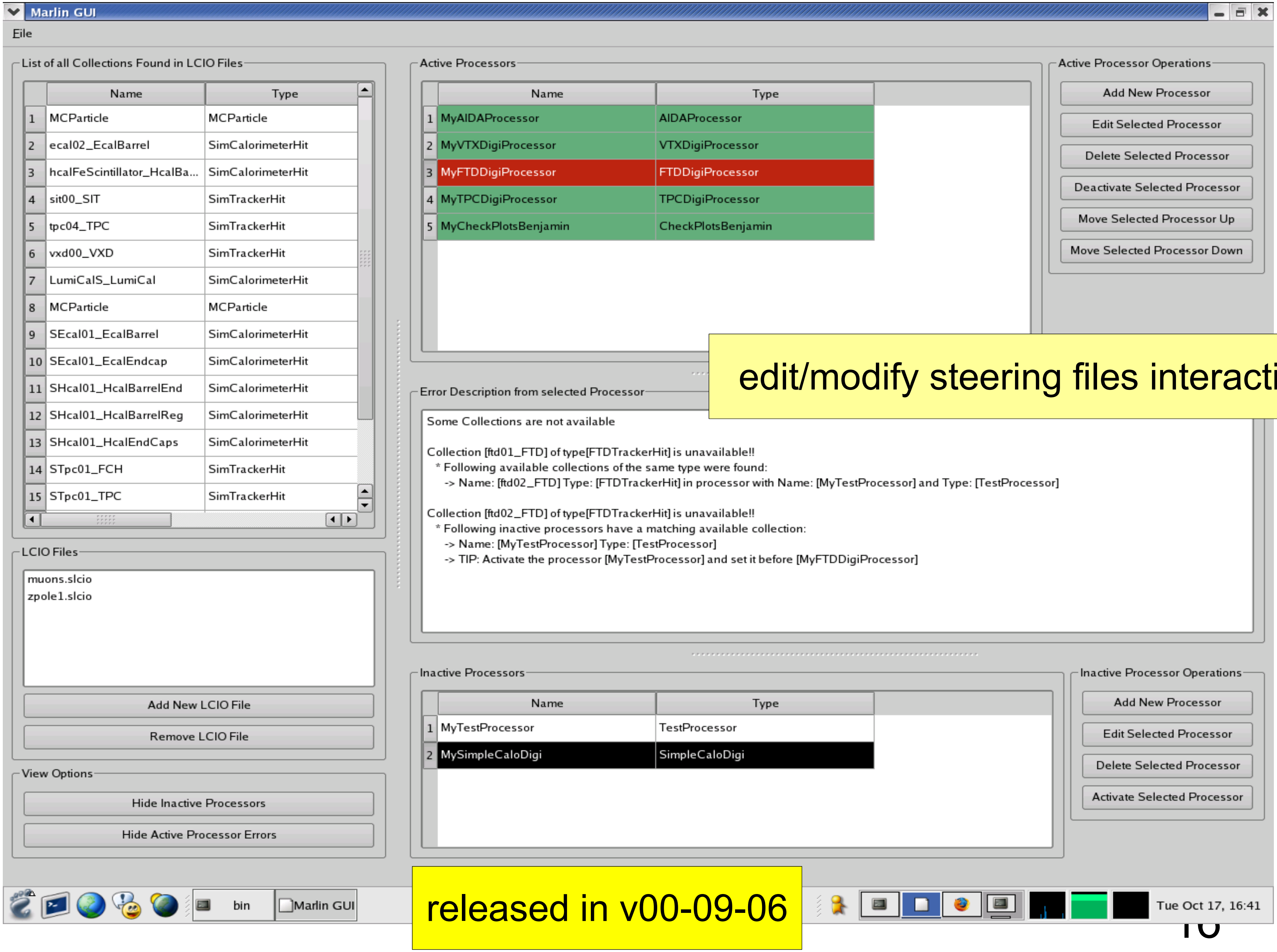


```
LCIO Available Collections:
LumiCalS_LumiCal                        SimCalorimeterHit             zpole10evt.slcio
MCParticle                              MCParticle                    zpole10evt.slcio
SEcal01_EcalBarrel                      SimCalorimeterHit             zpole10evt.slcio
SEcal01_EcalEndcap                      SimCalorimeterHit             zpole10evt.slcio
SHcal01_HcalBarrelReg                   SimCalorimeterHit             zpole10evt.slcio
SHcal01_HcalEndCaps                     SimCalorimeterHit             zpole10evt.slcio
STpc01_FCH                              SimTrackerHit                 zpole10evt.slcio
STpc01_TPC                              SimTrackerHit                 zpole10evt.slcio
ftd01_FTD                               SimTrackerHit                 zpole10evt.slcio
sit00_SIT                               SimTrackerHit                 zpole10evt.slcio
vxd00_VXD                               SimTrackerHit                 zpole10evt.slcio

Active Processors:
MyAIDAProcessor                         AIDAProcessor                  [ Active ]
MyVTXDigiProcessor                      VTXDigiProcessor               [ Active ]
MyFTDDigiProcessor                      FTDDigiProcessor               [ Active ]
MyTPCDigiProcessor                      TPCDigiProcessor               [ Active ]
MyMokkaCaloDigi                         MokkaCaloDigi                  [ Active : Some Collections are not available ]
MyTrackCheater                          TrackCheater                   [ Active ]
MyBbrKalFit                             BbrKalFit                      [ Active : Processor is not build in this Marlin binary ]
MyClusterCheater5_3                     ClusterCheater5_3              [ Active : Some Collections are not available ]
MyTrackwiseClustering                   TrackwiseClustering            [ Active ]
MyWolf                                  Wolf                           [ Active ]
MyWolfLEP                               Wolf                           [ Active : Some Collections are not available ]
MySimpleTimer                           SimpleTimer                    [ Active ]
MyGenericViewer                         GenericViewer                  [ Active ]

Inactive Processors:
MyCheckPlotsBenjamin                    CheckPlotsBenjamin             [ Inactive : Processor is not build in this Marlin binary ]
MySimpleCaloDigi                        SimpleCaloDigi                 [ Inactive ]
MAbsCalibr                              AbsCalibr                      [ Inactive ]
MyLEPTrackingProcessor                  LEPTrackingProcessor           [ Inactive ]
MyClusterCheater                        ClusterCheater                 [ Inactive ]
MyClusterOverlap                        ClusterOverlap                 [ Inactive : Processor is not build in this Marlin binary ]
MyPPF4                                  PPF4                           [ Inactive : Processor is not build in this Marlin binary ]
MyLCIOOutputProcessor                   LCIOOutputProcessor            [ Inactive ]


Processor [MyMokkaCaloDigi] of type [MokkaCaloDigi] has following errors:

Collection [SHcal01_HcalBarrelEnd] of type [SimCalorimeterHit] is unavailable!!
   * Following available collections of the same type were found:
      -> [Name: LumiCalS_LumiCal] [Type: SimCalorimeterHit] in LCIO file: zpole10evt.slcio
      -> [Name: SEcal01_EcalBarrel] [Type: SimCalorimeterHit] in LCIO file: zpole10evt.slcio
      -> [Name: SEcal01_EcalEndcap] [Type: SimCalorimeterHit] in LCIO file: zpole10evt.slcio
      -> [Name: SHcal01_HcalBarrelReg] [Type: SimCalorimeterHit] in LCIO file: zpole10evt.slcio
      -> [Name: SHcal01_HcalEndCaps] [Type: SimCalorimeterHit] in LCIO file: zpole10evt.slcio
```

J.Engels (EUDET)

released in v00-09-06

new development: MarlinGUI


Marlin GUI
File
Add New Processor
Edit Selected Processor
Delete Selected Processor
Deactivate Selected Processor
Move Selected Processor Up
Move Selected Processor Down
Add New LCIO File
Remove LCIO File
Hide Inactive Processors
Hide Active Processor Errors
Add New Processor
Edit Selected Processor
Delete Selected Processor
Activate Selected Processor
bin
Marlin GUI
Tue Oct 17, 16:41

List of all Collections Found in LCIO Files

	Name	Type
1	MCParticle	MCParticle
2	ecal02_EcalBarrel	SimCalorimeterHit
3	hcalFeScintillator_HcalBa...	SimCalorimeterHit
4	sit00_SIT	SimTrackerHit
5	tpc04_TPC	SimTrackerHit
6	vxd00_VXD	SimTrackerHit
7	LumiCalS_LumiCal	SimCalorimeterHit
8	MCParticle	MCParticle
9	SEcal01_EcalBarrel	SimCalorimeterHit
10	SEcal01_EcalEndcap	SimCalorimeterHit
11	SHcal01_HcalBarrelEnd	SimCalorimeterHit
12	SHcal01_HcalBarrelReg	SimCalorimeterHit
13	SHcal01_HcalEndCaps	SimCalorimeterHit
14	STpc01_FCH	SimTrackerHit
15	STpc01_TPC	SimTrackerHit

LCIO Files

muons.slcio
zpole1.slcio

View Options

Active Processors

	Name	Type
1	MyAIDAProcessor	AIDAProcessor
2	MyVTXDigiProcessor	VTXDigiProcessor
3	MyFTDDigiProcessor	FTDDigiProcessor
4	MyTPCDigiProcessor	TPCDigiProcessor
5	MyCheckPlotsBenjamin	CheckPlotsBenjamin

Active Processor Operations

Error Description from selected Processor

Some Collections are not available

Collection [ftd01_FTD] of type[FTDTrackerHit] is unavailable!!
 * Following available collections of the same type were found:
 -> Name: [ftd02_FTD] Type: [FTDTrackerHit] in processor with Name: [MyTestProcessor] and Type: [TestProcessor]

Collection [ftd02_FTD] of type[FTDTrackerHit] is unavailable!!
 * Following inactive processors have a matching available collection:
 -> Name: [MyTestProcessor] Type: [TestProcessor]
 -> TIP: Activate the processor [MyTestProcessor] and set it before [MyFTDDigiProcessor]

Inactive Processors

	Name	Type
1	MyTestProcessor	TestProcessor
2	MySimpleCaloDigi	SimpleCaloDigi

Inactive Processor Operations

edit/modify steering files interactively

released in v00-09-06

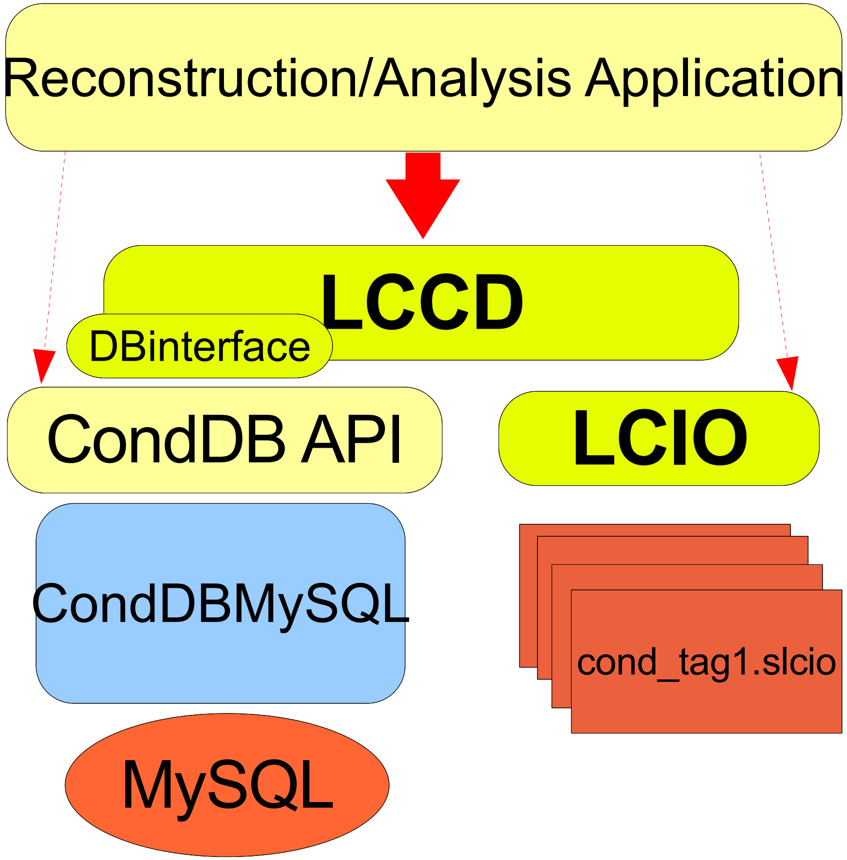
Marlin work in progress

- modified Makefiles/build procedure:
  - everything is now built in $MARLINWORKDIR
  - libs, bin, *.o files
  - $MARLINWORKDIR if not set $MARLIN is used (as before)
- experimental code to rewind the lcio data files, e.g. for calibration runs where you want to use the first N events for getting some initial calibration constants
  - needs more work and thought (input welcome)
- looking into making Marlin more portable
  - OSX, windows,...
  - investigating **cmake**

LCCD

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

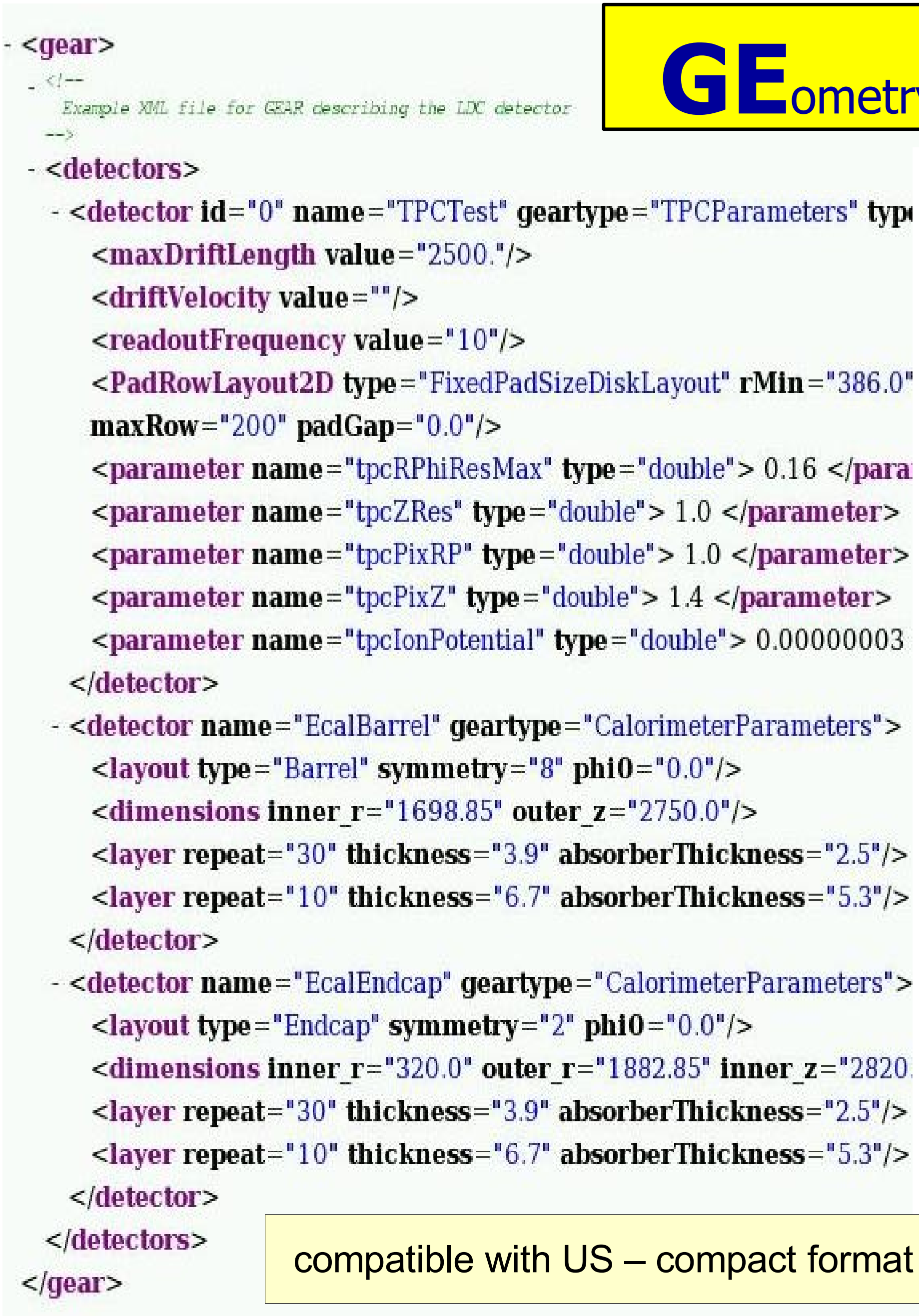
- Reading conditions data
  - from conditions database
  - from simple LCIO file
  - from LCIO data stream
  - from dedicated LCIO-DB file
- Writing conditions data
- tag conditions data
- Browse the conditions database
  - through creation of LCIO files
    - vertically (all versions for timestamp)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



LCCD is used by Calice and TPC groups for the conditions data of the ongoing testbeam studies

Gear

GEometry API for Reconstruction



```
- <gear>
  - <!--
      Example XML file for GEAR describing the LDC detector
    -->
  - <detectors>
    - <detector id="0" name="TPCTest" geartype="TPCParameters" typ
        <maxDriftLength value="2500."/>
        <driftVelocity value=""/>
        <readoutFrequency value="10"/>
        <PadRowLayout2D type="FixedPadSizeDiskLayout" rMin="386.0"
        maxRow="200" padGap="0.0"/>
        <parameter name="tpcRPhiResMax" type="double"> 0.16 </para
        <parameter name="tpcZRes" type="double"> 1.0 </parameter>
        <parameter name="tpcPixRP" type="double"> 1.0 </parameter>
        <parameter name="tpcPixZ" type="double"> 1.4 </parameter>
        <parameter name="tpcIonPotential" type="double"> 0.00000003
      </detector>
    - <detector name="EcalBarrel" geartype="CalorimeterParameters">
        <layout type="Barrel" symmetry="8" phi0="0.0"/>
        <dimensions inner_r="1698.85" outer_z="2750.0"/>
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    - <detector name="EcalEndcap" geartype="CalorimeterParameters">
        <layout type="Endcap" symmetry="2" phi0="0.0"/>
        <dimensions inner_r="320.0" outer_r="1882.85" inner_z="2820.
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    </detectors>
</gear>
```

compatible with US – compact format

- well defined geometry definition for reconstruction that
  - is flexible w.r.t different detector concepts
  - has high level information needed for reconstruction
  - provides access to material properties
- **abstract interface (a la LCIO)**
  - concrete implementation based on XML files
  - and Mokka-CGA

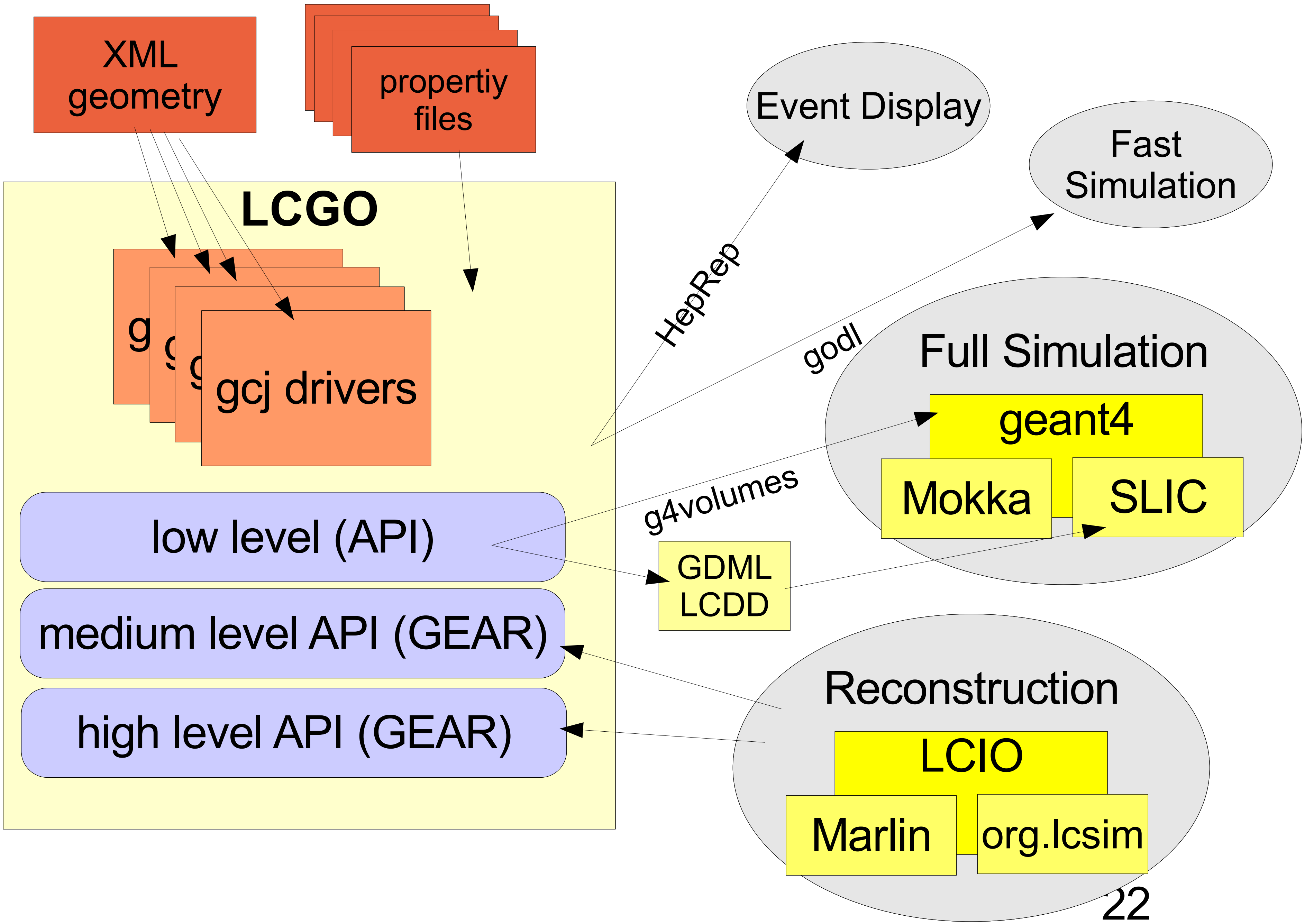
Gear status

- version v00-03
  - main detectors: TPC, Hcal, Ecal and VXT (new) interfaces defined and implemented
  - + free form user parameters for other detectors
  - description of TPC prototypes (rectangular pad plane)
  - description of calo prototype
- GearCGA (Mokka/geant4) - material properties
  - detailed material properties for every point (and distance)
- related work: MokkaGear
  - extract geometry information in Mokka drivers when detector is built in memory for simulation
  - use Gear to create XML files for reconstruction
  - need corresponding code in driver -> to be done for tbeams
  - -> have only one source of geometry information

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
    - **-> transition from GEAR should be very easy**
  - encourage/increase interoperability between systems
  - have no known principle short comings: “everything should be possible”

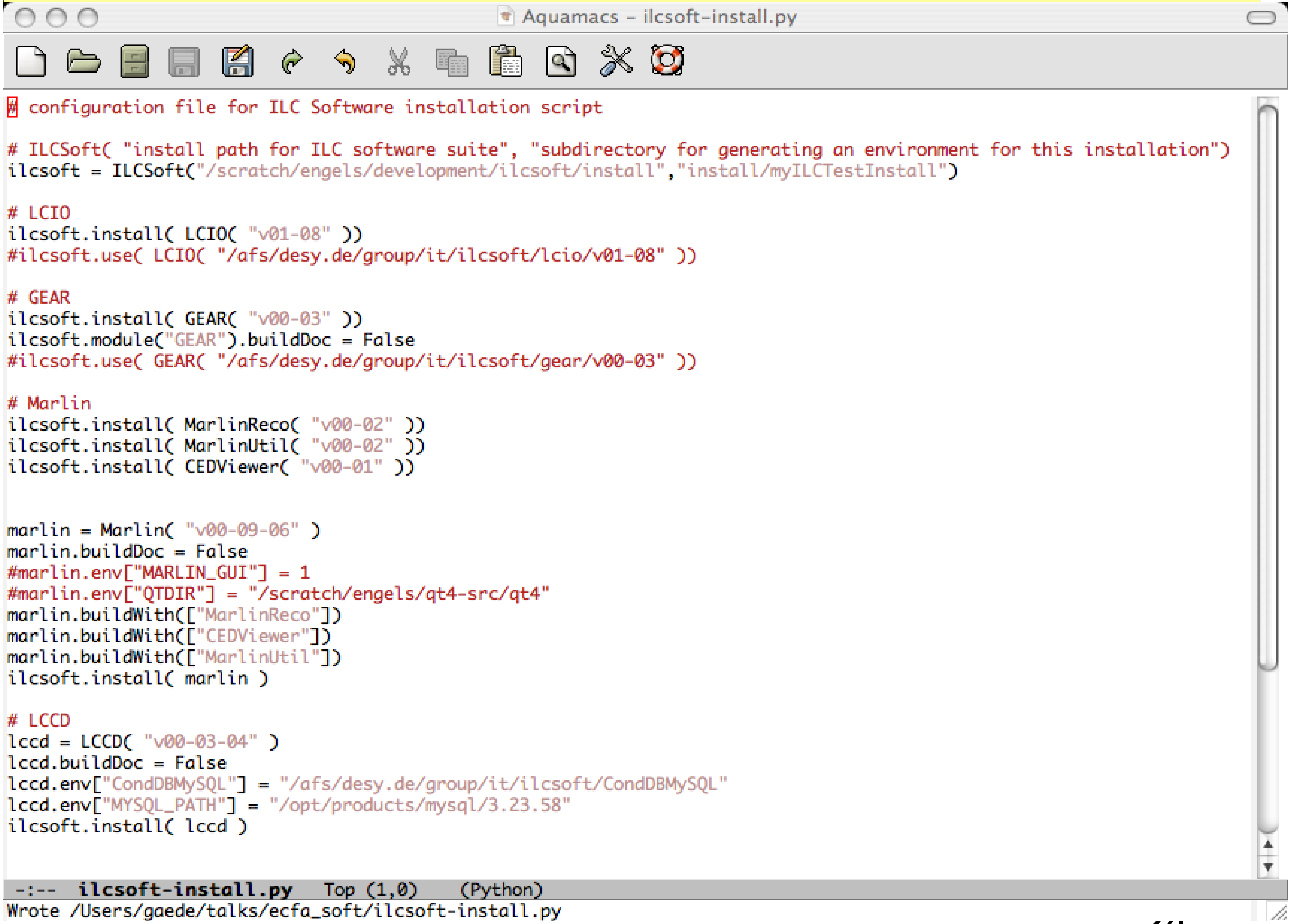
LCGO implementation prelim.



ilcsoft installation script

- ongoing work
- develop a python script that allows to install the full suite of core ILC software tools:
  - configure the versions of tools you want to install
  - check/configure the environment
  - take (optional) dependencies into account
  - allow to combine with existing packages,e.g.
    - CLHEP, gsl, cernlib,...
- **ideal: start script on empty disk – go to lunch – come back and run Marlin et al.**
- to be released soon
- work done by Jan Engels

ilcsoft installation script



```
# configuration file for ILC Software installation script

# ILCSoft( "install path for ILC software suite", "subdirectory for generating an environment for this installation")
ilcsoft = ILCSoft("/scratch/engels/development/ilcsoft/install","install/myILCTestInstall")

# LCIO
ilcsoft.install( LCIO( "v01-08" ))
#ilcsoft.use( LCIO( "/afs/desy.de/group/it/ilcsoft/lcio/v01-08" ))

# GEAR
ilcsoft.install( GEAR( "v00-03" ))
ilcsoft.module("GEAR").buildDoc = False
#ilcsoft.use( GEAR( "/afs/desy.de/group/it/ilcsoft/gear/v00-03" ))

# Marlin
ilcsoft.install( MarlinReco( "v00-02" ))
ilcsoft.install( MarlinUtil( "v00-02" ))
ilcsoft.install( CEDViewer( "v00-01" ))


marlin = Marlin( "v00-09-06" )
marlin.buildDoc = False
#marlin.env["MARLIN_GUI"] = 1
#marlin.env["QTDIR"] = "/scratch/engels/qt4-src/qt4"
marlin.buildWith(["MarlinReco"])
marlin.buildWith(["CEDViewer"])
marlin.buildWith(["MarlinUtil"])
ilcsoft.install( marlin )

# LCCD
lccd = LCCD( "v00-03-04" )
lccd.buildDoc = False
lccd.env["CondDBMySQL"] = "/afs/desy.de/group/it/ilcsoft/CondDBMySQL"
lccd.env["MYSQL_PATH"] = "/opt/products/mysql/3.23.58"
ilcsoft.install( lccd )
```

```
-:--  ilcsoft-install.py   Top (1,0)    (Python)
Wrote /Users/gaede/talks/ecfa_soft/ilcsoft-install.py
```

Summary

- core software framework exist:
  - LCIO – persistency & data model
  - Marlin – application framework
  - LCCD – conditions data toolkit
  - GEAR – geometry (for reconstruction&analysis)
- new developments:
  - Marlin - MarlinGUI
    - consistency and interactive creation of (xml) steering files
  - LCIO
    - user runtime extensions (and relations)
  - ilcinstall
    - full ilc core software installation tool (under development)
  - LCGO – new geometry framework (planned)

your input is needed for improvement of software and addition of new features !

Marlin core features

- fully configurable through steering files:
  - program flow
  - input parameters (processor based and global)
- self-documenting:
  - ./bin/Marlin -x

    prints example steering file with

    all available processors with their parameters and example/default values
- AIDA interface for histogramming
  - easy creation of histograms through absract interface
  - AIDAJNI/JAIDA, RAIDA (root based), ...
- configurable output
  - drop collections by name/type
- simple examples
  - user processor template, GNUmakefile,...
- easily extensible
  - makefiles 'automatically' include user packages with processors
- integration with: GEAR, LCCD, CED