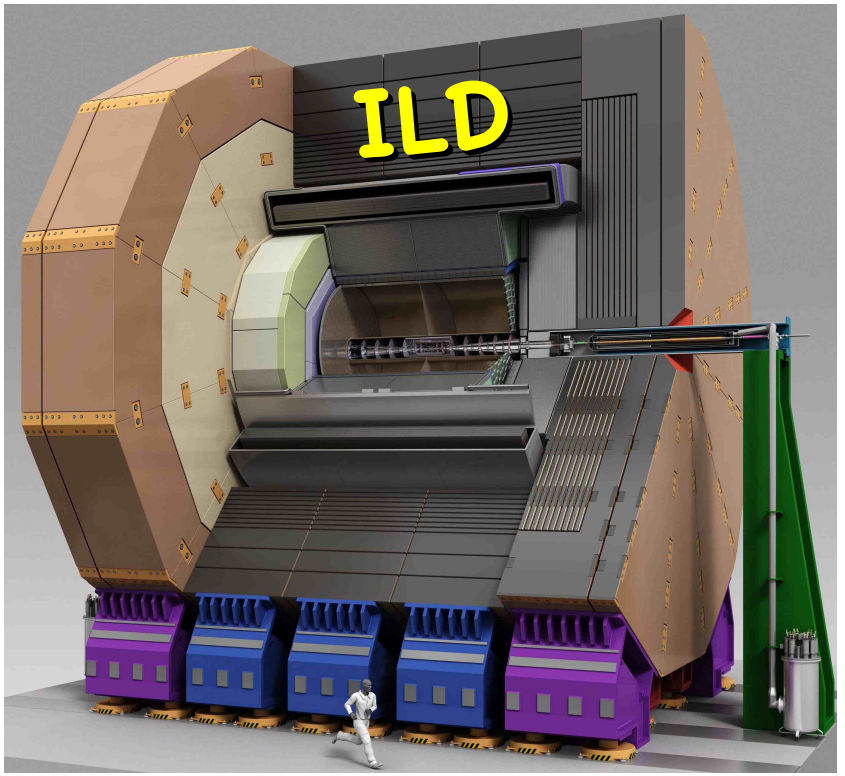
New developments in the iLCSoft framework

Frank Gaede, DESY

LCWS 2011
Granada, Spain, Sep 26-30, 2011

Outline

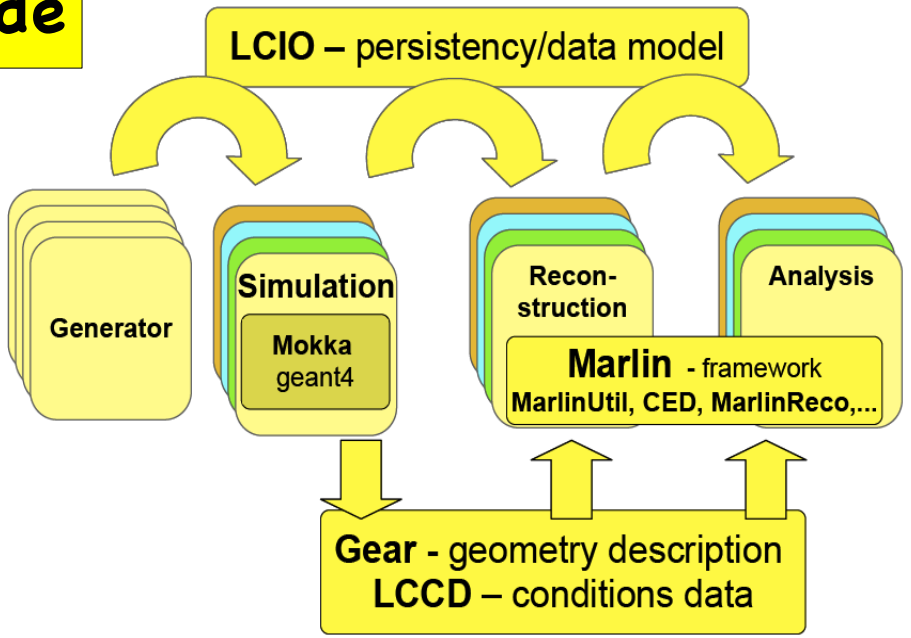
- overview of iLCSoft
- new in latest release: v01-12 (partly v01-11)
  - ilcutil
  - LCIO v02-00
  - Gear
  - Mokka
  - Marlin
  - CED
  - MarlinTrk
- Summary & Outlook



iLCSoft framework - Overview

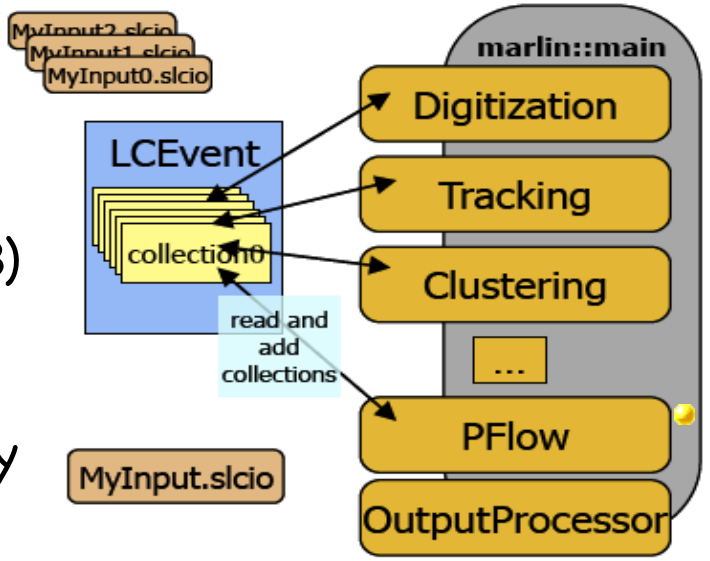
http://ilcsoft.desy.de

- **Mokka** (LLR)
  - geant4 simulation application
- **LCIO** (DESY/SLAC)
  - international standard for persistency format / event data model
- **Marlin**
  - core application framework for reconstruction & data analysis
- **GEAR** geometry package f. reconstruction
- **LCCD**
  - conditions
  - data toolkit (DB)
- **CED**
  - 3d event display



- complete framework used in Monte Carlo & 'real experiments':
  - **ILD detector concept** studies
  - **Calice** calo testbeam
  - **LC-TPC** testbeam
  - EUDET - **Pixel Telescope**

synergies between testbeam and global detector optimization



iLCSoft release v01-12

CED	v01-03
CEDViewer	v01-03
CLHEP	2.0.4.5
CondDBMySQL	ILC-0-9-5
Druid	1.8
Eutelescope	v00-06-03
KalTest	v01-02
KalDet	v01-02
LCFIVertex	v00-06
LCFI_MokkaBasedNets	v00-01
Marlin	v01-01
MarlinPandora	v00-06
MarlinReco	v00-30
MarlinTPC	v00-06
MarlinUtil	v01-04
Mokka	mokka-07-07
MokkaDBConfig	v03-02

Overlay	v00-11
PandoraPFANew	v00-07
QT	4.2.2
RAIDA	v01-06-01
StandardConfig	v03-00
cernlib	2006
dcap	1.9.5-5
gear	v01-00
gsl	1.14
lccd	v01-02
lcio	v02-00
mysql	5.0.45
root	5.28.00f
ilcutil	v00-02
MarlinTrk	**v01-00**
MarlinKinFit	**v00-01**
MarlinFastJet	**v00-02**

updated
new
this talk

development release targeted at getting the software into shape for the DBD

afs reference installations

- provide reference installations in afs for usage from anywhere on ScientificLinux and compatible platforms:

/afs/desy.de/project/ilcsoft/sw/_OS_/v01-12

_OS_:	**i386_gcc34_sl4**	# i386 CPU, 32 bit, gcc3.4, SL4 and compatible
	i386_gcc41_sl5	# i386 CPU, 32 bit, gcc4.1, SL5 and compatible
	x86_64_gcc41_sl5	# i686 CPU, 64 bit, gcc4.1, SL5 and compatible

  - you can directly run from these installations, .eg:

```
. /afs/desy.de/project/ilcsoft/sw/x86_64_gcc41_sl5/v01-12/init_ilcsoft.sh

  Marlin mysteer.xml
```

  - you can link your own libraries against these
- plan to have other OSs in the future (as needed)

Note: older releases (<v01-09) at /afs/desy.de/ilcsoft/

new package ILCUTIL

- (meta) package with utility packages:
- ILCSOFT_CMAKE_MODULES
  - (Previously known as CMakeModules)
  - cmake modules and utility macros.
  - most iLCSoft packages depend on this
- ILCTEST
  - C++ utility headers
  - cmake macros
  - for unit and integration tests of iLCSoft packages
- streamlog
  - logging classes for C++ applications
    - used to live in Marlin -> can now be used in other packages
- -> can be extended with other useful general purpose utilities

simplified use of CMake in iLCSoft

- iLCSoft uses CMake as build tool
- now greatly improved:
  - deprecated: BuildSetup.cmake, $PKG_HOME, -BUILD_WITH, LoadPackage.cmake and CheckDeps.cmake
- simply use: FindPackage()

Typical FIND_PACKAGE usage

```
- FIND_PACKAGE( Marlin REQUIRED)
- INCLUDE_DIRECTORIES( ${Marlin_INCLUDE_DIRS})
- LINK_LIBRARIES( ${Marlin_LIBRARIES})
```

-> also includes needed packages LCIO, Gear, RAIDA, etc !!

Version checking

```
- FIND_PACKAGE( ROOT 5.28 REQUIRED)
- FIND_PACKAGE( LCCD 1.2 EXACT )
```

Using COMPONENTS

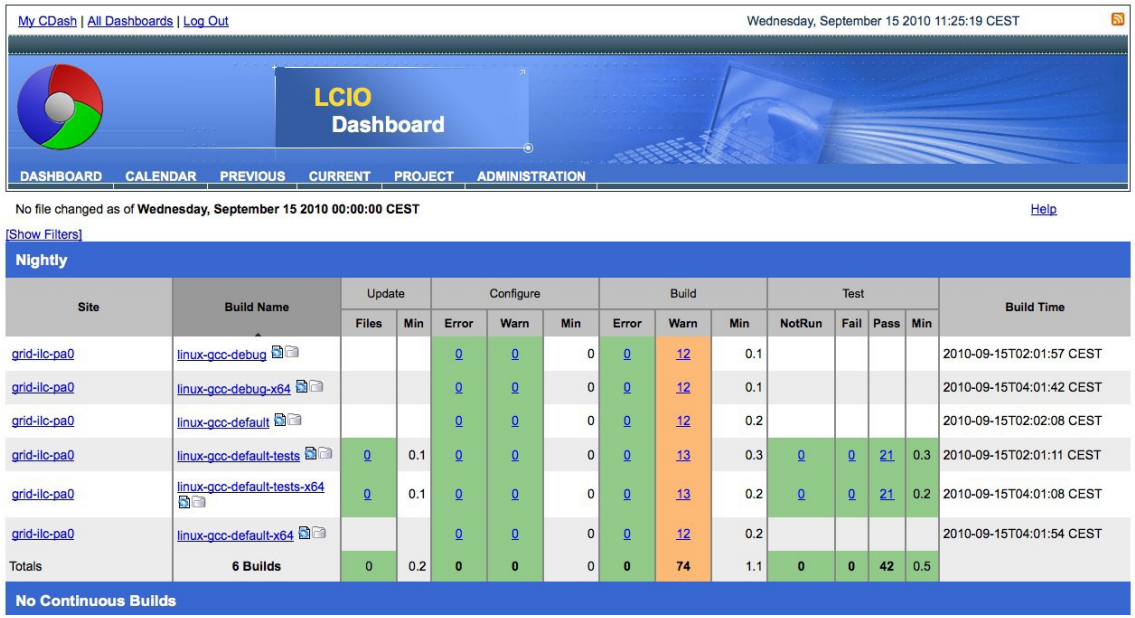
```
- FIND_PACKAGE( ROOT 5.28 REQUIRED COMPONENTS Gdml Geom XMLIO )
- LINK_LIBRARIES( ${ROOT_LIBRARIES} )
- LINK_LIBRARIES( ${ROOT_COMPONENT_LIBRARIES} )
- LINK_LIBRARIES( ${ROOT_GDML_LIBRARY} )
```

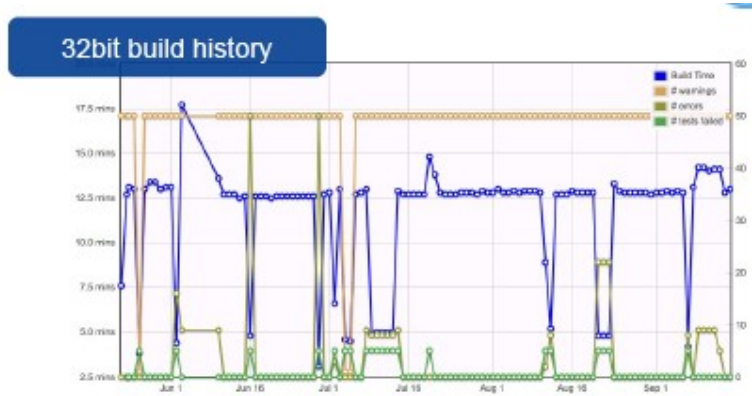
for details see: http://ilcagenda.linearcollider.org/getFile.py/access?contribId=1&resId=0&materialId=slides&confId=4950

ILCTest

- generic test system for iLCSoft:
  - unit tests
  - integration tests
  - physics test
- added some unit tests to most packages (run in Nightly Builds)
  - result browsable on dashboard
  - can be added to any iLCSoft package

```
// first line in your c++ source file
static ILCTest ilctest = ILCTest( "hello_world" );
...
ilctest.log( "hello world test" ); // a log message
...
If( x != 42 ){ ilctest.error("wrong answer!!") ; }
...
cout << ilctest.last_test_status() << endl; // prints "FAILED"
...
If( r > 3 ){ ilctest.fatal_error("this is a fatal error. program will quit now!") ; }
```





LCIO v02-00

- after LOI decided to have major new LCIO release "2.0"
- goal: improve usability of LCIO and address some short comings while being fully backward compatible
- planned/requested features:
- **simplify using LCIO with ROOT -> Done** (v01-12-01)
- **direct access to events -> Done** (v01-51)
- **improving the event data model -> Done** (v01-60, v02-00)
  - partial reading of events -> postponed
  - splitting of events over files -> postponed

- LCIO v02-00 has been released:
- svn co svn://svn.freehep.org/lcio/tags/v02-00

LCIO v02-00 - new features/extensions

- moved to SVN code repository
  http://java.freehep.org/svn/repos/lcio/list
- browse code changes online
- added method to count events
  - LCReader::getNumberOfEvents()
  - tool: $LCIO/bin/lcio_event_counter
- added definitions specific to ILD to UTIL/ILDConf.h
  - -> allows to encode: subdetector, side, layer, module, sensor in cellID0
- EDM extensions:
  - float[3] MCParticle::getSpin()
  - int[2] MCParticle::getColorFlow()
    - also written by Whizzard now
  - SimCalorimeterHit::getStepPosition(int i)
    - needed for SDHCAL digitization
  - Cluster::getEnergyError()
  - int (Sim)TrackerHit::getCellID0()
  - int (Sim)TrackerHit::getCellID1()
    - allows to encode details of the measurement module in the hits -> needed for tracking package

LCIO v2 Track & Trackstates

- lcio Track now has **multiple TrackStates**
- will store four canonical TSs:
  - AtIP, AtFirstHit, AtLastHit, AtCalo
- TS returned either by
  - identifier
  - or closest to given point
- mostly backward compatible

virtual	**~TrackState** () *Destructor.*
virtual int	**getLocation** () const =0 *The location of the track state.*
virtual float	**getD0** () const =0 *Impact paramter of the track in (r-phi).*
virtual float	**getPhi** () const =0 *Phi of the track at the reference point.*
virtual float	**getOmega** () const =0 *Omega is the signed curvature of the track in [1/mm].*
virtual float	**getZ0** () const =0 *Impact paramter of the track in (r-z).*
virtual float	**getTanLambda** () const =0 *Lambda is the dip angle of the track in r-z at the reference point.*
virtual const **FloatVec** &	**getCovMatrix** () const =0 *Covariance matrix of the track parameters.*
virtual const float *	**getReferencePoint** () const =0 *Reference point of the track parameters.*

virtual const **TrackStateVec** &	**getTrackStates** () const =0 *Returns track states associated to this track.*
virtual const **TrackState** *	**getClosestTrackState** (float x, float y, float z) const =0 *Returns track state closest to the given point.*
virtual const **TrackState** *	**getTrackState** (int location) const =0 *Returns track state for the given location - or NULL if not found.*
virtual const **TrackerHitVec** &	**getTrackerHits** () const =0 *Optionaly (check/set flag(LCIO::TRBIT_HITS)==1) return the hits that have been used to create this track.*

LCIOv2: 1d and 2d TrackerHits

- need new tracker hit classes to properly describe 1d and 2d measurements (pixels/TPC and **strips**)
- **TrackerHitPlanar**
  - x, y, z - 'space point'
  - u(theta, phi) , v(theta, phi) - measurement directions (spanning vectors in the plane)
  - du, dv - measurement errors
  - -> to be used for 1d and 2d (dv is strip length in 1d case)
- **TrackerHitCylindrical**
  - x, y, z - 'space point'
  - Xc, Yc - center of cylinder (parallel to z)
    - (cylinder radius: R = sqrt((x-x_c)^2 + (y-y_c)^2))
  - dphi, dz - measurement errors
  - -> to be used for 1d and 2d
- these also implement the TrackerHit interface (x,y,z, cov) for backward compatibility and code reusability (eg in event display)

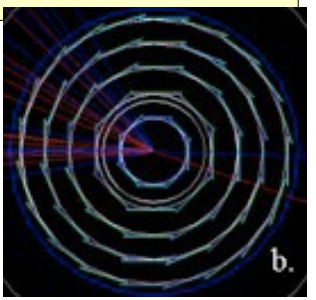
a ROOT dictionary for LCIO

- LCIO comes with a ROOT dictionary for all LCIO classes - with this one can: (since v01-12-01)
  - use LCIO classes in ROOT macros
  - write simple ROOT trees, e.g. std::vector<MCParticleImpl*>
  - use TTreeDraw for quick interactive analysis of LCObjects:

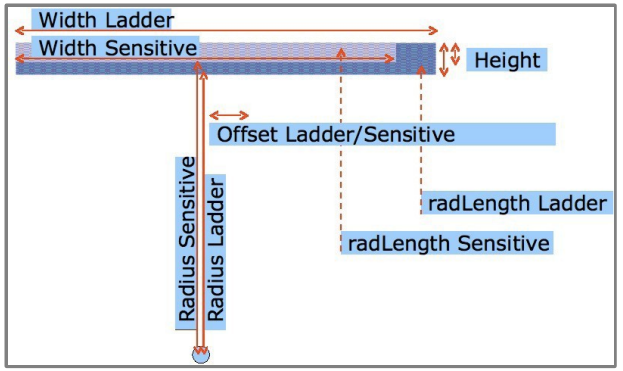
```
//---gamma conversions:
TCut isPhoton("MCParticlesSkimmed.getPDG()==22" ) ;
LCIO->Draw("MCParticlesSkimmed._endpoint[][0]:
            MCParticlesSkimmed._endpoint[][1]",isPhoton ) ;
```

  - write complete LCIO events in one ROOT branch
  - see: $LCIO/examples/cpp/rootDict/README for details & help
- -> we are interested in feedback from the users if this is a reasonable way to work with ROOT & LCIO
- other option: implement ROOT I/O for LCIO (.rlcio) !?

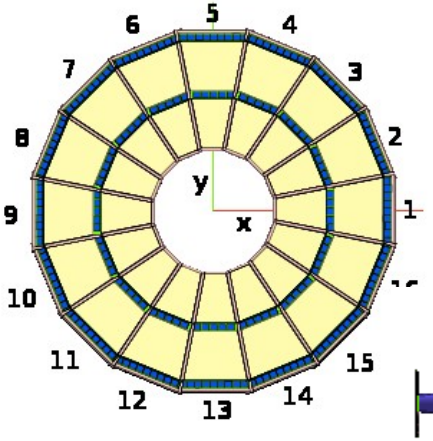
GEAR - new detector parameters

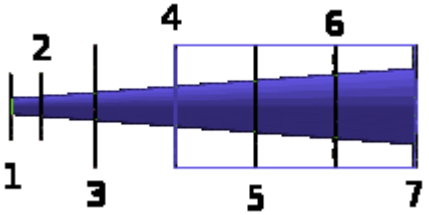


- added SIT and SET parameters - similar to VXD
  - describe (silicon) planar wafers along z-axis with phi-symmetry in placement and support material
  - renamed VXDParameters and VXDLayerLayout to ZPlanarParameters and ZPlanarLayerLayout
  - should be backward compatible through typedefs...



- added new FTDParameters and FTDLayerLayout (J.Duarte)
  - describe (silicon) disk detectors
  - made from petals
  - allow for tilting of petals (discouraged) or
  - staggering in z (preferred)





both needed to describe the now much more realistic and detailed Si-tracking simulation

Gear material description

- Gear interface to material from the first release (GearPointProperties/DistanceProperties) – implemented in
  - gearcga (geant4), geartgeo (ROOT)
- typically too exhaustive for reconstruction/fitting where one needs a simplified material description:
  - average material in simplified shapes (surfaces)
  - get overall material budget right

- -> introduced SimpleMaterial section in Gear parameters:
  - SimpleMaterial(Name, A, Z, density, radLength, interactionLength)
  - need to add code to Mokka drivers to write these materials

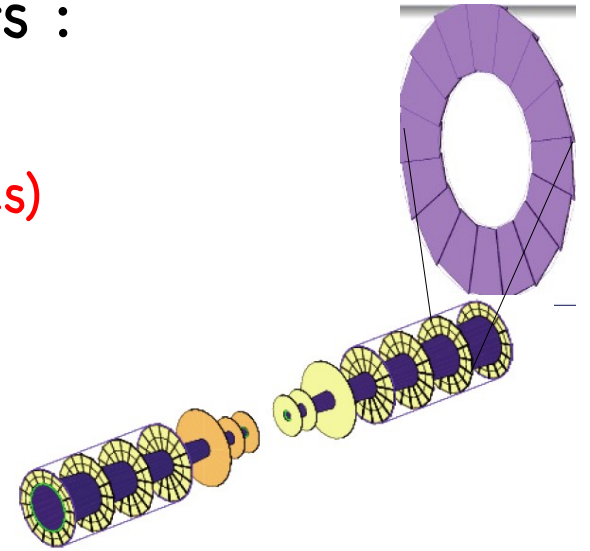
- in the midterm future, hope to improve geometry/material description based on results from AIDA WP2

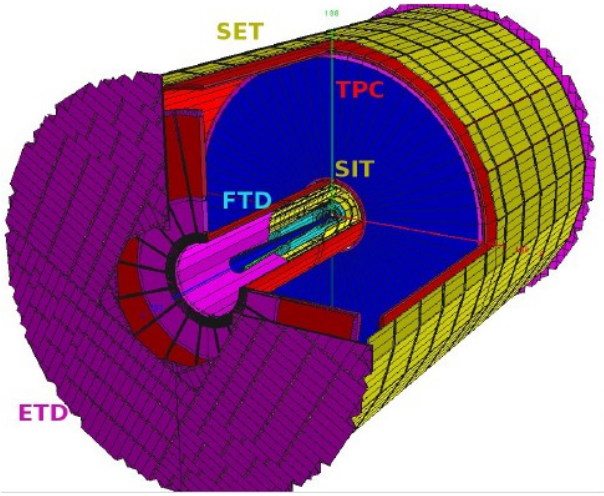
recent developments in Mokka

- major rewrite of some sub detector drivers :
  - SIT, SET, ETD - FTD - Muon
  - increased level of detail and realism (incl. services)
- made existing drivers more realistic:
  - TPC, AHCal, Ecal
- new drivers (technology options):
  - SDHCal, SciEcal
- added overall services and cables
- new models under development:

ILD_01_pre02	- AHCal	and	Si-Ecal
ILD_01_SDH_pre00	- SDHCal	and	Si-Ecal
ILD_01_SciW_pre00	- AHCal	and	Scintillator-Ecal

- next steps:
  - finalize and debug these models
  - adopt new Gear materials





Marlin v01-01

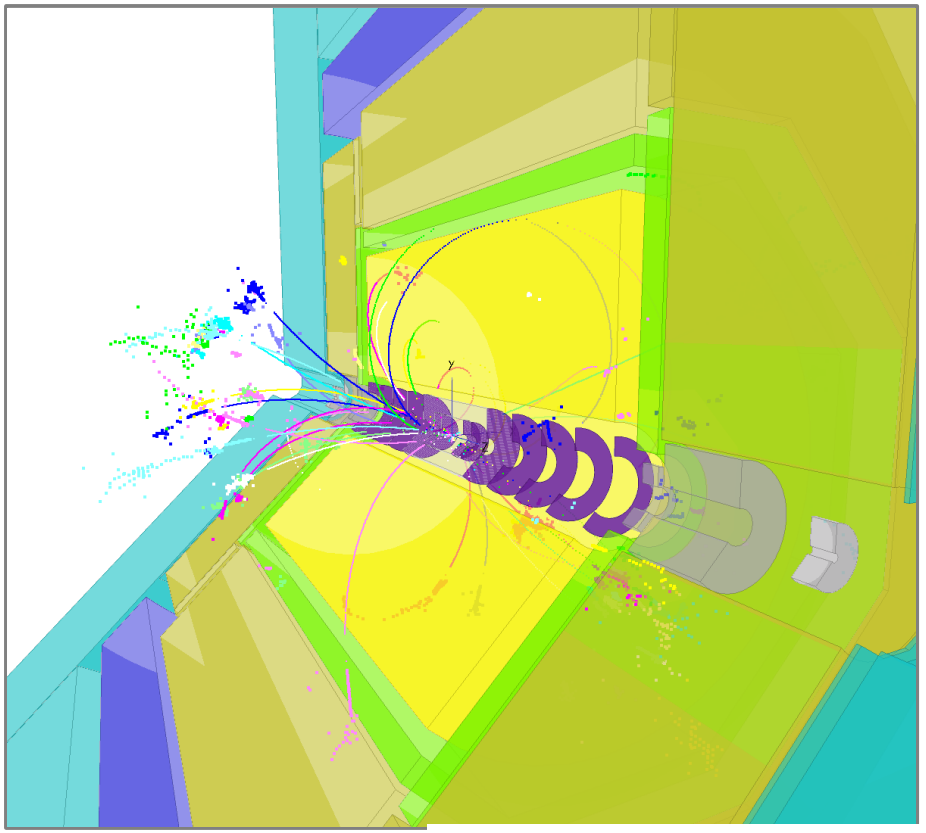
- added command line parameters
  - long standing feature request ...
- Marlin -h

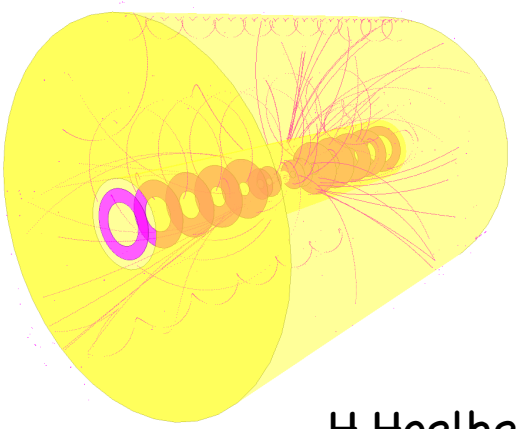
```
Dynamic command line options may be specified in order to overwrite individual steering file parameters, e.g.:
    Marlin --global.LCIOInputFiles="input1.slcio input2.slcio" --global.GearXMLFile=mydetector.xml
           --MyLCIOOutputProcessor.LCIOWriteMode=WRITE_APPEND --MyLCIOOutputProcessor.LCIOOutputFile=out.slcio steer.xml
```

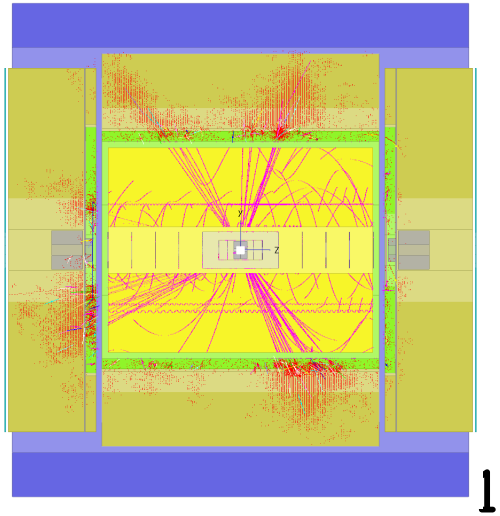
- can overwrite every parameter from steering file with --ProcName.ParameterName=Value
- useful for batch processing scripts, etc.

new features in CED event display

- many new features in CED, CEDViewer & MarlinCED :
  - added a New View with
    - 3d transparent surfaces
    - cut open detector
  - save display settings
  - turn on/off detector components
  - new projections:
    - r-phi ("F")
    - r-z ("S")
  - toggle view of axes
  - ...
  - detailed User Manual



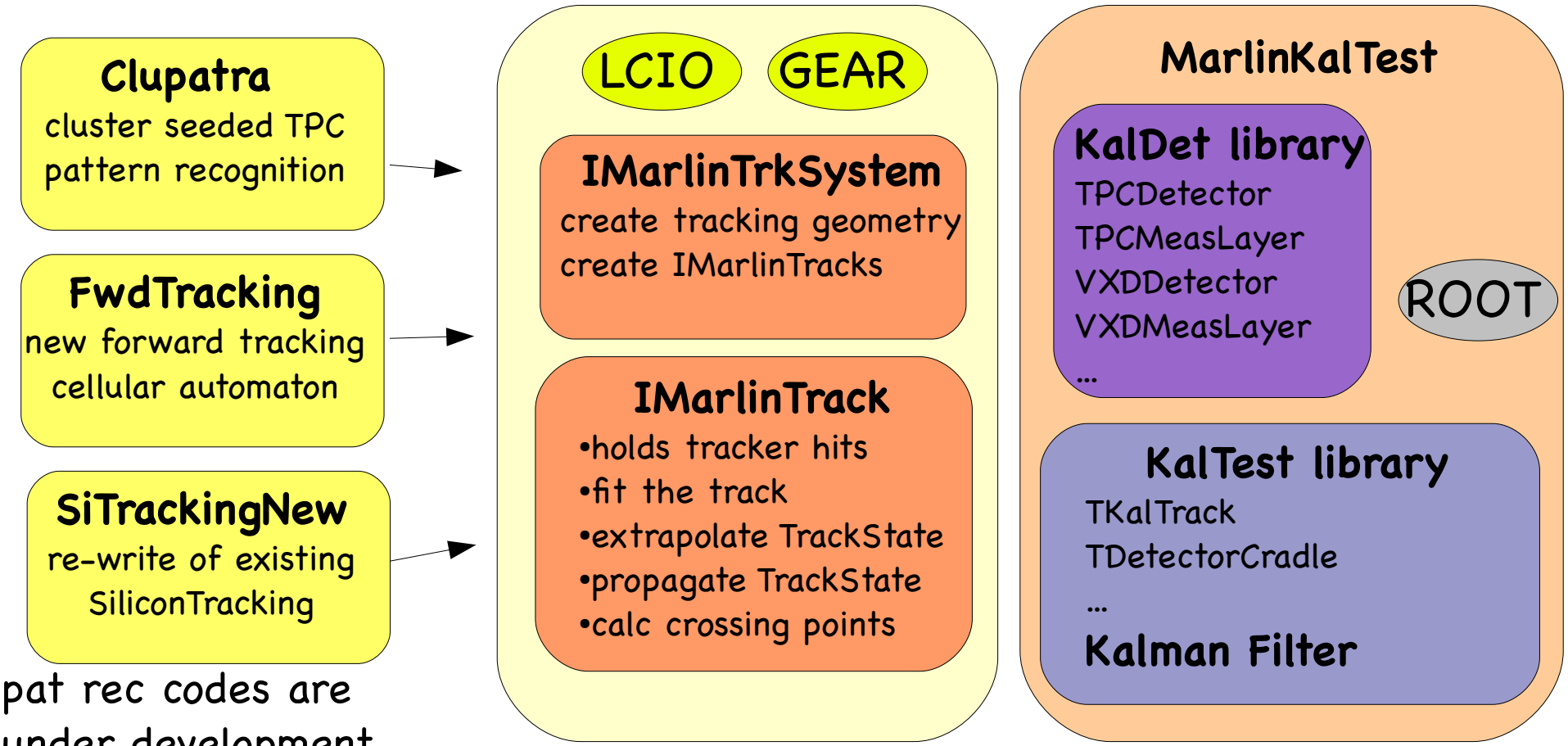




H.Hoelbe

MarlinTrk

- need common framework for developing new tracking code (TPC, Silicon, Fwd)
- would like to have loose coupling between patrec and fitting
- defined abstract interface IMarlinTrk and implement using KalTest/KalDet
  - other fitters might follow (GenFit, ...)
- serves as tests case for writing a generic tracking package in AIDA



pat rec codes are under development

iLCSoft status

plan for core software after LOI (shown at ALCPG09) :

- merge goodies from JSF into framework ✓
- develop a test system ✓
- develop new GRID production system ✓
- improve the geometry description ✓
- improve the reconstruction (tracking & PFA) ✓
- develop LCIOv2 ✓
- improve the simulation ✓

- plan for iLCSoft core tools for DBD mainly fullfilled
- focus on improving/finalizing the simulation and reconstruction

Summary & Outlook

- very active development in iLCSoft framework as preparation for the DBD – focusing on core tools
- met plans for core tools made after LOI
- Thanks to the many people that have contributed to this !

- now need to focus on improving/finalizing the simulation and reconstruction tools
- **we are in good shape – but a lot of work still to be done until DBD !**

- plan to continue to provide iLCSoft as software tool beyond the DBD for LC detector R&D
- continue to improve it – also in context of AIDA WP2