Testbeam software for LC-TPC

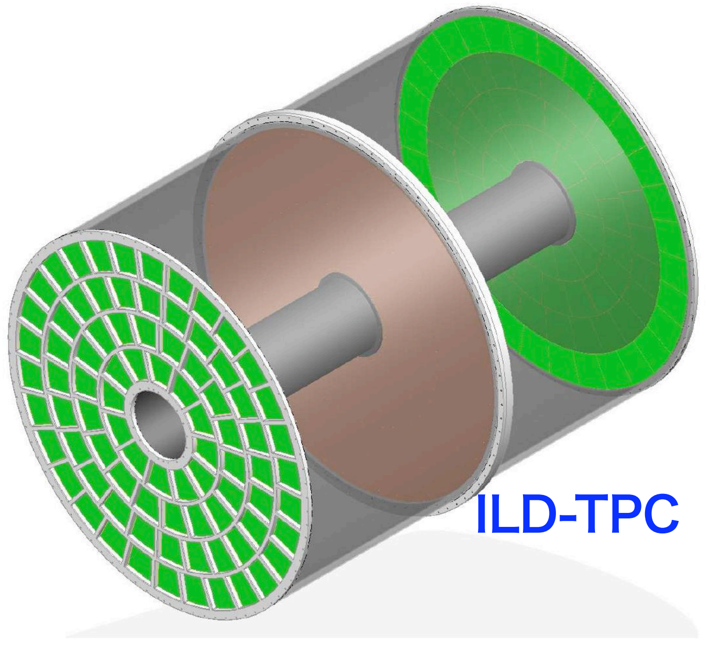
- Data reconstruction w/ MarlinTPC -

Katsumasa Ikematsu (KEK/IPNS)

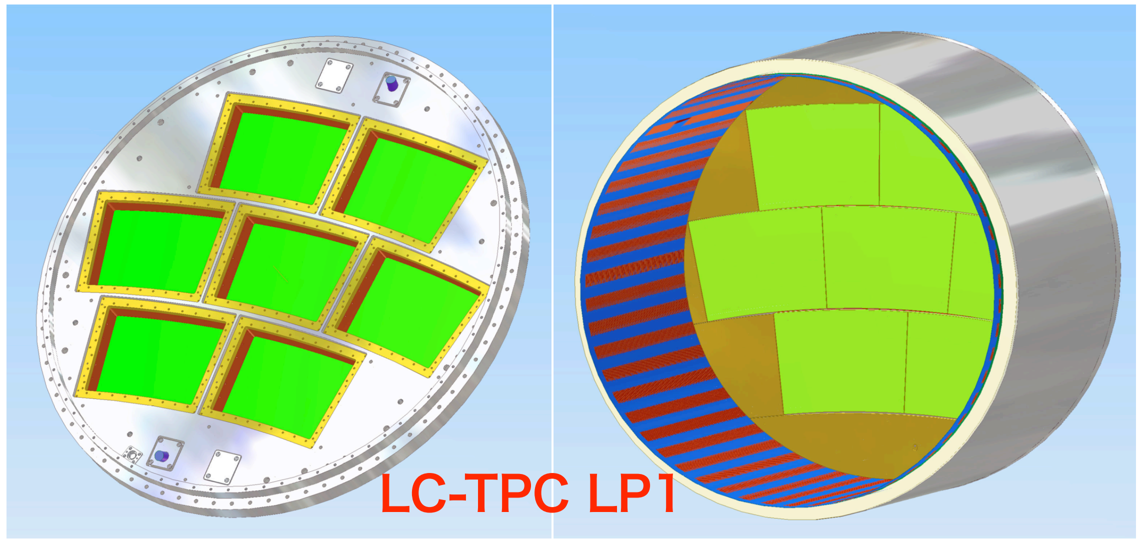
ILD workshop 2010 software meeting

27/01/2010 @ LLR - Ecole polytechnique

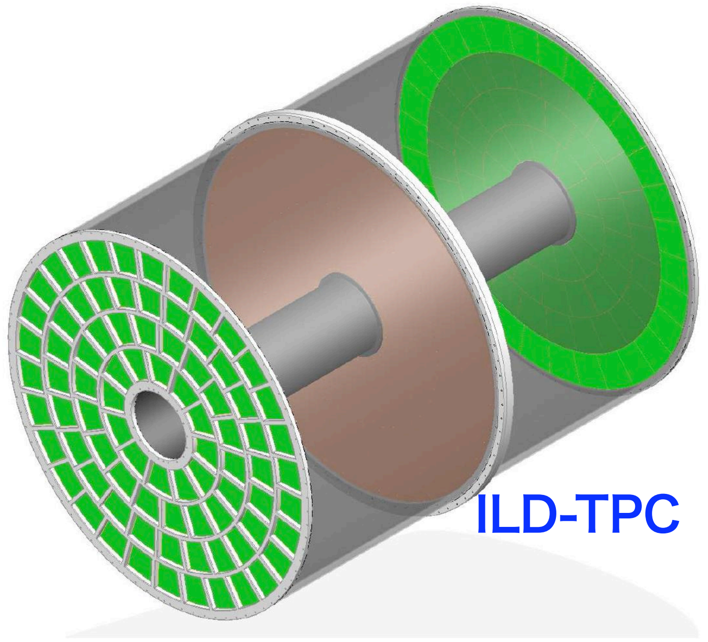
LC-TPC R&D studies



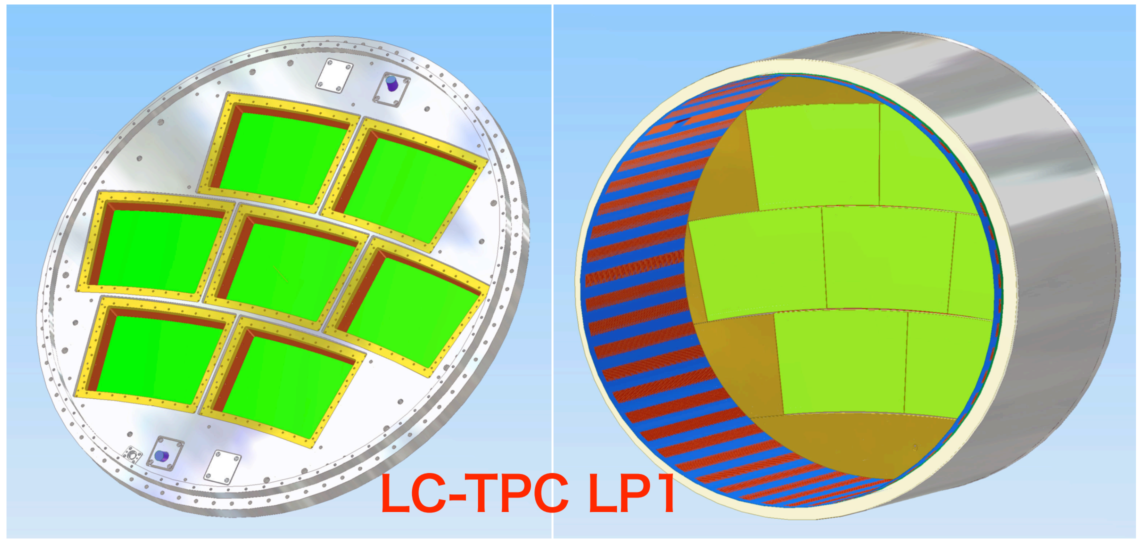
- Purpose of the LC-TPC large prototype 1 (LP1)
  - Build and operate a large prototype (Øin: 72cm, Ldrift: 61cm) in conjunction with EUDET, which allows any MPGD technology, to test manufacturing techniques for MPGD endplates, field cage and electronics
  - Measurements using DESY testbeam & magnetic field (PCMAG) to confirm small prototype results with larger scale device in particular: spatial resolution, dE/dx, homogeneity of large MPGD surfaces, long term stability
  - Demonstrate measurement of 6GeV electron beam momentum over 70cm track length (~10k channels), including development of corrections procedure in the presence of inhomogeneous magnetic field
- Asian group brought GEM modules to LP1
  - Already tested w/o gate-GEM plane
  - Will be exposed w/ gate-GEM + double-GEM soon



LC-TPC R&D studies



- Purpose of the LC-TPC large prototype 1 (LP1)
  - Build and operate a large prototype (Ø$_{in}$: 72cm, L$_{drift}$: 61cm) in conjunction with EUDET, which allows any MPGD technology, to test manufacturing techniques for MPGD endplates, field cage and electronics
  - Measurements using DESY testbeam & magnetic field (PCMAG) to confirm small prototype results with larger scale device in particular: spatial resolution, dE/dx, homogeneity of large MPGD surfaces, long term stability
  - Demonstrate measurement of 6GeV electron beam momentum over 70cm track length (~10k channels), including development of corrections procedure in the presence of inhomogeneous magnetic field

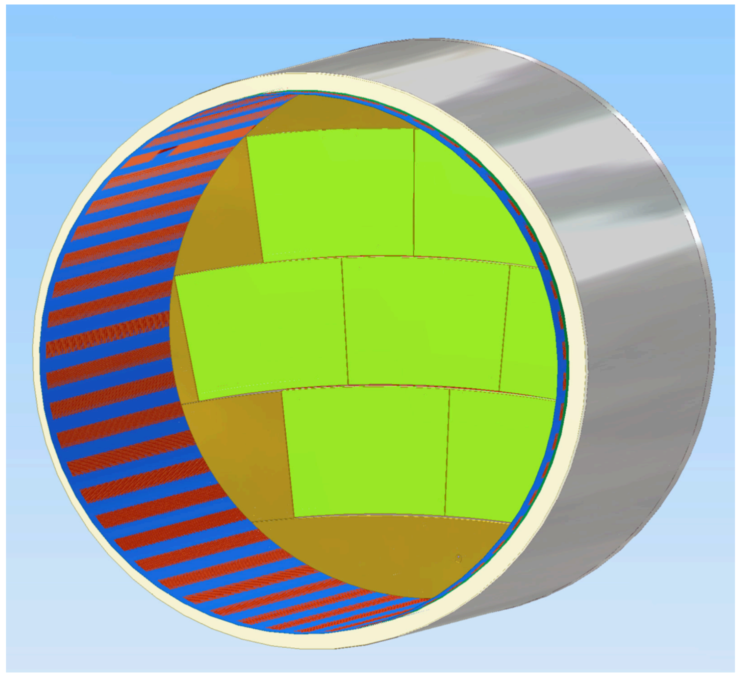


-

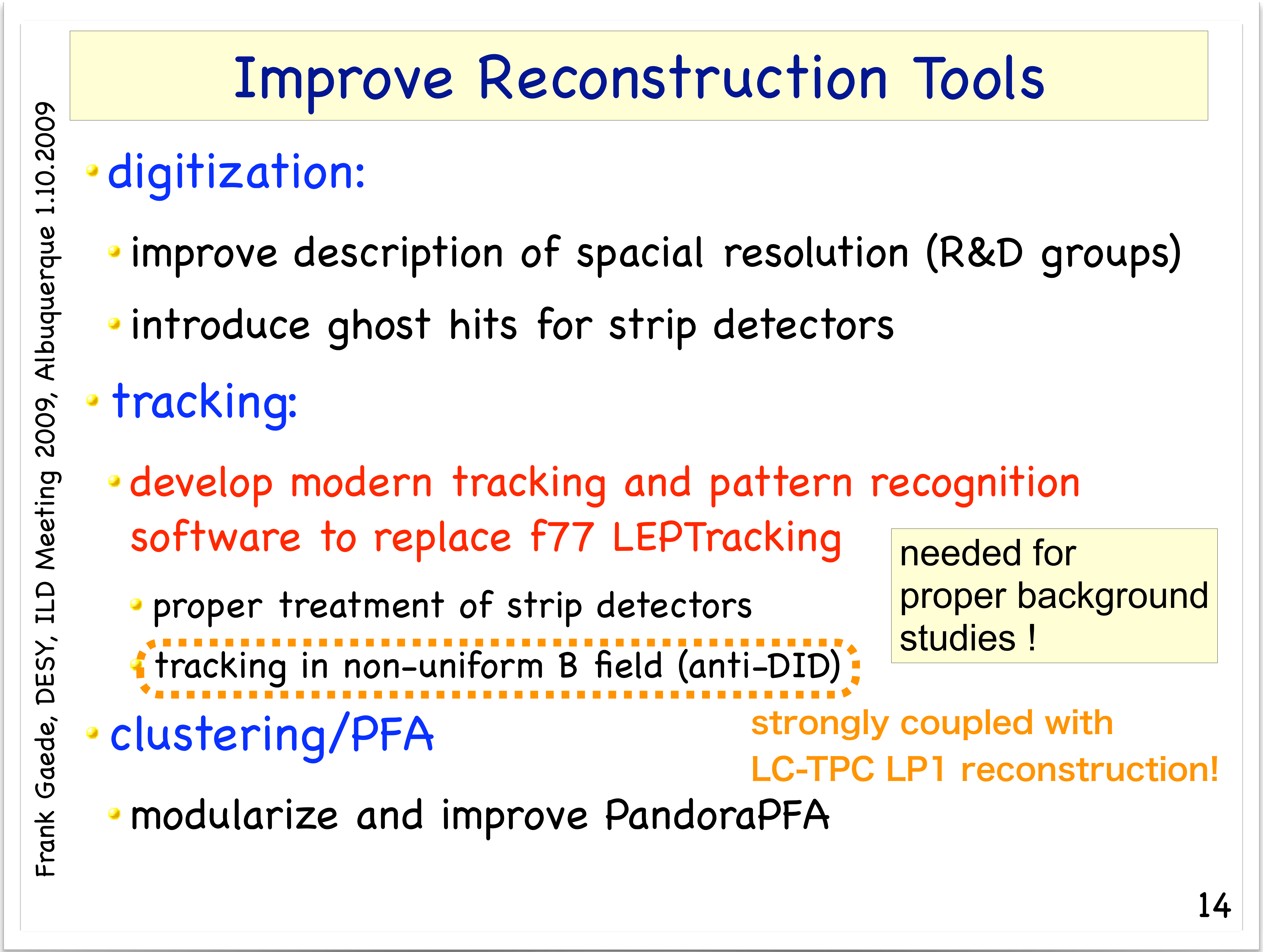
In this talk only asian status will be presented...
Inputs from German & French colleagues
are highly welcome!!

Motivation for our software R&D

- LC-TPC LP1 testbeam data analyses
  - Multi-module tracking in the presence of significant distortion due to non-uniform B-field (PCMAG ~ Anit-DID field in the ILC environment); need to establish the correction procedure
  - Resultant momentum resolution to be consistent with our design goal when extrapolated to ILD-TPC
  - Effects of module boundaries
  - Gating in the multi-module environment



- Feed back to ILD-TPC design
- Feed back to ILD tracking software
  - For the moment, the ILD tracking code is FORTRAN (based on LEP2 tracking code). It is useful if we can replace it with the C++ code with higher functionalities imported from the LP1 software



Frank Gaede, DESY, ILD Meeting 2009, Albuquerque 1.10.2009

Improve Reconstruction Tools

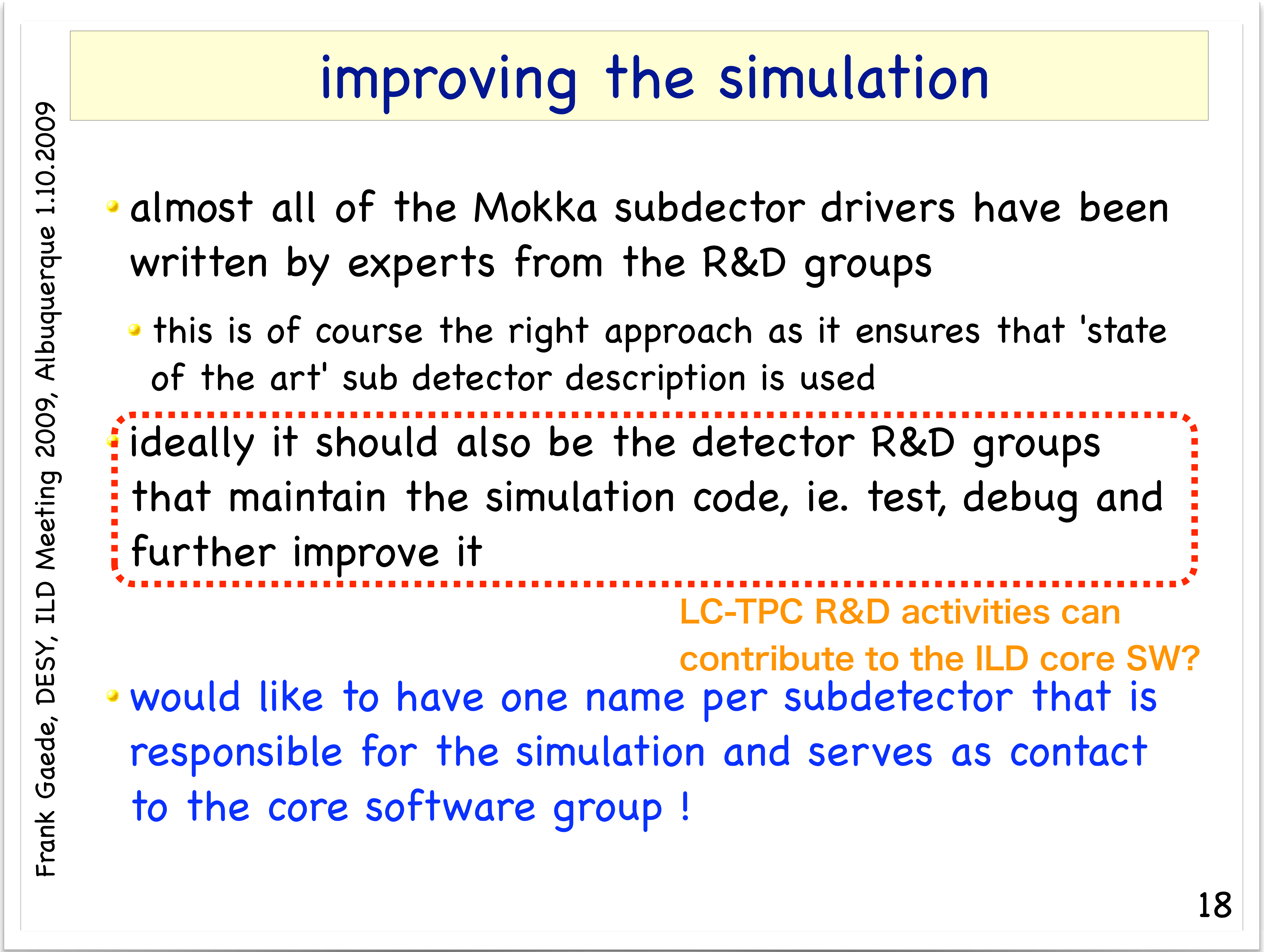
- digitization:
  - improve description of spacial resolution (R&D groups)
  - introduce ghost hits for strip detectors
- tracking:
  - develop modern tracking and pattern recognition software to replace f77 LEPTracking
    - proper treatment of strip detectors
    - tracking in non-uniform B field (anti-DID)

needed for proper background studies !

strongly coupled with LC-TPC LP1 reconstruction!

- clustering/PFA
  - modularize and improve PandoraPFA

14



improving the simulation

- almost all of the Mokka subdector drivers have been written by experts from the R&D groups
  - this is of course the right approach as it ensures that 'state of the art' sub detector description is used
- ideally it should also be the detector R&D groups that maintain the simulation code, ie. test, debug and further improve it

LC-TPC R&D activities can contribute to the ILD core SW?

- would like to have one name per subdetector that is responsible for the simulation and serves as contact to the core software group !

Frank Gaede, DESY, ILD Meeting 2009, Albuquerque 1.10.2009

18

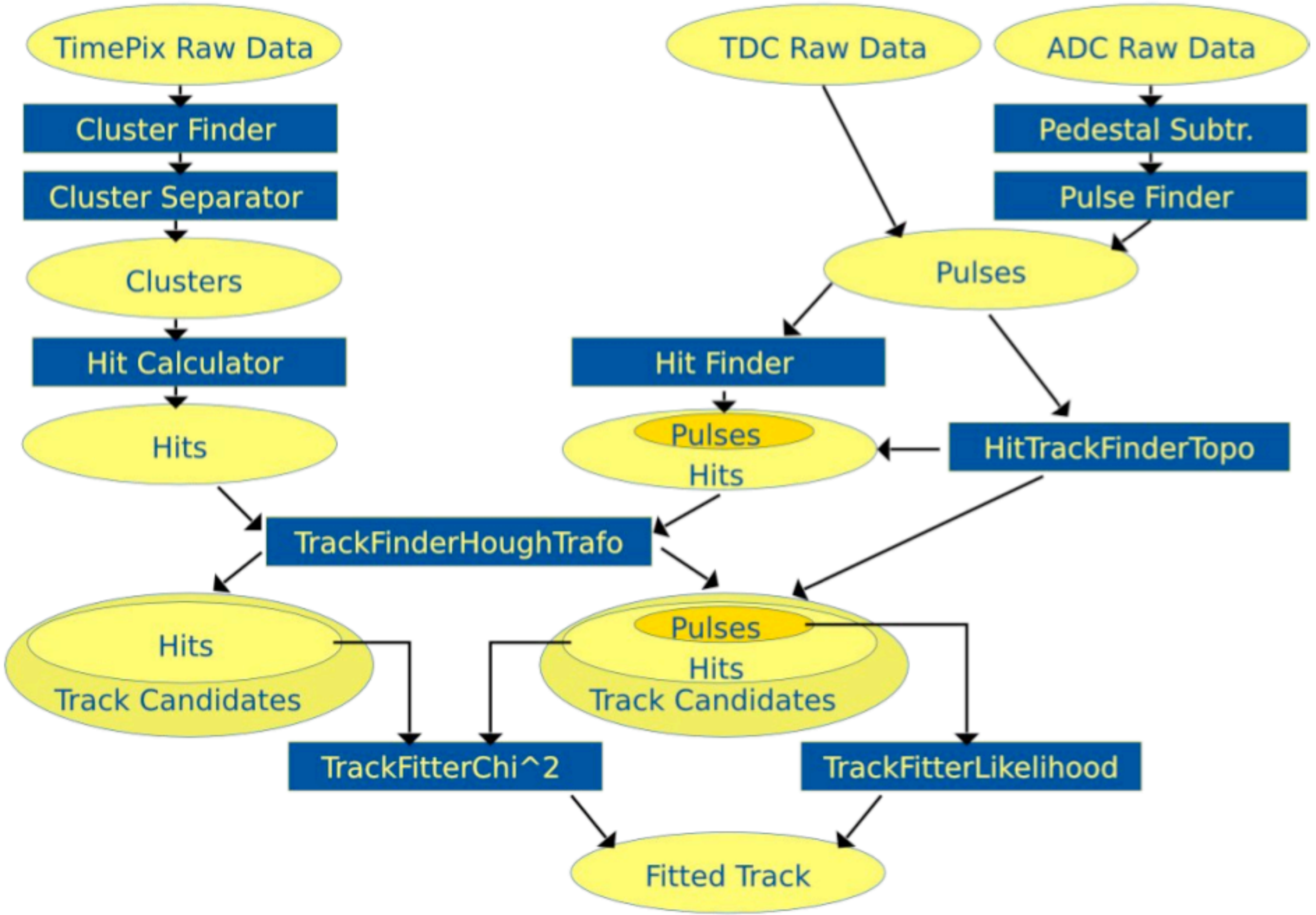
What is MarlinTPC ?

- Marlin based simulation, digitisation, reconstruction and analysis code for the TPC
- Goal: To get a highly modular reconstruction and analysis framework for ILC TPC R&D with standardized interfaces between its modules
  - This ensures that despite of the large diversity of readout structures, electronics, amplification system, etc. much code can be shared among the groups and that different algorithms developed by different people or data taken by different groups can be easily compared
- MarlinTPC is an implementation of the TPC data model (LCIO provides data class as the Event data model)

Data Class	**Description**
LCEvent	Contains collections of one event (bunch crossing)
LCCollection	Collection of data classes of a certain type e. g. TrackerRawData
MCParticle	Particle from the MC generator
SimTrackerHit	Charge deposition in the detector
TrackerRawData	ADC values from the TPC
TrackerData	Calibrated raw data
TrackerPulse	Charge and time of a pulse in one electronics channel
TrackerHit	3D hit with charge information
Track	Helix parametrisation of the fitted track
LCGenericObject	User defined data class

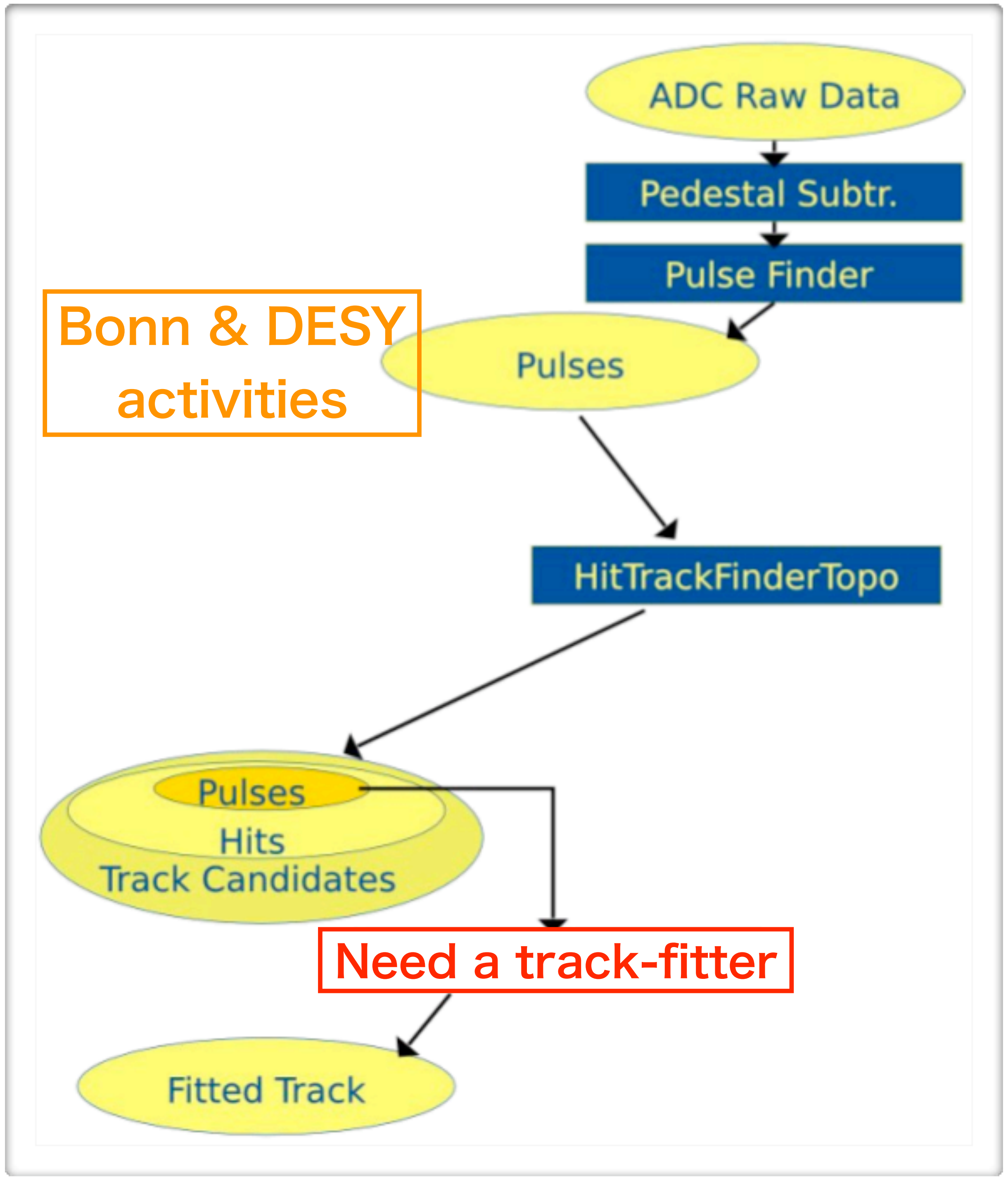
MarlinTPC reconstruction chain

- Highly modular and independent of specific detector
  - Prototypes and large ILC detector TPCs
  - MicroMEGAS, GEMs and Anode Wires
  - Pad and Pixel (TimePix) readout
  - ADC and TDC read-out electronics



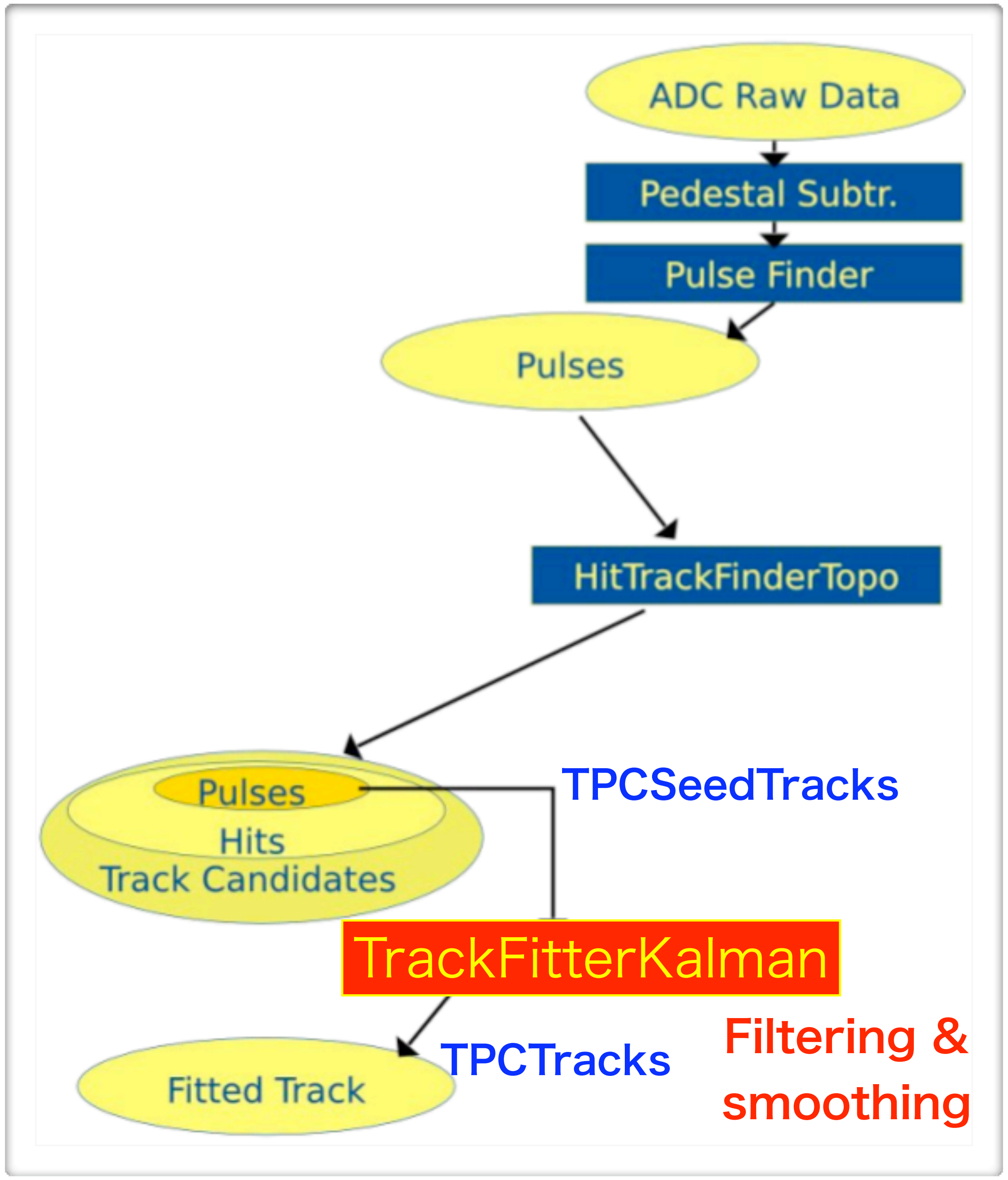
Reconstruction for Asian GEM module

- List of processors (needed for Asian GEM module data reconstruction)
  - ConditionsProcessor (TPCConditions, TPCPedestal & TPCChannelMapping)
    - Provides access to conditions data transparently from LCIO files or a databases, using LCCD
    - At the moment TPCChannelMapping only
  - TrackerRawDataToDataConverterProcessor
    - Converts the TrackerRawData to TrackerData without processing the values
  - ADCPulseConverterProcessor
    - Convert zero-suppressed ADC raw data to pulses
  - ChannelMapperProcessor
    - Changes cellID to map from hardware channel numbers to sowftware/logical channel numbers



Reconstruction for Asian GEM module (cont)

- List of processors (needed for Asian GEM module data reconstruction)
  - HitTrackFinderTopoProcessor
    - Calculates TrackerHits from TrackerPulses
  - TrackSeederProcessor
    - Calculates seed track parameters from in the TrackerHits in the track candidates collection
  - **TrackFitterKalmanProcessor**
    - **Our work!!**
  - HepRepOutputProcessor
    - Generates a HepRep Output file for a HepRep based event display program
    - HepRep XML file which can be displayed e. g. with Wired/JAS3 or HepRApp (.jar)
  - LCIOOutputProcessor
    - Writes the current event to the specified LCIO output file
    - Needs to be the last Active Processor
    - Split output file if size in 1992294 kB exceeds



Reconstruction for Asian GEM module (cont)

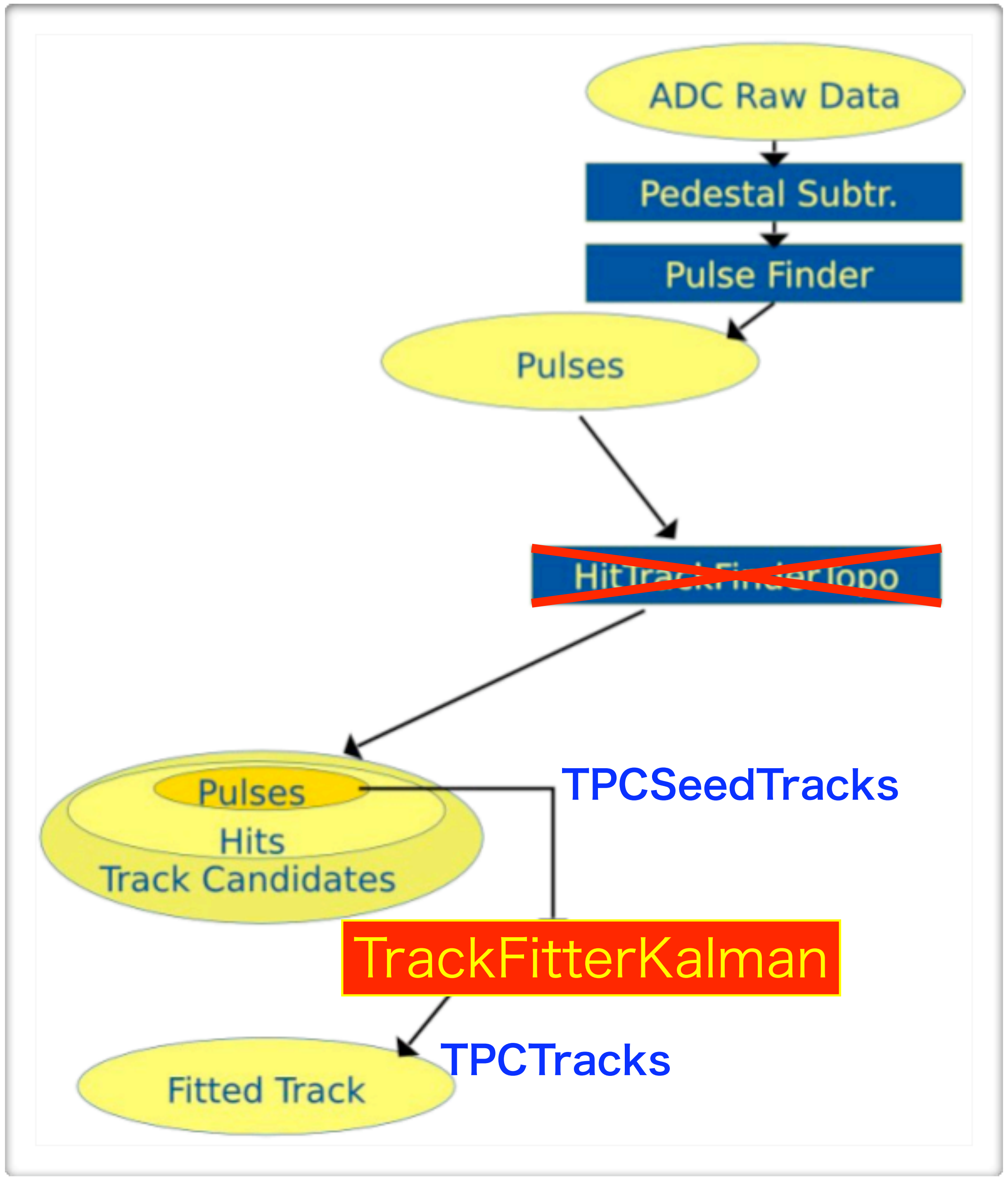
- List of processors (needed for Asian GEM module data reconstruction)
  - HitTrackFinderTopoProcessor
    - **Trying to replace w/ combinatorial KF scheme**

-

French & Canadian group
are also interested in this work.
We'll have a camp at CEA Saclay
just after the ILD workshop!

-

    - Our work!!
  - HepRepOutputProcessor
    - Generates a HepRep Output file for a HepRep based event display program
    - HepRep XML file which can be displayed e. g. with Wired/JAS3 or HepRApp (.jar)
  - LCIOOutputProcessor
    - Writes the current event to the specified LCIO output file
    - Needs to be the last Active Processor
    - Split output file if size in 1992294 kB exceeds



Why Kalman filter track-fitting ?

- Need to take into account the inhomogeneous field (not only the LP1 condition but also the ILC condition due to Anti-DID field)
  - Track model can change site to site which allows B-field variation along a particle trajectory!

- No need for calculation of Matrix inversion
  - In general the matrix inversion takes time under the space point measurements of ~ 200 sampling!

- Track connection between sub-detectors (multiple scattering and energy loss in the boundaries) can be considered

- Beneficial to calculate geometric mean resolution by using “Inverse Kalman filter” (in/out target row hit)

Discussion list (instead of a summary)

- For LP1 testbeam data analyses
  - Develop (porting work) a track finder taking into account the multi-module configuration (existence of module boundaries)
  - Non-helical track model with B-field map implementation
  - Distortion correctors
  - Direct access to objects like tracks, hits, clusters, pulses using LCIO-v2 RIO (.rlcio) ??
  - Direct access to events, zoom up/down, rotate, ... not only for data monitoring (3D event display, various plots etc.) but also debugging track finder/fitter codes !!
- For feed back to ILD-TPC design
- For feed back to ILD tracking software
  - To replace f77 LEPTracking

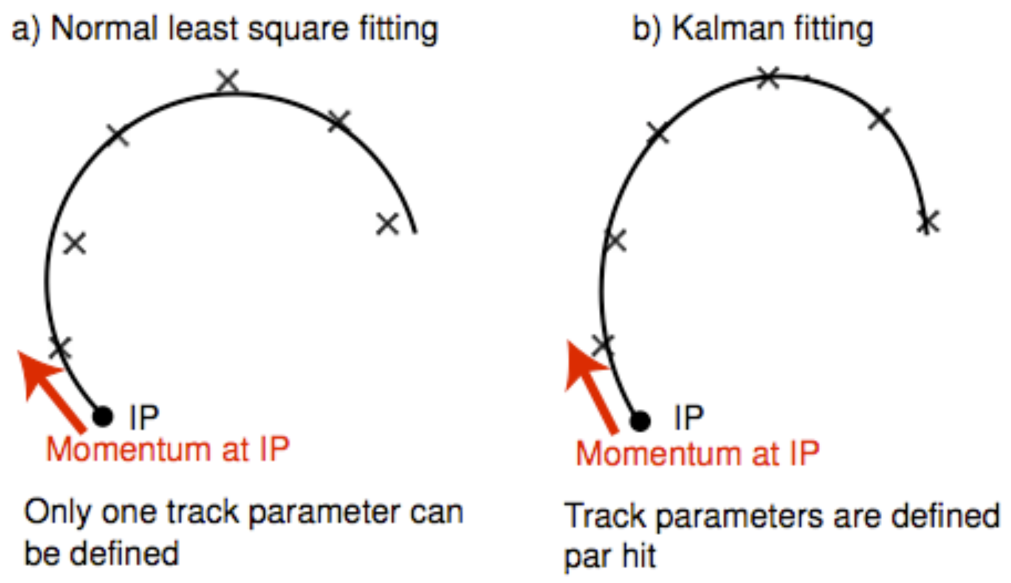
Backup slides

Excerpt from our steering XML file

```
<marlin>
<!--##########################################
    #                                        #
    #     steering file for reconstruction   #
    #                                        #
    ##########################################-->

 <execute>
  <processor name="MyAIDAProcessor"/>
  <processor name="MyConditionsProcessor"/>
  <processor name="MyTrackerRawDataToDataConverterProcessor"/>
  <processor name="MyADCPulseConverterProcessor"/>
  <processor name="MyChannelMapperProcessor"/>
  <processor name="MyHitTrackFinderTopoProcessor"/>
  <processor name="MyTrackSeederProcessor"/>
  <processor name="MyTrackFitterKalmanProcessor"/>
  <processor name="MyHepRepOutputProcessor"/>
  <processor name="MyLCIOOutputProcessor"/>
 </execute>

 <global>
  <parameter name="LCIOInputFiles"> readout-7049.000.slcio </parameter>
  <!-- limit the number of processed records (run+evt): -->
  <parameter name="MaxRecordNumber" value="3" />
  <parameter name="SkipNEvents" value="0" />
  <parameter name="SupressCheck" value="false" />
  <parameter name="GearXMLFile"> gear_LP_TPC_GEM_7module.xml </parameter>
  <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> DEBUG  </parameter>
 </global>

...
...

 <processor name="MyTrackFitterKalmanProcessor" type="TrackFitterKalmanProcessor">
  <!-- The the name of the input collection of track candidates (default: TPCSeedTracks) -->
  <parameter name="InputSeedTracks" type="string" lcioInType="Track">TPCSeedTracks </parameter>
  <!-- The name of the output collection with the fitted tracks (default: TPCTracks) -->
  <parameter name="OutputTracks" type="string" lcioOutType="Track">TPCTracks </parameter>
  <!-- if not 0 the output hits collection is set transient (default: 0) -->
  <parameter name="SetOutputTransient" type="int">0 </parameter>
 </processor>

...
...
```

Disclaimer

- Skip an explanation for the principle of Kalman filter
  - but there is a good document and a good slide (talk by Keisuke)
    - http://www-jlc.kek.jp/subg/offl/kaltest/doc/ReferenceManual.pdf
    - http://www-jlc.kek.jp/subg/offl/kaltest/doc/kalman.pdf


a) Normal least square fitting
IP
Momentum at IP
Only one track parameter can be defined
b) Kalman fitting
IP
Momentum at IP
Track parameters are defined par hit

- Our implementation depends on 2 external C++ class libraries (should be loaded via $MARLIN_DLL)
  - Kalman filter engine: KalTest (**Kal**man filter **T**racking **E**nvironment w/ **S**tandard **T**est suits)
    - http://jlccvs.kek.jp/cgi-bin/cvsweb.cgi/KalTest/?cvsroot=CDC
  - User defined application-specific detector class library: KalDet
    - http://jlccvs.kek.jp/cgi-bin/cvsweb.cgi/KalDet/?cvsroot=CDC

Typical Procedure for Tracking

outermost innermost

$$(1) \longrightarrow (2) \longrightarrow \cdots \longrightarrow (k-1) \longrightarrow (k) \longrightarrow \cdots (n)$$

filtering

$$\begin{tikzcd}
 & \boldsymbol{a}_2^1 \arrow[d] & & \boldsymbol{a}_{k-1}^{k-2} \arrow[d] & \boldsymbol{a}_k^{k-1} \arrow[d] & & \boldsymbol{a}_n \\
\boldsymbol{a}_1 \arrow[ur] & \boldsymbol{a}_2 \arrow[ur] & \cdots \arrow[ur] & \boldsymbol{a}_{k-1} \arrow[ur] & \boldsymbol{a}_k \arrow[ur] & \cdots \arrow[ur] &
\end{tikzcd}$$

prediction

$$(1) \cdots \longleftarrow (k) \longleftarrow (k+1) \longleftarrow \cdots \longleftarrow (n-1) \longleftarrow (n)$$

$$\begin{tikzcd}
\boldsymbol{a}_1^n \cdots & \boldsymbol{a}_k^n \arrow[l] & \boldsymbol{a}_{k+1}^n \arrow[l] & \cdots \arrow[l] & \boldsymbol{a}_{n-1}^n \arrow[l] & \boldsymbol{a}_n \arrow[l] \arrow[d, phantom, sloped, "\equiv"] \\
 & & & & & \boldsymbol{a}_n^n
\end{tikzcd}$$

smoothing

Extrapolation

$$\boldsymbol{a}_n \longrightarrow \boldsymbol{a}_{n+1}^n = \boldsymbol{a}_{IP} \qquad \boldsymbol{a}_0^n = \boldsymbol{a}_{\mathrm{cal}} \longleftarrow \boldsymbol{a}_1^n$$

prediction prediction

Alignment, Resolution Study, etc.

Need to eliminate point (k)

$$(1)\cdots\cdots\cdots(k-1) \quad (k) \quad (k+1)\cdots\cdots\cdots(n)$$

$\Downarrow$ Inverse Kalman Filter

$\boldsymbol{a}_k^{n*}$ Reference Track Param.

$\Downarrow$

$\boldsymbol{h}_k(\boldsymbol{a}_k^{n*})$ Expected Hit Position

$\Downarrow$

$\boldsymbol{r}_k^{n*} = \boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{a}_k^{n*})$ Residual to Look At

KalTest

- Kalman filter Tracking Environment w/ Standard Test suits
  - A Kalman Filter Package written in C++. It is based on ROOT and provides basic libraries for track fitting with Kalman filter technique. The package is distributed with some sample test programs to illustrate their usage.

- Kalman filter library features
  - **KalLib**: Kalman filter defines a generic procedure and has a much wider scope than track fitting. This implies necessity for a library of generic abstract base classes that implement the generic algorithm of the Kalman Filter.
  - **KalTrackLib**: By inheriting from the generic base classes in KalLib and implementing their pure virtual methods for track fitting purpose, we then realize a Kalman-filter-based track fitter library. However, KalTrackLib should not depend on any particular track model or shape or coordinate system of a measurement layer according to the above guideline.
  - **GeomLib**: hence separate out geometry classes that provide track model (helix, straight line, ...) and surfaces (cylinder, hyperboloid, flat plane, etc.) as a geometry library.

  - Since the tracking system of a collider detector usually consists of various components such as a vertex detector (VD), an intermediate tracker (IT), a central tracker (CT), etc., which have different shapes and coordinate systems, the software package for Kalman-filter-based track fitting should be able to accommodate a measurement layer with any shape and/or coordinate system. Considering possible extrapolation of a track to an outer tracking device such as a muon detector, it is also desirable that the package can handle site-to-site change of the magnetic field.

KalTest (requirement for user)

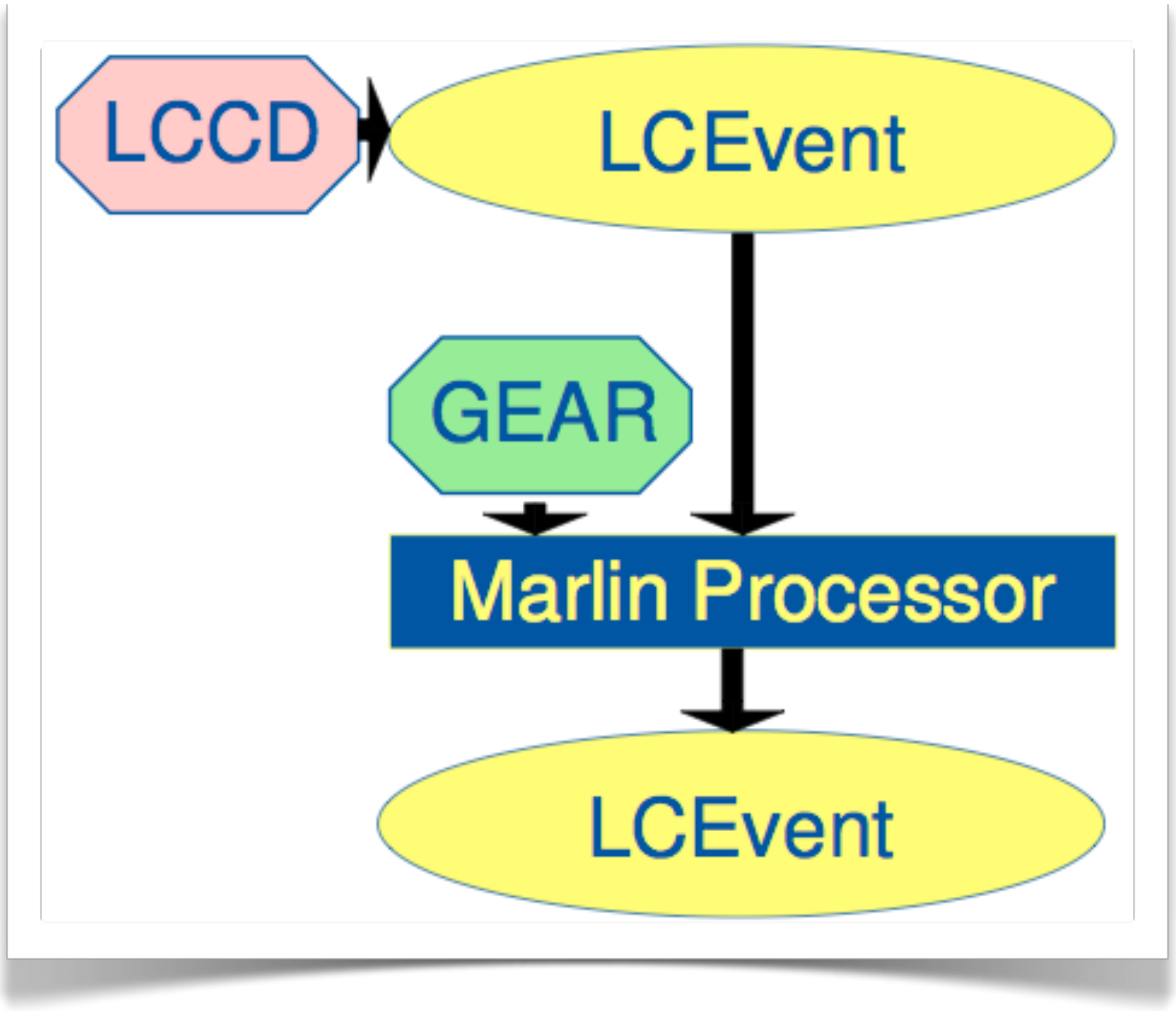
- Require minimum number of user-implemented classes to the following three
  - **MeasLayer**: a measurement layer that multiply inherits from an abstract measurement layer class `TVMeasLayer` and a shape class derived from `TVSurface` in GeomLib.
  - **KalDetector**: an array class derived from `TVKalDetector` that holds the user-defined MeasLayers with any shape and/or coordinate system. Notice that this also defines material distributions in the tracker.
  - **Hit**: a coordinate vector class as defined by the MeasLayer, which inherits from `TVTrackHit`.

KalDet

- provides user-defined classes required by KalTest
- **kern**: application-specific abstract classes
- **lctpc/lp1gem**: provides LC-TPC LP1-JGEM modules' “KalHit”, “MeasLayer”, and “KalDetector”
- Can put your detector detector for example, **othertpc/medi-tpc** etc.

The Marlin framework

- Modular Analysis and Reconstruction for the LINear collider
  - C++ reconstruction framework for LCIO data
  - Controls the data flow
  - Each computing task is performed by a “processor”
  - Interface between the processors: LCEvent
  - Programme flow is controlled with XML steering files
  - Provides an interface to GEAR and LCCD

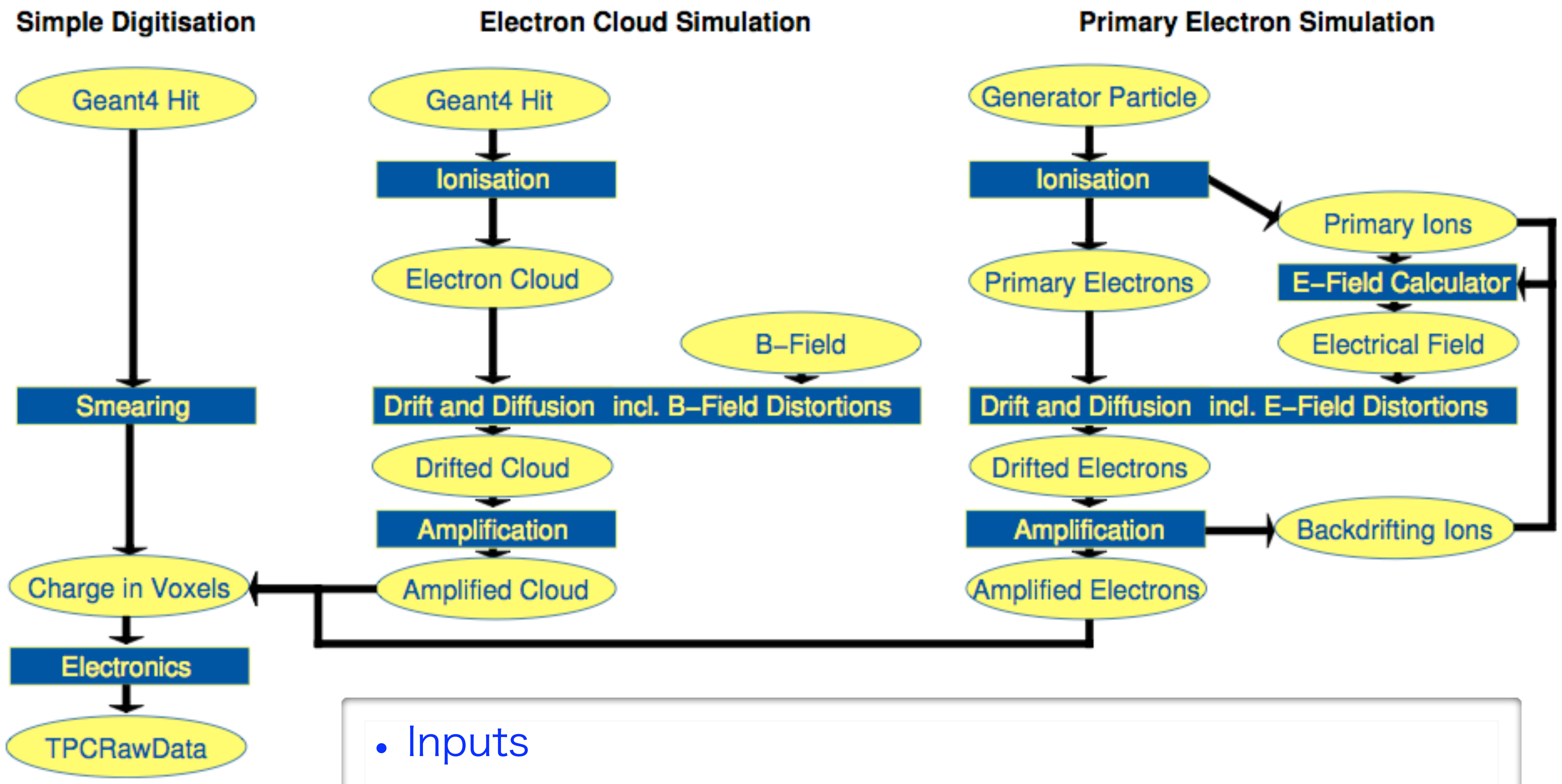


A steering file mechanism allows to activate the needed processors.

These are then called for every event using the `LCEvent` as

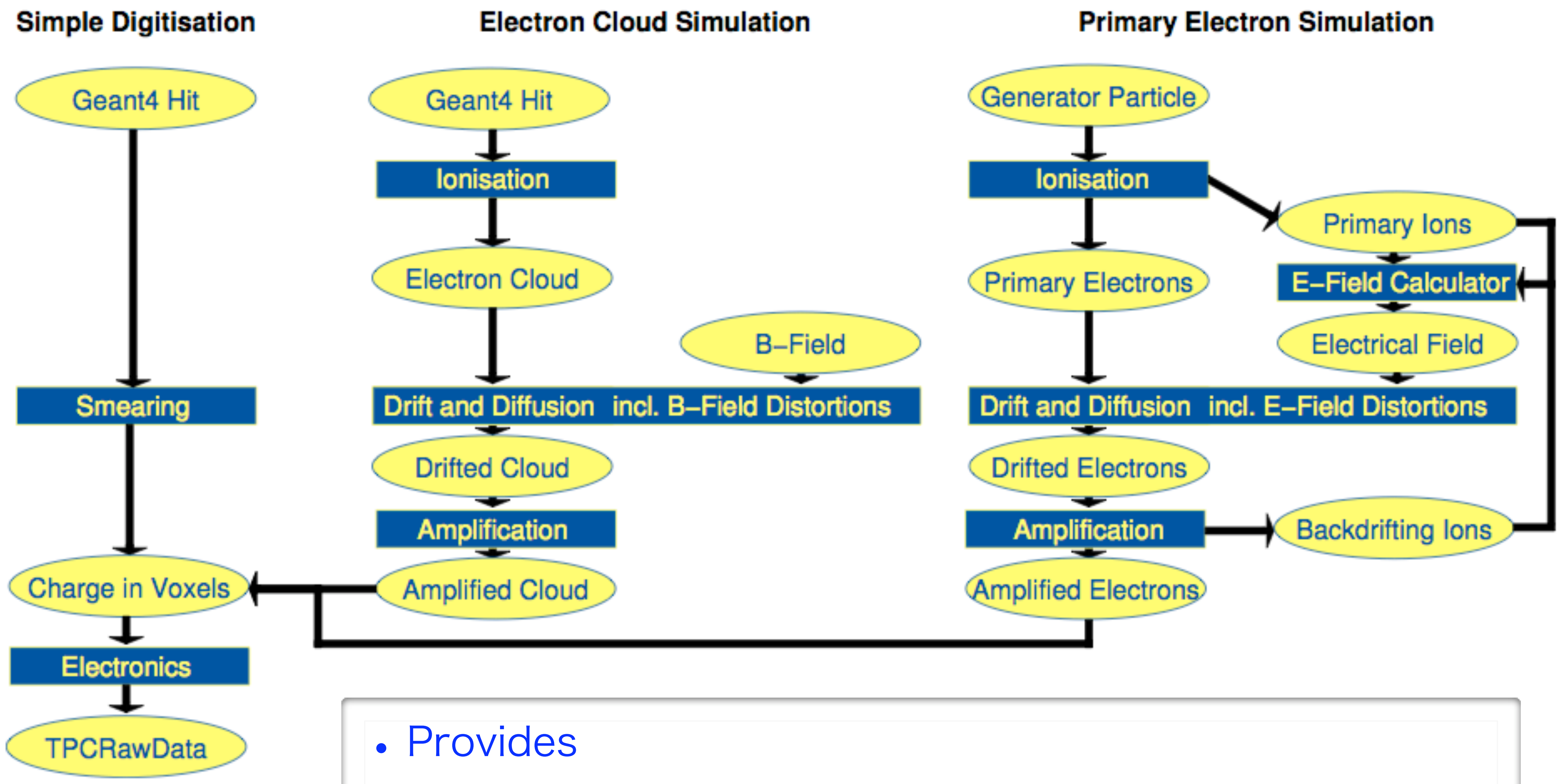
container for input and output data in terms of `LCCollections`.

Simulation / Digitization



- Inputs
  - Single electrons from detailed simulation (inside MarlinTPC): production, drift, amplification and pulse shaping of electronics
  - Mokka (Geant4) Hits: smearing (+ voxel) or more detailed electron cloud simulation

Simulation / Digitization



- Provides
  - TrackerRawData for use in reconstruction
  - Read-out specific data
  - Event pile-up
  - Ion backdrift (Handling of field distortions)

What needs to be done…

- ★ Ideally aim to incorporate background into analyses as the default
  - Beam background
  - Two-photon background

- ★ To do this requires:
  - New tracking code !
    - TPC patrec (old f77 code) needs replacing
    - SiliconTracking not optimised for background
  - Proper simulation/reconstruction of silicon strip detector
    - Need to account for stereo strip layers in SIT/FTD (currently, artificially combine into "point")
    - Reconstruction code for FTD – combinatorics potentially large
  - New digitisation code? + TPC pad hits ?
    - Treatment (possibly parametric) of clusters in pixel detectors
  - Definition of two photon samples
  - More realistic treatment of BX tagging in reconstruction
  - Realistic plan how to implement into analyses (speed issues)