DigiSim
A digitization simulation package

Guilherme Lima
for the software group at NIU

NORTHERN ILLINOIS
UNIVERSITY

ALCPG-Cal meeting
May 09, 2005

Talk Outline

- DigiSim: Purpose and requirements
- Package description
- Existing modifiers
- Usage and configuration
- Demo for tile HCal
- Current status

DigiSim

- Goal: a program to parametrically simulate the signal propagation and digitization processes for the ILC detector simulation
  ☞ an essential tool for comparing different detector technologies

- DigiSim role is to convert the simulated data (energy depositions and hit timings) into the same format AND *as close as possible* to real data from readout channels, while preserving all MC information from input data files

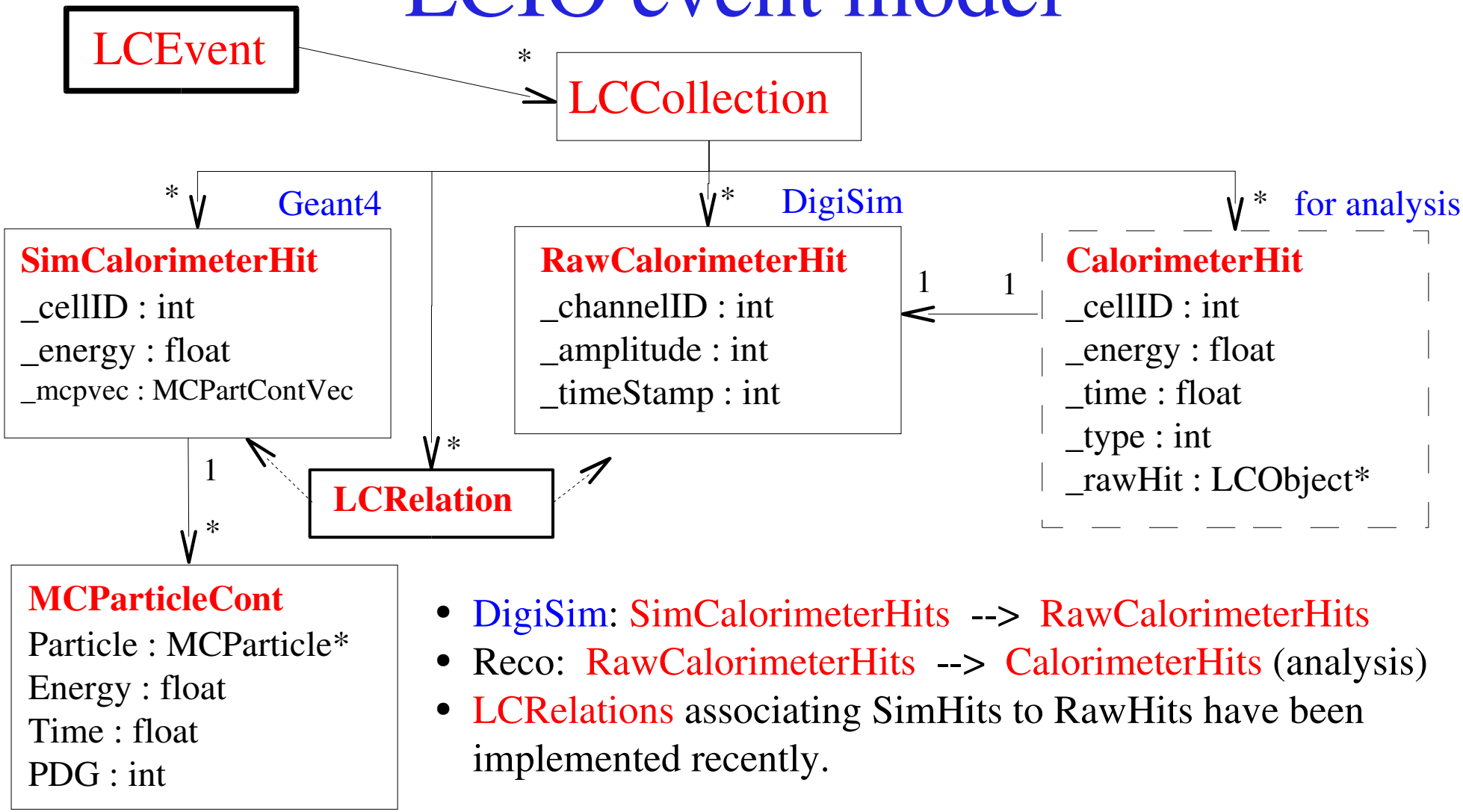
  – *As close as possible* means that all known effects from digitization process should be taken into account, if possible: cell ganging, inefficiencies, noise, crosstalks, hot and dead channels, non-linearities, attenuation, etc.

- Same reco / analysis software can be used for MC and real data.

Requirements and choices

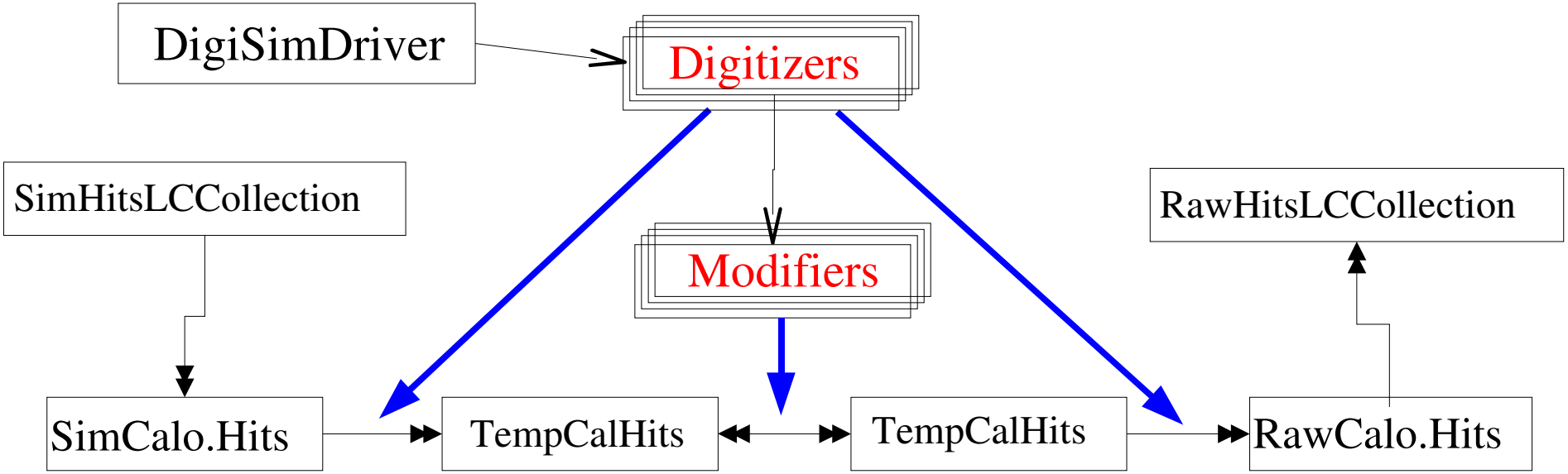
- Basic requirements:
  - Object oriented design to simplify maintenance and implementation of new functionality
  - Intended not only for the full ILC detector, but also for the test beam (TB), as a testbed for full detector digitizaton
  - To be run either standalone, or as an on-the-fly preprocessor to reconstruction and/or analysis program
  - Originally in C++/Marlin framework for the test beam, it has been recently ported to the Java/Lcsim framework
- Marlin chosen as the C++ framework

LCIO event model



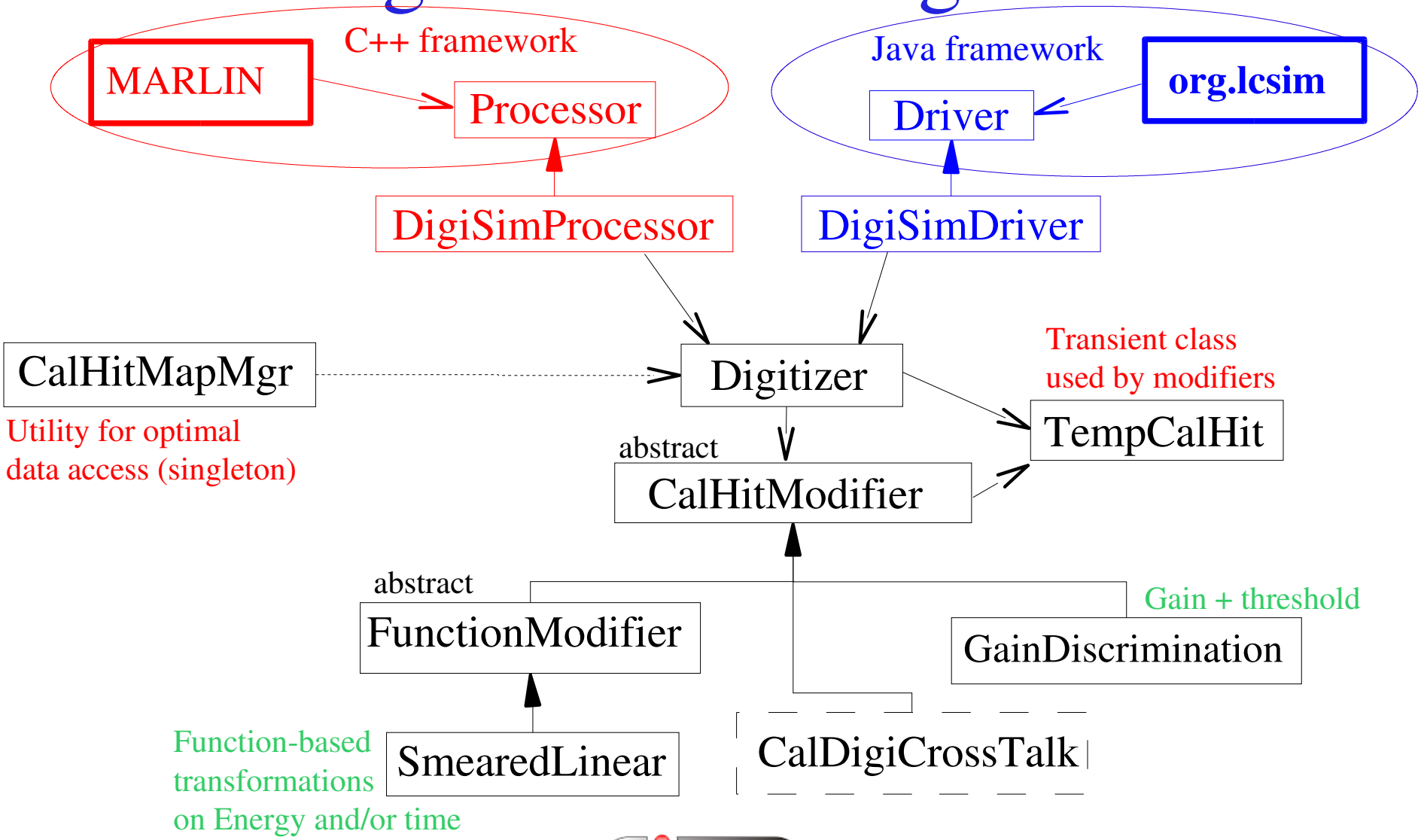
- DigiSim: SimCalorimeterHits --> RawCalorimeterHits
- Reco: RawCalorimeterHits --> CalorimeterHits (analysis)
- LCRelations associating SimHits to RawHits have been implemented recently.

DigiSim event loop

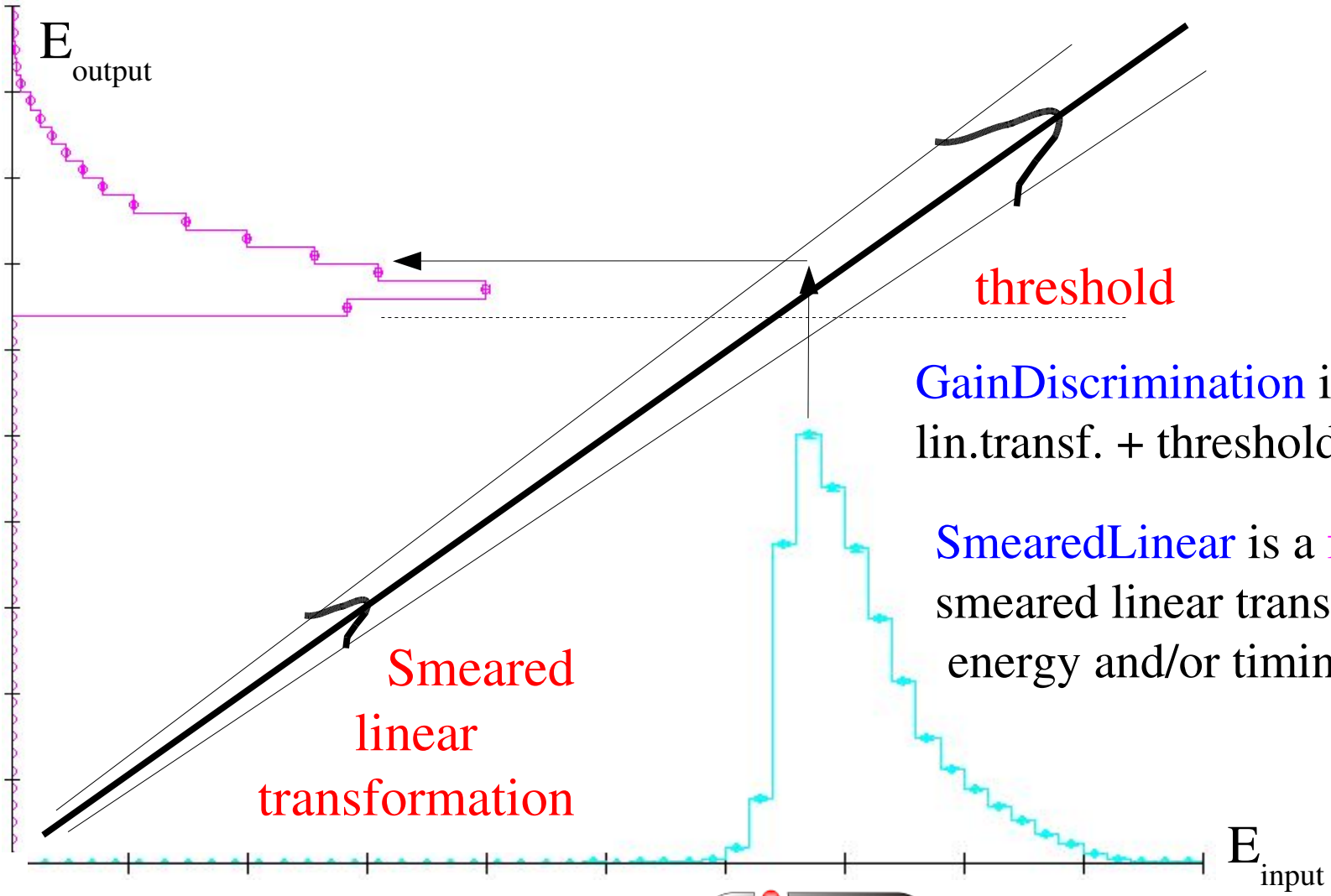


- Calorimeter hits are shown here, but the same overall structure could also be used for tracker hits
- TempCalHits are both input and output to each modifier (allows for chaining)
- All processing is controlled by the Digitizers (one per subdetector)
- Modifiers are configured at run time, via the Marlin steering file (efficiencies, noise ratio, etc.)
- New modifiers can be easily created for new functionality (more info later)

DigiSim class diagrams



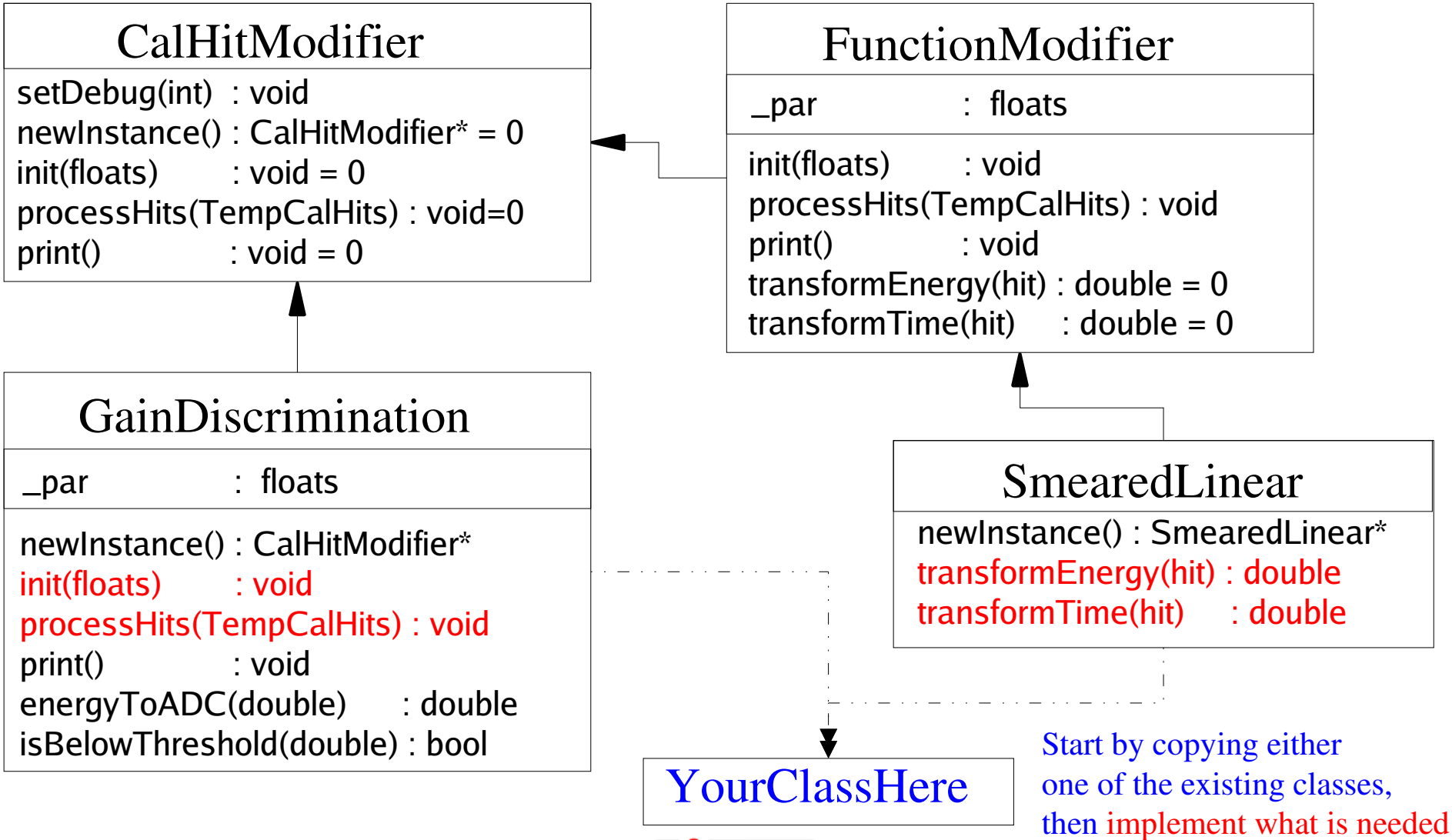
Existing modifiers



GainDiscrimination is a smeared lin.transf. + threshold on energy

SmearedLinear is a function-based smeared linear transformation on energy and/or timing

Use existing modifiers or create new ones?



Start by copying either one of the existing classes, then implement what is needed

Simple example: numbers from the tile HCal

- Scintillation: 100 eV/photon $\rightarrow$ 10^{+4} photons/MeV

  flat factor of 10^{+7} photons/GeV $\rightarrow$ use GainDiscrimination

  Ex: For a MIP (Geant4): 0.9MeV (normal, 5mm thick) $\rightarrow$ 9000 photons

- SiPM detection efficiency:

  QE ~15%. Again, use GainDiscriminator

- Photon collection efficiency:

  9000 scint.photons/MIP $\rightarrow$ 15PE/MIP detected $\rightarrow$ 15/0.15 = 100 incident photons

  => Eff_{coll} = 100 inc. / 9000 tot.scint. = 0.0111

  Variance (Poisson): $\sigma_N^2 = \langle N \rangle$ $\rightarrow$ for $\langle N_{PE} \rangle = 15$, $\sigma_N / N \sim 26\%$

  Therefore: $Eff_{coll} = 0.0111 \pm 0.0029$ => use GainDiscrimination with smearing

- Saturation: a function-based modifier (derived from FunctionModifier, to be implemented soon)

Steering file example

```
##########################################
#  Example steering file for DigiSim
##########################################
.begin Global  ----------------------------------------
# specify one or more input files (in one or more lines)
LCIOInputFiles inputfile

# the active processors that are called in the given order
ActiveProcessors CalHitMapProcessor
ActiveProcessors EcalBarrelDigitizer
ActiveProcessors EcalEndcapDigitizer
ActiveProcessors HcalBarrelDigitizer
ActiveProcessors HcalEndcapDigitizer
ActiveProcessors OutputProcessor

# limit the number of processed records (run+evt):
MaxRecordNumber 500
.end Global  ----------------------------------------
```

Configuring processors and modifiers

```
##########################################################
# A subdetector digitizer.  It instantiates one or more calorimeter hit
#  "modifiers", which together represent the full digitization process
.begin HcalBarrelDigitizer
ProcessorType       DigiSimProcessor
InputCollection     HcalBarrHits
OutputCollection    HcalBarrRawHits
Raw2SimLinksCollection  HcalBarrRaw2sim
```

(As many as needed)

```
ModifierNames    HADlightYield HADlightCollEff SiPMQEandAmplif
# Parameters:
#modifierName   type                  gainNom gainSig  thresh   thrSig
HADlightYield   GainDiscrimination 10000000       0       0        0
HADlightCollEff GainDiscrimination   0.0111  0.0029       0        0
# 15% Qeff * 601 amplification
SiPMQEandAmplif GainDiscrimination    90.15       0       0        0

HADGainThresh   GainDiscrimination  1000000   50000     120      2.4

# A function-based modifier       ElinNom  ElinSig  TlinNom  TlinSig
HADDigiIdent    SmearedLinear           1        0        1        0
.end ----------------------------------------------------------------
```

Using JAS3 to look at output

JAS3

File Edit View Tuple Loop LCD LCIO Window Help

digisim.slcio

DataSets
digisim.slcio

Welcome | LCSim Event

Run:0 Event: 0

Event
TkrEndcapHits
EcalEndcapHits
VtxBarrHits
HcalBarrRaw2sim
HcalBarrHits
HcalEndcapHits
EcalBarrRaw2sim
EcalBarrRawHits
MuonBarrHits
LumEndcapHits
MCParticle
TkrBarrHits
HcalBarrRawHits
EcalBarrHits
MuonEndcapHits

From DigiSim

Collection: HcalBarrRawHits size:735 flags:8000000

CellID	Amplitude	TimeStamp
390	592	83
385	217	3768
385	577	646
384	662	155
390	708	4179
384	286	3251
385	11245	68
401	158	106
384	146	66
386	660	5088
385	9827	62
398	279	84
384	299	2764
391	1914	2323
385	1166	2624
396	2271	68
397	41681	70
401	561	105
388	1897	70

Analyzed 1 records in 901ms

4.59/5.59MB

Using JAS3 to look at output

JAS3

File Edit View Tuple Loop LCD LCIO Window Help

digisim.slcio

DataSets

Welcome | LCSim Event

Run:0 Event: 0

- Event
  - TkrEndcapHits
  - EcalEndcapHits
  - VtxBarrHits
  - HcalBarrRaw2sim
  - HcalBarrHits
  - HcalEndcapHits
  - EcalBarrRaw2sim
  - EcalBarrRawHits
  - MuonBarrHits
  - LumEndcapHits
  - MCParticle
  - TkrBarrHits
  - HcalBarrRawHits
  - EcalBarrHits
  - MuonEndcapHits

Collection: EcalBarrRaw2sim size:390 flags:80000000

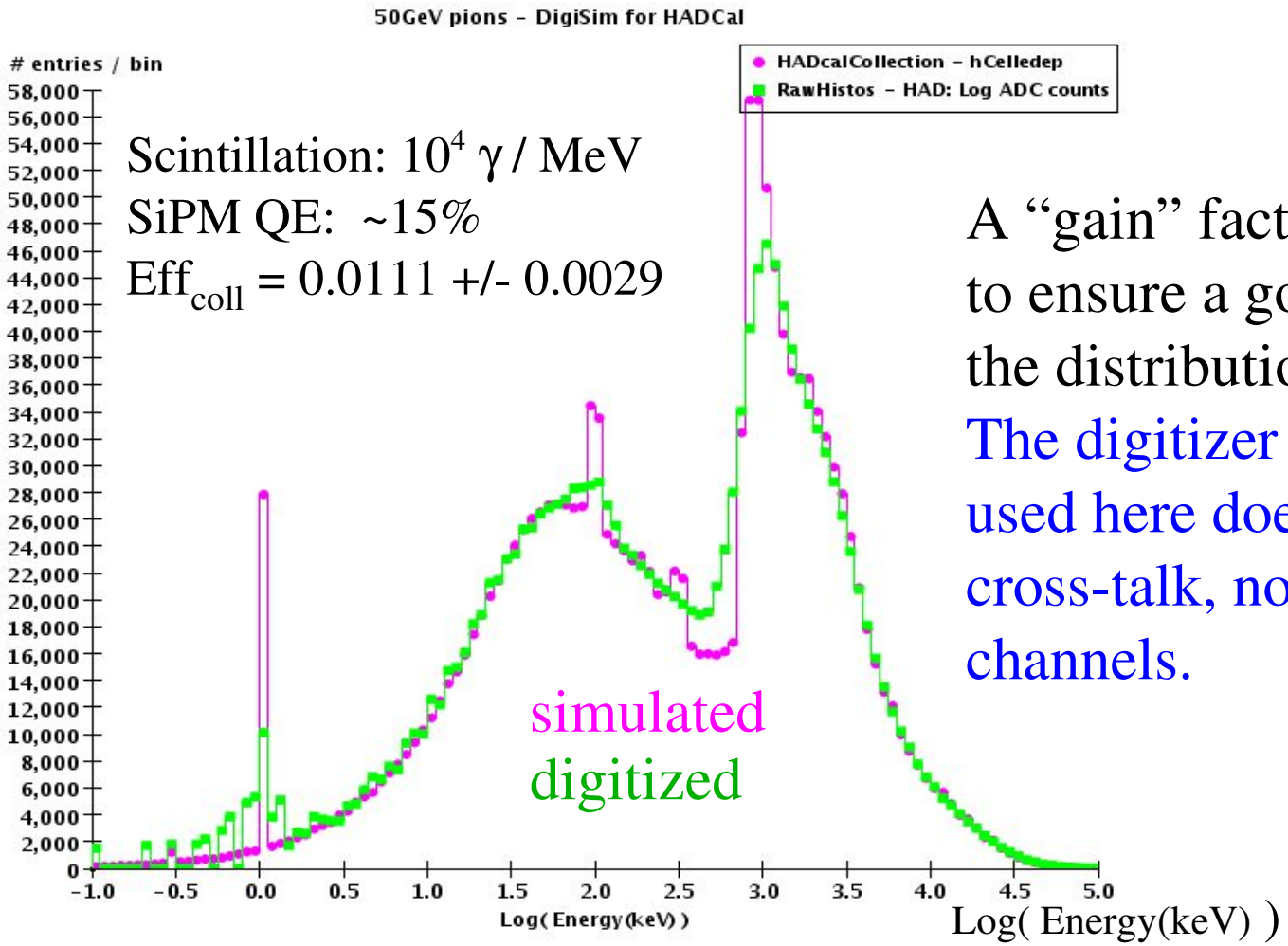
From	To	Weight
EcalBarrRawHits[0]	EcalBarrHits[313]	1.0000
EcalBarrRawHits[1]	EcalBarrHits[368]	1.0000
EcalBarrRawHits[2]	EcalBarrHits[129]	1.0000
EcalBarrRawHits[3]	EcalBarrHits[7]	1.0000
EcalBarrRawHits[4]	EcalBarrHits[149]	1.0000
EcalBarrRawHits[5]	EcalBarrHits[109]	1.0000
EcalBarrRawHits[6]	EcalBarrHits[111]	1.0000
EcalBarrRawHits[7]	EcalBarrHits[260]	1.0000
EcalBarrRawHits[8]	EcalBarrHits[372]	1.0000
EcalBarrRawHits[9]	EcalBarrHits[156]	1.0000
EcalBarrRawHits[10]	EcalBarrHits[177]	1.0000
EcalBarrRawHits[11]	EcalBarrHits[90]	1.0000
EcalBarrRawHits[12]	EcalBarrHits[91]	1.0000
EcalBarrRawHits[13]	EcalBarrHits[345]	1.0000
EcalBarrRawHits[14]	EcalBarrHits[154]	1.0000
EcalBarrRawHits[15]	EcalBarrHits[412]	1.0000
EcalBarrRawHits[16]	EcalBarrHits[409]	1.0000
EcalBarrRawHits[17]	EcalBarrHits[248]	1.0000
EcalBarrRawHits[18]	EcalBarrHits[192]	1.0000

Analyzed 1 records in 486ms

4.90/5.73MB

Crosstalk will produce weights < 1.0

HCal digitization (1st attempt)



Scintillation: 10^4 γ / MeV

SiPM QE: ~15%

Eff_{coll} = 0.0111 +/- 0.0029

A "gain" factor was used to ensure a good overlap of the distributions shown.

The digitizer configuration used here does not include cross-talk, noise or hot/dead channels.

Log(Energy(keV))

DigiSim Status

- A digitization simulation package, DigiSim, has been developed at NICADD/NIU
  - Both Java and C++ versions are available (same basic structure)
  - Generic modifiers available: GainDiscrimination and function-based SmearedLinear
  - LCRelations implemented to associate raw hits to corresponding simulated hits
- DigiSim can be run in either a stand-alone mode to produce persistent lcio output, or as an on-the-fly preprocessor to reconstruction/analysis.
  In the former case, raw hits and LCRelations are saved into the output LCIO files, in addition to all the (untouched) MC information present at DigiSim input.
- A first version of a real (though incomplete) tile HCal barrel digitizer exists.
  It can be used as an example to implement other subdetectors, like endcaps, ECal, RPCs/GEMs. See example presented in this talk.
- When reuse of existing generic modifiers is not appropriate, the creation of new modifiers is quite easy, by just copying one of the existing modifier classes and implementing the desired transformation.

DigiSim status (cont.)

- Both C++ and Java versions are available through official CVS servers
  C++: http://www-zeuthen.desy.de/lc-cgi-bin/cvsweb.cgi/calice_sim/digitization/digisim/?cvsroot=calice
  Java: released in the org.lcsim CVS repository. Download instructions from http://www.lcsim.org
- Some modifiers, like crosstalk and cell ganging, require neighborhood information from geometry-aware classes, which are currently under development.
- Digitization of tracking detectors hits can probably be implemented as easily as for calorimeter hits, as shown in this talk.
- Documentation available at http://nicadd.niu.edu/digisim/DigiSim.html, including download and building instructions (needs updates regarding the Java version)
- Send any questions or comments to me: lima at nicadd.niu.edu