CALICE testbeam data model and preparations

G.Mavromanolakis, University of Cambridge

Outline

- **Data models**
- **Data flowchart**
- **GUI application**

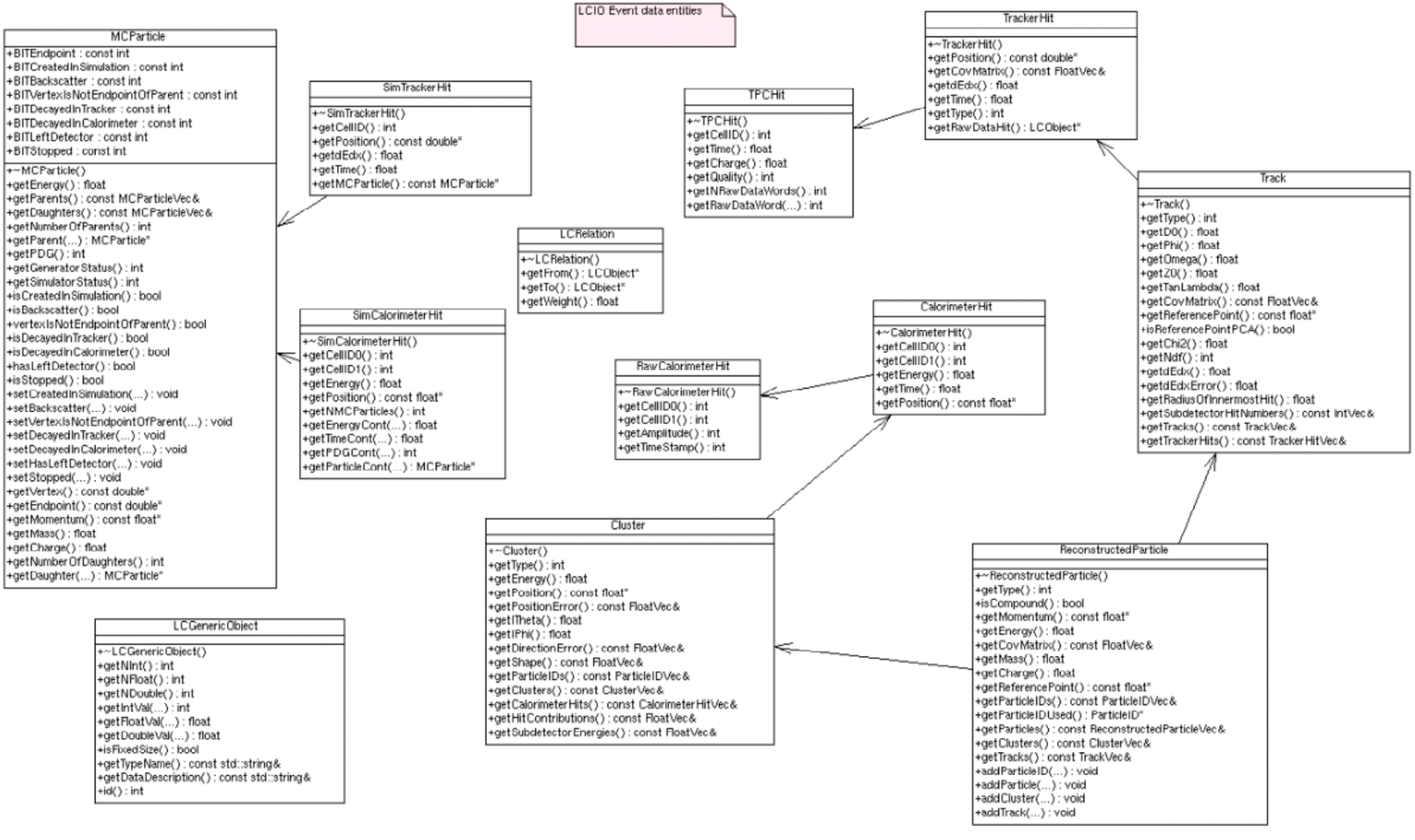
Testbeam data model

- .
  proposing a **data model** for the CALICE testbeam program
  - ▷ persistency
  - ▷ flexible implementation
  - ▷ simple user interface
  - ▷ efficiency

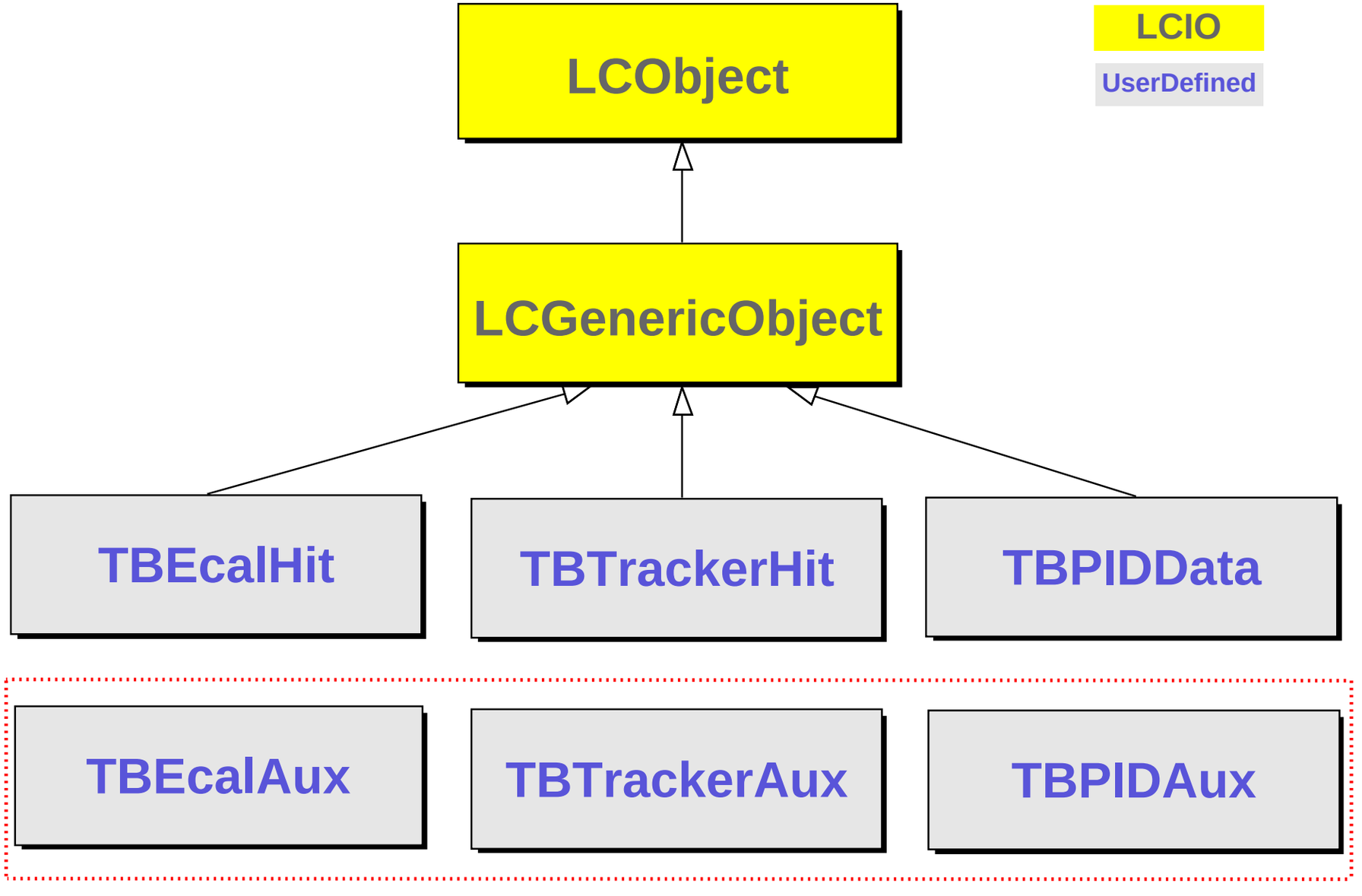
- .
  use LCIO and ROOT frameworks for some simple test implementation and benchmarking

- .
  general conversion scheme discussed
  (from raw/simulation data to analysis data)

LCIO event data entities (v01-03)



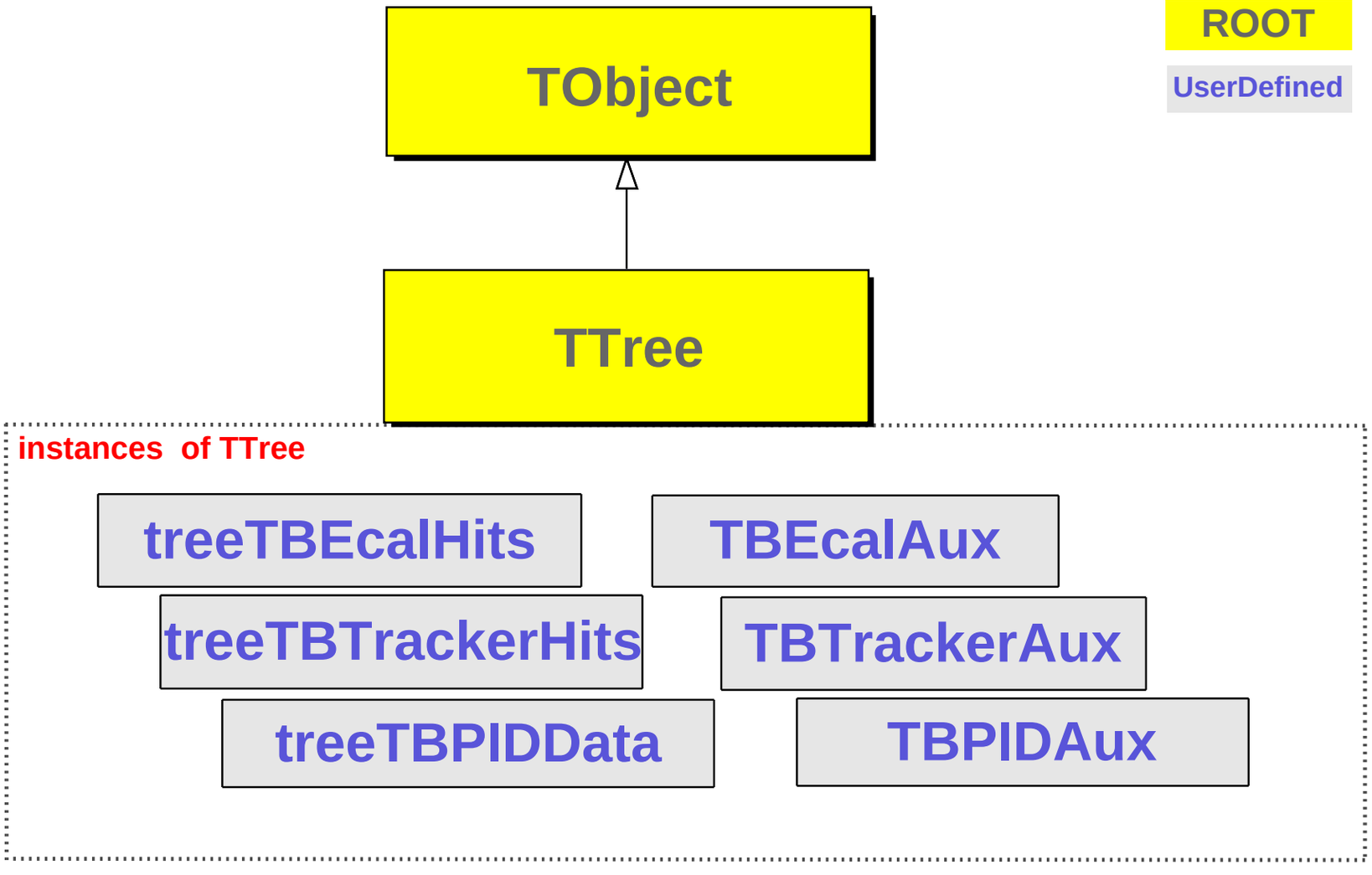
CALICE testbeam data model



example class TBEcalAux

```
//////////////////////////////////////////////////////////////////////////////
//
// author     : G.Mavromanolakis
//
// description: user defined concrete class derived from LCGenericObject class
//
// comments   : uses LCIO v01-03
//
//////////////////////////////////////////////////////////////////////////////


#ifndef class_TBEcalAux
#define class_TBEcalAux

#include "lcio.h"
#include "EVENT/LCGenericObject.h"
#include "EVENT/LCObject.h"

using namespace lcio;


//////////////////////////////////////////////////////////////////////////////
class TBEcalAux : public LCGenericObject
{
       private:
                  int theK;
                  int theL;
                  int theM;
                float theX;
                float theY;
                float theZ;

       public:
                  TBEcalAux();
                  TBEcalAux(int*,float*);
                  TBEcalAux(LCObject*);
                 ~TBEcalAux();

   const static std::string TypeName;
   const static std::string DataDescription;

     const static int NumOfIntegers;
     const static int NumOfFloats;
     const static int NumOfDoubles;
           static int counter;

                  int GetK();//..... User Interface
                  int GetL();//  .
                  int GetM();//  .
                float GetX();//  .
                float GetY();//  .
                float GetZ();//...

                  int getNInt() const;             //..... LCIO Interface
                  int getNFloat() const;           //  .   (LCGenericObject
                  int getNDouble() const;          //  .    virtual methods)
                  int getIntVal(int index) const;  //  .
                float getFloatVal(int index) const; //  .
               double getDoubleVal(int index) const;//  .
                 bool isFixedSize() const;         //  .
   const std::string& getTypeName() const;         //  .
   const std::string& getDataDescription()const ;  //...

};
//////////////////////////////////////////////////////////////////////////////


#endif
```

alternative ...

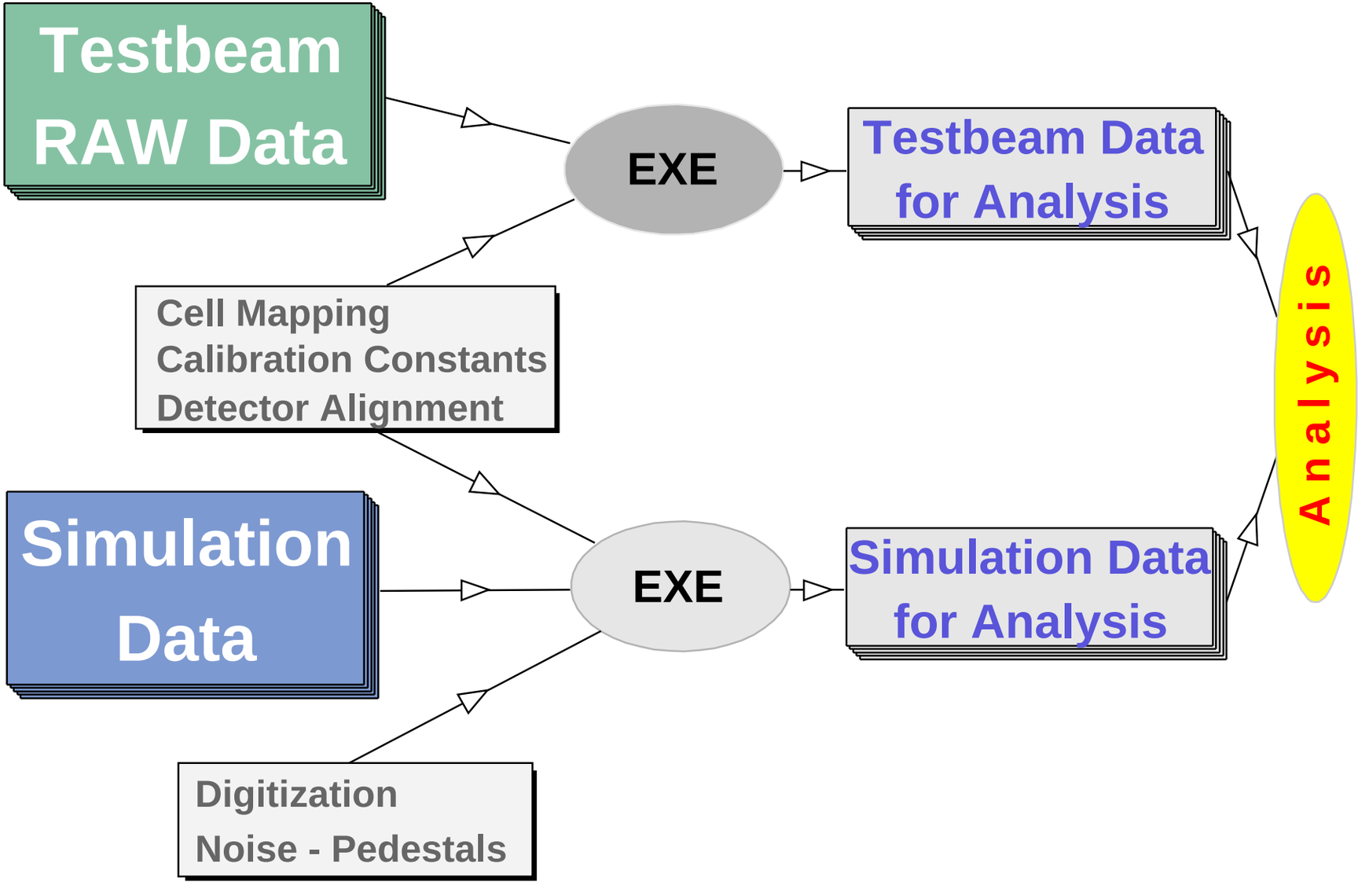


Benchmarks

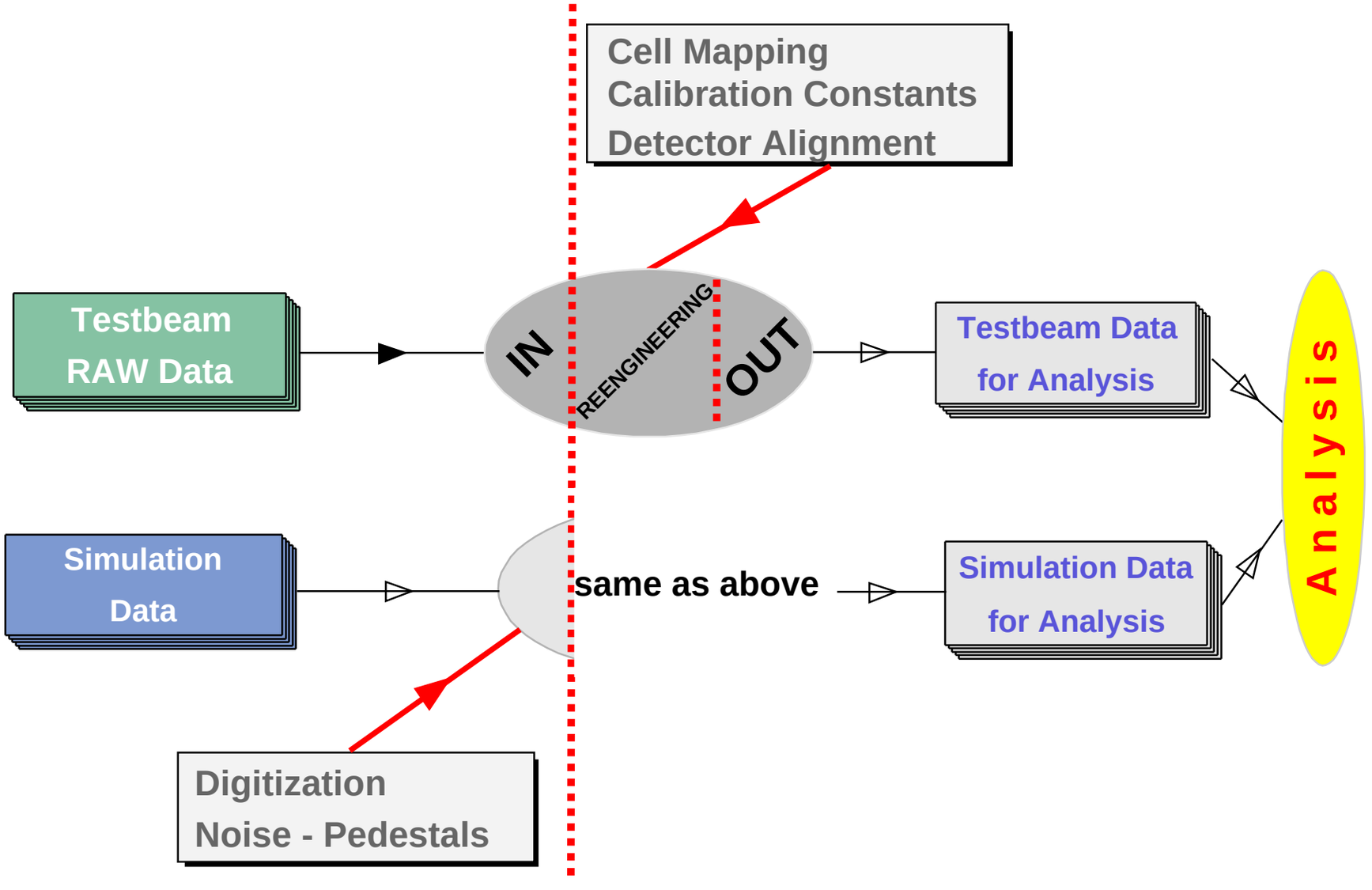
- **. configuration**
  - machine: Linux P4 2.66 GHz / 512 MB RAM
  - libs: ROOT v4.00/08 and LCIO v01-03
  - task: write/read 1 ROOT tree or 1 LCIO collection of N events $\times$ 100 hits (1 hit = 3 integers + 3 floats)

		LCIO	ROOT
100k events	**size (MB)**	**28**	**4**
	time write (sec)	**64**	**9**
	time read (sec)	**71**	**19**
500k events	**size (MB)**	**139**	**19**
	time write (sec)	**365**	**48**
	time read (sec)	**328**	**95**

Data flowchart



Data flowchart - details



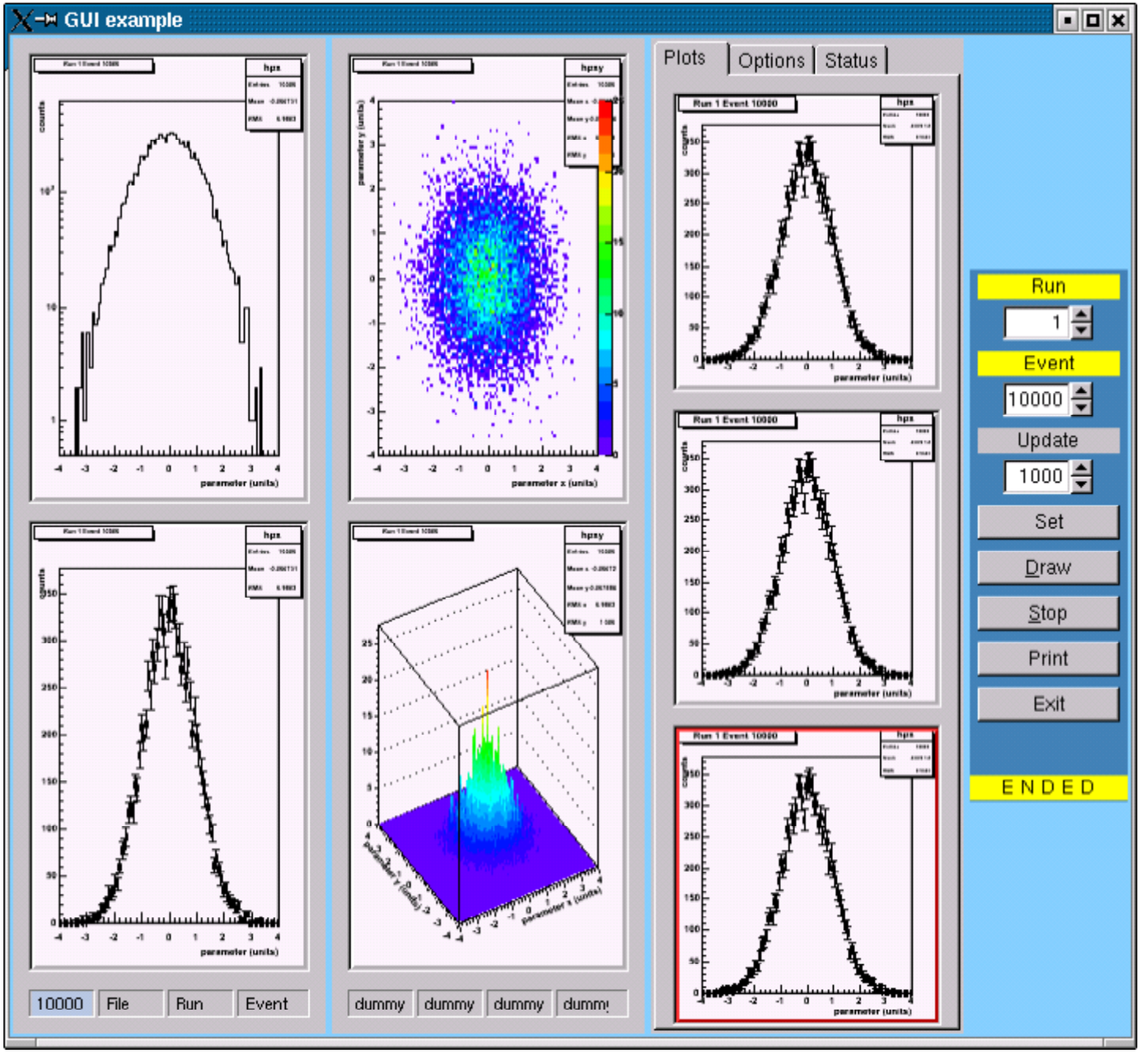
Graphical User Interface

- .
  cross-platform GUI application under development for
  - online histogramming
  - basic analysis
  - monitoring
  - event display

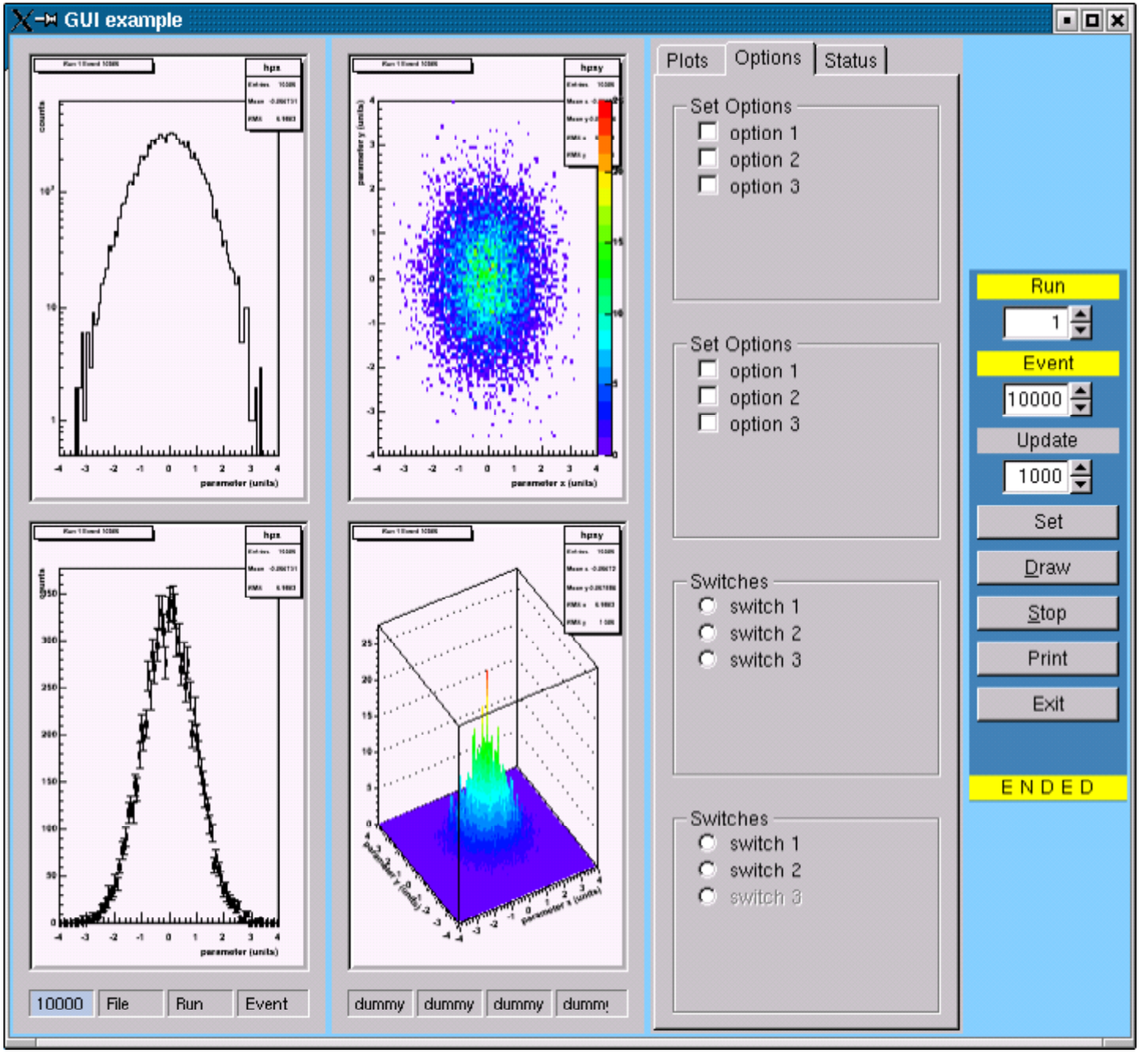
- .
  using ROOT GUI classes based on Xclass'95 widget library
  (`http://xclass.sourceforge.net`)

- .
  basic layout has been implemented and tested on
  Redhat 7.3 (gcc 3.2.3 or 2.96) with ROOT 4.00 or 3.10

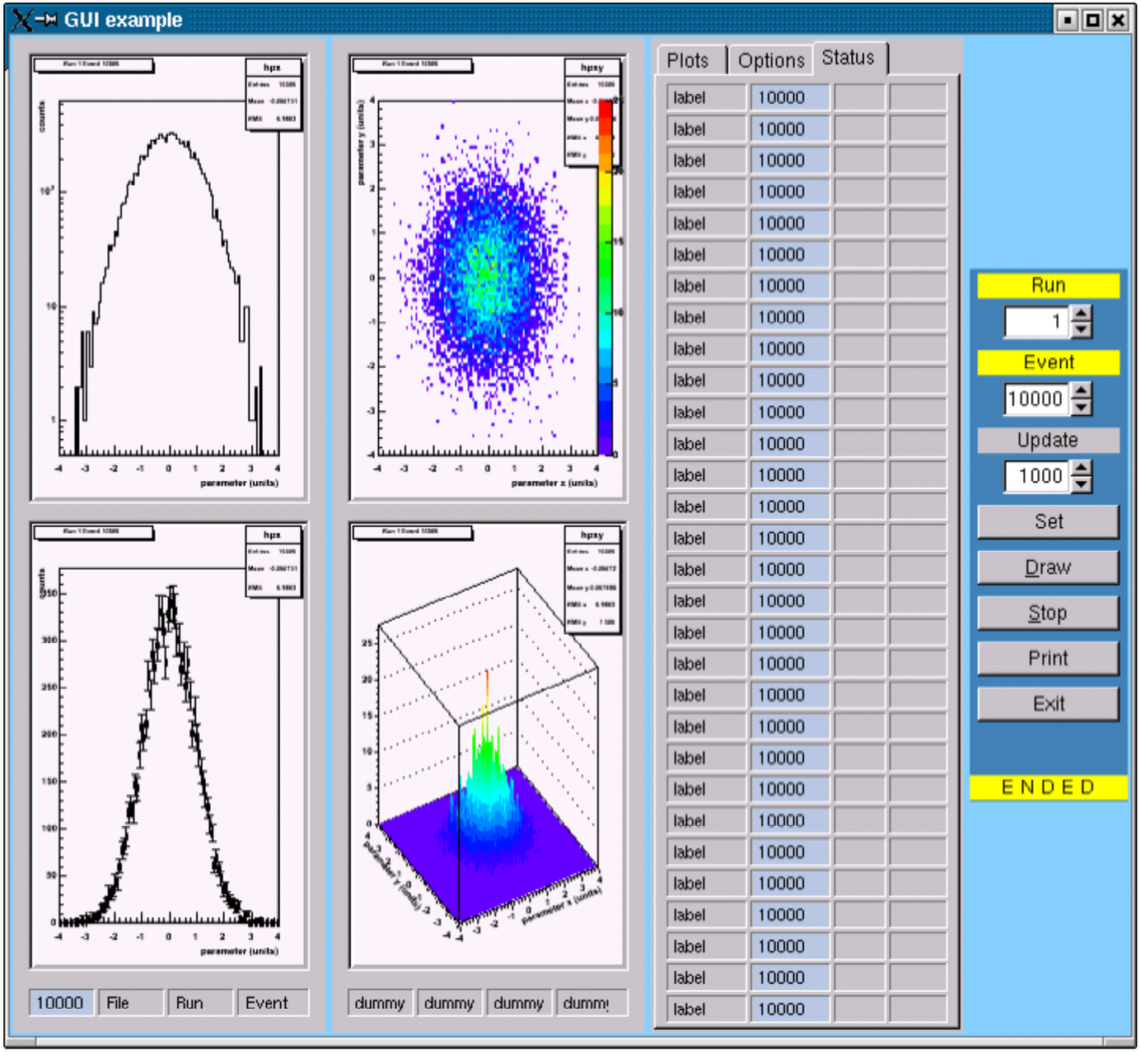
GUI: histograms and action widgets



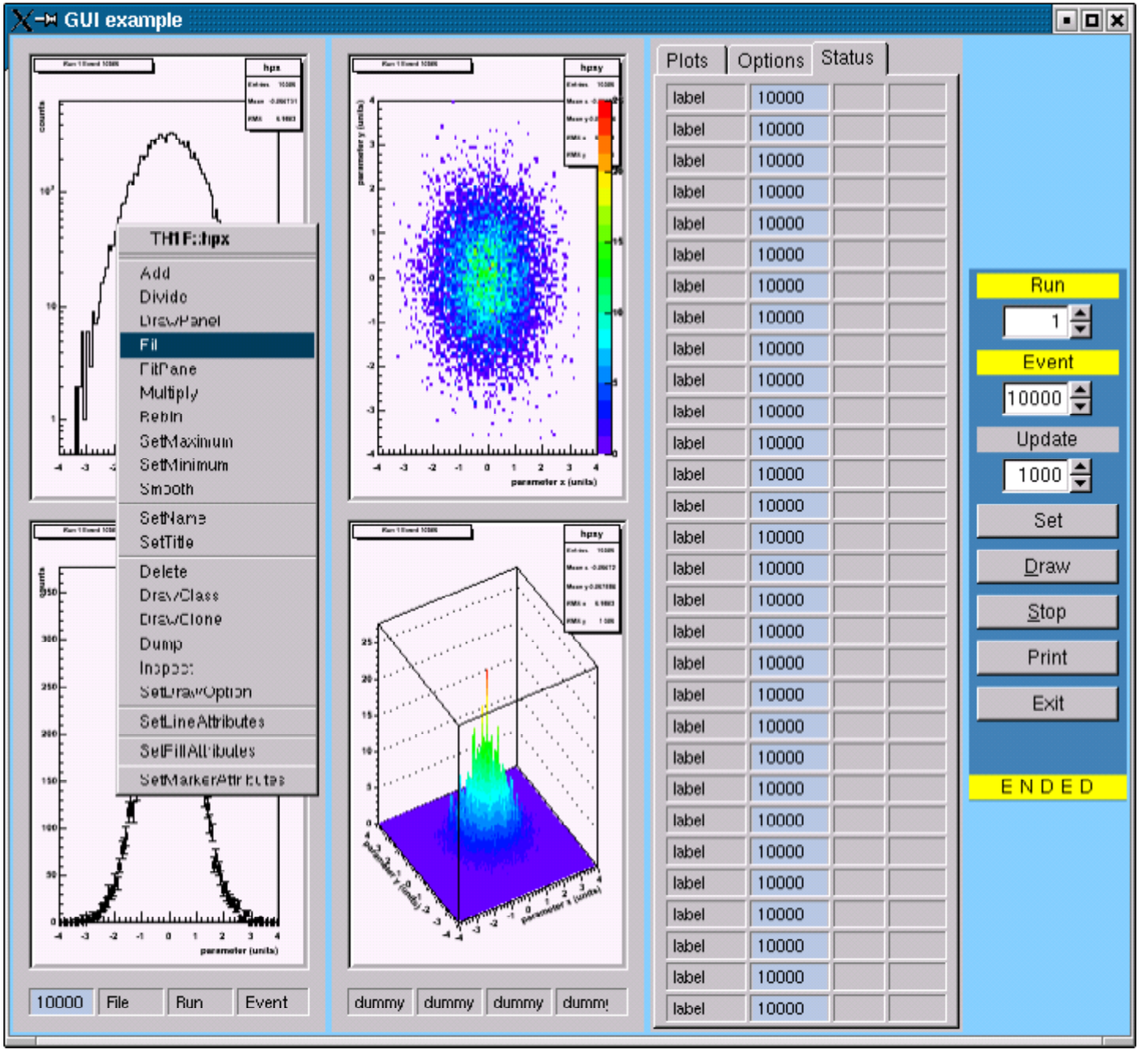
GUI: tabs for expanded info, settings



GUI: status table



GUI: active pads to invoke ROOT actions



Summary

- .
  : work on testbeam related software has started and rapid progress is expected

- .
  : the running and analysis phases of the testbeam program will be fully supported

Calorimeter clustering with gNIKI

general **N**odes **I**nterlaced **K**lustering **I**mplementation

G.Mavromanolakis, University of Cambridge

www.hep.phy.cam.ac.uk/~gmavroma/calice/gNIKI

Clustering with gNIKI

general Nodes Interlaced Klustering Implementation

► . **gNIKI**

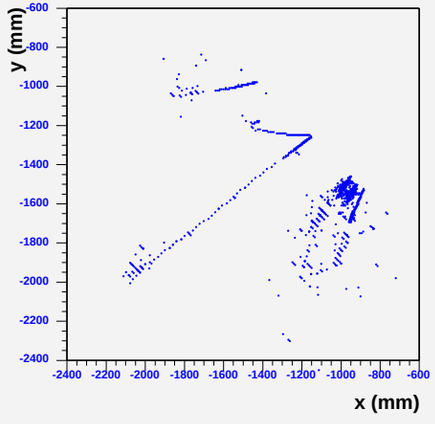
: algorithm based on minimal spanning tree theory to implement a "top-down and then bottom-up" approach to calorimeter clustering

► . **in brief**

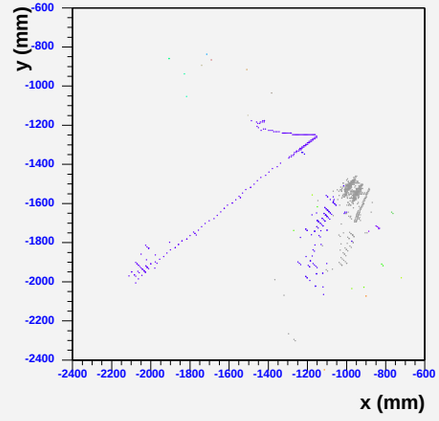
: use MST clustering algorithm with loose cut to perform coarse clustering

: then go through MST clusters found in previous step and refine using a cone-like energy flow clustering algorithm

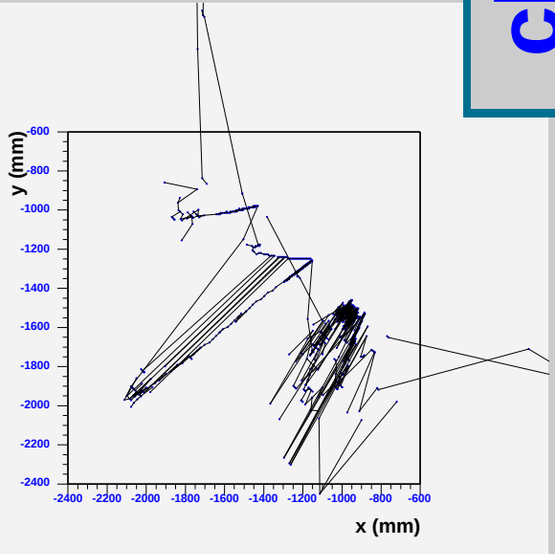
hits



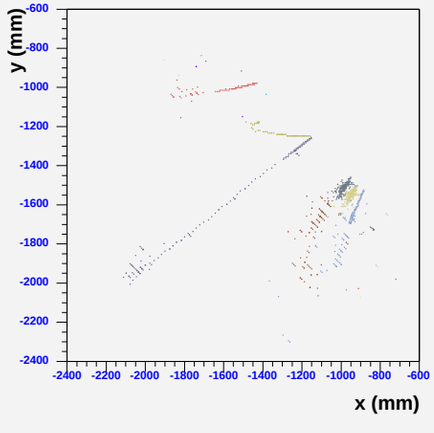
cluster MST



MST



recluster



d041130 **LC Note: ... ...**

Calorimeter clustering with gNIKI

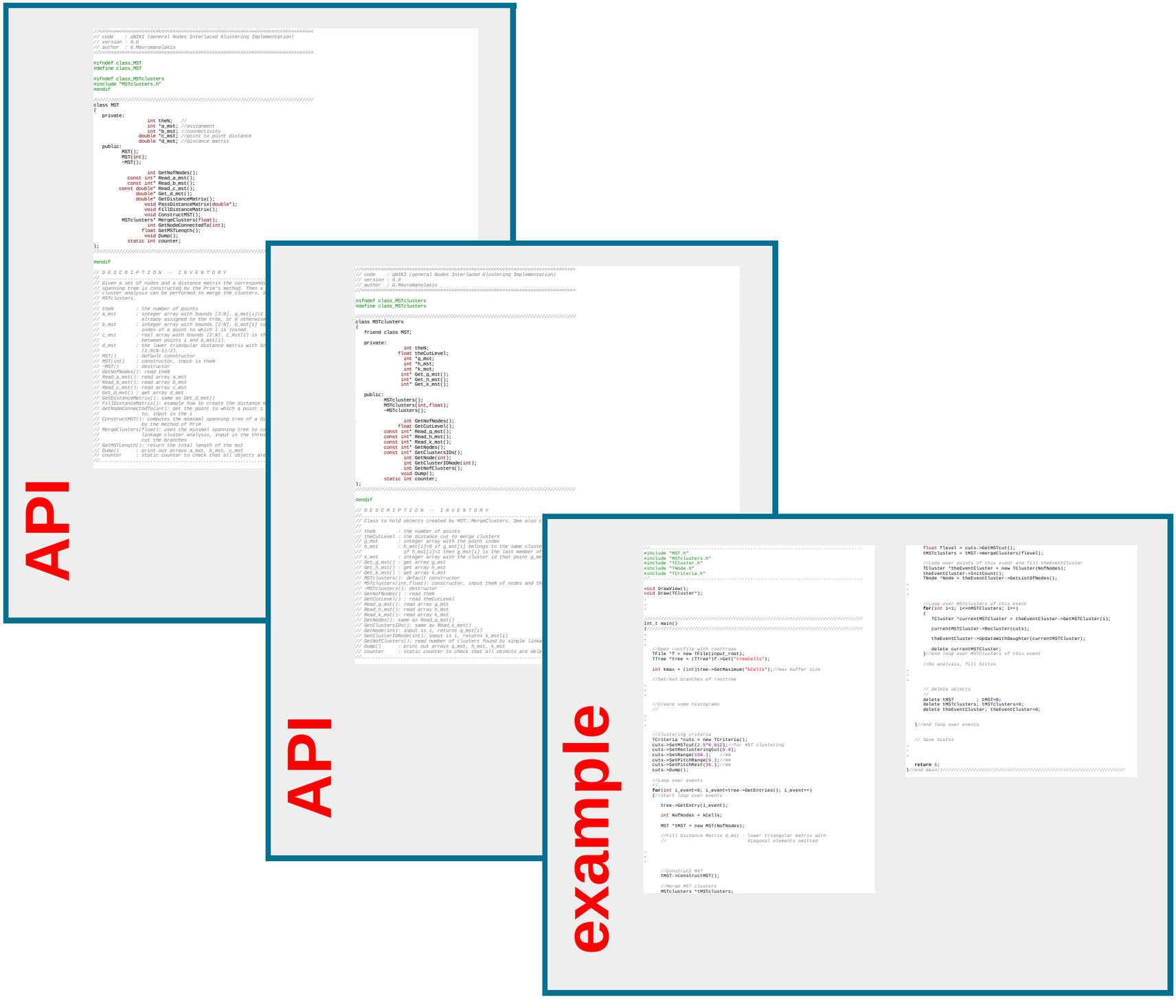
general Nodes Interlaced Klustering Implementation

G. Mavromanolakis *

University of Cambridge, Department of Physics
Cavendish Laboratory, High Energy Physics Group
Madingley Road, Cambridge, CB3 0HE, UK

Contents

* email: gmavroma@hep.phy.cam.ac.uk or gmavroma@mail.cern.ch


API
API
example

gNIKI

► **. release version 0.0**

: standalone C++ code arranged in 6 classes
(programmatically not the best code you've ever seen but it works)

: get source from
www.hep.phy.cam.ac.uk/∼gmavroma/calice/gNIKI

: example applications on howto link and read data from
ROOT or LCIO included

► **.**

: write-up as LC Note in preparation (hopefully available soon),
in the meantime a list of talks and documents are on the webpage