The LCG Software
and the ROOT Framework

LCG seminar

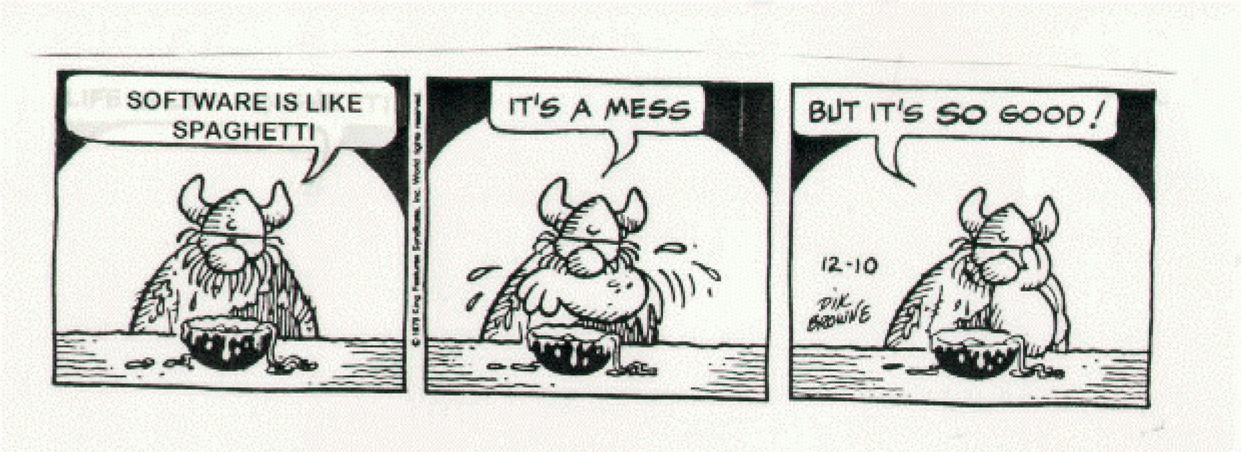
CERN- 7 November

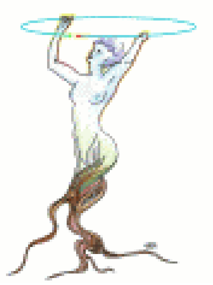
René Brun

ftp://root.cern.ch/root/lcgtalk.ppt

Plan of talk

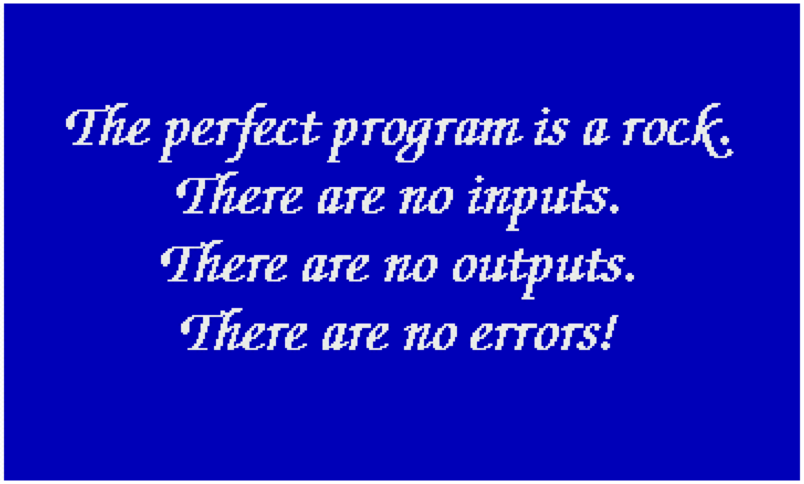
- General considerations
- ROOT status & plans
  - what we presented at the Blueprint RTAG
  - required developments
  - modularity issues
- Relations with LCG
  - Where we agree
  - Possible problems
  - Scenario 1, 2, 3





General Considerations

Some time-invariant messages



The perfect program is a rock.
There are no inputs.
There are no outputs.
There are no errors!

Some basic rules (1)

When starting a new project, many unexpected obstacles. Must explore many ways before building the final path.

Some basic rules (2)

The development model is crucial.

A "well-designed" system that looks pretty on paper, but not validated early enough by users may end up into a very fat and unusable system.

Some basic rules (3)

Cooperation between projects is difficult.

Different development styles, Team experience.

One more reason for rapid prototyping. Do not wait the end to see the product. Develop the product with users.

HEY, I THINK WE'RE REALLY CLOSE ON OUR SPECS!

Some basic rules (4)

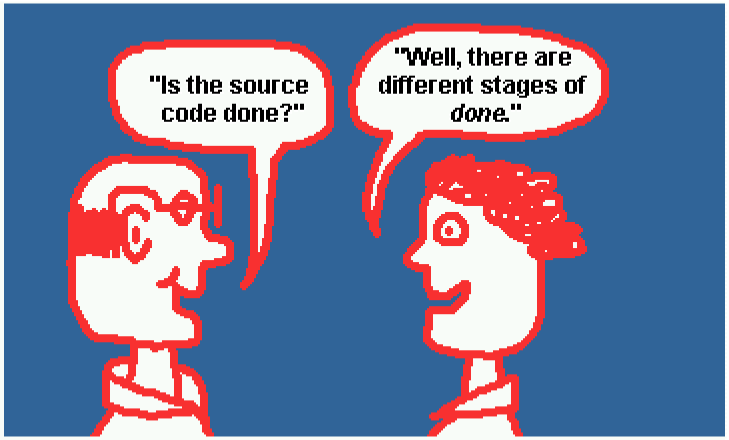
If you have more than pi^2 years experience in managing software projects, multiply by pi your estimation to complete your next project

else multiply by pi^2 R.Brun

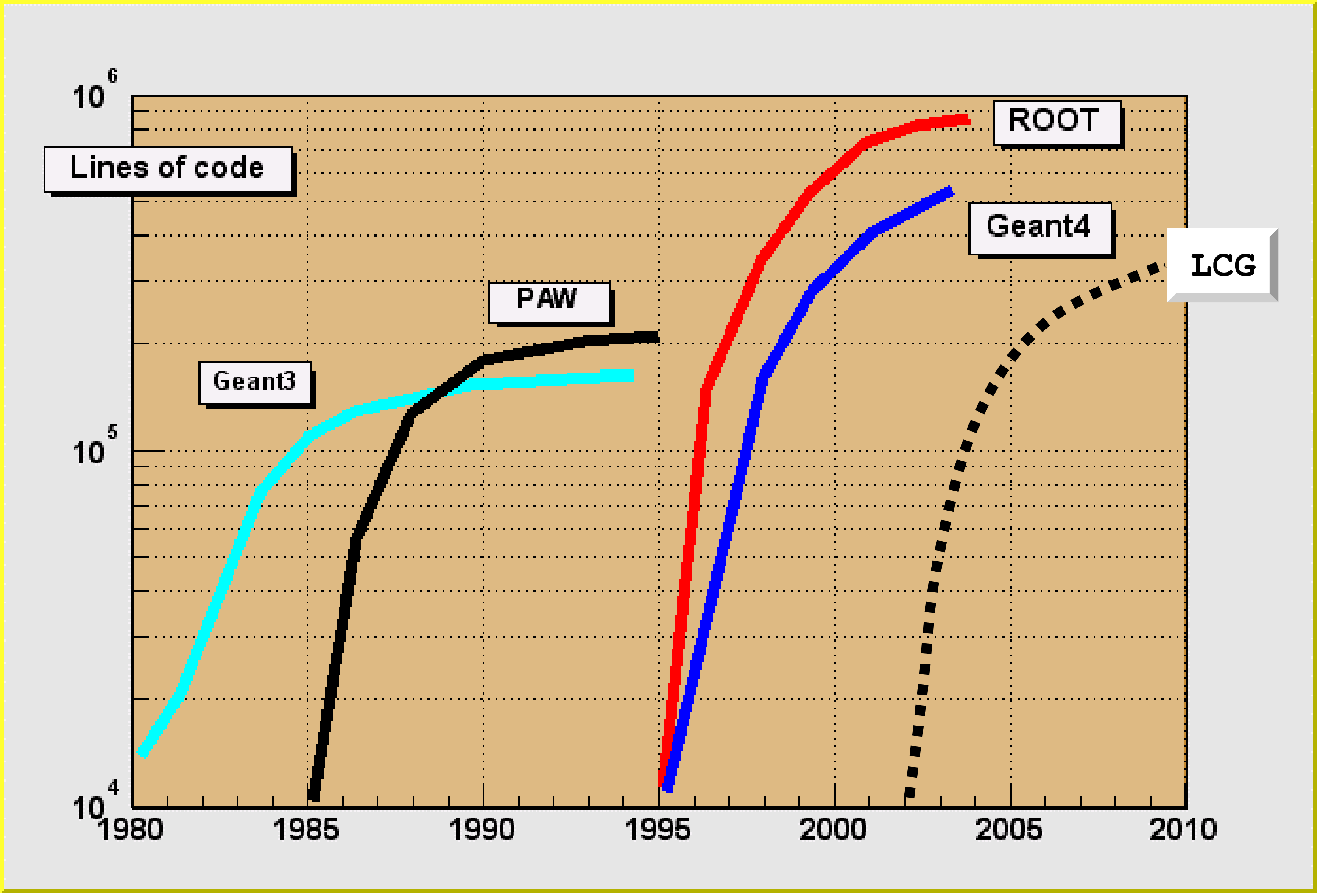
Nine women working together have never made a baby in one month.

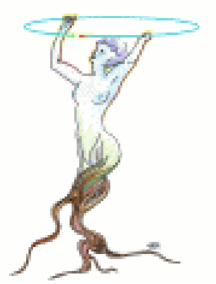
If T is the time to implement a system and N the number of participants $T = a/N - b*N^2$

Ideal team size 3 to 6 people


"Is the source code done?"
"Well, there are different stages of *done*."

Time to develop





ROOT status/plans

The team

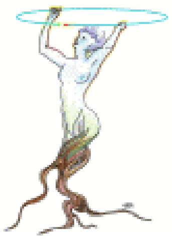
Measuring success

The main components

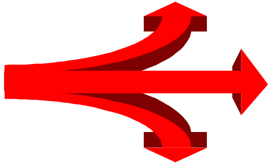
The development plans

Project History

8 years !!



- Jan 95: Thinking/writing/rewriting/???
- November 95: Public seminar, show Root 0.5
- Spring 96: decision to use CINT
- Jan 97: Root version 1.0
- Jan 98: Root version 2.0
- Mar 99: Root version 2.21/08 (1st Root workshop FNAL)
- Feb 00: Root version 2.23/12 (2nd Root workshop CERN)
- Mar 01: Root version 3.00/06
- Jun 01: Root version 3.01/05 (3rd Root workshop FNAL)
- Jan 02: Root version 3.02/07 (LCG project starts: RTAGs)
- Oct 02: Root version 3.03/09 (4th Root workshop CERN)



ROOT Project basic principles

- Born with minimal CERN staff (by constraint)
- Try to involve as many developers as possible from outside CERN ==>Open Source project
- Try to get as many users asap to validate the ideas. We knew many PAW users.
- Release early, Release often principle
- Target maximum portability (OS & compilers)
- Target maximum functionality and simplicity
- Roottalk newsgroup: vital importance

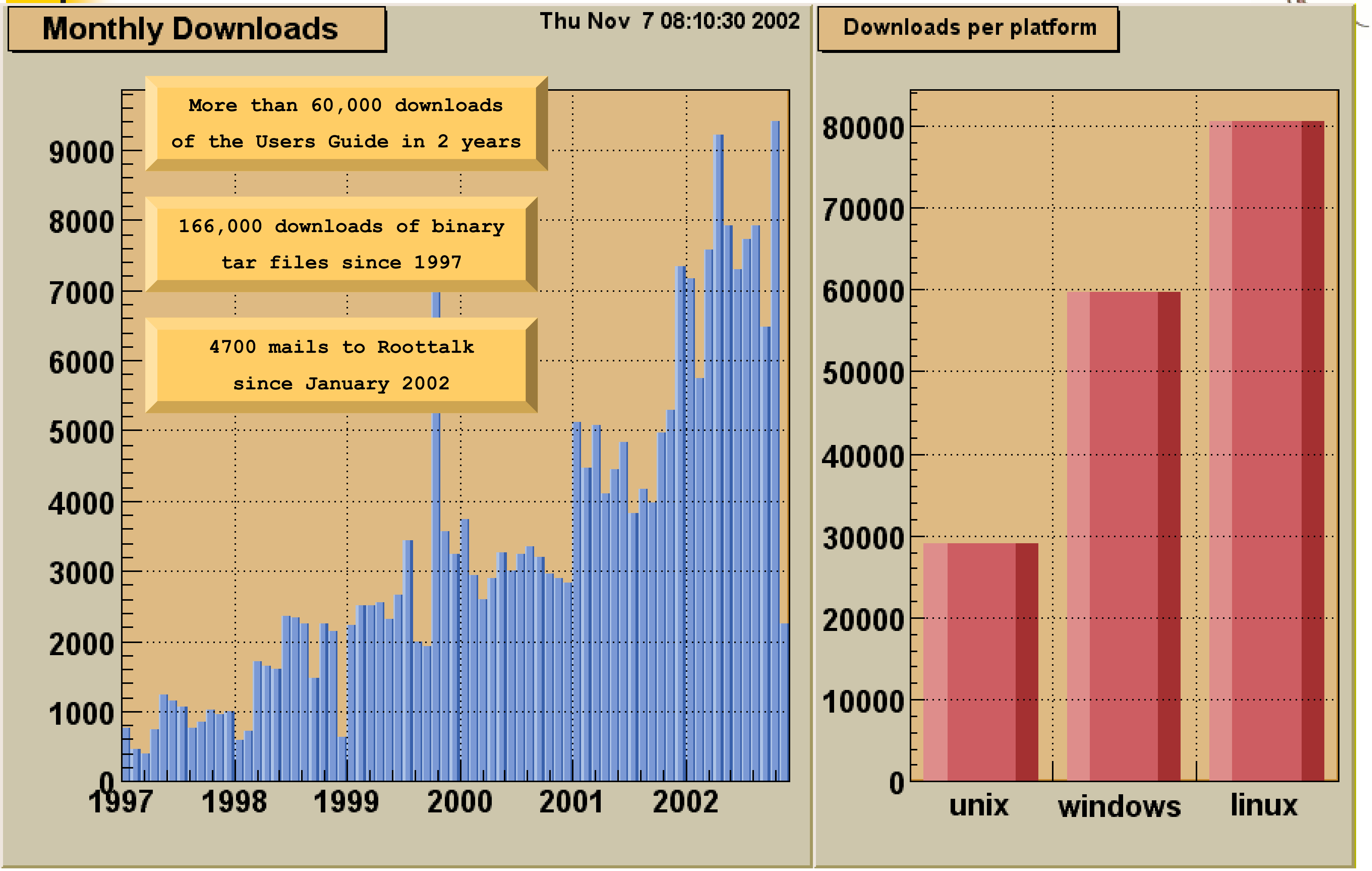
ROOT Team & Associates

- ROOT Team:
  - Ilka Antcheva (LCG staff) (since 1st Aug 2002)
  - Rene Brun: new SFT group and Alice part time
  - Philippe Canal (FNAL/CD) (since 1998)
  - Olivier Couet CERN/IT/API (from PAW) (since 1st Jun 2002)
  - Gerardo Ganis (LCG/EP/SFT) starting just now
  - Masa Goto (Agilent technologies/Japan)
  - Valeriy Onuchin (LCG project associate) (since 1st Feb 2002)
  - Fons Rademakers: Alice and new SFT group
- External Associates
  - Bertrand Bellenot (Alcan) Win32gdk (since June 2000)
  - Maarten Ballintijn (MIT) PROOF (since Sep 2001)
  - Andrei Gheata: (Alice) Geometry package (since Sep 2001)

ROOT Team & Associates (2)

- Now in the LCG
  - Valery Fine (BNL/Atlas) TVirtualX/Qt
  - Victor Perevoztchikov (BNL/Atlas) STL, foreign classes
- More than 50 important contributions from people spending a substantial fraction of their time on the project. See **$ROOTSYS/README/CREDITS**
- Special thanks to Suzanne Panacek who did a great job with the ROOT Users Guide, tutorials, lectures.
  - **Printed copies of the Users Guide in my office**.
- Many thanks to FNAL computing Division for the continuous support of the project since 1998.

ROOT ftp Downloads

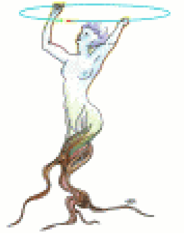


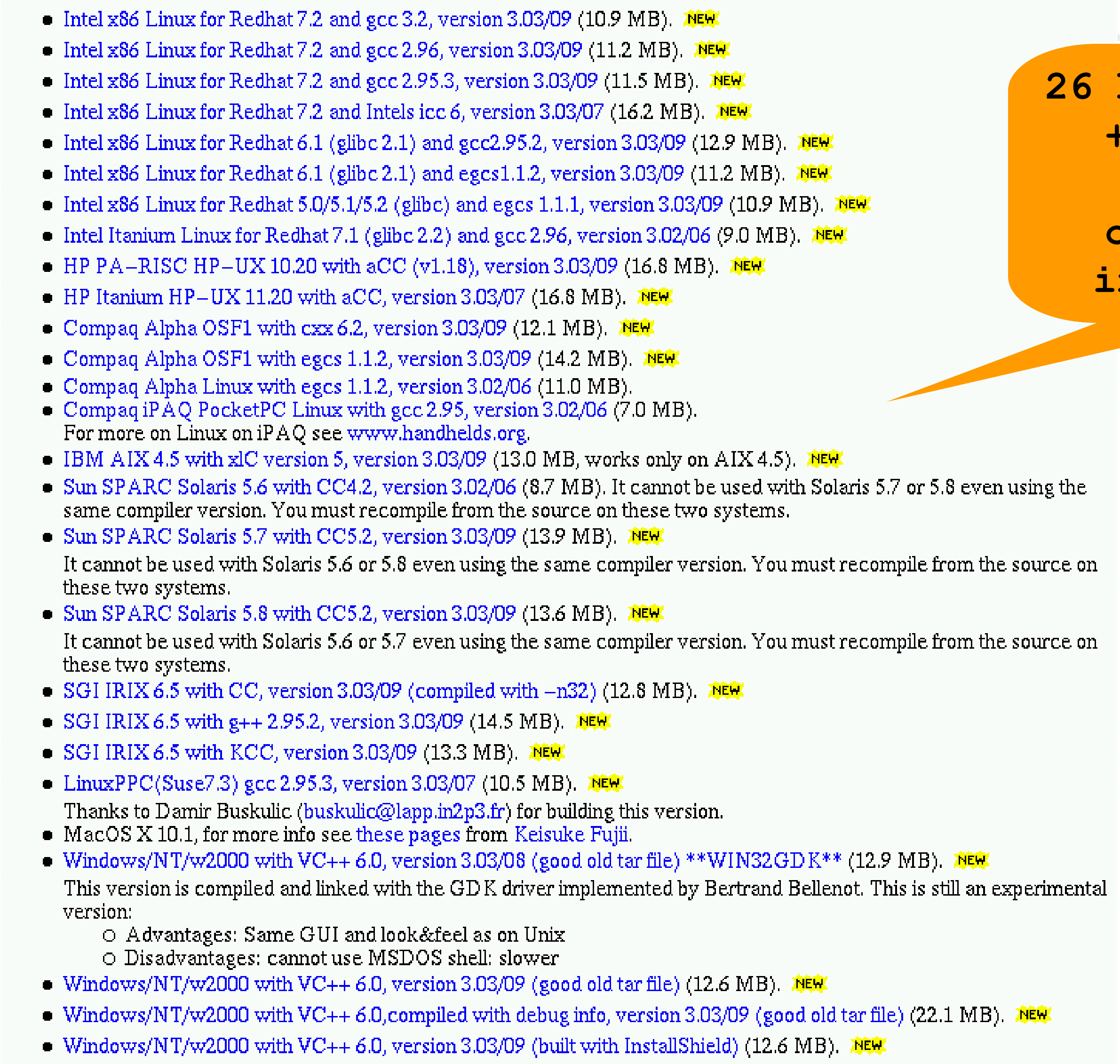
More than 60,000 downloads of the Users Guide in 2 years

166,000 downloads of binary tar files since 1997

4700 mails to Roottalk since January 2002

Source & Binary distributions





- Intel x86 Linux for Redhat 7.2 and gcc 3.2, version 3.03/09 (10.9 MB). NEW
- Intel x86 Linux for Redhat 7.2 and gcc 2.96, version 3.03/09 (11.2 MB). NEW
- Intel x86 Linux for Redhat 7.2 and gcc 2.95.3, version 3.03/09 (11.5 MB). NEW
- Intel x86 Linux for Redhat 7.2 and Intels icc 6, version 3.03/07 (16.2 MB). NEW
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and gcc2.95.2, version 3.03/09 (12.9 MB). NEW
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and egcs1.1.2, version 3.03/09 (11.2 MB). NEW
- Intel x86 Linux for Redhat 5.0/5.1/5.2 (glibc) and egcs 1.1.1, version 3.03/09 (10.9 MB). NEW
- Intel Itanium Linux for Redhat 7.1 (glibc 2.2) and gcc 2.96, version 3.02/06 (9.0 MB). NEW
- HP PA–RISC HP–UX 10.20 with aCC (v1.18), version 3.03/09 (16.8 MB). NEW
- HP Itanium HP–UX 11.20 with aCC, version 3.03/07 (16.8 MB). NEW
- Compaq Alpha OSF1 with cxx 6.2, version 3.03/09 (12.1 MB). NEW
- Compaq Alpha OSF1 with egcs 1.1.2, version 3.03/09 (14.2 MB). NEW
- Compaq Alpha Linux with egcs 1.1.2, version 3.02/06 (11.0 MB).
- Compaq iPAQ PocketPC Linux with gcc 2.95, version 3.02/06 (7.0 MB).
  For more on Linux on iPAQ see www.handhelds.org.
- IBM AIX 4.5 with xlC version 5, version 3.03/09 (13.0 MB, works only on AIX 4.5). NEW
- Sun SPARC Solaris 5.6 with CC4.2, version 3.02/06 (8.7 MB). It cannot be used with Solaris 5.7 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.7 with CC5.2, version 3.03/09 (13.9 MB). NEW

  It cannot be used with Solaris 5.6 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.8 with CC5.2, version 3.03/09 (13.6 MB). NEW

  It cannot be used with Solaris 5.6 or 5.7 even using the same compiler version. You must recompile from the source on these two systems.
- SGI IRIX 6.5 with CC, version 3.03/09 (compiled with –n32) (12.8 MB). NEW
- SGI IRIX 6.5 with g++ 2.95.2, version 3.03/09 (14.5 MB). NEW
- SGI IRIX 6.5 with KCC, version 3.03/09 (13.3 MB). NEW
- LinuxPPC(Suse7.3) gcc 2.95.3, version 3.03/07 (10.5 MB). NEW

  Thanks to Damir Buskulic (buskulic@lapp.in2p3.fr) for building this version.
- MacOS X 10.1, for more info see these pages from Keisuke Fujii.
- Windows/NT/w2000 with VC++ 6.0, version 3.03/08 (good old tar file) **WIN32GDK** (12.9 MB). NEW

  This version is compiled and linked with the GDK driver implemented by Bertrand Bellenot. This is still an experimental version:
  - Advantages: Same GUI and look&feel as on Unix
  - Disadvantages: cannot use MSDOS shell: slower
- Windows/NT/w2000 with VC++ 6.0, version 3.03/09 (good old tar file) (12.6 MB). NEW
- Windows/NT/w2000 with VC++ 6.0,compiled with debug info, version 3.03/09 (good old tar file) (22.1 MB). NEW
- Windows/NT/w2000 with VC++ 6.0, version 3.03/09 (built with InstallShield) (12.6 MB). NEW

26 binary tar files
+ all possible
combinations
of OS/Compiler
in the Makefile

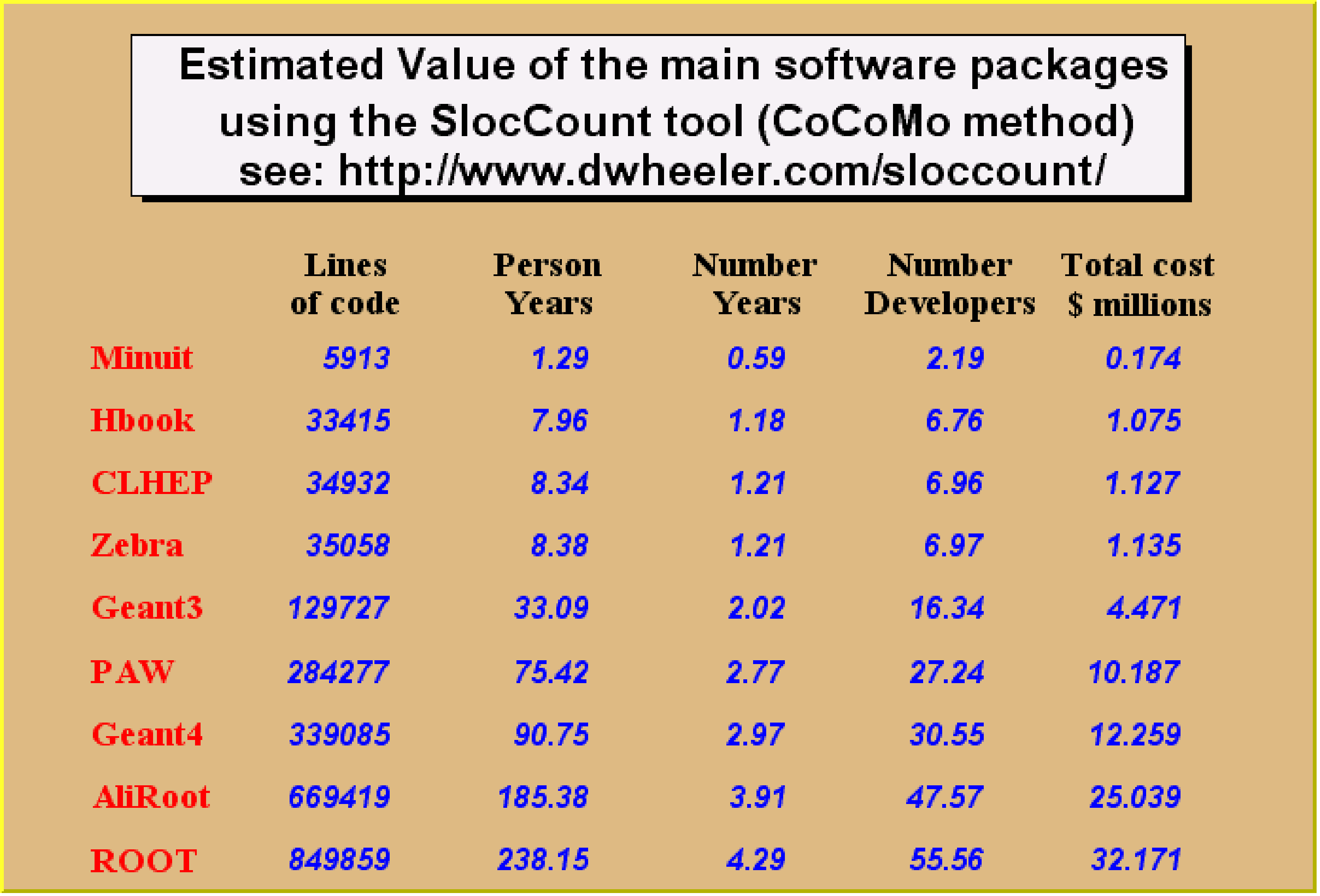
Unix(es)

Linux

Windows

MacosX

Cost to develop



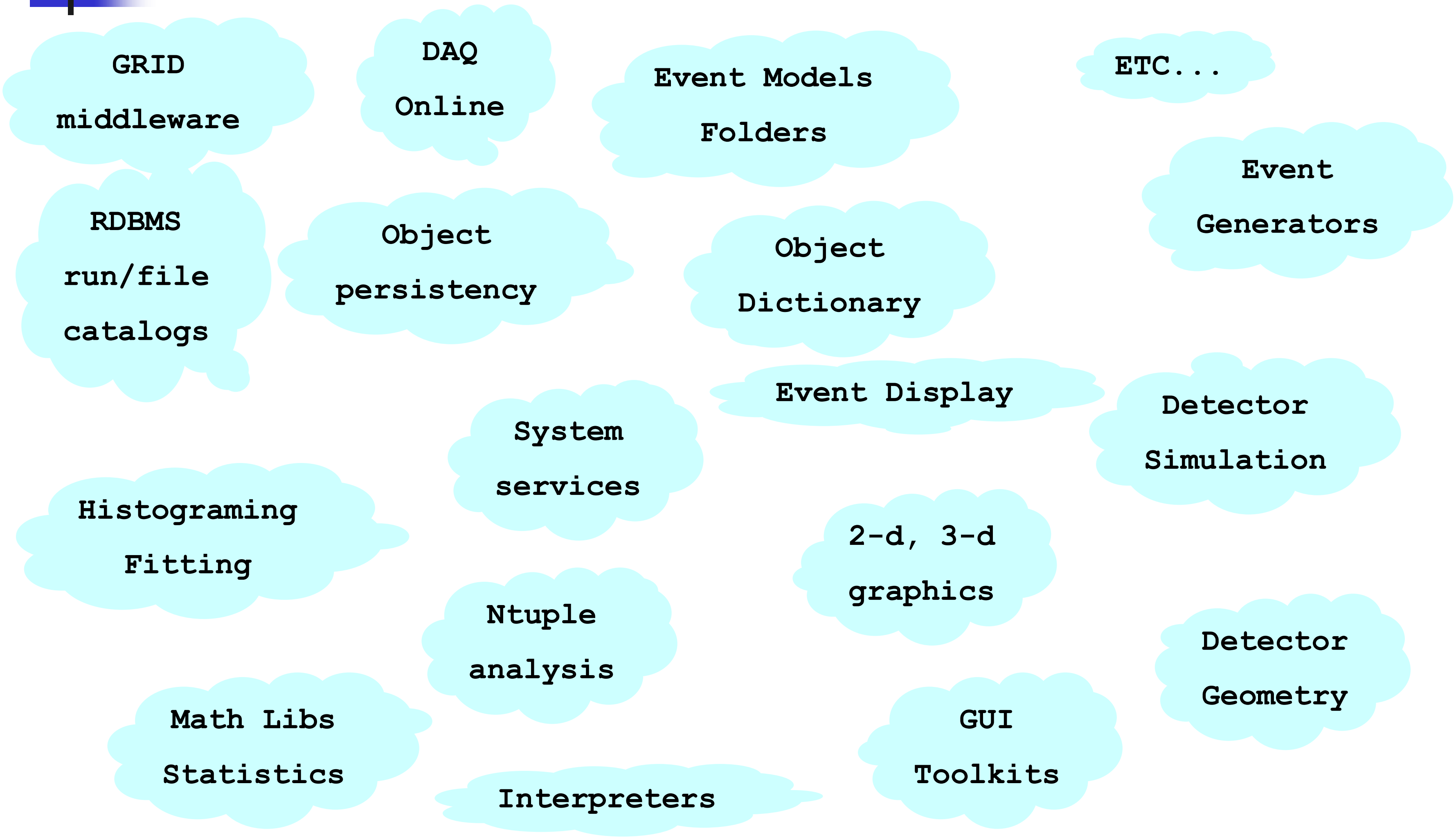
Estimated Value of the main software packages using the SlocCount tool (CoCoMo method) see: http://www.dwheeler.com/sloccount/

	Lines of code	Person Years	Number Years	Number Developers	Total cost $ millions
Minuit	5913	1.29	0.59	2.19	0.174
Hbook	33415	7.96	1.18	6.76	1.075
CLHEP	34932	8.34	1.21	6.96	1.127
Zebra	35058	8.38	1.21	6.97	1.135
Geant3	129727	33.09	2.02	16.34	4.471
PAW	284277	75.42	2.77	27.24	10.187
Geant4	339085	90.75	2.97	30.55	12.259
AliRoot	669419	185.38	3.91	47.57	25.039
ROOT	849859	238.15	4.29	55.56	32.171

ROOT : Powerful & Light-weight

- Tar ball < 15 Mbytes
- Source < 7 Mbytes
- Install from binary file : 1 minute
- No external dependencies
- Fast start-up (1 second)
- Low memory occupancy (< 15 Mbytes)
- Highly optimized algorithms for histograming, fitting, I/O, Tree queries, etc.
- and yet runs everywhere

The main software areas



GRID middleware

DAQ Online

Event Models Folders

ETC...

Event Generators

RDBMS run/file catalogs

Object persistency

Object Dictionary

Event Display

Detector Simulation

System services

Histograming Fitting

2-d, 3-d graphics

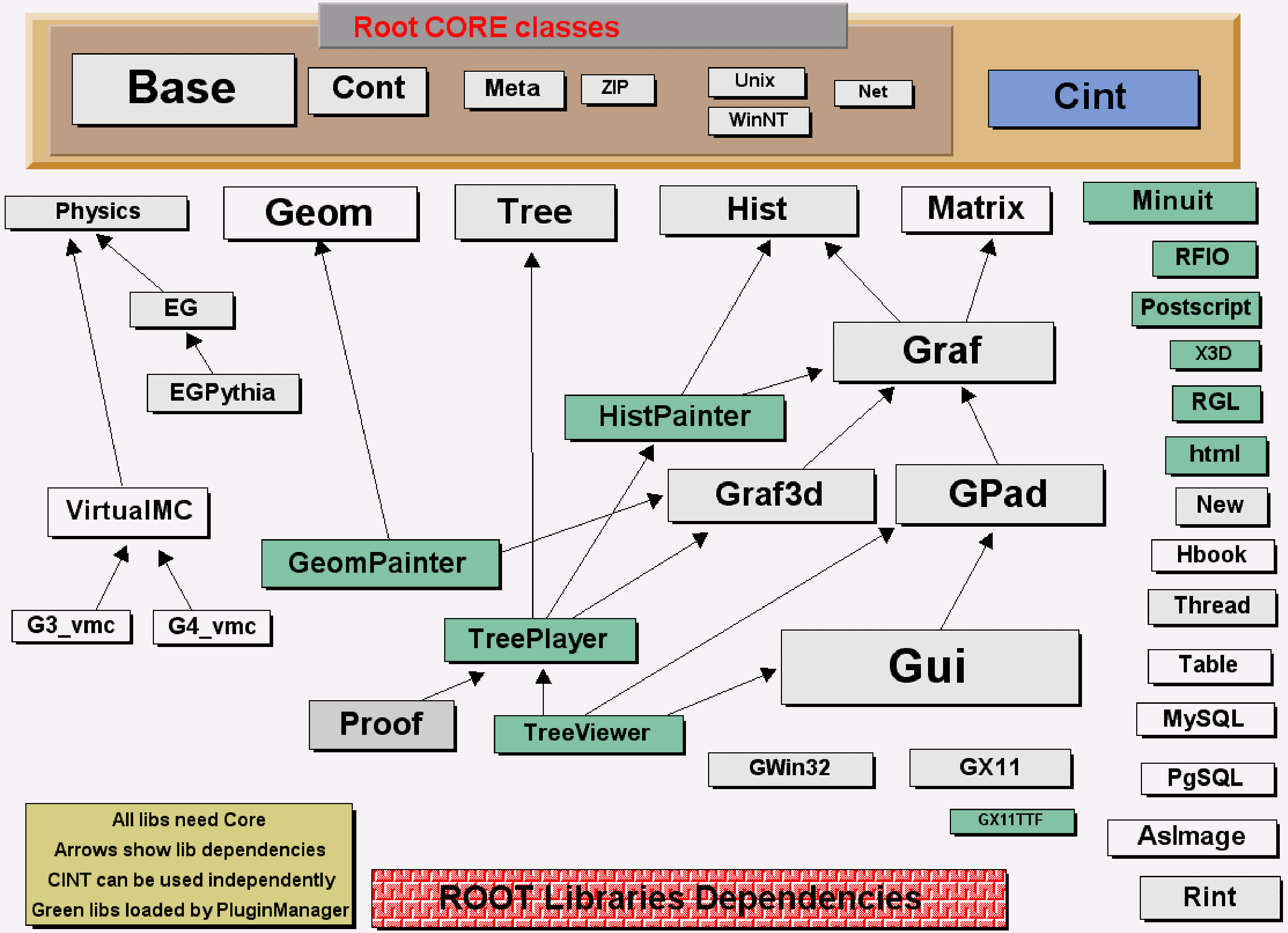
Ntuple analysis

Detector Geometry

Math Libs Statistics

GUI Toolkits

Interpreters



Root CORE classes

Base

Cont

Meta

ZIP

Unix

WinNT

Net

Cint

Physics

Geom

Tree

Hist

Matrix

Minuit

RFIO

EG

Postscript

Graf

X3D

EGPythia

HistPainter

RGL

html

VirtualMC

Graf3d

GPad

New

GeomPainter

Hbook

Thread

G3_vmc

G4_vmc

TreePlayer

Gui

Table

Proof

TreeViewer

MySQL

GWin32

GX11

PgSQL

GX11TTF

AsImage

Rint

All libs need Core
Arrows show lib dependencies
CINT can be used independently
Green libs loaded by PluginManager

ROOT Libraries Dependencies

ROOT libs

ROOT

`ls -l $ROOTSYS/lib`

```
 203856  libASImage.so
1273308  libCint.so
5658143  libCore.so
 419481  libEG.so
 152912  libEGPythia.so
 160874  libEGPythia6.so
 162181  libEGVenus.so
 326000  libGX11.so
 183065  libGX11TTF.so
2306421  libGeom.so
 158895  libGeomPainter.so
1019977  libGpad.so
1602106  libGraf.so
1028762  libGraf3d.so
3669409  libGui.so
1605344  libHbook.so
1940222  libHist.so
 332268  libHistPainter.so
 114970  libHtml.so
```

```
 167670  libMC.so
 580851  libMatrix.so
 319945  libMinuit.so
 268321  libMySQL.so
  21981  libNew.so
  88438  libPgSQL.so
 336736  libPhysics.so
 196318  libPostscript.so
 576691  libProof.so
 681086  libRFIO.so
2017467  libRGL.so
 177657  libRint.so
  35410  libSRPAuth.so
1120731  libTable.so
 312785  libThread.so
1067715  libTree.so
 356186  libTreePlayer.so
 409350  libTreeViewer.so
 155664  libX3d.so
```

TOTAL = 30.7 MBytes

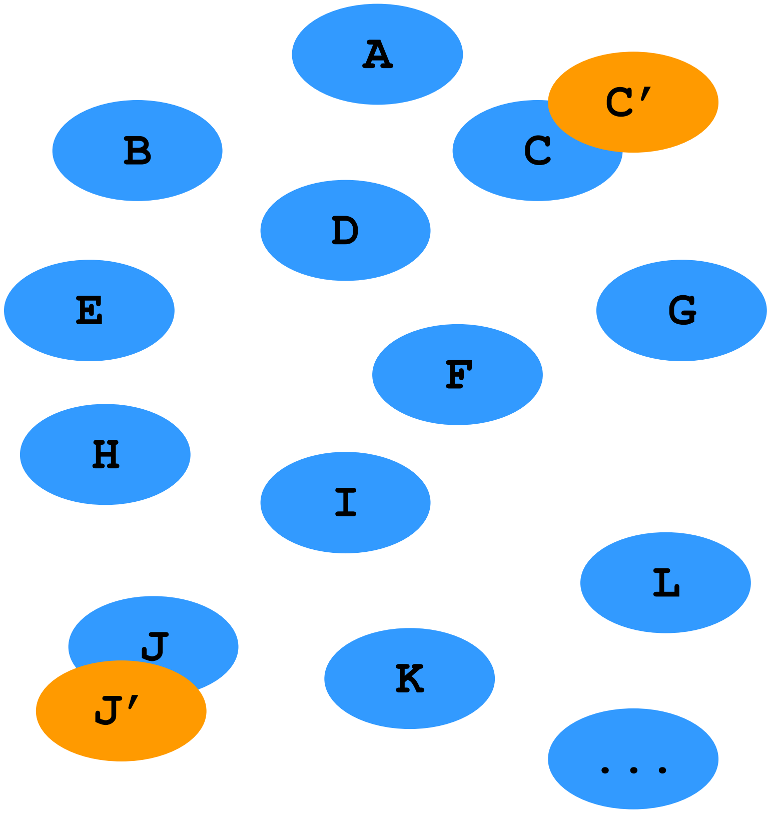
CERNLIB

`ls -l /cern/pro/lib`

```
 1434404  libgrafX11.a
 1046944  libgraflib.a
 4981896  libkernlib.a
 2002460  libmathlib.a
11849762  libpacklib.a
 4350440  libpawlib.a
```

TOTAL = 25.2 MBytes

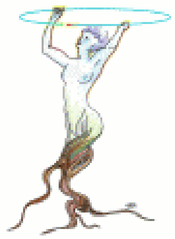
The naive component model

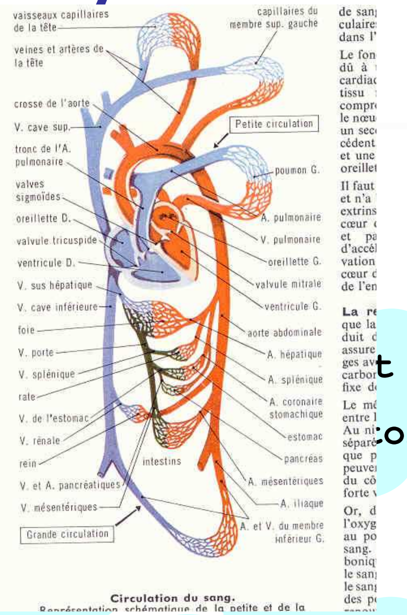


PROs: In principle easy to add or replace a component because of weak coupling

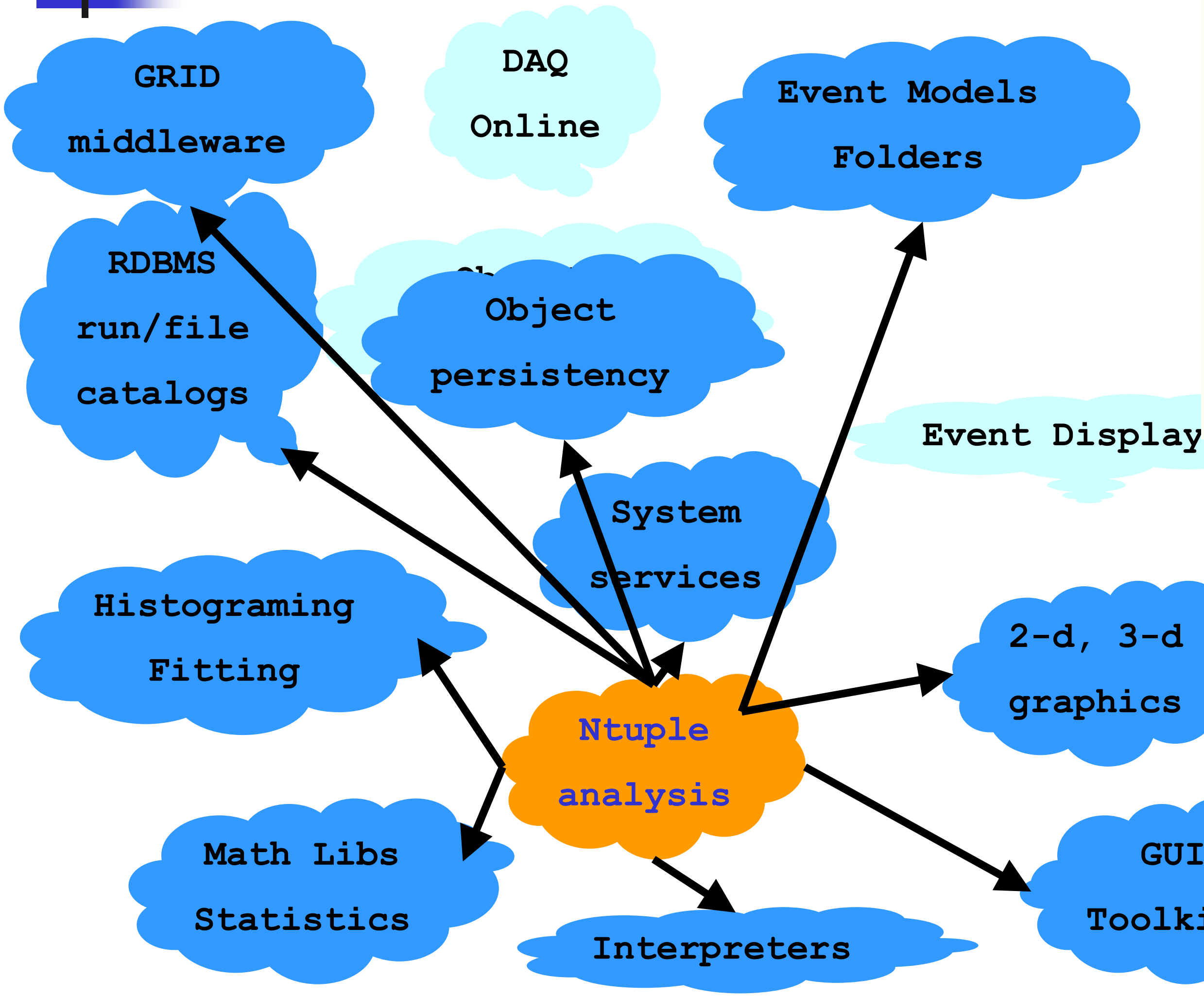
In reality, you simply postpone the integration problem if the number of components N is big eg N> 50

Any box connected to many boxes





Circulation du sang.



GRID middleware

DAQ Online

Event Models Folders

RDBMS run/file catalogs

Object persistency

Event Display

System services

Histograming Fitting

2-d, 3-d graphics

Ntuple analysis

Math Libs Statistics

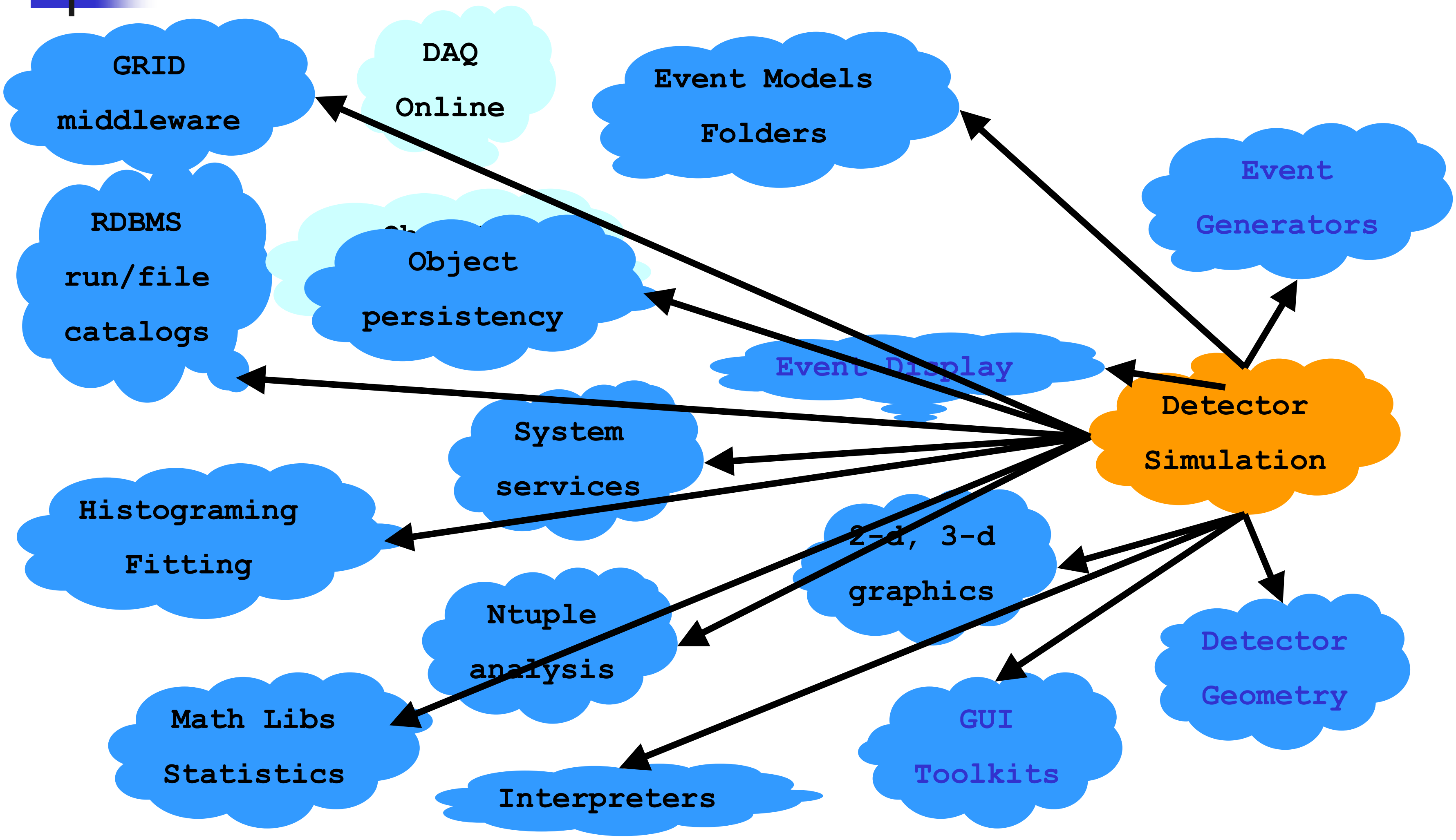
Interpreters

GUI Toolkits

Dectector Simulation

Detector Geometry

Any box connected to many boxes



GRID middleware

DAQ Online

Event Models Folders

Event Generators

RDBMS run/file catalogs

Object persistency

Event Display

Detector Simulation

System services

Histograming Fitting

2-d, 3-d graphics

Ntuple analysis

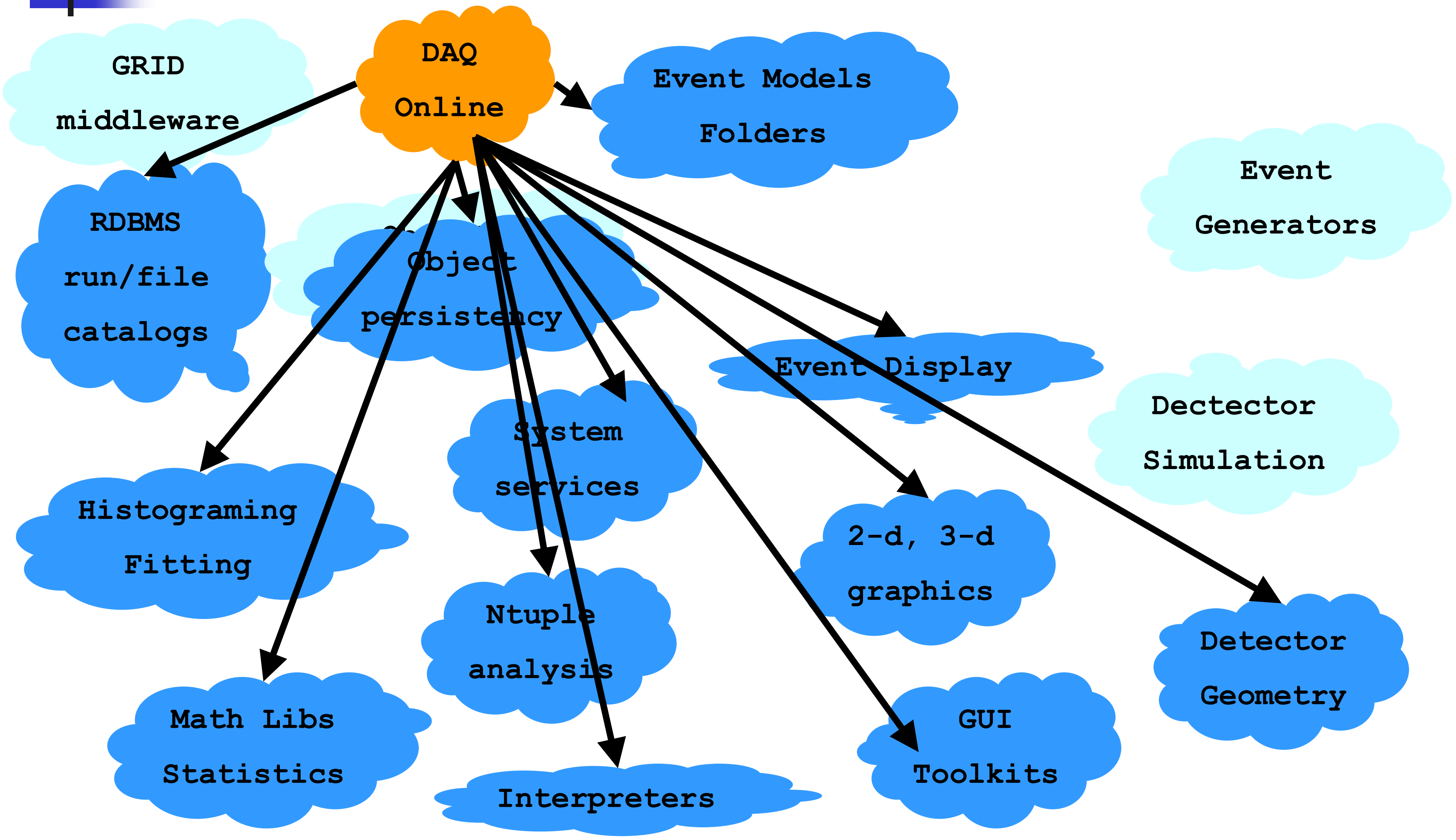
Detector Geometry

Math Libs Statistics

GUI Toolkits

Interpreters

Any box connected to many boxes



GRID middleware

DAQ Online

Event Models Folders

Event Generators

RDBMS run/file catalogs

Object persistency

Event Display

Dectector Simulation

System services

Histograming Fitting

2-d, 3-d graphics

Ntuple analysis

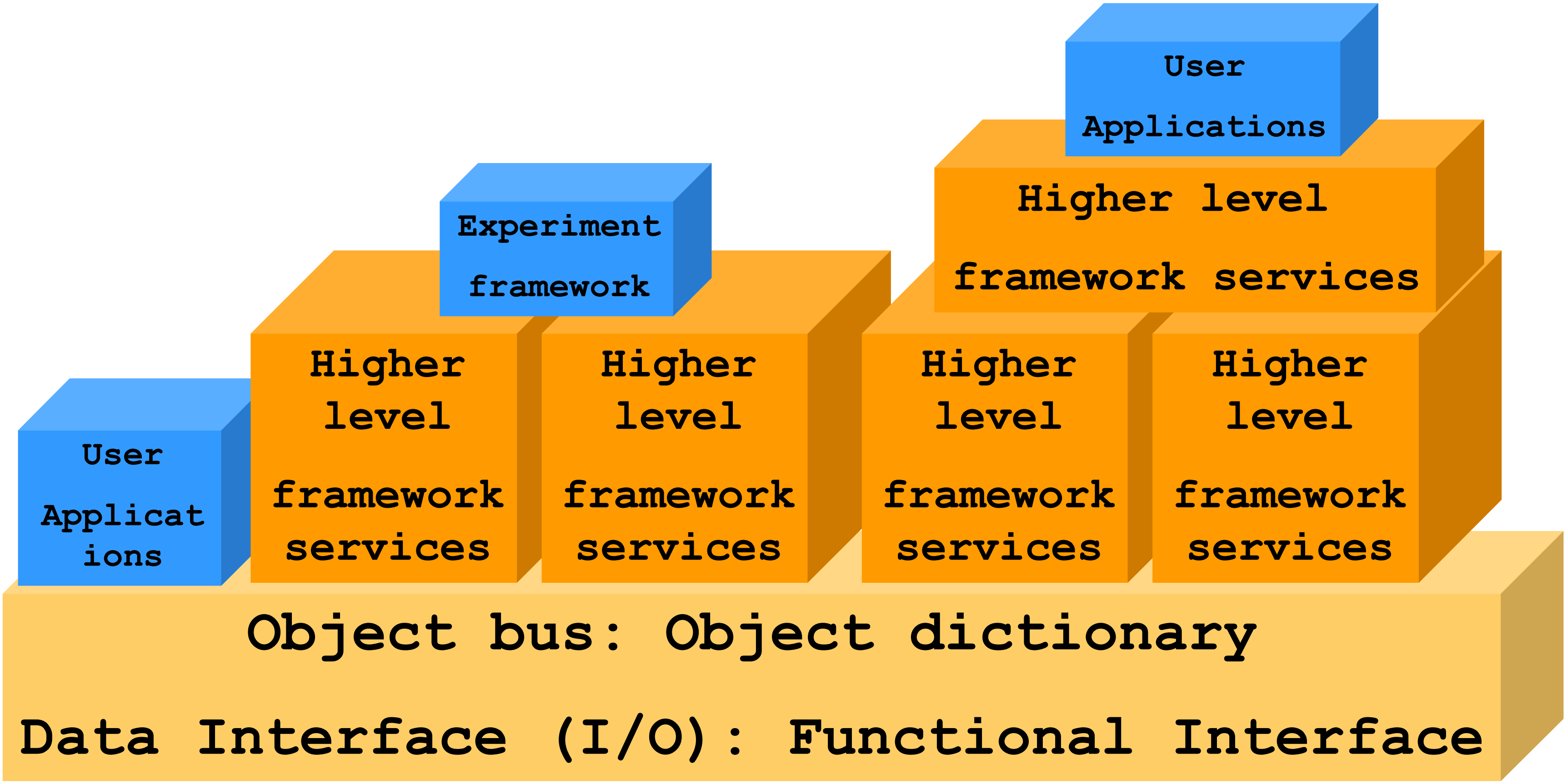
Detector Geometry

Math Libs Statistics

GUI Toolkits

Interpreters

Framework with Object bus



User Applications

Higher level framework services

Experiment framework

User Applications

Higher level framework services

Higher level framework services

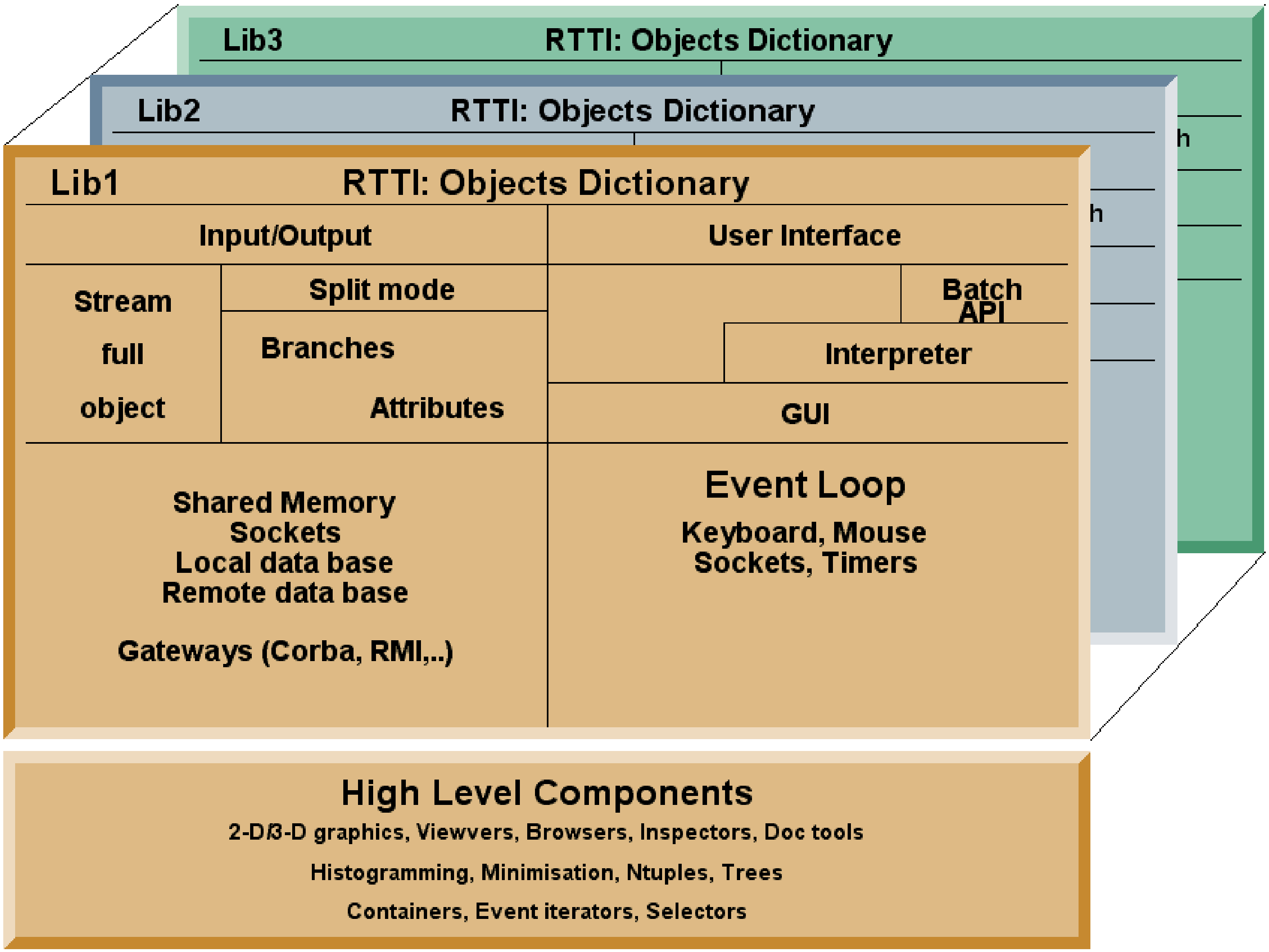
Higher level framework services

Higher level framework services

Object bus: Object dictionary

Data Interface (I/O): Functional Interface

Framework: Basic components



Lib3 RTTI: Objects Dictionary

Lib2 RTTI: Objects Dictionary

Lib1 RTTI: Objects Dictionary

Input/Output

Stream full object

Split mode

Branches Attributes

Shared Memory
Sockets
Local data base
Remote data base

Gateways (Corba, RMI,..)

User Interface

Batch API

Interpreter

GUI

Event Loop

Keyboard, Mouse
Sockets, Timers

High Level Components

2-D/3-D graphics, Viewvers, Browsers, Inspectors, Doc tools

Histogramming, Minimisation, Ntuples, Trees

Containers, Event iterators, Selectors

/user/brun/root/slides/barcelona/framework.C

Object Persistency (in a nutshell)

- Two I/O modes supported (Key and Trees).
- Key access: simple object streaming mode.
  - A ROOT file is like a Unix directory tree
  - Very convenient for objects like histograms, geometries, mag.field, calibrations
- Trees
  - A generalization of ntuples to objects
  - Designed for storing events
  - split and no split modes
  - query processor
- Chains: Collections of files containing Trees
- ROOT files are self-describing
- Interfaces with RDBMS also available
- Access to remote files (RFIO, DCACHE, GRID)

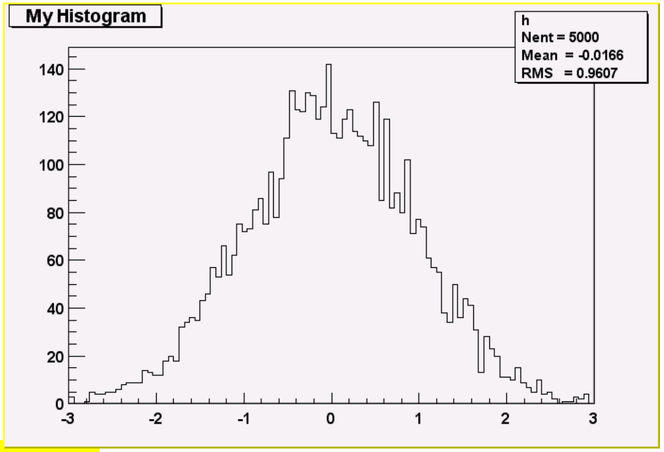
ROOT I/O : An Example

Program Writing

```
TFile f(“example.root”,”new”);
TH1F h(“h”,”My histogram”,100,-3,3);
h.FillRandom(“gaus”,5000);
h.Write();
```

Program Reading

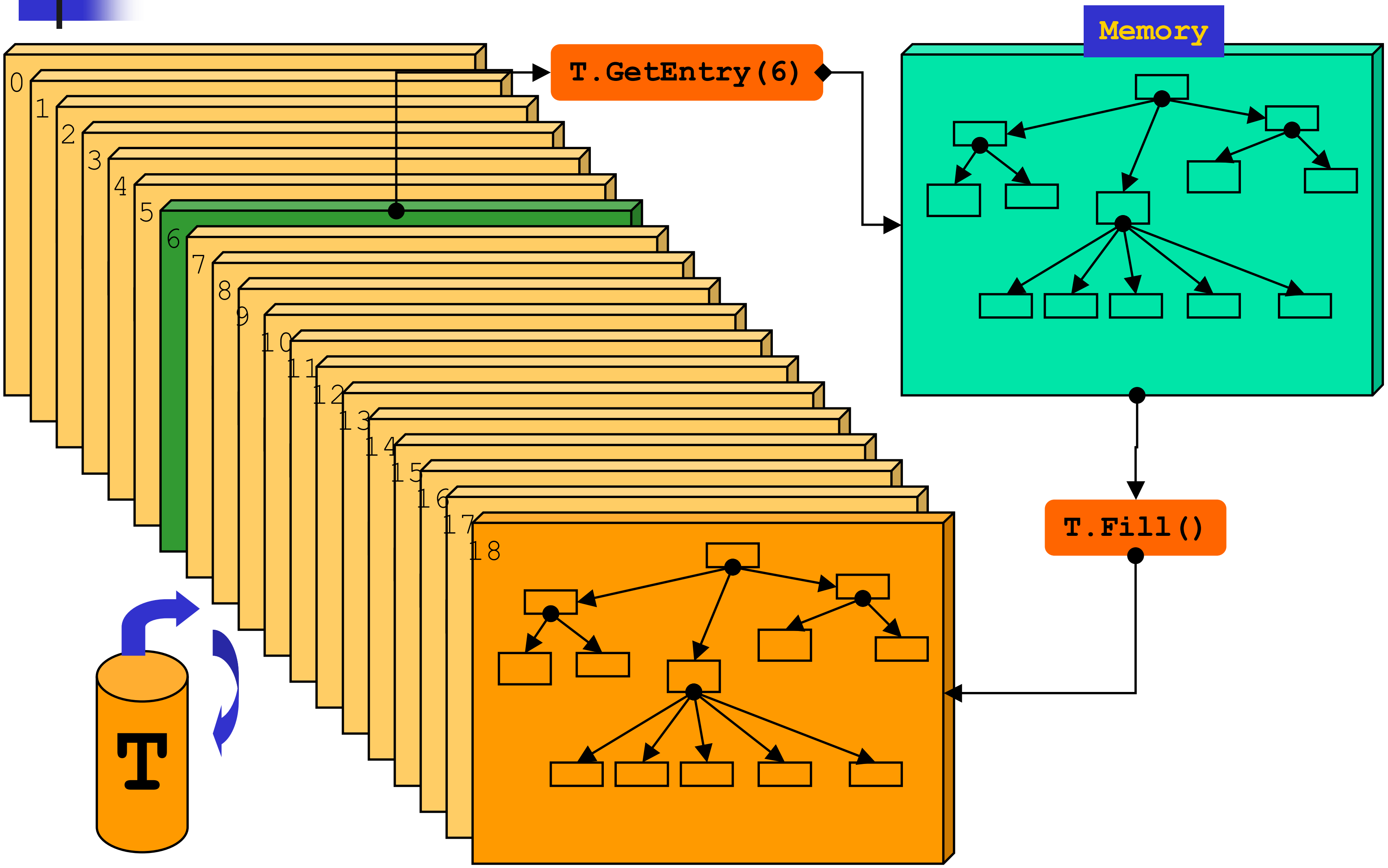
```
TFile f(“example.root”);
TH1F *h = (TH1F*)f.Get(“h”):
h->Draw();
f.Map();
```



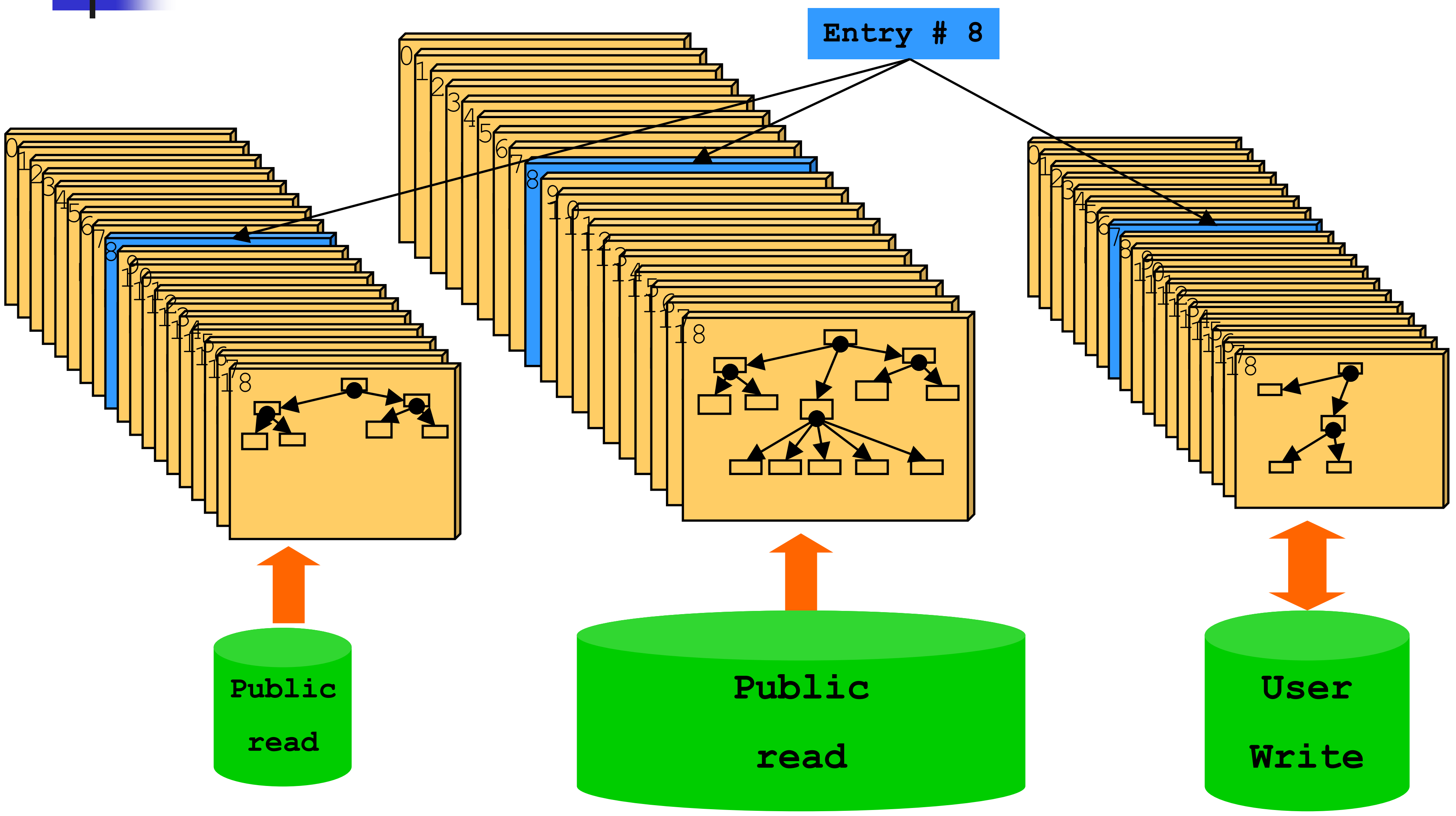
```
20010831/171903  At:64       N=90       TFile
20010831/171941  At:154      N=453      TH1F          CX =  2.09
20010831/171946  At:607      N=2364     StreamerInfo  CX =  3.25
20010831/171946  At:2971     N=96       KeysList
20010831/171946  At:3067     N=56       FreeSegments
20010831/171946  At:3123     N=1        END
```

Memory <--> Tree

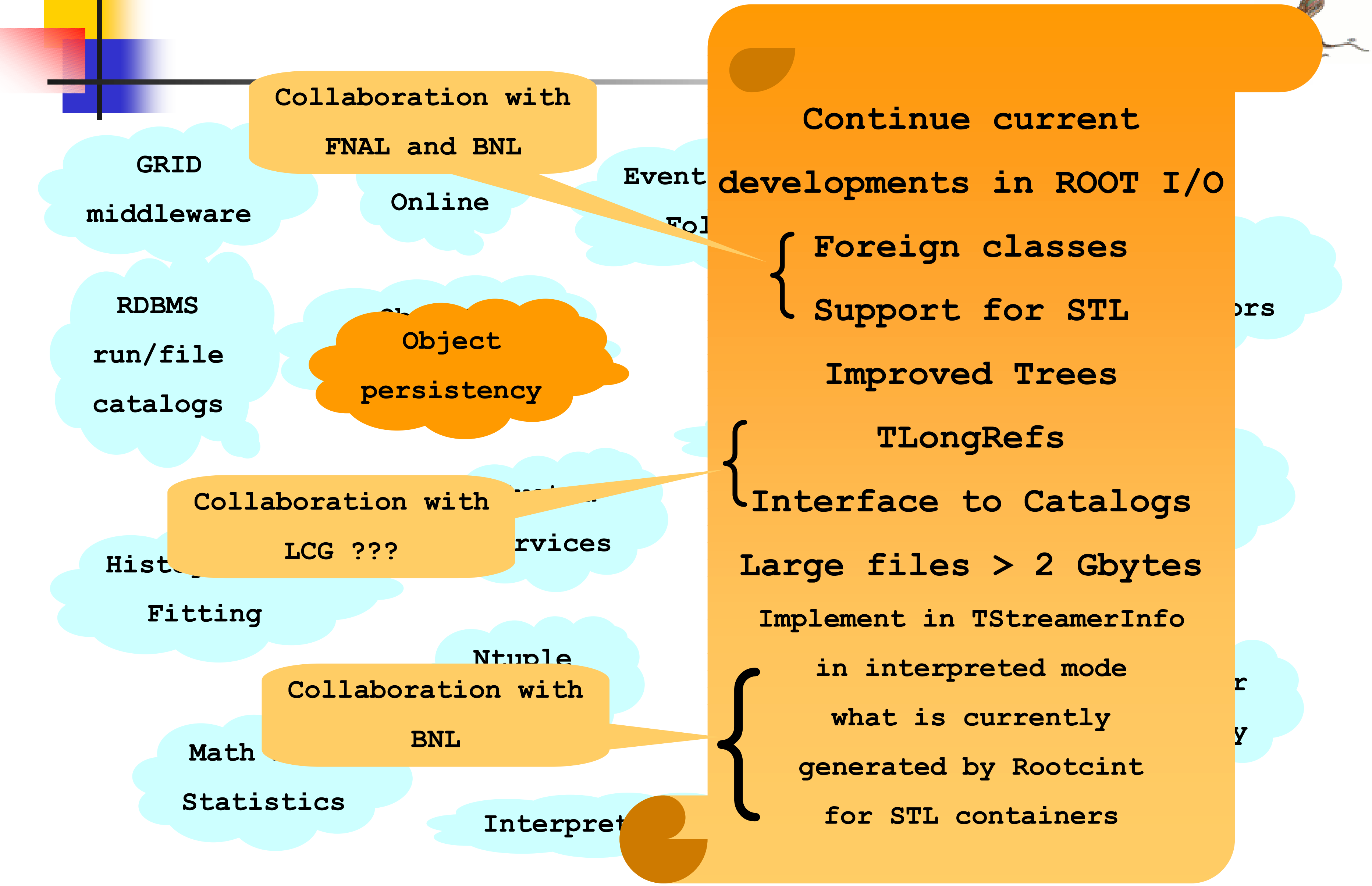
Each Node is a branch in the Tree



Tree Friends



Object Persistency



Collaboration with FNAL and BNL

Collaboration with LCG ???

Collaboration with BNL

Continue current developments in ROOT I/O

- Foreign classes
- Support for STL

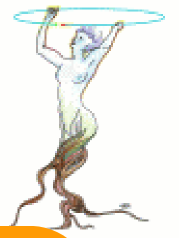
Improved Trees

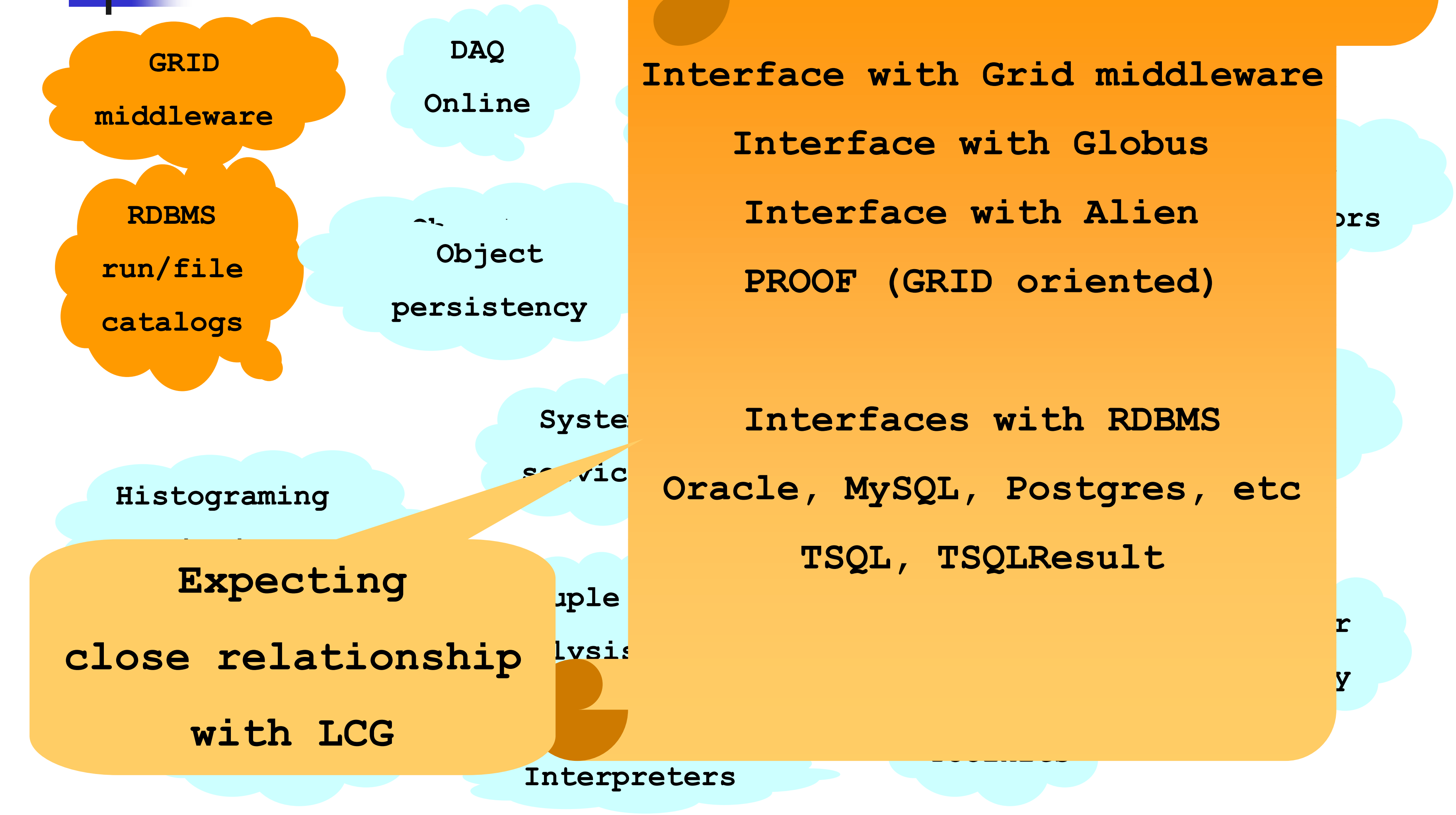
- TLongRefs
- Interface to Catalogs

Large files > 2 Gbytes

Implement in TStreamerInfo in interpreted mode what is currently generated by Rootcint for STL containers

GRIDS Run/file catalogs





GRID middleware

DAQ Online

RDBMS run/file catalogs

Object persistency

Syste

se vic

Histograming

uple

lysis

ors

r

y

Interpreters

Interface with Grid middleware

Interface with Globus

Interface with Alien

PROOF (GRID oriented)

Interfaces with RDBMS

Oracle, MySQL, Postgres, etc

TSQL, TSQLResult

Expecting close relationship with LCG

DAQ Online

GRID middleware

DAQ Online

Event Fo

RDBMS

ors

Electronic logbook at FNAL
well integrated with ROOT

System

JavaRoot interface
possibly interesting

analysis

Math Libs
Statistics

Interpret

Multi-Threading support

Shared memory

Sockets/Monitors

Client/Server

Network classes

System interface/Signals

Interpreter

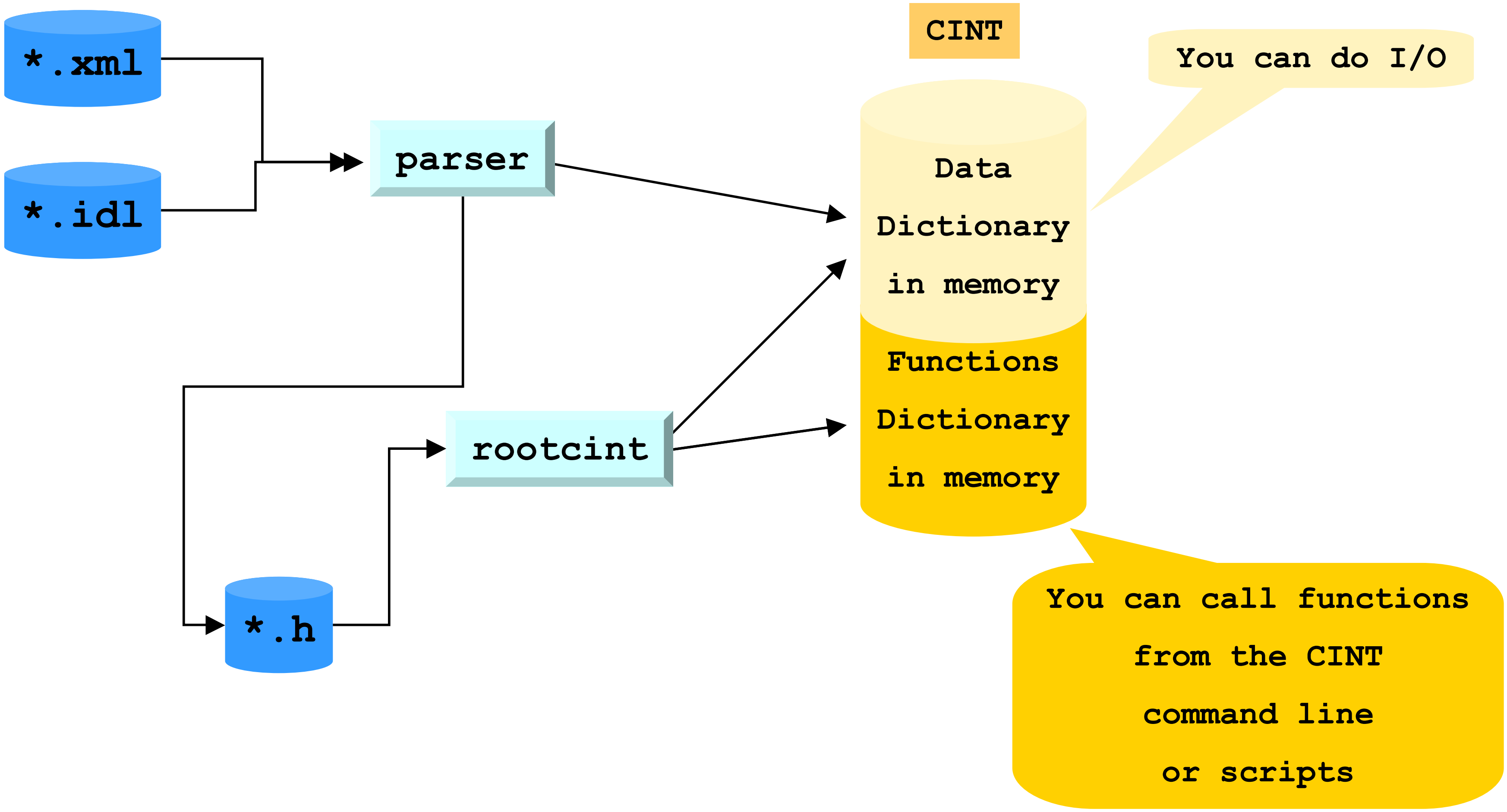
Histograming

Event Display

Browsers/Inspectors

Persistency

Object Dictionary(ies)



*.xml

*.idl

parser

*.h

rootcint

CINT

Data Dictionary in memory

Functions Dictionary in memory

You can do I/O

You can call functions from the CINT command line or scripts

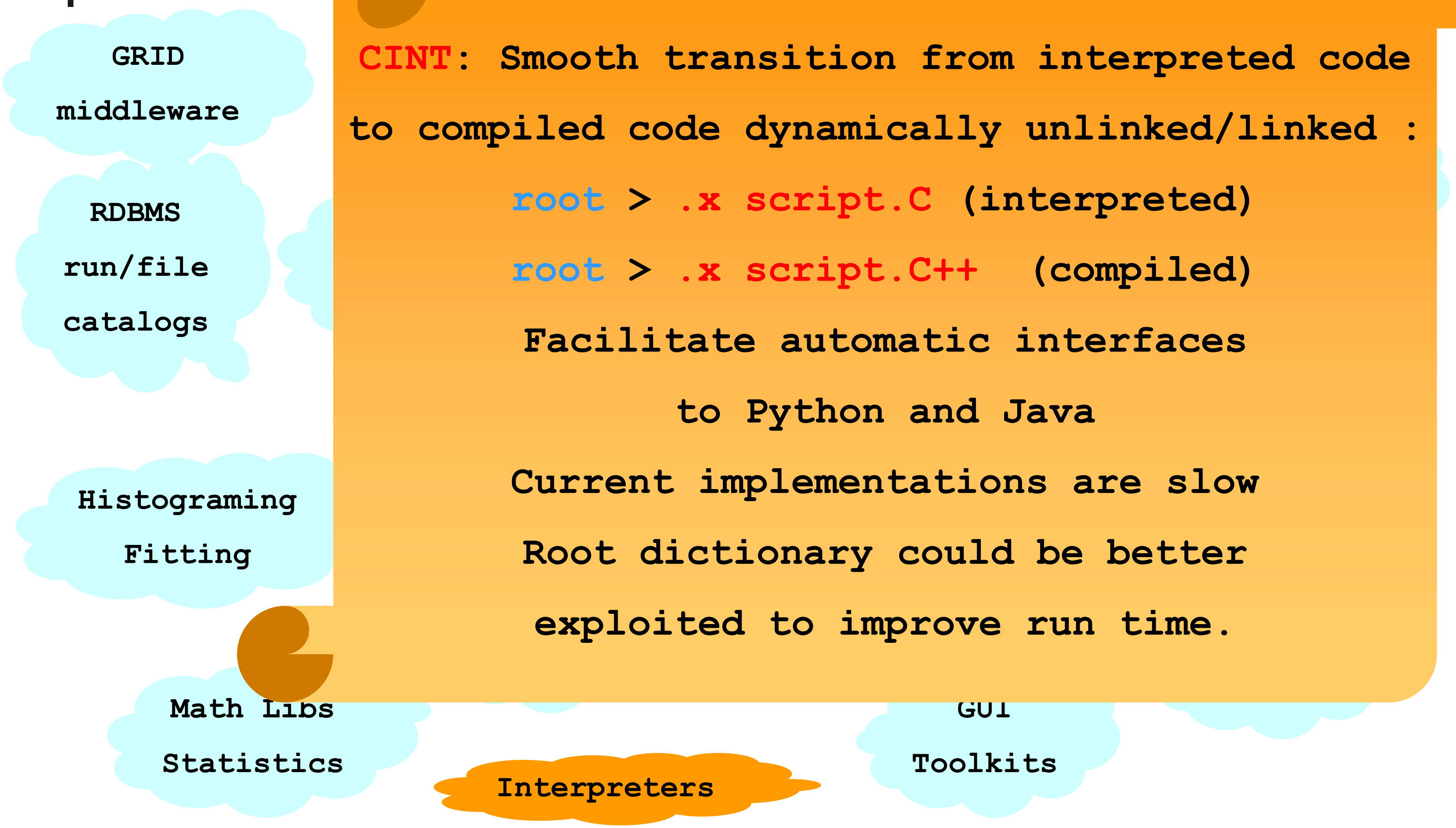
Object Dictionary

- Follow the new C++ standard XTI proposal in the area of introspection (eXtended Type Information)
- Too much emphasis so far on the transient class dictionary. (may be we can have one in common!!)
- The real difficulty is the support for automatic schema evolution (persistent views and relationship with current transient classes).
- We did not discuss enough this essential area.
- Remove as much as possible automatic generated code in favor of dynamic interpretation. Reading files without the original classes.

Interpreters

- It was surprising to see that 3 of the 4 architects did not express any interest in the CINT command line. I may have a distorted perception of the user requirements. Please speak now on this important subject.
- Python (like Perl) is a nice scripting language, but is inferior to CINT for what we want to do! A scripting language that can be compiled by a native compiler is essential.
  - In PAW it was not possible to compile a 5000 lines kumac.
- It will happen to Python what we have seen with Iris Explorer a few years ago. It can eat a lot of manpower and users silently ignoring it.
- RootPython is an interesting tool to extend ROOT to services already interfaced with Python.
- JavaRoot could be very interesting for Java-based projects that need an interface to ROOT services.

CINT, Python, Java, C#



CINT: Smooth transition from interpreted code to compiled code dynamically unlinked/linked :

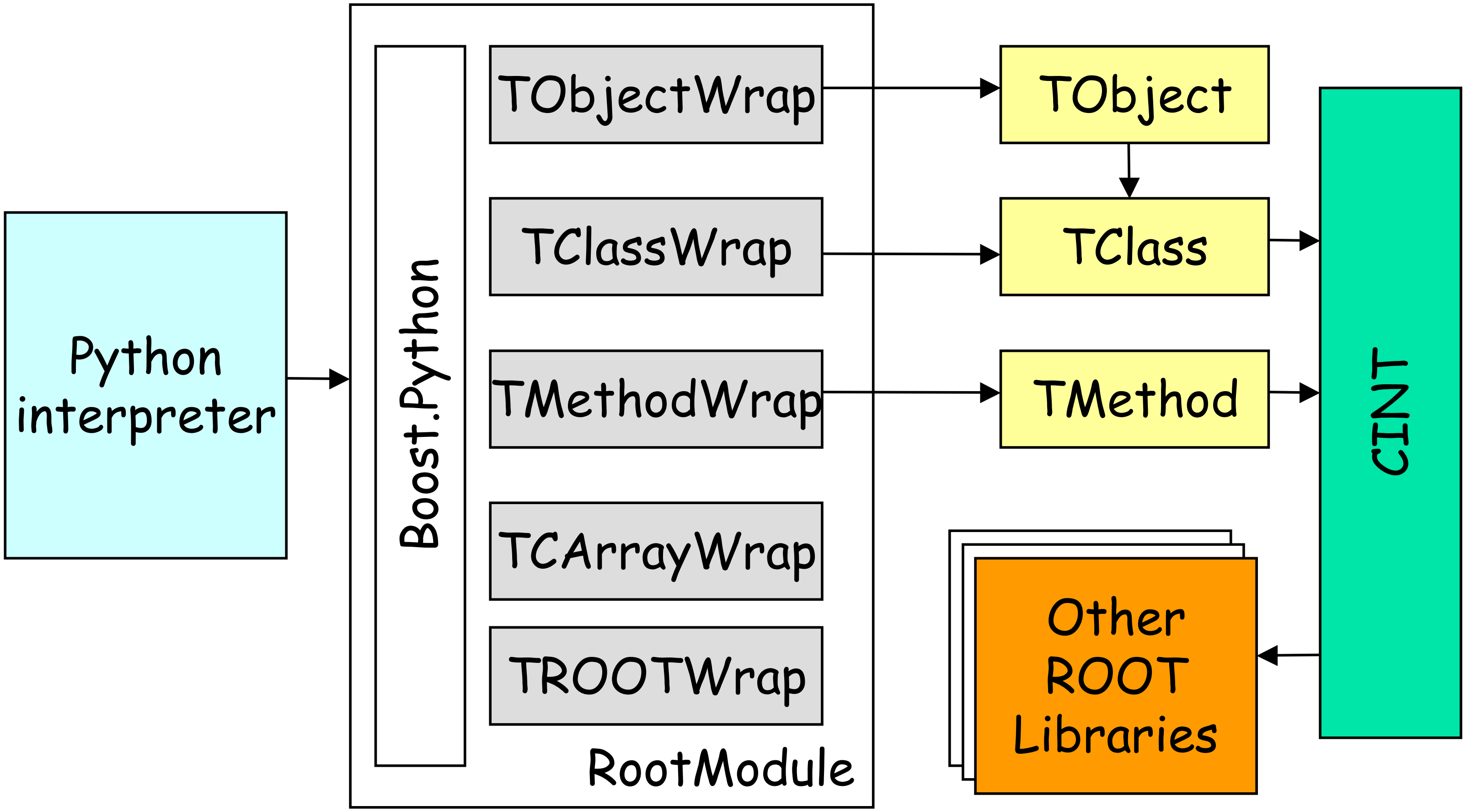
```
root > .x script.C (interpreted)
root > .x script.C++  (compiled)
```

Facilitate automatic interfaces to Python and Java

Current implementations are slow

Root dictionary could be better exploited to improve run time.

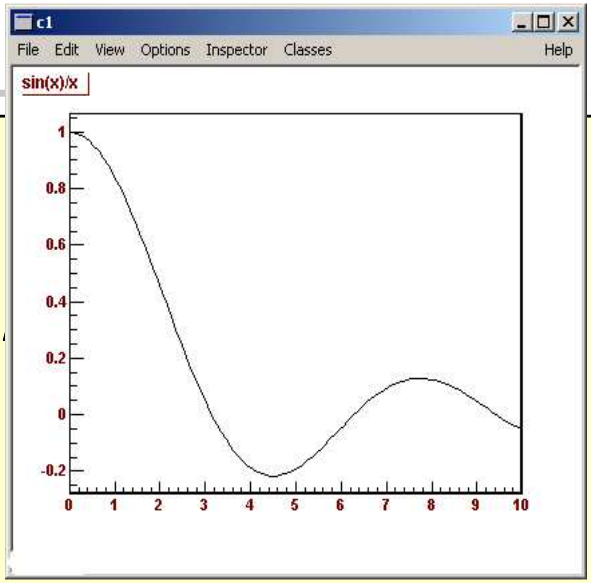
RootPython (Pere Mato)



Example

Python

```
C:\> python
...
>>> from rootmodule import *
>>> f1 = TF1('func1','sin(x)/x',0,
>>> f1.Eval(3)
0.047040002686622402
>>> f1.Derivative(3)
-0.34567505667199266
>>> f1.Integral(0,3)
1.8486525279994681
>>> f1.Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
```



- No much difference between CINT and Python !

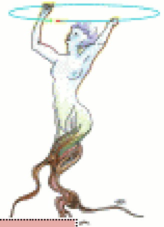
P.Mato

CINT

```
TF1 f1("func1","sin(x)/x",0,10)
f1.Eval(3)
f1.Derivative(3)
f1.Integral(0,3)
f1.Draw()
```

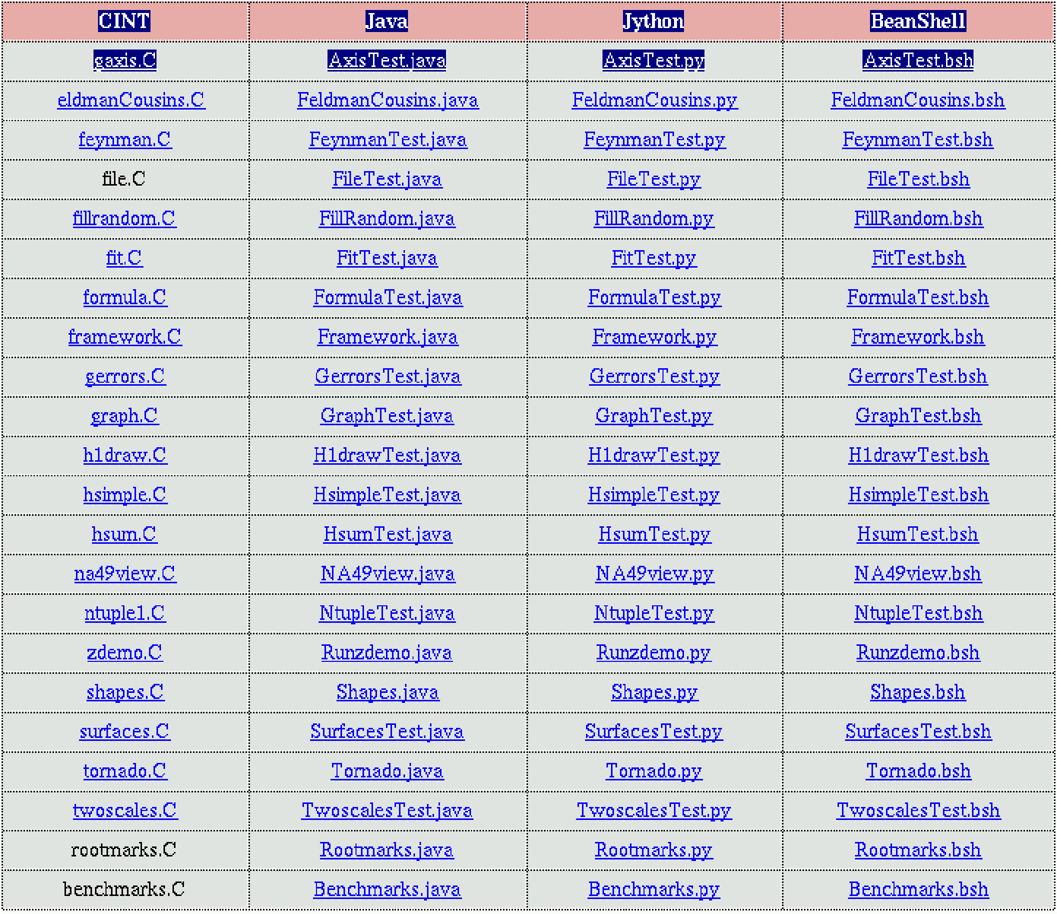
JavaRoot

http://sarkar.home.cern.ch/sarkar/jroot/main.html

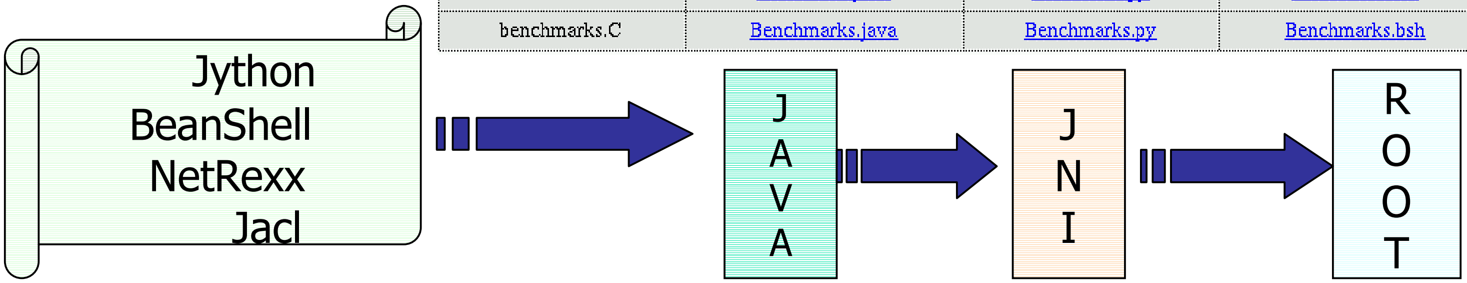


Extend Java with Root libraries

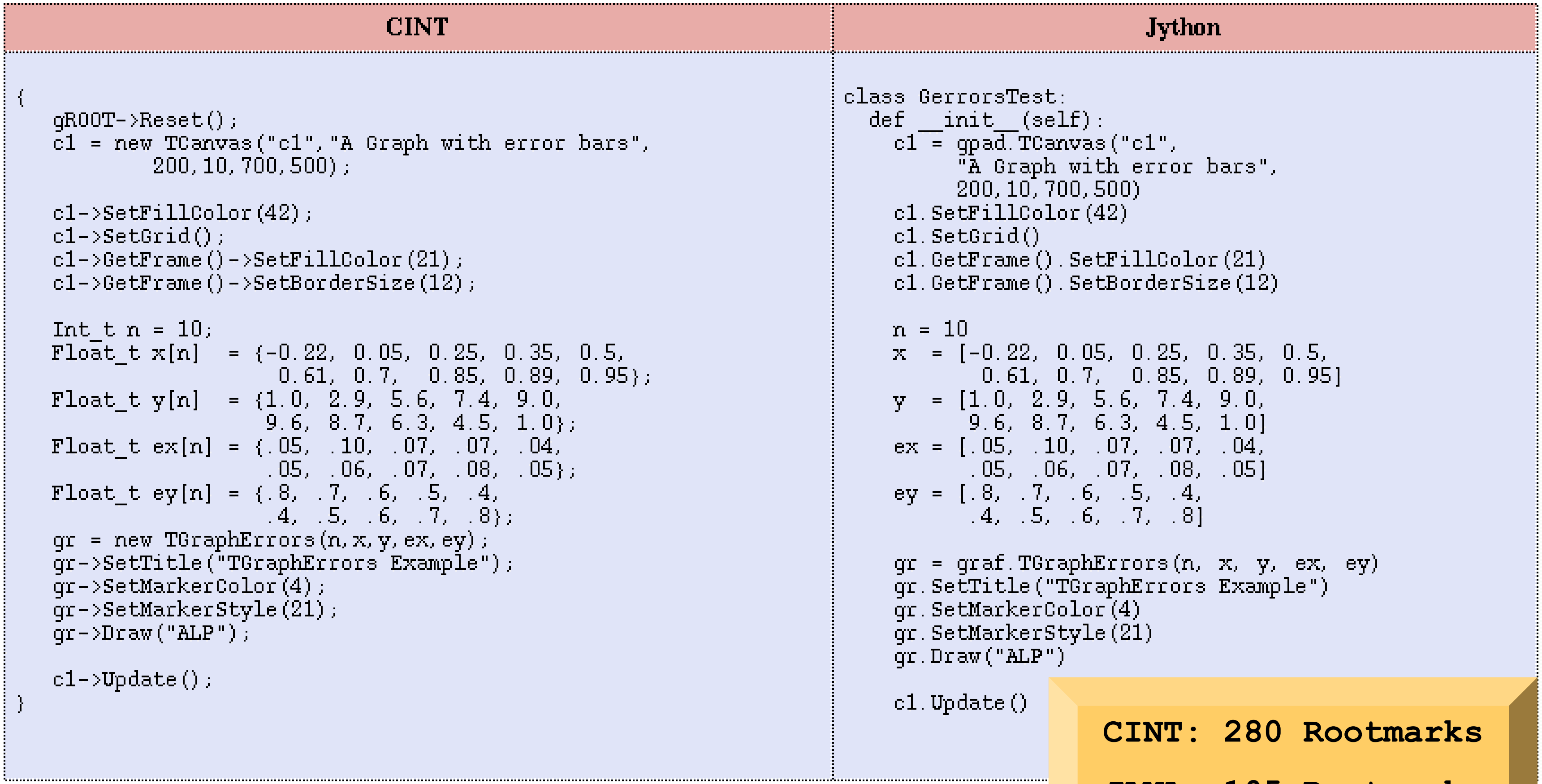
- Java gets a matured Histograming API, Fitting and Physics analysis classes, a HEP specific Socket programming API
- Root reaches an even wider audience, finds a number of interpreter and scripting environments that are (re)implemented in Java (Jython, BeanShell etc.)



CINT	Java	Jython	BeanShell
gaxis.C	AxisTest.java	AxisTest.py	AxisTest.bsh
eldmanCousins.C	FeldmanCousins.java	FeldmanCousins.py	FeldmanCousins.bsh
feynman.C	FeynmanTest.java	FeynmanTest.py	FeynmanTest.bsh
file.C	FileTest.java	FileTest.py	FileTest.bsh
fillrandom.C	FillRandom.java	FillRandom.py	FillRandom.bsh
fit.C	FitTest.java	FitTest.py	FitTest.bsh
formula.C	FormulaTest.java	FormulaTest.py	FormulaTest.bsh
framework.C	Framework.java	Framework.py	Framework.bsh
gerrors.C	GerrorsTest.java	GerrorsTest.py	GerrorsTest.bsh
graph.C	GraphTest.java	GraphTest.py	GraphTest.bsh
h1draw.C	H1drawTest.java	H1drawTest.py	H1drawTest.bsh
hsimple.C	HsimpleTest.java	HsimpleTest.py	HsimpleTest.bsh
hsum.C	HsumTest.java	HsumTest.py	HsumTest.bsh
na49view.C	NA49view.java	NA49view.py	NA49view.bsh
ntuple1.C	NtupleTest.java	NtupleTest.py	NtupleTest.bsh
zdemo.C	Runzdemo.java	Runzdemo.py	Runzdemo.bsh
shapes.C	Shapes.java	Shapes.py	Shapes.bsh
surfaces.C	SurfacesTest.java	SurfacesTest.py	SurfacesTest.bsh
tornado.C	Tornado.java	Tornado.py	Tornado.bsh
twoscales.C	TwoscalesTest.java	TwoscalesTest.py	TwoscalesTest.bsh
rootmarks.C	Rootmarks.java	Rootmarks.py	Rootmarks.bsh
benchmarks.C	Benchmarks.java	Benchmarks.py	Benchmarks.bsh



CINT vs Java/Python/Jython



CINT

```
{
   gROOT->Reset();
   c1 = new TCanvas("c1","A Graph with error bars",
           200,10,700,500);

   c1->SetFillColor(42);
   c1->SetGrid();
   c1->GetFrame()->SetFillColor(21);
   c1->GetFrame()->SetBorderSize(12);

   Int_t n = 10;
   Float_t x[n]  = {-0.22, 0.05, 0.25, 0.35, 0.5,
                     0.61, 0.7,  0.85, 0.89, 0.95};
   Float_t y[n]  = {1.0, 2.9, 5.6, 7.4, 9.0,
                    9.6, 8.7, 6.3, 4.5, 1.0};
   Float_t ex[n] = {.05, .10, .07, .07, .04,
                    .05, .06, .07, .08, .05};
   Float_t ey[n] = {.8, .7, .6, .5, .4,
                    .4, .5, .6, .7, .8};
   gr = new TGraphErrors(n,x,y,ex,ey);
   gr->SetTitle("TGraphErrors Example");
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(21);
   gr->Draw("ALP");

   c1->Update();
}
```

Jython

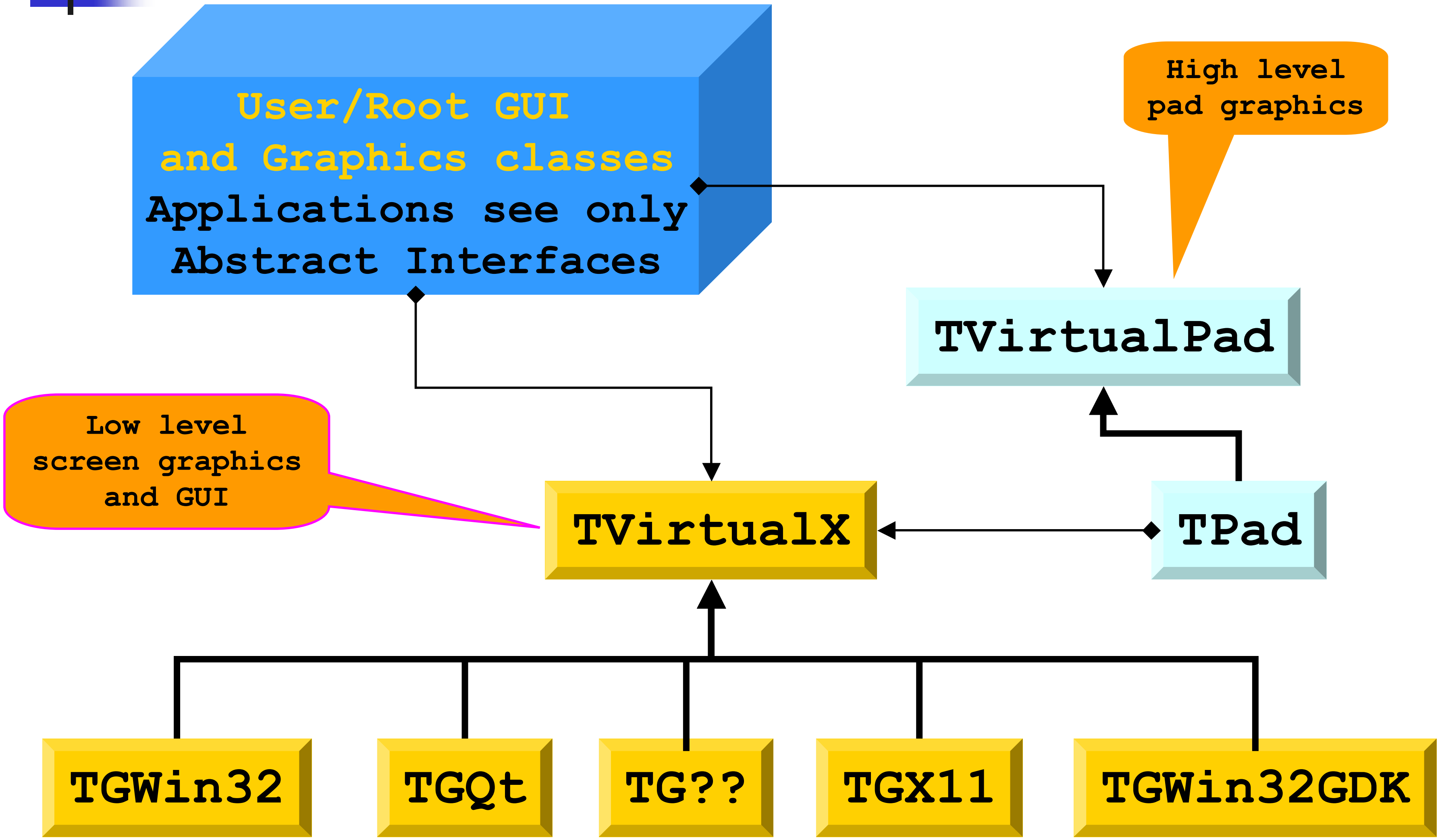
```
class GerrorsTest:
  def __init__(self):
    c1 = gpad.TCanvas("c1",
         "A Graph with error bars",
         200,10,700,500)
    c1.SetFillColor(42)
    c1.SetGrid()
    c1.GetFrame().SetFillColor(21)
    c1.GetFrame().SetBorderSize(12)

    n = 10
    x  = [-0.22, 0.05, 0.25, 0.35, 0.5,
           0.61, 0.7,  0.85, 0.89, 0.95]
    y  = [1.0, 2.9, 5.6, 7.4, 9.0,
          9.6, 8.7, 6.3, 4.5, 1.0]
    ex = [.05, .10, .07, .07, .04,
          .05, .06, .07, .08, .05]
    ey = [.8, .7, .6, .5, .4,
          .4, .5, .6, .7, .8]

    gr = graf.TGraphErrors(n, x, y, ex, ey)
    gr.SetTitle("TGraphErrors Example")
    gr.SetMarkerColor(4)
    gr.SetMarkerStyle(21)
    gr.Draw("ALP")

    c1.Update()
```

CINT: 280 Rootmarks

JAVA: 105 Rootmarks

JYTHON: 52 Rootmarks

Gui/Graphics strategy



User/Root GUI and Graphics classes

Applications see only Abstract Interfaces

High level pad graphics

Low level screen graphics and GUI

GUI Toolkits

Consolidate the TVirtualX interface

Complete TVirtualX/Qt implementation

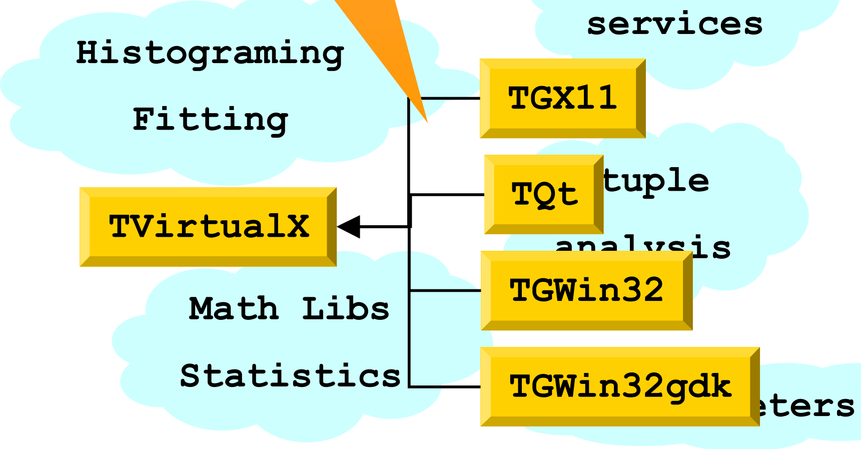
TVirtualX/Win32-GDK (free on Windows)

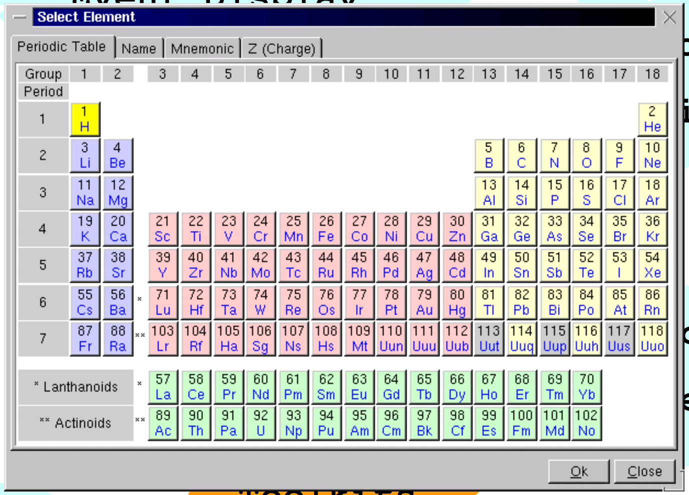
Export script from a running GUI

Build GUI from a given script

GUI editor/builder

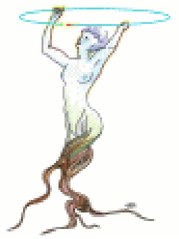
The ROOT event loop has proven to work with all known graphics systems: X11, Xt, Motif, Qt, Open Inventor, etc





Group	1	2		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Period																			
1	1 H																		2 He
2	3 Li	4 Be												5 B	6 C	7 N	8 O	9 F	10 Ne
3	11 Na	12 Mg												13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
4	19 K	20 Ca		21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
5	37 Rb	38 Sr		39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
6	55 Cs	56 Ba	*	71 Lu	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	**	103 Lr	104 Rf	105 Ha	106 Sg	107 Ns	108 Hs	109 Mt	110 Uun	111 Uuu	112 Uub	113 Uut	114 Uuq	115 Uup	116 Uuh	117 Uus	118 Uuo
* Lanthanoids			*	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb		
** Actinoids			**	89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No		

Math Libs & Statistics



In ROOT today

- TVector2,3
- TLorentzRotation
- TLorentzVector
- TRandom,2,3
- TMatrix
- TMath
- TFeldmanCousins
- TPrincipal
- TMultidimFit
- TConfidenceLevel
- TFractionFitter

Can generate random numbers
from basic distributions; gaus, poisson, etc
from parametric analytic functions 1,2,3-d
from histograms, 1,2,3-d

Matrix package maintained by E. Offermann (Rentec)

A collection of many algorithms
CERNLIB, Numerical Recipes in C/C++

Many algorithms classes
developed by a huge user community
See recent FNAL meeting
and effort organized within ACAT

Would like to see an interface
to GSL
to Numerical Recipes in C++
Collaboration with
Fred James, Louis Lyons, Sherry Towers

GRI
middle

RDBM
run/fi
catal

Histo
Fi

Math Libs
Statistics

ne

ct
tency

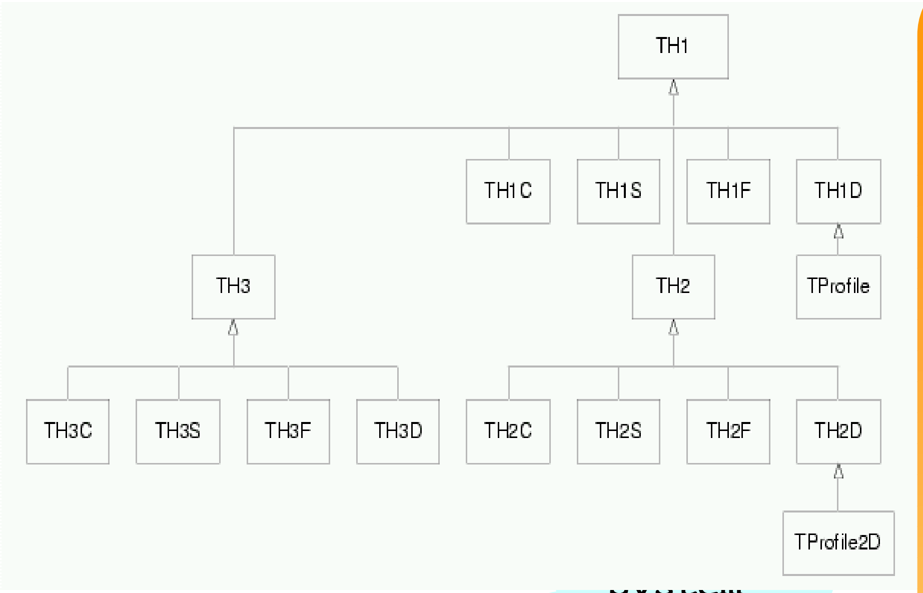
System
services

Ntuple
anal

Event

Event Display

Histograming & Fitting



Much more than HBOOK/PAW

Fix & var bin size for 1-d, 2-d 3-D

Profile 1, 2 & 3D

All kinds of projections, slices

Errors for all dims

Filling with strings (auto sort)

time axis

associated fitting

Random n. generation

+ auto binning

+ auto addition

support for parallelism

system

services

Histograming

Fitting

Fairly complete

no requests

for extensions

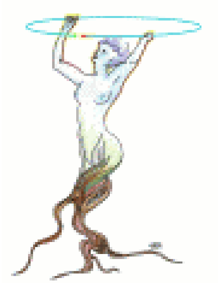
analysis

Fitting

new ideas

collaborators

Interpreters

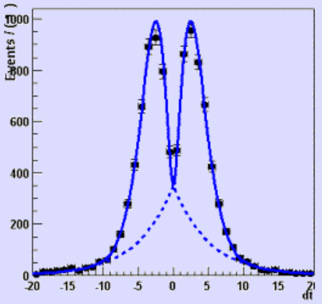


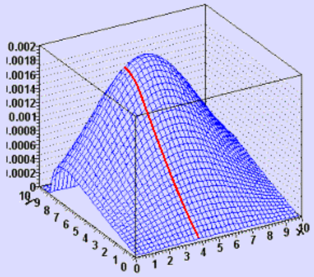
RooFit

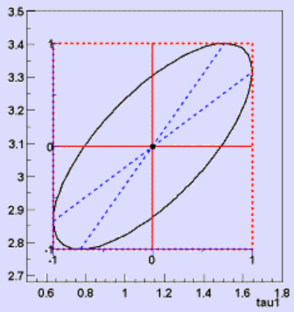
A general purpose tool kit for data modeling

Wouter Verkerke (UC Santa Barbara)

David Kirkby (UC Irvine)





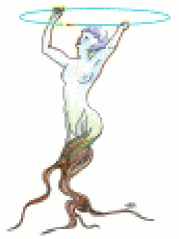


ftp://root.cern.ch/root/R2002/verkerke.ppt

Development and Use of RooFit in BaBar

- Development
  - RooFit started as RooFitTools (presented at ROOT2001) in late 1999
    - Original design was rapidly stretched to its limits
  - Started comprehensive redesign early 2001
    - New design was released to BaBar users in Oct 2001 as RooFit
    - Extensive testing & tuning of user interface in the past year
  - RooFit released on SourceForge in Sep 2002
- Current use
  - Almost all BaBar analysis requiring a non-trivial fit now use RooFit or are in the process of switching to RooFit, e.g.
    - CP violation and mixing in hadronic decays (‘sin2β’)
    - B-Mixing in di-lepton events, $D^{*}\ell\nu$ events
    - Measurement of sin2$\alpha_{(eff)}$ from $B \rightarrow \rho\ \pi$, $B \rightarrow \pi\ \pi$
    - Searches for rare decays ($B \rightarrow \phi\ K_S$, $\eta'\ K_S$, …)
  - **Typical fit complexity**
    - **30 – 70 floating parameters**
    - **4-8 dimensions**
    - **PDF consists of 1000-10000 objects**
    - **Dataset of 500-100000 events**

Ntuples & Trees analysis



PAW-like queries on attributes

`tree.Draw("varx","sqrt(x*y) <z")`

`tree.Draw("event.tracks.GetPt()")`

+

Tree browser and viewer

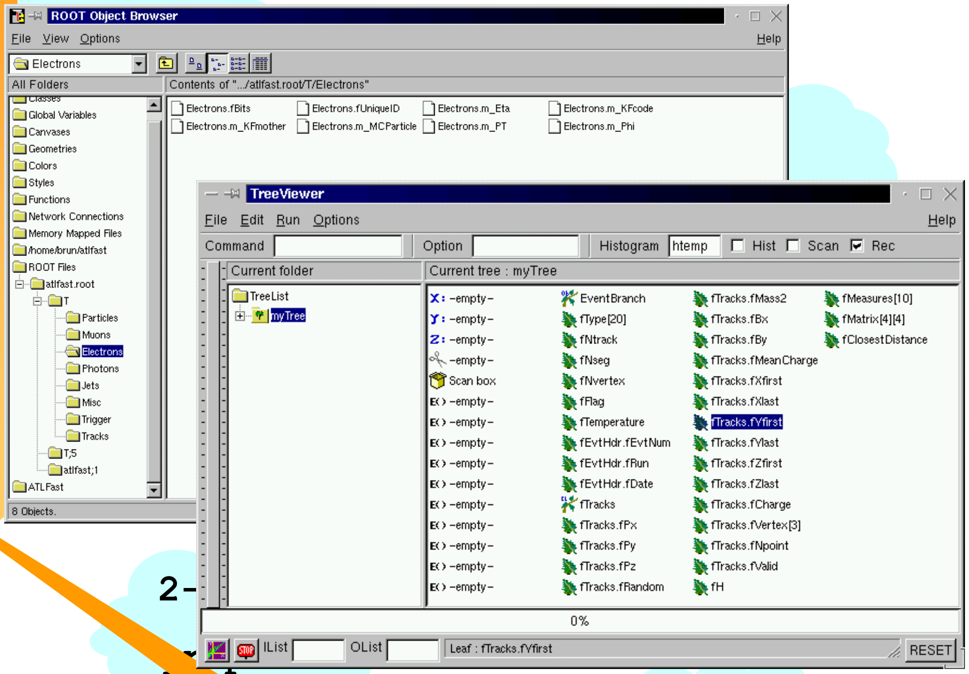
+MakeClass

(generation of skeleton analysis code)

+MakeSelector

same as MakeClass for PROOF

Collaboration with FNAL



Contents of ".../atlfast.root/T/Electrons"

Electrons.fBits
Electrons.fUniqueID
Electrons.m_Eta
Electrons.m_KFcode
Electrons.m_KFmother
Electrons.m_MCParticle
Electrons.m_PT
Electrons.m_Phi

Classes
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
Particles
Muons
Electrons
Photons
Jets
Misc
Trigger
Tracks
T;5
atlfast;1
ATLFast

8 Objects.

Command
Option
Histogram htemp
☐ Hist ☐ Scan ☑ Rec

Current folder
TreeList
myTree

Current tree : myTree

X: -empty-
Y: -empty-
Z: -empty-
-empty-
Scan box
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-

EventBranch
fType[20]
fNtrack
fNseg
fNvertex
fFlag
fTemperature
fEvtHdr.fEvtNum
fEvtHdr.fRun
fEvtHdr.fDate
fTracks
fTracks.fPx
fTracks.fPy
fTracks.fPz
fTracks.fRandom

fTracks.fMass2
fTracks.fBx
fTracks.fBy
fTracks.fMeanCharge
fTracks.fXfirst
fTracks.fXlast
fTracks.fYfirst
fTracks.fYlast
fTracks.fZfirst
fTracks.fZlast
fTracks.fCharge
fTracks.fVertex[3]
fTracks.fNpoint
fTracks.fValid
fH

fMeasures[10]
fMatrix[4][4]
fClosestDistance

0%

IList
OList
Leaf : fTracks.fYfirst

Histograming

Fitting

services

2-

Ntuple

analysis

Detector

Geometry

Math Libs

Statistics

GUI

Toolki

Interpreters

Collaboration with MIT

Parallel ROOT Facility

- The PROOF system allows:
  - parallel analysis of trees in a set of files
  - parallel analysis of objects in a set of files
  - parallel execution of scripts

  on clusters of heterogeneous machines
- Its design goals are:
  - transparency, scalability, adaptability
- Prototype developed in 1997 as proof of concept, full version nearing completion now
- Collaboration between core ROOT group at CERN and MIT Heavy Ion Group

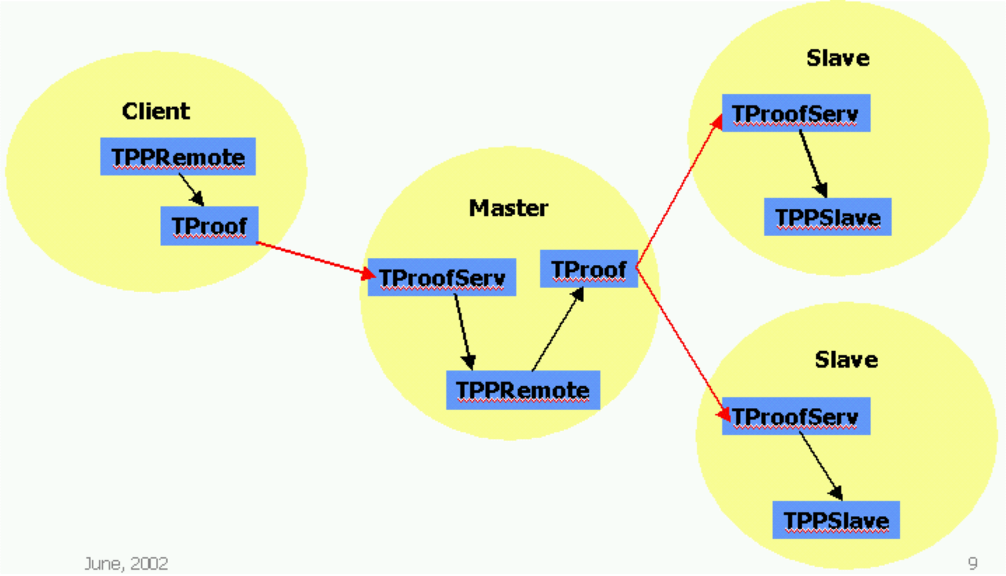
Running a PROOF Job

```
// Analyze TChains in parallel

gROOT->Proof();
TChain *chain = new TChain(“AOD");
chain->Add("lfn://alien.cern.ch/alice/prod2002/P2001*");
. . .
chain->Process(“myselector.C”);
```

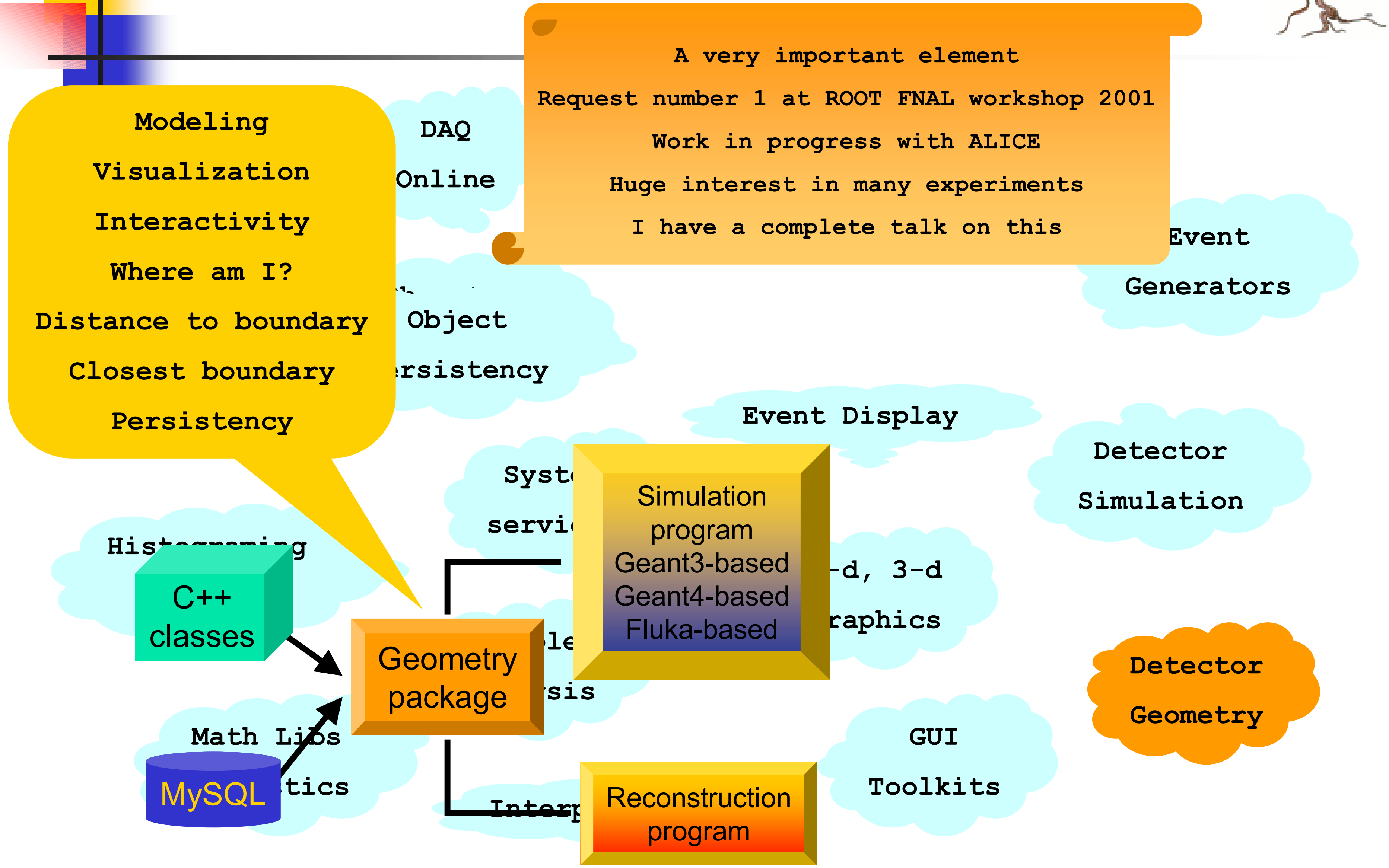
```
// Analyze generic data sets in parallel

gROOT->Proof();
TDSet *objset = new TDSet("MyEvent", "*", "/events");
objset->Add("lfn://alien.cern.ch/alice/prod2002/file1");
. . .
objset->Add(set2003);
objset->Process(“myselector.C++”);
```

PROOF Demo at ROOT workshop

- Client machine
  - PIII @ 1GHz / 512 MB
  - Standard IDE disk
- Cluster with 15 nodes at CERN
  - Dual PIII @ 800 MHz / 384 MB
  - Standard IDE disk
- Cluster with 4 nodes at MIT
  - Dual AthlonMP @ 1.4GHz / 1GB
  - Standard IDE disk



Detector geometry



A very important element

Request number 1 at ROOT FNAL workshop 2001

Work in progress with ALICE

Huge interest in many experiments

I have a complete talk on this

Modeling

Visualization

Interactivity

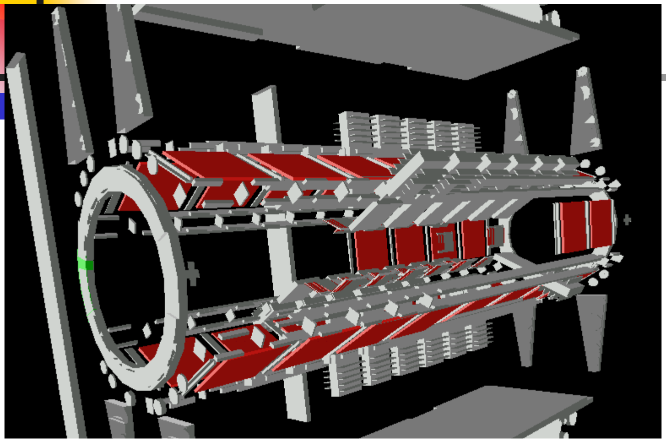
Where am I?

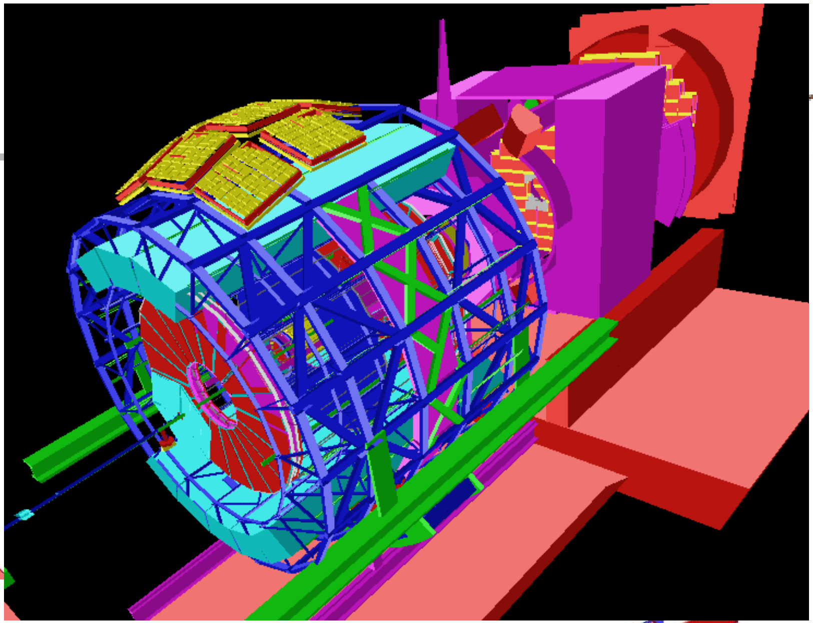
Distance to boundary

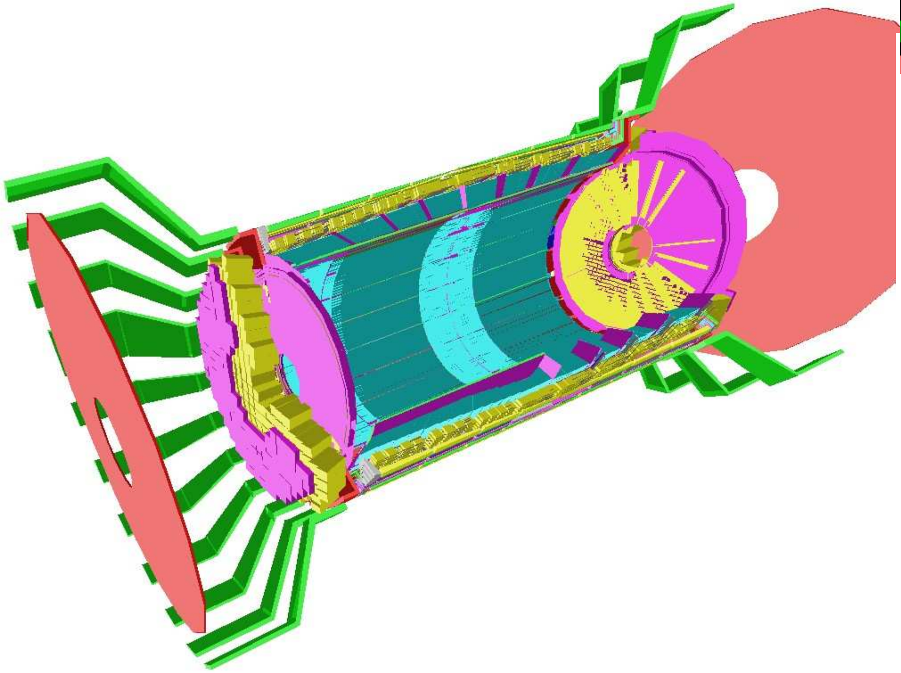
Closest boundary

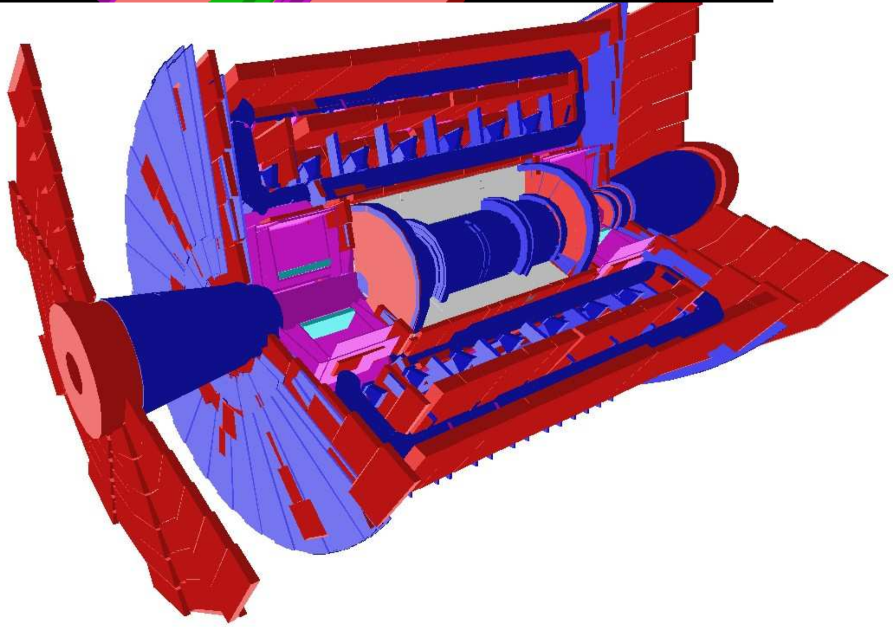
Persistency

Some detectors in ROOT geometry





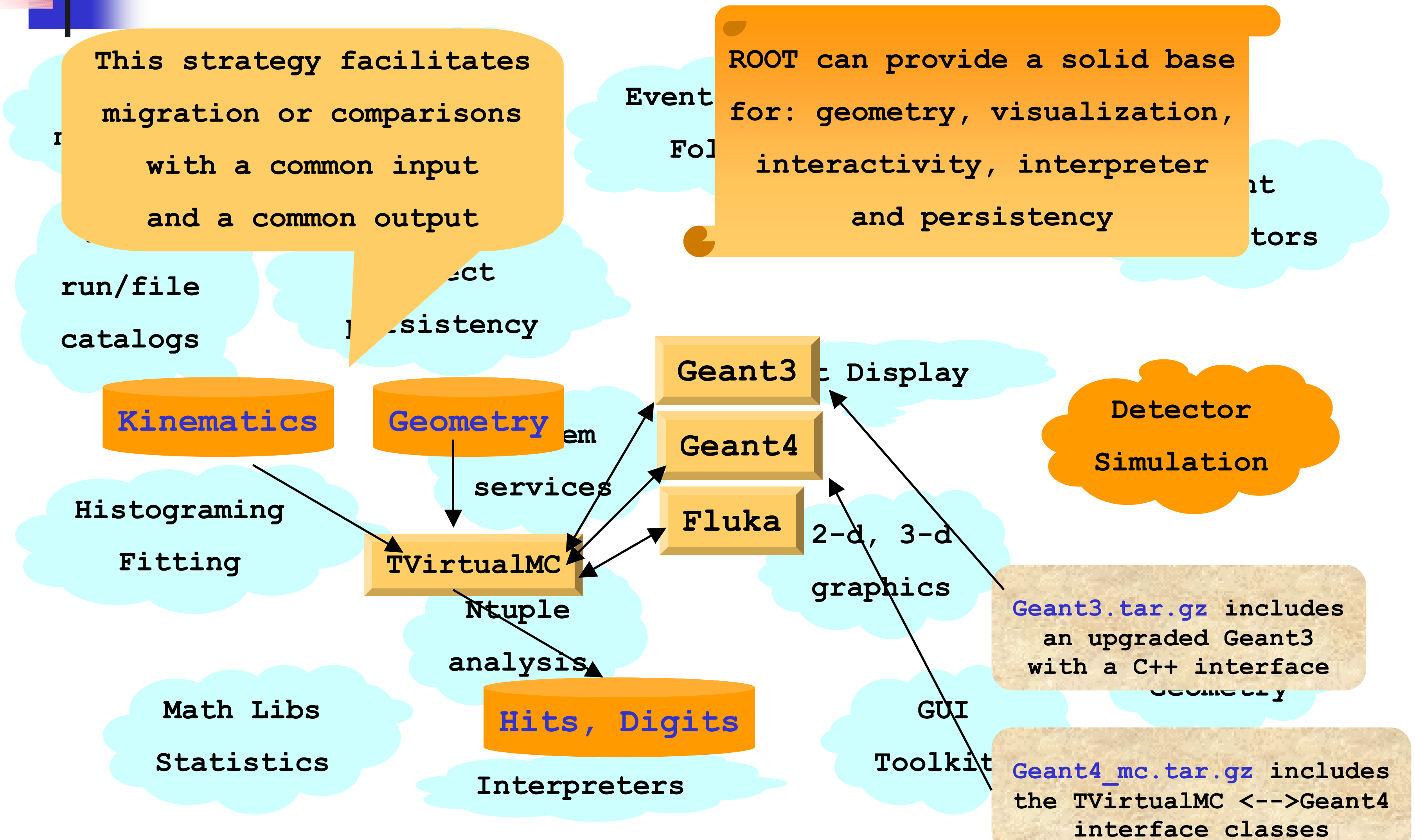




TGeo performance vs Geant3

	Number nodes	gtmedi physics	Root physics	Geant3/ Root	gtmedi random	Root random	Geant3/ Root
Gexam1	425	3.08	1.84	1.67	6.60	4.39	1.50
Gexam3	86	2.87	2.15	1.33	3.47	2.50	1.38
Gexam4	12781	2.51	2.20	1.14	12.09	11.18	1.08
Brahms	2649	5.82	3.04	1.91	4.17	1.93	2.16
Tesla	15370	6.56	5.58	1.17	12.95	7.15	1.81
CDF	24422	14.81	4.31	3.43	20.94	5.85	3.57
Minos_near	30988	30.93	20.99	1.47	21.57	13.70	1.57
BTeVEcal	52	1.57	1.08	1.45	1.78	0.73	2.43
BTeV	295310	45.27	25.88	1.75	197.06	26.83	7.34
CMSEcal	251713	5.60	1.81	3.09	5.69	1.74	3.27
CMS	1166310	33.57	8.76	3.83	39.09	24.98	1.56
LHCb	1533488	7.98	6.75	1.18	12.58	2.89	4.35
Alice	3080198	11.50	8.63	1.33	11.45	7.28	1.57
Atlas	29046966	8.90	9.94	0.89	32.48	23.39	1.38

Detector Simulation

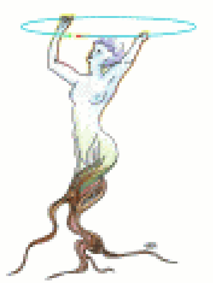


This strategy facilitates migration or comparisons with a common input and a common output

ROOT can provide a solid base for: geometry, visualization, interactivity, interpreter and persistency

Geant3.tar.gz includes an upgraded Geant3 with a C++ interface

Geant4_mc.tar.gz includes the TVirtualMC <-->Geant4 interface classes



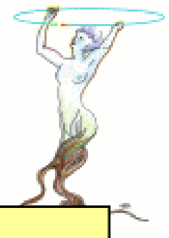
Modularity

Ignominy plots

Decoupling components

The Plug-in Manager

Package Metrics (Ignominy)



from Lassi Tuura

Project	Release	Packages	Average # of direct dependencies	Cycles (Packages Involved)	# of levels	ACD*	CCD*	NCCD*	Size
Anaphe	3.6.1	**31**	**2.6**	**--**	8	5.4	167	**1.3**	630/170k
ATLAS	1.3.2	**230**	**6.3**	**2 (92)**	96	70	16211	**10**	1350k
	1.3.7	**236**	**7.0**	**2 (92)**	97	77	18263	**11**	1350k
CMS/ORCA	4.6.0	**199**	**7.4**	**7 (22)**	**35**	24	4815	**3.6**	420k
	6.1.0	**385**	**10.1**	**4 (9)**	**29**	37	14286	**4.9**	580k
CMS/COBRA	5.2.0	**87**	**6.7**	**4 (10)**	19	15	1312	**2.7**	180k
	6.1.0	**99**	**7.0**	**4 (8)**	20	17	1646	**2.9**	200k
CMS/IGUANA	2.4.2	**35**	**3.9**	**--**	6	5.0	174	**1.2**	150/38k
	3.1.0	**45**	**3.3**	**1 (2)**	8	6.1	275	**1.3**	150/60k
Geant4	4.3.2	**108**	**7.0**	**3 (12)**	21	16	1765	**2.8**	680k
ROOT	2.25/05	**30**	**6.4**	**1 (19)**	22	19	580	**4.7**	660k

**) John Lakos, Large-Scale C++ Programming*

- Size = total amount of source code (roughly—not normalised across projects!)
- ACD = average component dependency (~ libraries linked in)
- CCD = sum of single-package component dependencies over whole release: test cost
- **NCCD = Measure of CCD compared to a balanced binary tree**
  - < 1.0: structure is flatter than a binary tree (= independent packages)
  - > 1.0: structure is more strongly coupled (vertical or cyclic)
  - **Aim: Minimise NCCD for given software/functionality** (good toolkit: ~ 1.0)

Plug-in Manager

Extended in

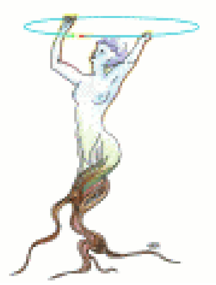
3.03/09

- Where are plug-ins used?

```
TFile *rf = TFile::Open("rfio://castor.cern.ch/alice/aap.root")

TFile *df = TFile::Open("dcache://main.desy.de/h1/run2001.root")
```

- Previously dependent on "magic strings" in source, e.g. in TFile.cxx:
- Adding case or changing strings requires code change and recompilation. Not user customizable.
- Currently 29 plug-ins are defined for 20 different (abstract) base classes
- No magic strings in code anymore

The Plug-in Manager is expected to solve most problems reported by Lassi



Relations with LCG

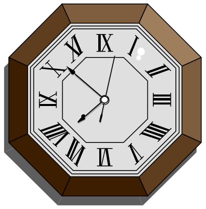
Where we agree

Possible problems

Wishes

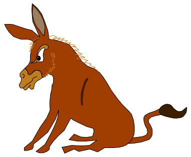
The LCG Project

- Must be a success
- Because success is not guaranteed
  - we better start on solid grounds
  - time is very very very critical



- The LCG has an opportunity to capitalize on the success of ROOT.

- But, there is a potential danger to see
  - parallel developments and conflicts
  - software “balkanisation” and “saucissonage”
  - <u>wrong balance in manpower between projects and experiments</u>



Proposal to Blueprint RTAG in June

The existing set of ROOT libs is the starting core of the LCG software. Because the system is already widely distributed and used, it guarantees the initial acceptance of a wide community.

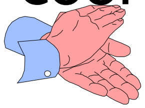
We invite architects and key developers to review the current organisation of libraries and to propose an evolution if it proves necessary.

This proposal was rejected at the level of the architects (3:1) In fact, it was never discussed on pure technical grounds.

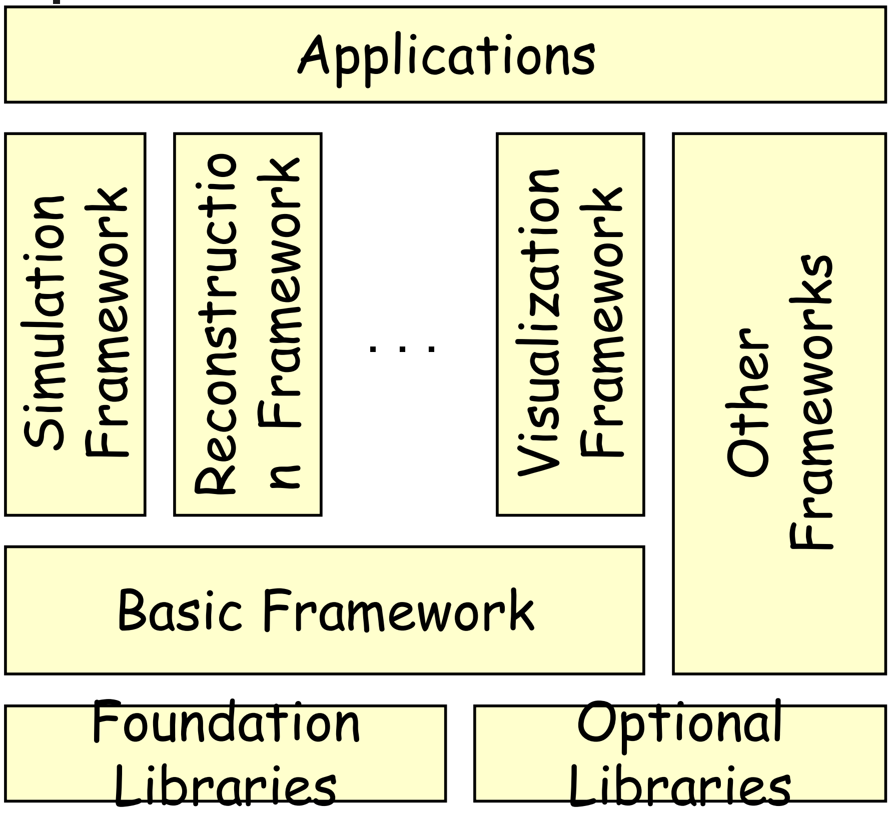
Blueprint RTAG & ROOT

The ROOT data analysis framework is widely used in HENP and beyond, and is being heavily used by the LHC experiments and the LCG. We see the LCG software as a user of ROOT; a user with a very close relationship with the ROOT team. While the ROOT team is highly attuned and responsive to the needs of the LHC experiments, it also supports a large and diverse non-LHC community (including many major HENP experiments) with its own requirements, not least the stability of ROOT itself.
It is impractical for LCG software architecture and development to be tightly coupled to ROOT and vice versa. We expect the user-provider relationship to work much better. The ROOT team has an excellent record of responsiveness to users. So while ROOT will be used at the core of much LCG software for the foreseeable future, there will always be a 'line' with ROOT proper on one side and LCG software on the other.
ROOT itself will grow and change over time. Decisions on making use of ROOT in the implementation of LCG software components should be made on a case by case basis, driven by the circumstances. Despite the user-provider relationship, LCG software may nonetheless place architectural, organizational or other demands on ROOT. For example, the library organization and factorization of ROOT will impact component interdependencies in LCG software employing ROOT implementations and may drive changes in the organization and/or factorization.

Blueprint RTAG impressions

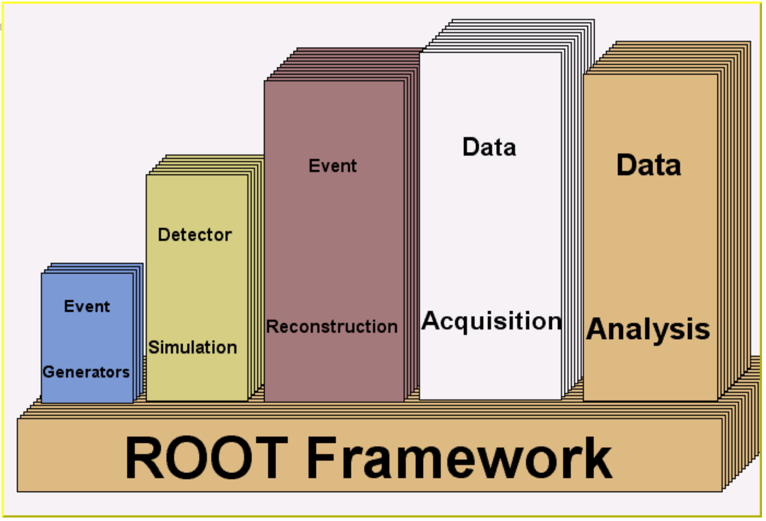
- We had useful meetings, sometime hot meetings.
- An opportunity to discuss many topics (first time since many years!). Dialog is better than wars.
- There is nothing wrong if there are some divergences. Uniformity is a bad sign.
- We had input from architects and some experts. I hope that we will get feedback now from end users.
- Torre, as application area coordinator, has done a great job in just a few months. 
- There are a few but important points where I positively disagree with him (next slides).

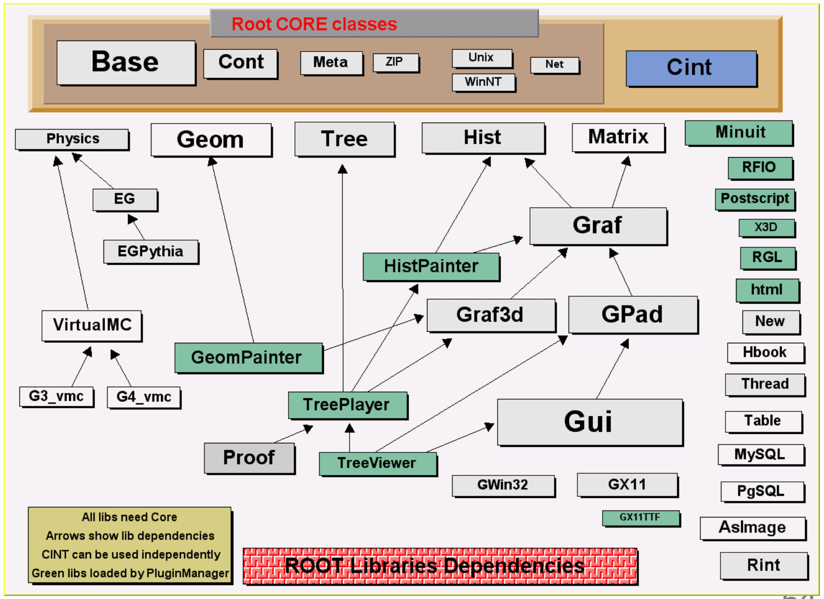
Software Structure



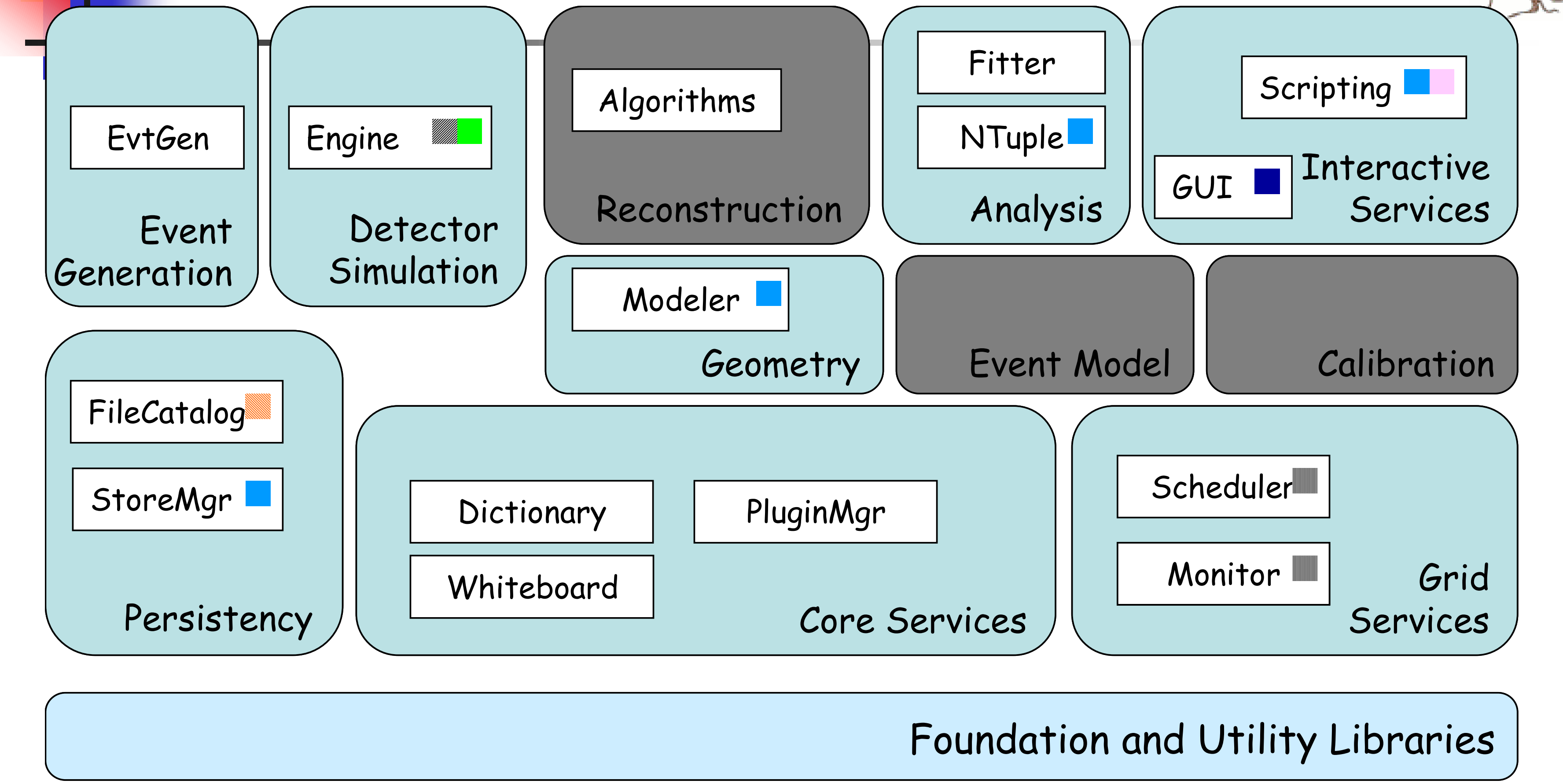
Blueprint RTAG 2002

Picture from 1995





Domain Decomposition



EvtGen

Event Generation

Engine ▨🟩

Detector Simulation

Algorithms

Reconstruction

Fitter

NTuple 🟦

Analysis

Scripting 🟦🟪

GUI ■

Interactive Services

Modeler 🟦

Geometry

Event Model

Calibration

FileCatalog 🟧

StoreMgr 🟦

Persistency

Dictionary

Whiteboard

PluginMgr

Core Services

Scheduler ▦

Monitor ▦

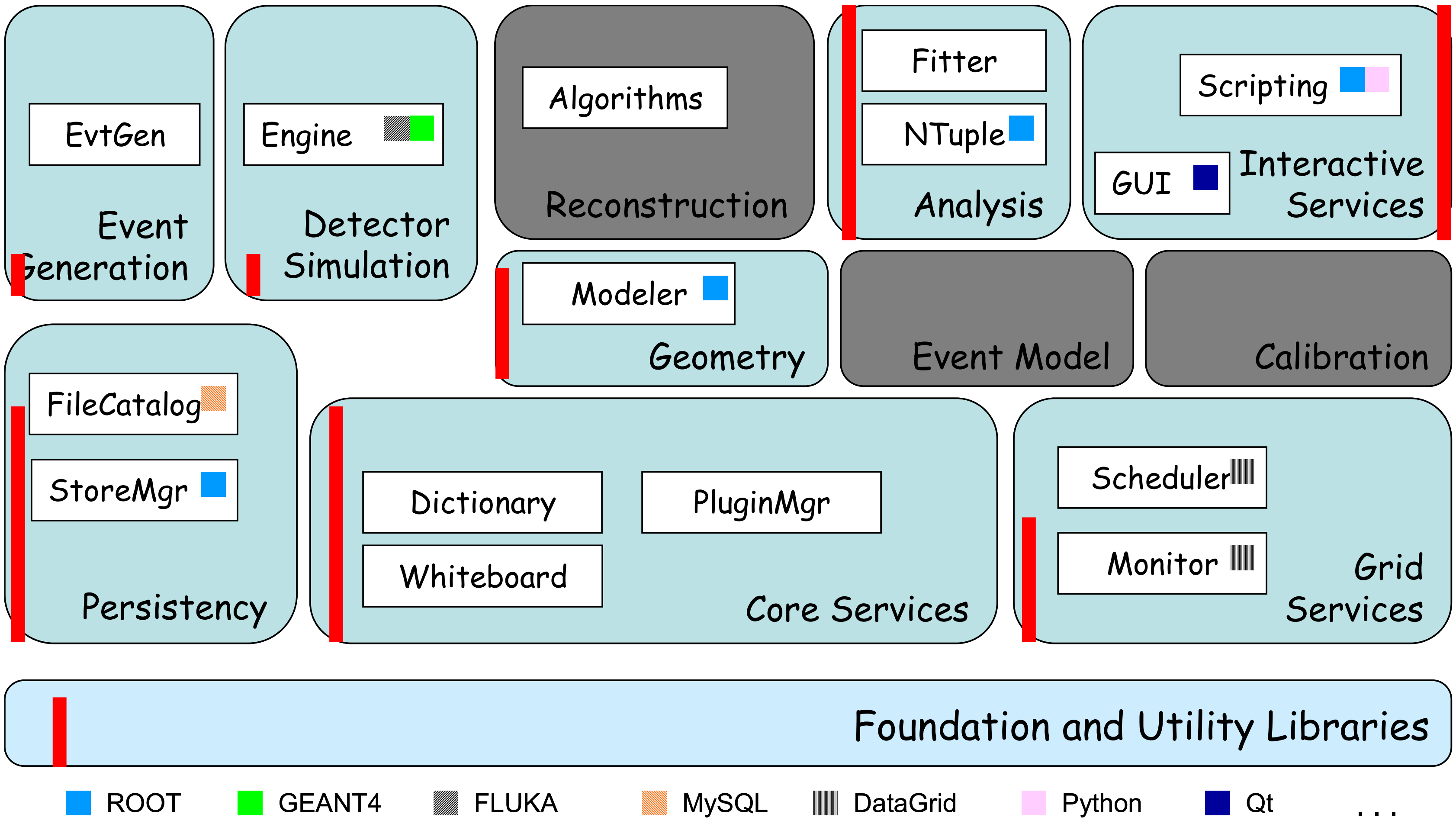
Grid Services

Foundation and Utility Libraries

🟦 ROOT 🟩 GEANT4 ▨ FLUKA 🟧 MySQL ▦ DataGrid 🟪 Python ■ Qt . . .

Products mentioned are examples; not a comprehensive list

Domain Decomposition
What ROOT covers in red



EvtGen

Event Generation

Engine

Detector Simulation

Algorithms

Reconstruction

Fitter

NTuple

Analysis

Scripting

GUI

Interactive Services

Modeler

Geometry

Event Model

Calibration

FileCatalog

StoreMgr

Persistency

Dictionary

PluginMgr

Whiteboard

Core Services

Scheduler

Monitor

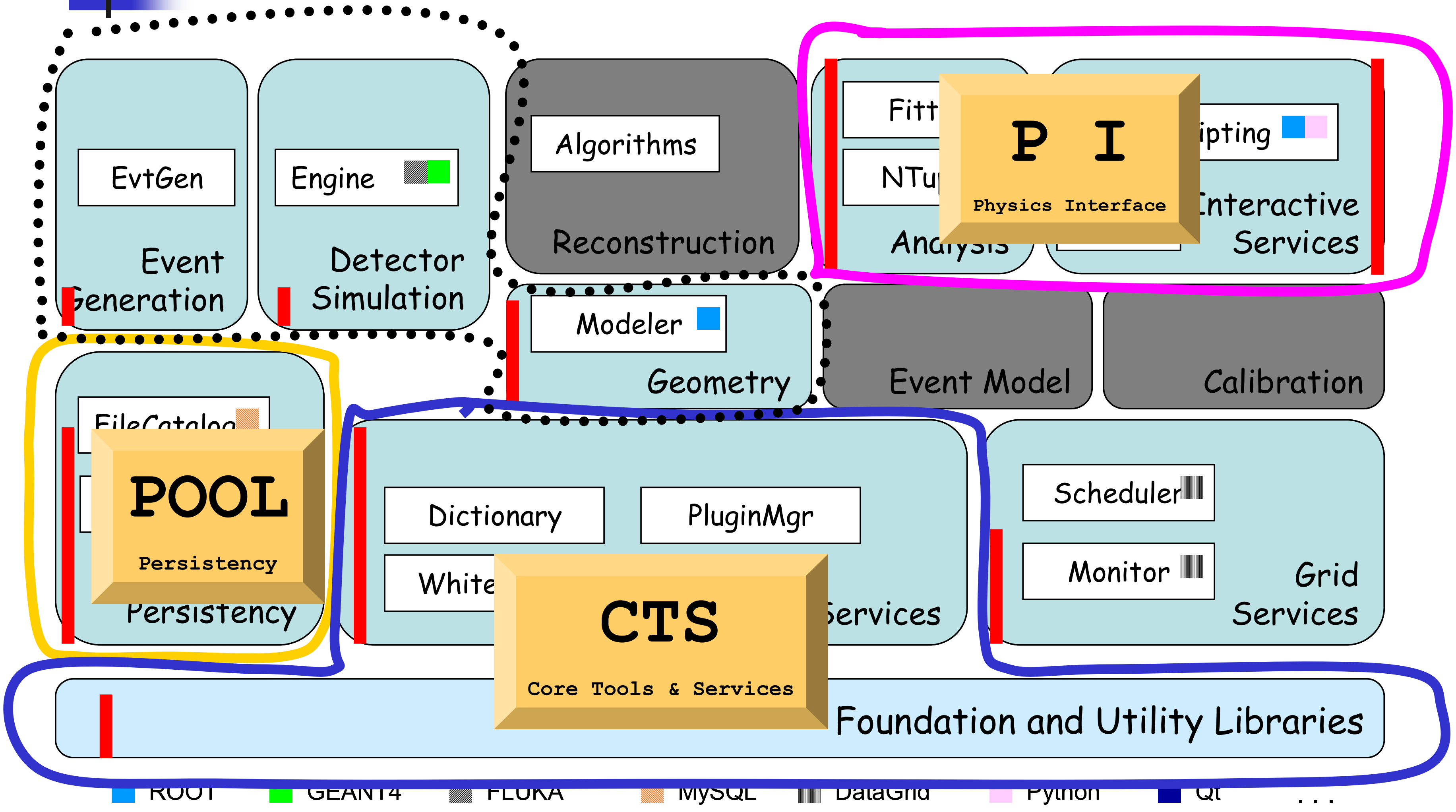
Grid Services

Foundation and Utility Libraries

🟦 ROOT 🟩 GEANT4 ▨ FLUKA ▨ MySQL ▩ DataGrid ■ Python ■ Qt . . .

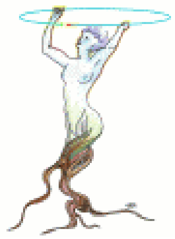
Domain Decomposition

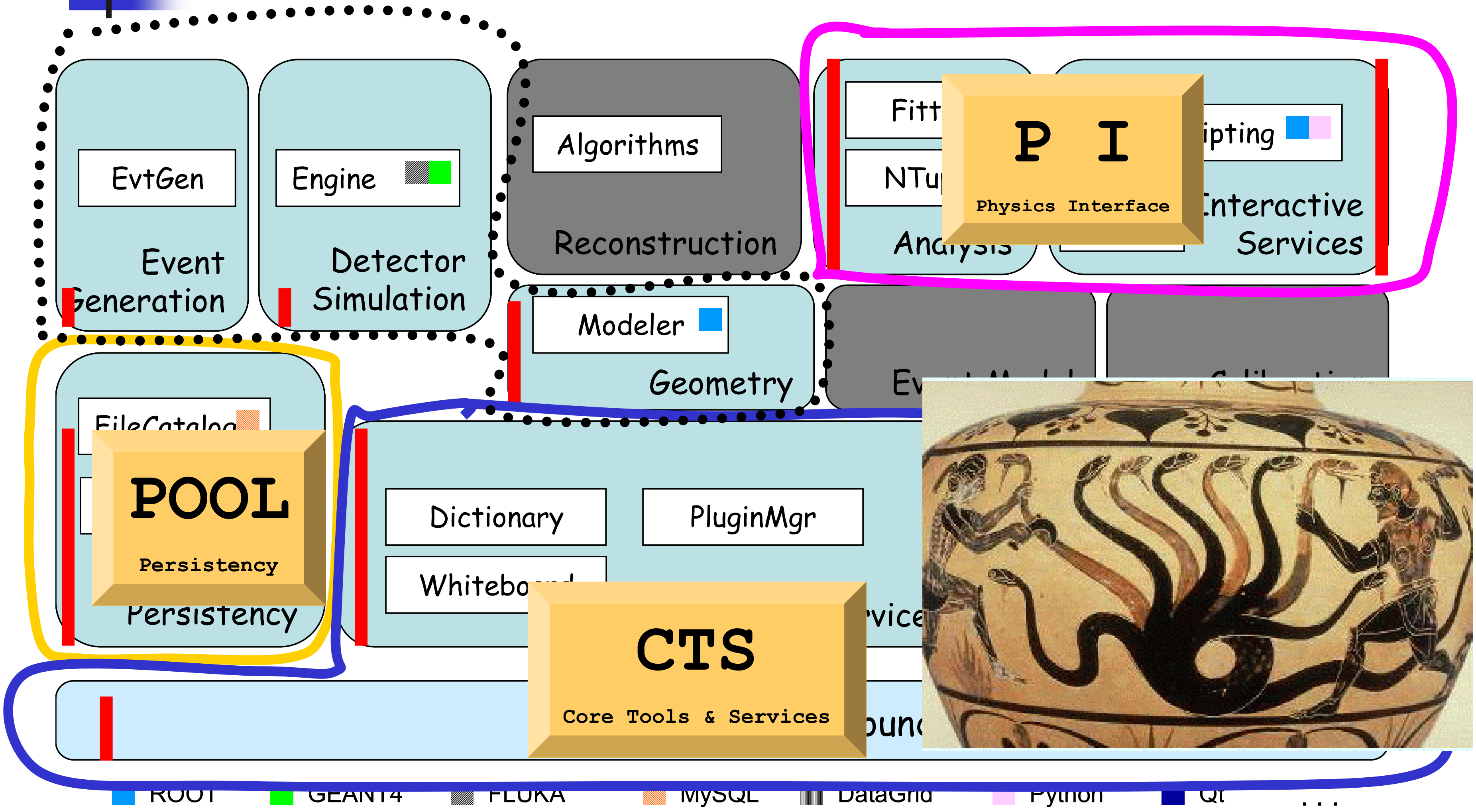
LCG proposed projects



Domain Decomposition

LCG proposed projects



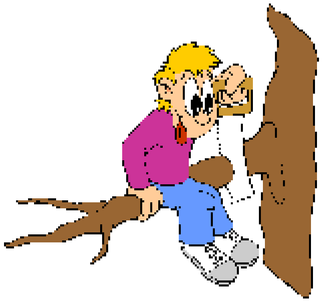


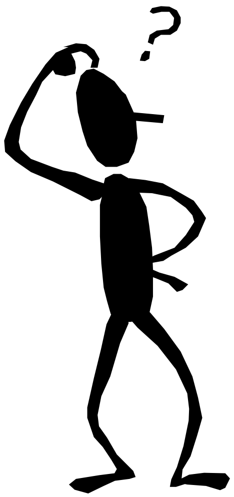
LCG proposed projects

- Scenario 1
- Scenario 2
- Scenario 3

Two extreme views on the LCG proposal

A simple alternative model





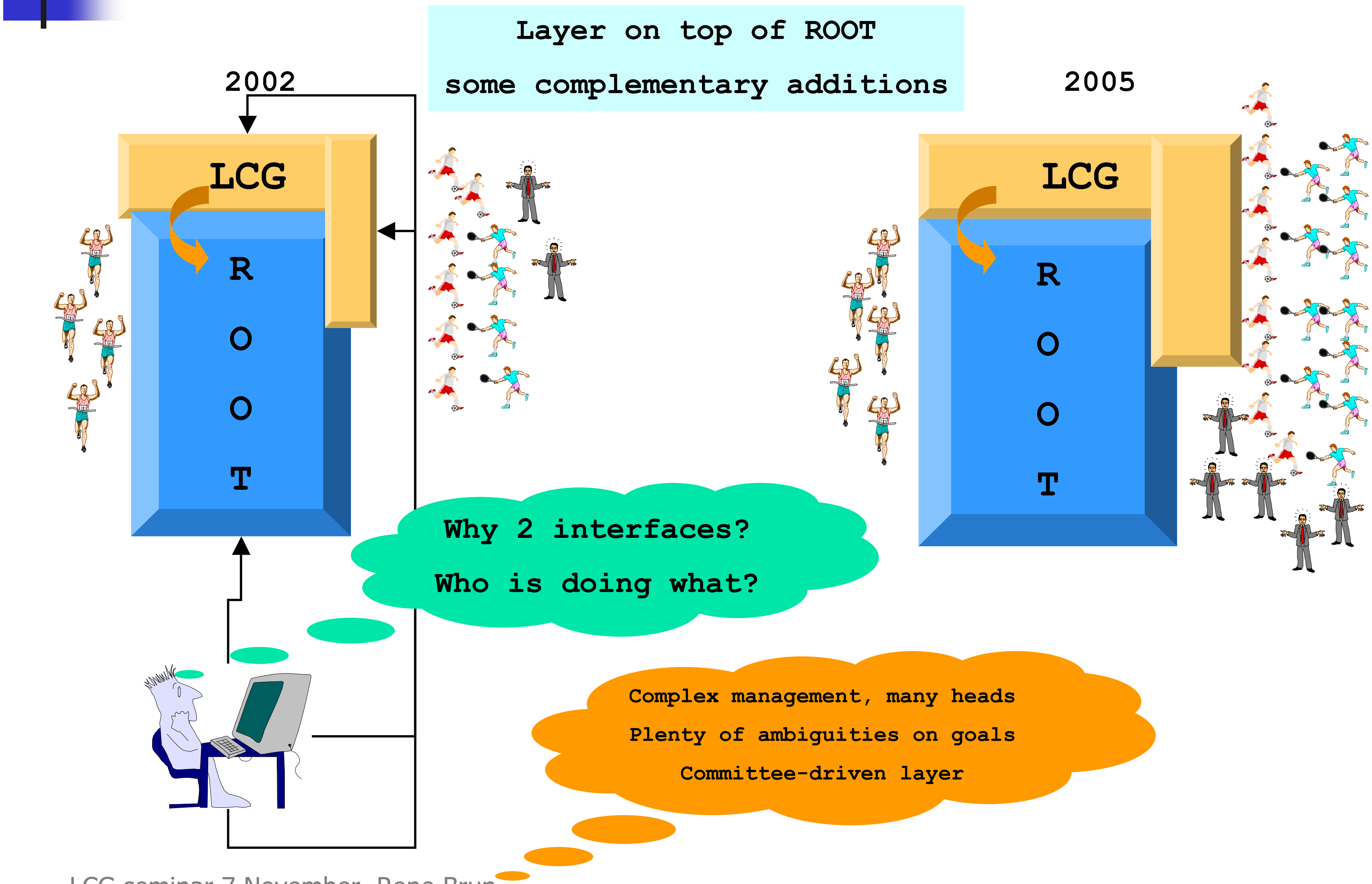
We more or less agree on the domain decomposition.

We agree that RTAGS are necessary and useful..

The mapping between domains and responsibilities is wrong

LCG - ROOT : Scenario 1

My understanding of the LCG proposal



Layer on top of ROOT

some complementary additions

Why 2 interfaces?

Who is doing what?

Complex management, many heads

Plenty of ambiguities on goals

Committee-driven layer

LCG - ROOT : Scenario 1 comments

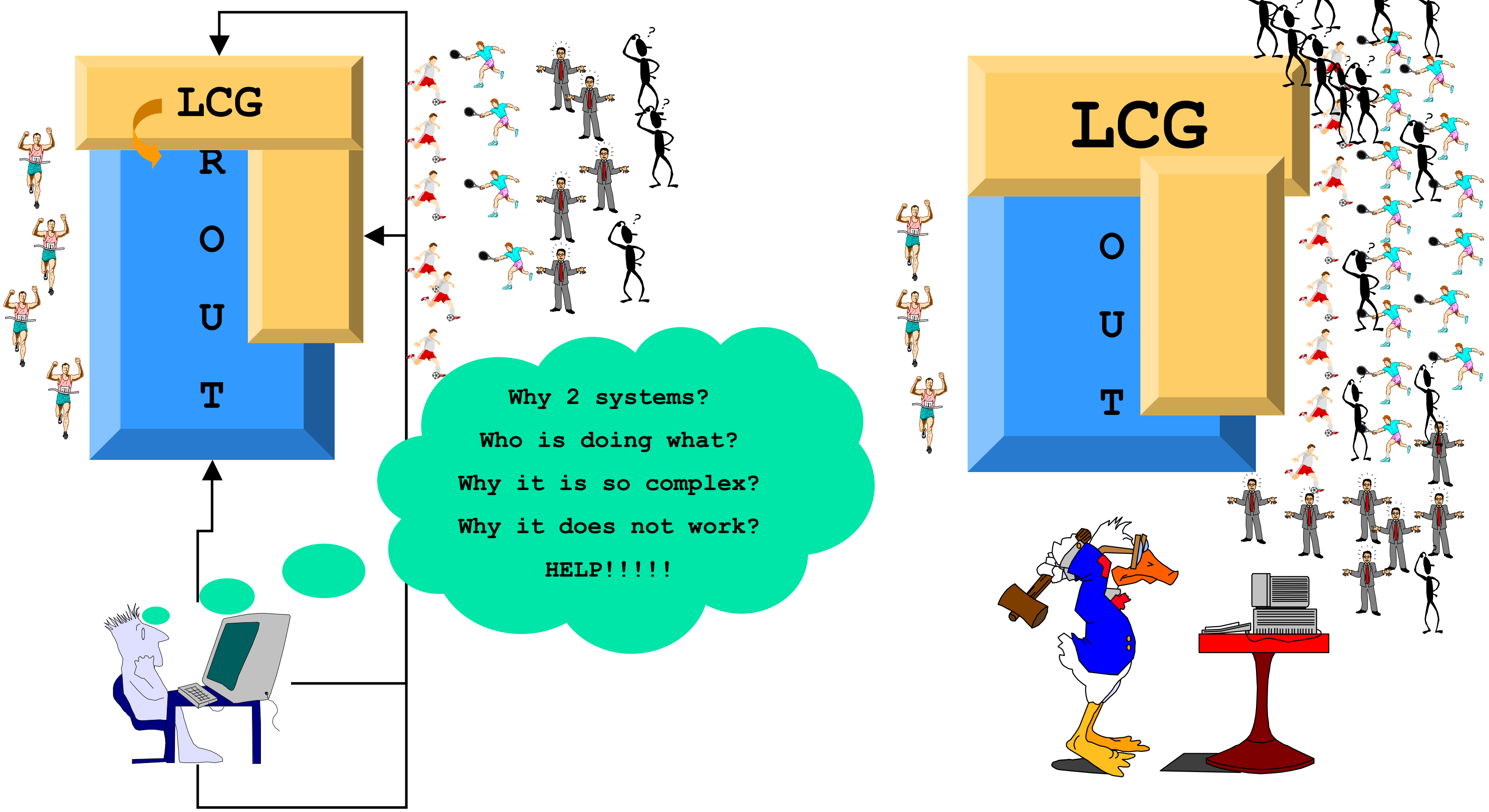
- Assuming that S1 is the way to go, we will provide the best possible service (as provider).
- We anticipate a committee-driven system that will not be in the interest of the users.
- It will be difficult to maintain a coherent view between ROOT and the other projects working in the same domains.
- Cannot be a motivating environment because you cannot be creative in this context.
- Motivation will come from the large ROOT users base (LHC being a fraction only).

LCG - ROOT : Scenario 2

The conditions for a failure

Layer on top of ROOT in 2002

becoming an alternative in 2005



2002

2005

Why 2 systems?

Who is doing what?

Why it is so complex?

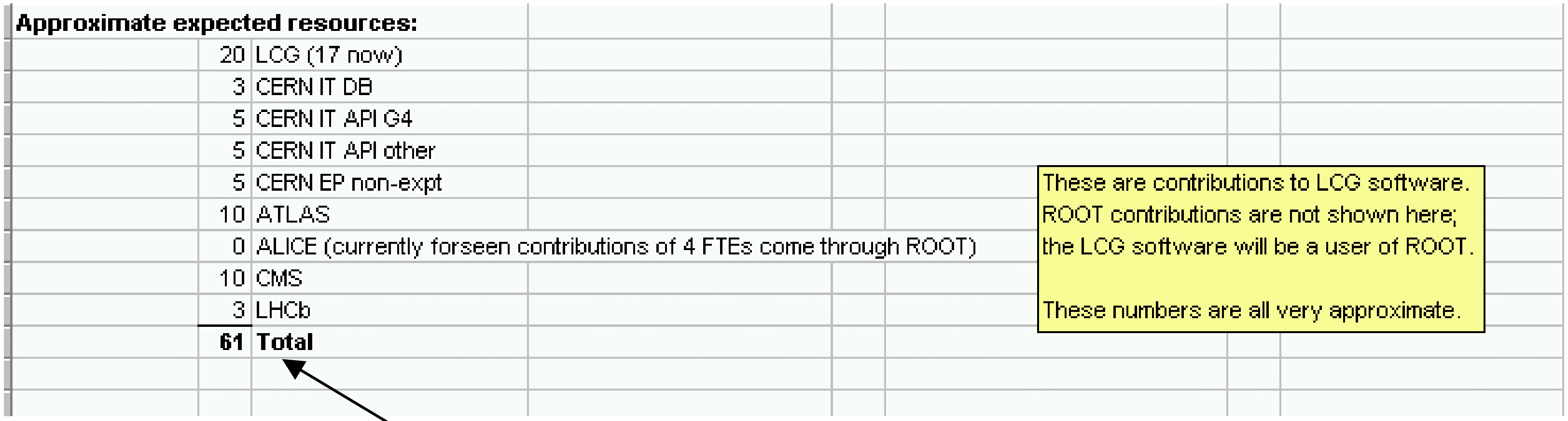
Why it does not work?

HELP!!!!!

LCG - ROOT : Scenario 2 Comments

- The risk is that Scenario 1 may drift gradually to Scenario 2.
- User-Provider relation at the beginning
- Parts of ROOT copied to the new project
- Two parallel projects
- ROOT: seen as a pain in the neck
- We better stop ROOT now in the LCG context if the "secret idea" is to follow this scenario.

Personnel resources

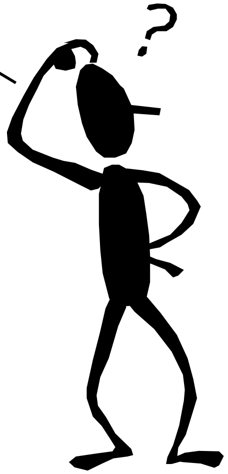


Approximate expected resources:

20	LCG (17 now)
3	CERN IT DB
5	CERN IT API G4
5	CERN IT API other
5	CERN EP non-expt
10	ATLAS
0	ALICE (currently forseen contributions of 4 FTEs come through ROOT)
10	CMS
3	LHCb
61	**Total**

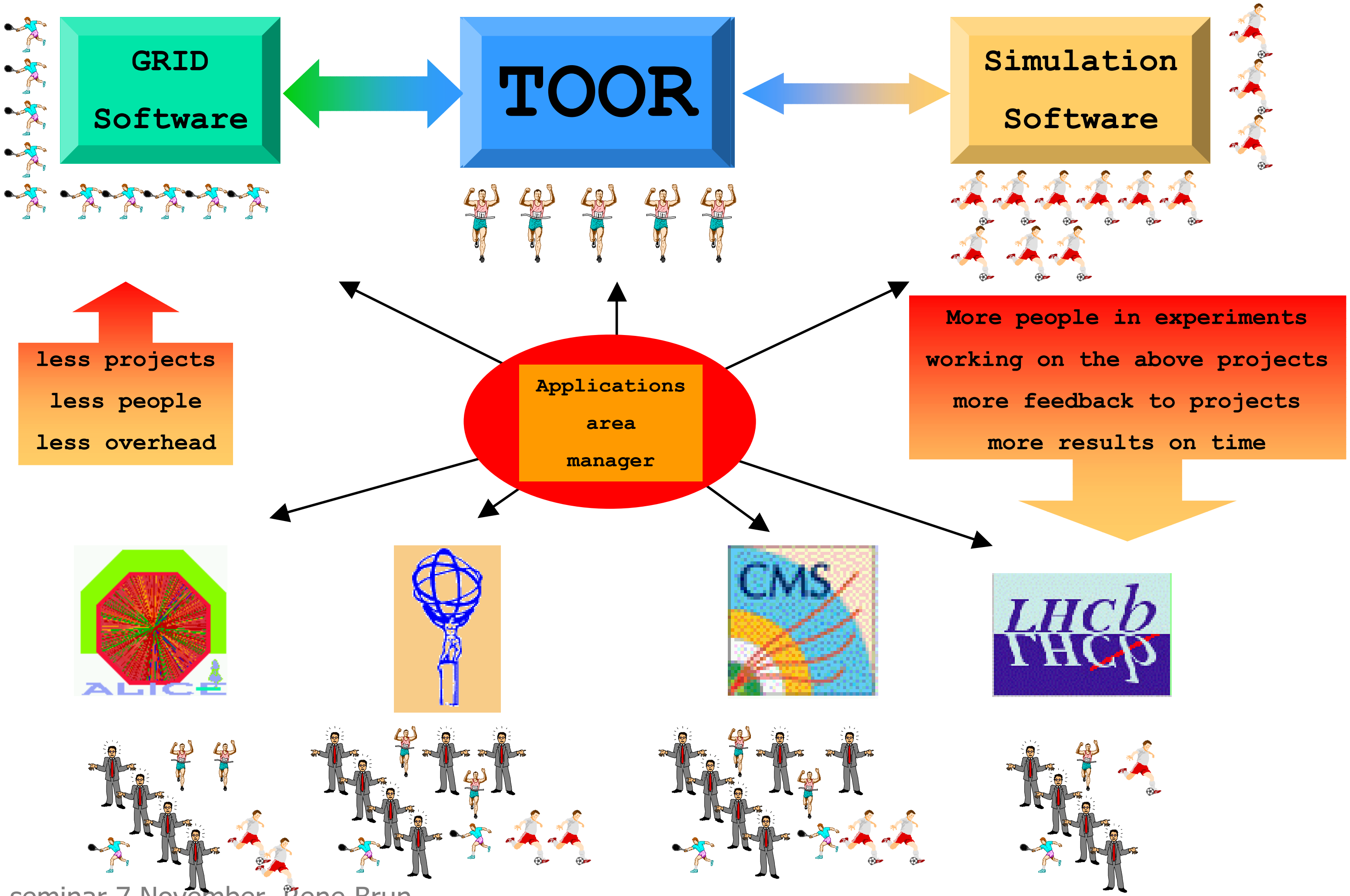
These are contributions to LCG software.
ROOT contributions are not shown here;
the LCG software will be a user of ROOT.

These numbers are all very approximate.



LCG - ROOT : Scenario 3

The conditions for success



GRID Software

TOOR

Simulation Software

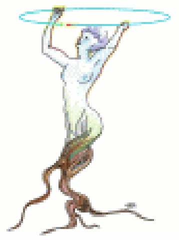
less projects
less people
less overhead

Applications area manager

More people in experiments
working on the above projects
more feedback to projects
more results on time

LCG - ROOT : Scenario 3 comments

- More realistic mapping of existing responsibilities
- Clear path for ROOT and users
- Minimize possible divergences
- Facilitate the task of the applications coordinator, but gives him more responsibility.
- Easier to monitor progress
- Increase useful manpower in experiments



Summary 1

- We have developed a simple and powerful framework now in use by thousands of people in most HEP labs.
- We have many outside contributors.
- As already expressed to the Blueprint architects, we are willing to discuss the evolution of ROOT in close cooperation with the LHC experiments.
- Scenario 1 & 2 will end-up into fights and fat systems. It is our duty to ring the bell!
- We invite end-users in LHC experiments to reassess the situation in view of the realistic scenario 3.

Summary 2

- Software is not a technical problem. It is mainly a sociological problem.
- It is easy to write many lines of code. It is more difficult to give a momentum to a system.
- Our goal should not be to write software per se, but to make sure we deliver a simple, robust, coherent framework in time for the LHC.
- We must create the conditions to remove all possible ambiguities on the path to follow.

Users must give feedback now, not in 2005.