ROOT and ROOT4STAR framework

or

Software for “C++ speaking” people

V. Fine

OO approach

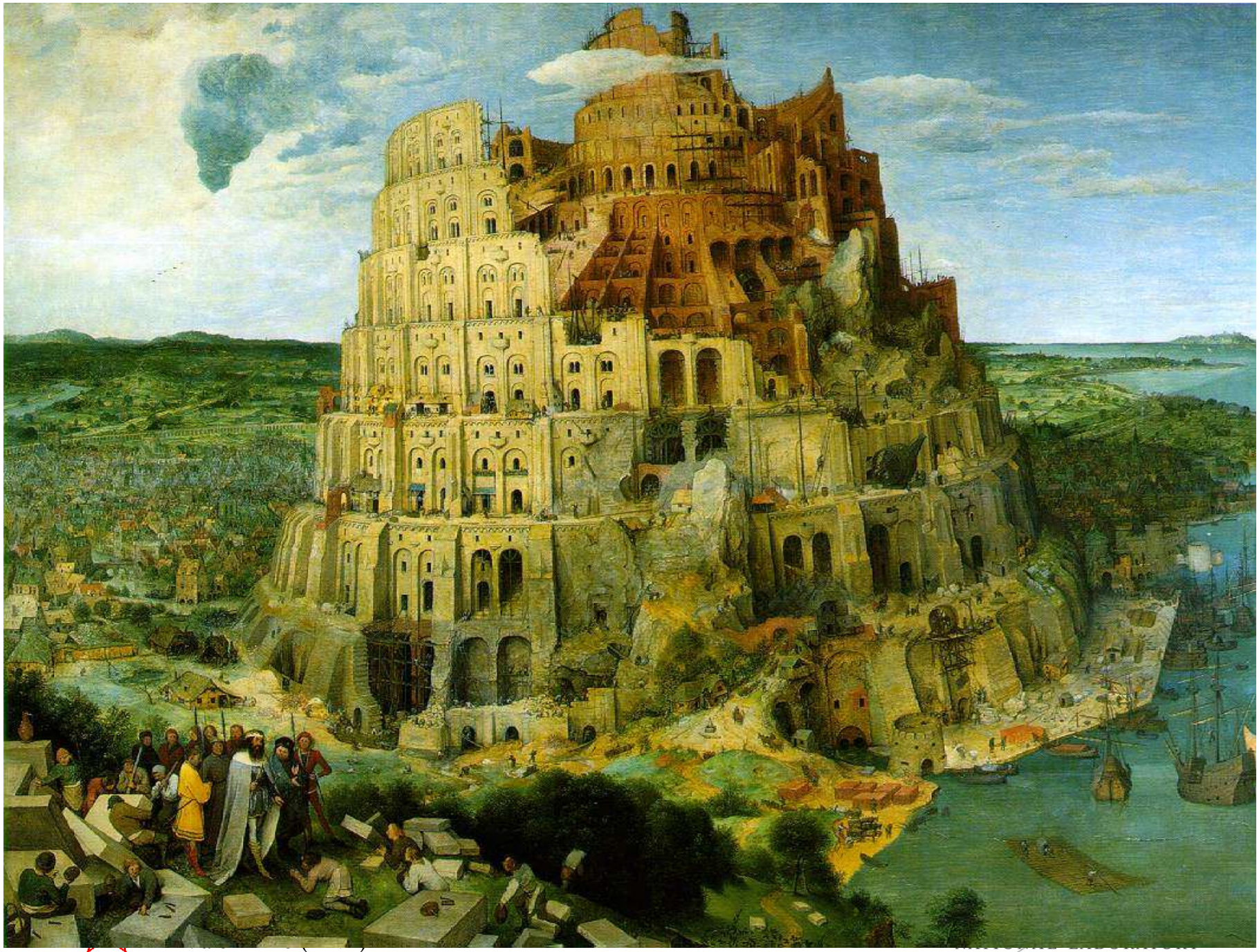
Object-Oriented (OO) programming had been identified and adopted by HEP communities as an efficient and powerful approach to developing capable, robust, maintainable software in this environment.

C++ adoption left us alone with a language that has no built-n tools to provide I/O and debug the code

Encapsulated data access via class methods (no direct access to bare data)

The hidden methods – access via “virtual method” / “abstract class”

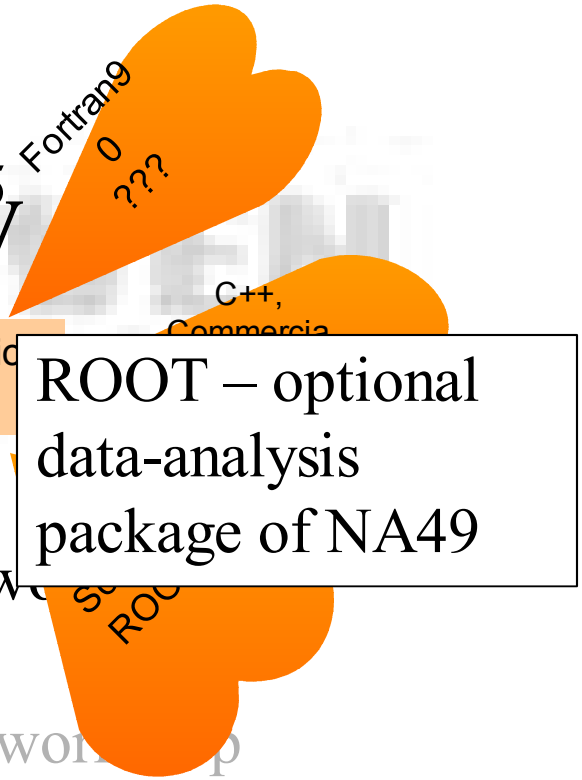
An it is a tool for community that collects data and WRITE the code to save and analyze it.



Project History

- 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 Intl Root w… FNAL)
- Feb 00: Root version 2.23/12 (2nd Intl Root wor… CERN)
- Sep 00: Root version 2.25/03
- Dec 00: Root version 3.00/01
- Jun 01: 3rd International Root workshop at FNAL

In 1994, fundamental divergence of opinions in Applicatio… Software Group in IT. The PAW/Geant3 team is dismantled.

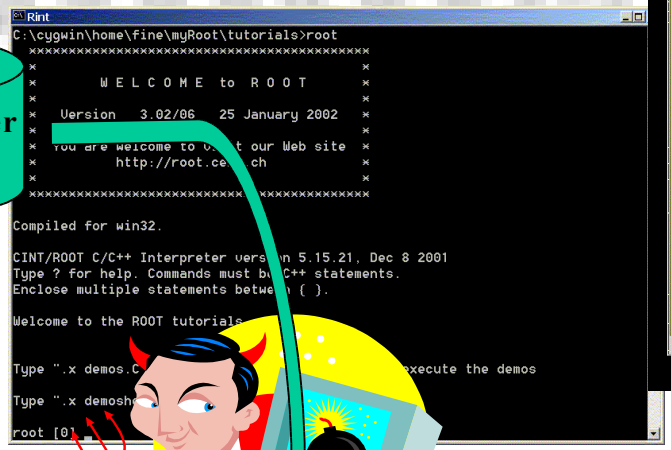


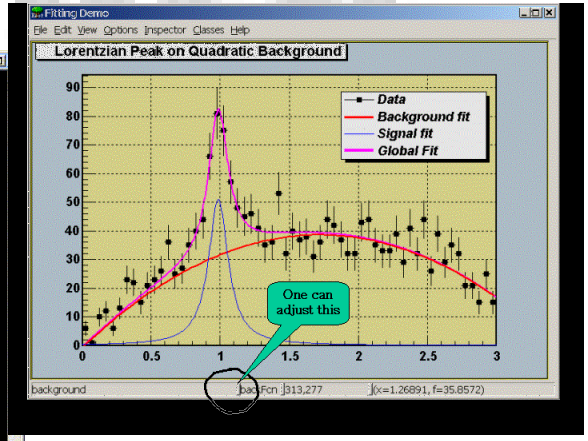
ROOT – optional data-analysis package of NA49

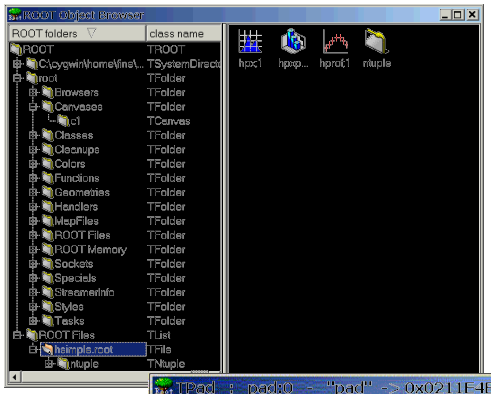
See: "Evolution with ROOT" by Rene Brun, HepVIS'01

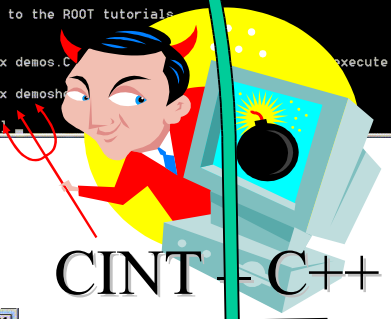
ROOT overview

C++ header files

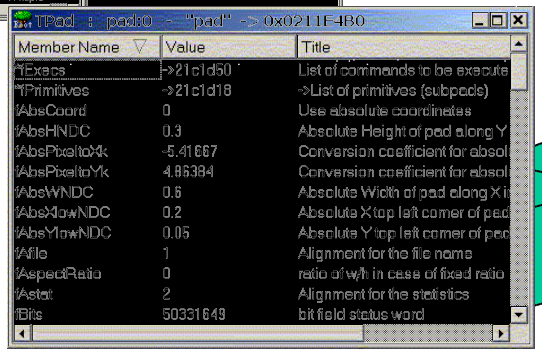


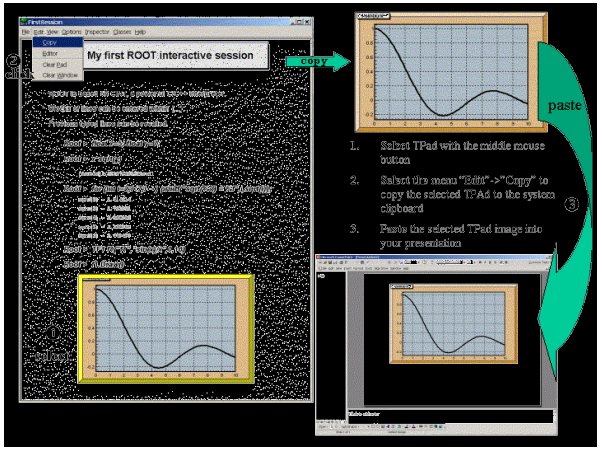






CINT - C++





ROOT's Services/Utilities

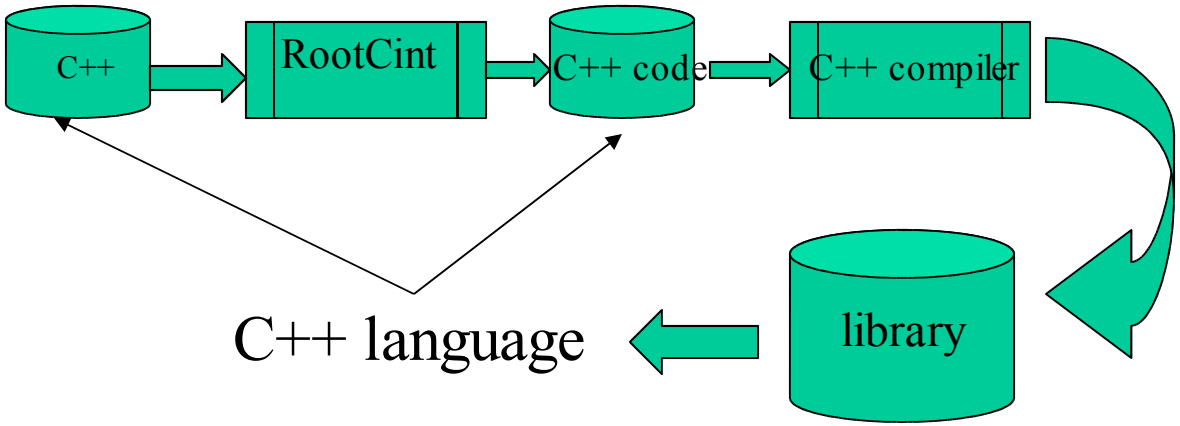
- Histogramming and Fitting
- Graphics (2D, 3D)
- I/O to file or socket: specialized for histograms, Ntuples (Trees)
- Collection Classes and Run Time Type Identification
- User Interface
  - GUI: Browsers, Panels, Tree Viewer
  - Command Line interface: C++ interpreter CINT
  - Script Processor (C++ compiled $\Leftrightarrow$ C++ interpreted)

How to apply ROOT to solve the concrete task.

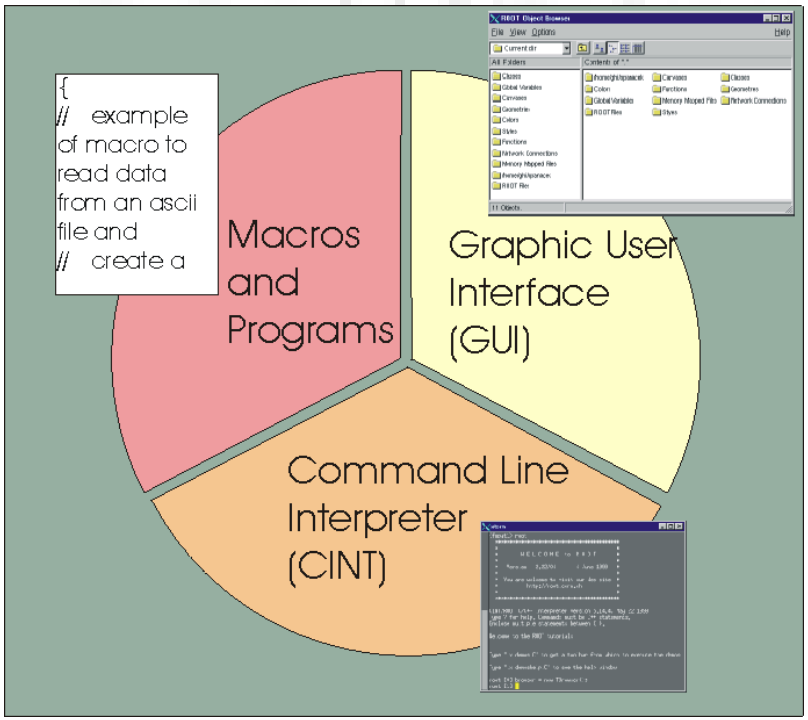
- Build OO model.
- Express it C++ language.
- Create a ROOT/Cint dictionary (which is a plain C++ code)
- Compile that dictionary to create a share library
- Create ROOT macro to load the brand-new library and instantiate your class object

It is a recurrent task and needs time to be completed (may be infinite one:-)

“pure” ROOT integration



Three User Interfaces



- GUI
  windows, buttons, menus
- Root Command line
  CINT (C++ interpreter)
- Macros, applications,
  libraries (C++ compiler
  and interpreter)

See: "Evolution with ROOT" by Rene Brun, HepVIS'01

STAR reconstruction framework*

* "*… a set of cooperating classes that make up a reusable design for a specific class of software …*"
by **Erich Gamma**,et al.
"Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Pub Co, 1995.

"The framework dictates the architecture of your application. It defines the over-all structure, its partitioning into classes and objects, the key responsibilities thereof, how the classes and objects collaborate, and the thread of control. "

A framework predefines these design parameters so **physicists can** design their solutions using a proven programming model and can **concentrate on the specifics of their applications**.

Object-Oriented Design Issues

http://www.kitware.com/vtkData/WhatIsVTK.html

Visualization Toolkit

(GE Corp. R&D)

3.3 Object-Oriented Design Issues

To the OO purist, the design of our visualization system poses some problems. In usual OO design, data structures and methods are encapsulated into objects. In our design, **algorithms** (i.e., methods) and **datasets** (i.e., data structures) are **encapsulated separately**.

Our departure from what might be considered a purer OO design is based on three factors.

- **First**, combining complex algorithms and datasets into a single object would result in excessively large objects. The simplicity and modularity of the resulting design would be compromised.
- **Second**, combining algorithms and datasets into objects would result in repeating code, since the implementation of an algorithm for different data types often differs only in regions of data access.
- **Third**, users naturally view algorithms as objects that operate on data objects. Thus the design is comfortable to users, which is a key element of good system design.

Other researchers differ from our view.

STAR framework is designed to support the chained components, which can themselves be composite sub-chains, with components (*"makers"*) managing "*datasets*" they have created and are responsible for.

An **TDataSet** class from which data sets and makers inherit allows the construction of hierarchical organizations of components and data, and centralizes almost all system tasks:

- **data set navigation**,
- **I/O, database access**,
- **inter-component communication.**

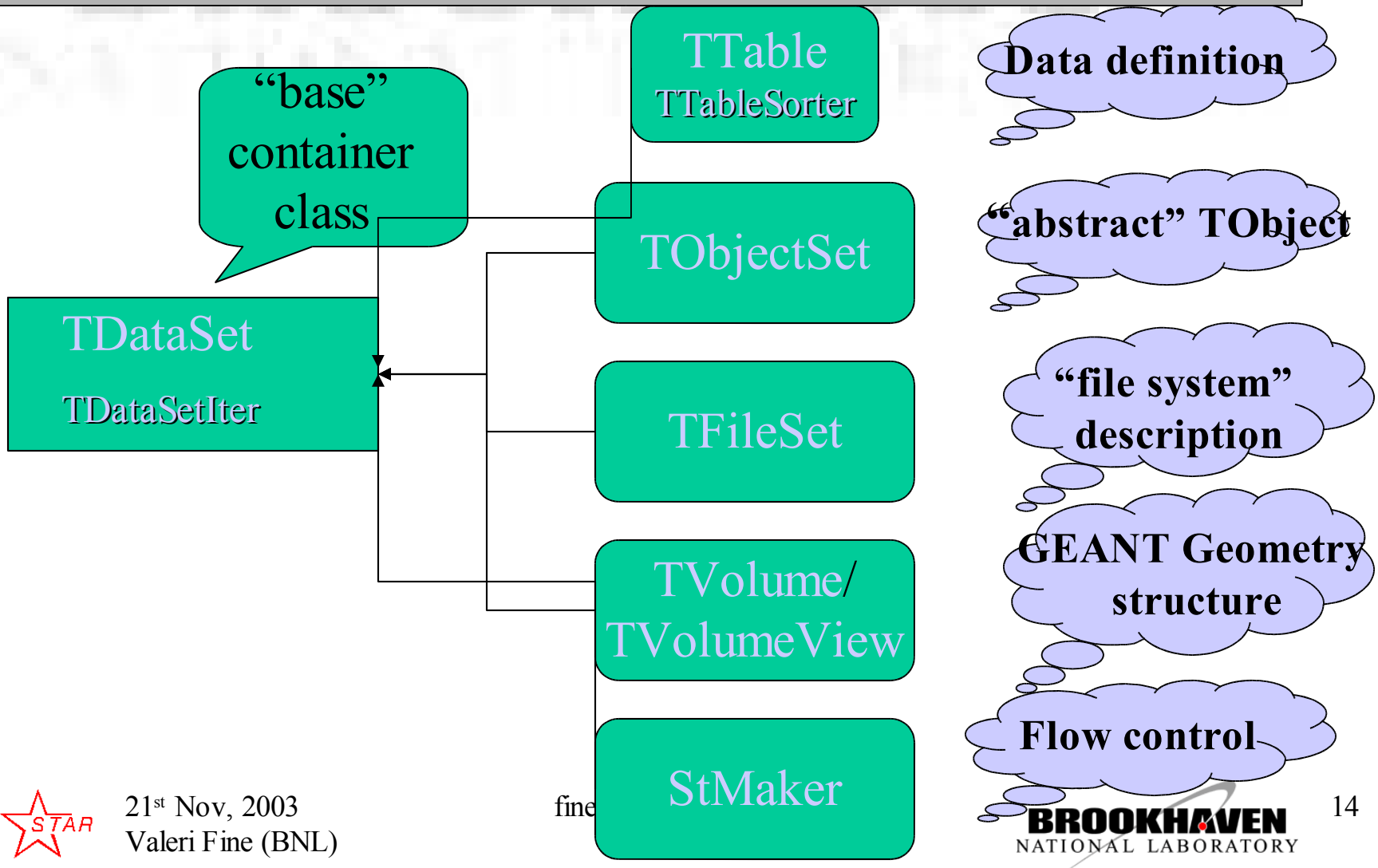
Basic TDataSet properites

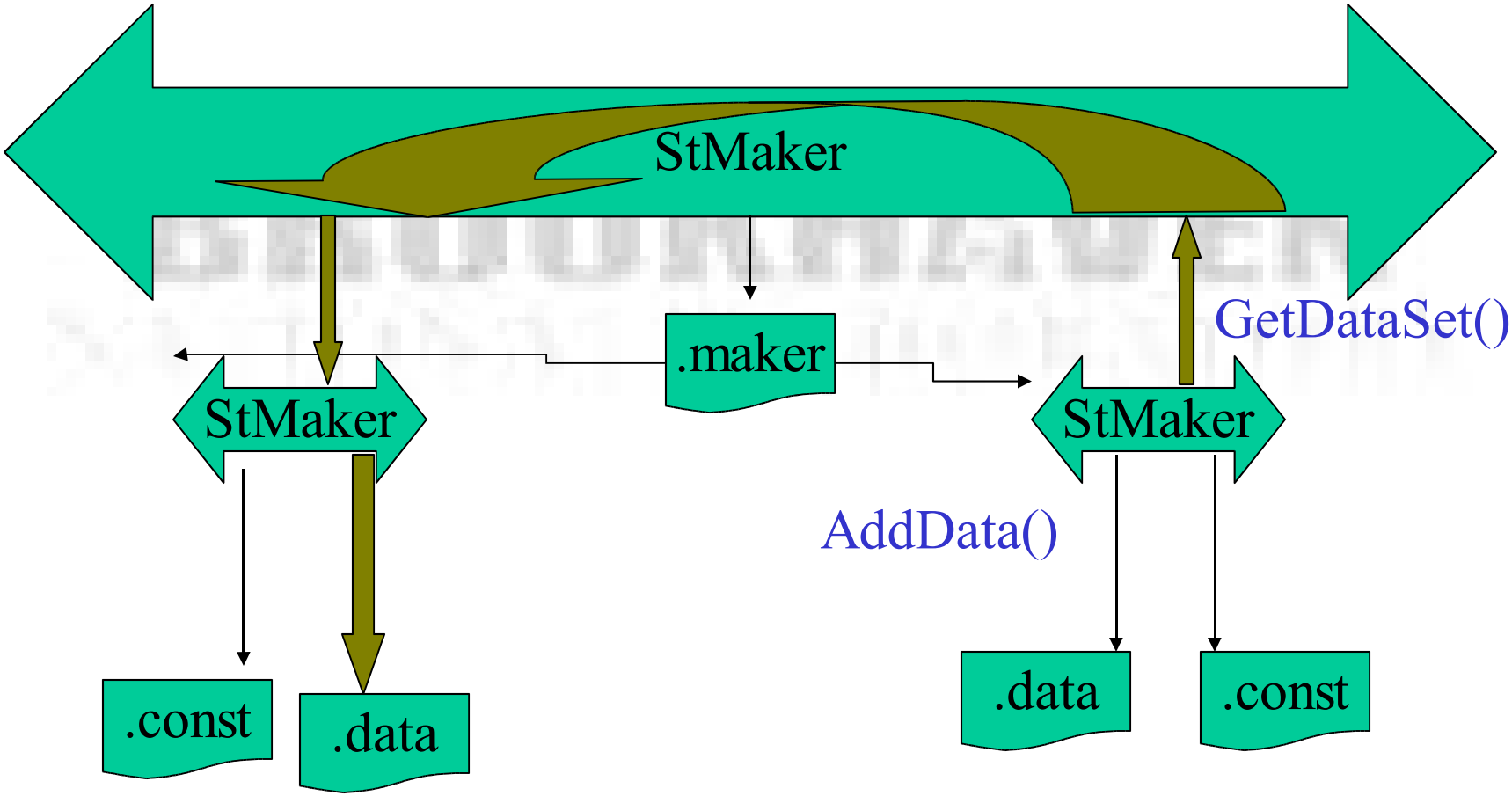
TDataSet object ::= the "named" collection of TDataSet objects

- ***Dataset Member***. Any object from the collection above is called "***DataSet Member***"
- ***Structural member***. The *"Dataset Member"* is its *"Structural member"* if its "back pointer" points to this object
- ***Dataset Owner*** (parent). We will say this **TDataSet** object "owns" (or is an owner / parent of) another **TDataSet** object if the last one is its *"Structural Member*"
- ***Associated member***. If some object is not *"Structural member"* of this object we will say it is an *"Associated Member"* of this dataset
- ***Orphan dataset***. If some dataset is a member of NO other **TDataSet** object it is called an *"orphan"* dataset object

OO model of the STAR simulation / reconstruction chain:

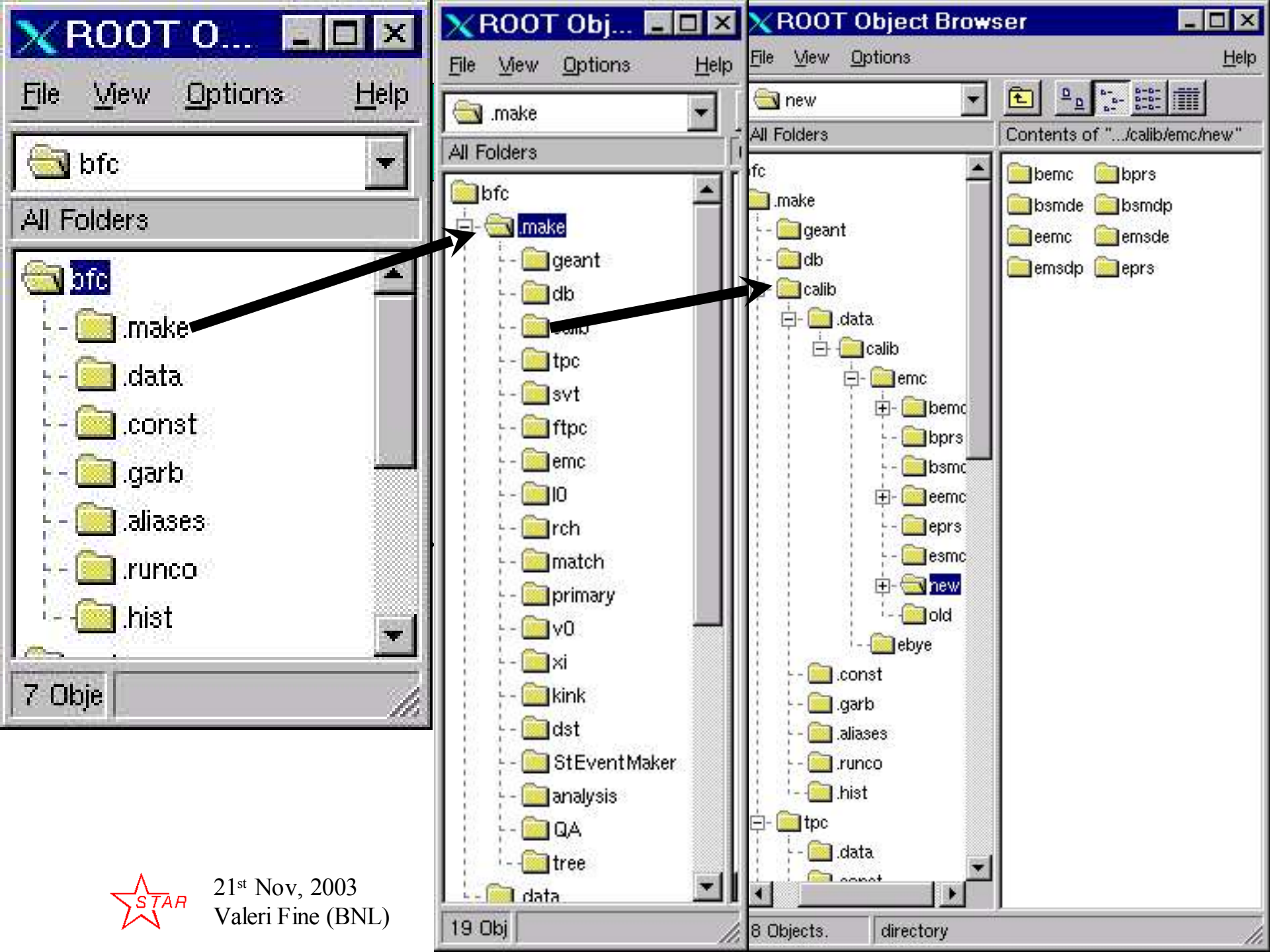
TDataSet object ::= the "named" collection of TDataSet objects





1. Init()
2. Make()
   2.1 InitRun()

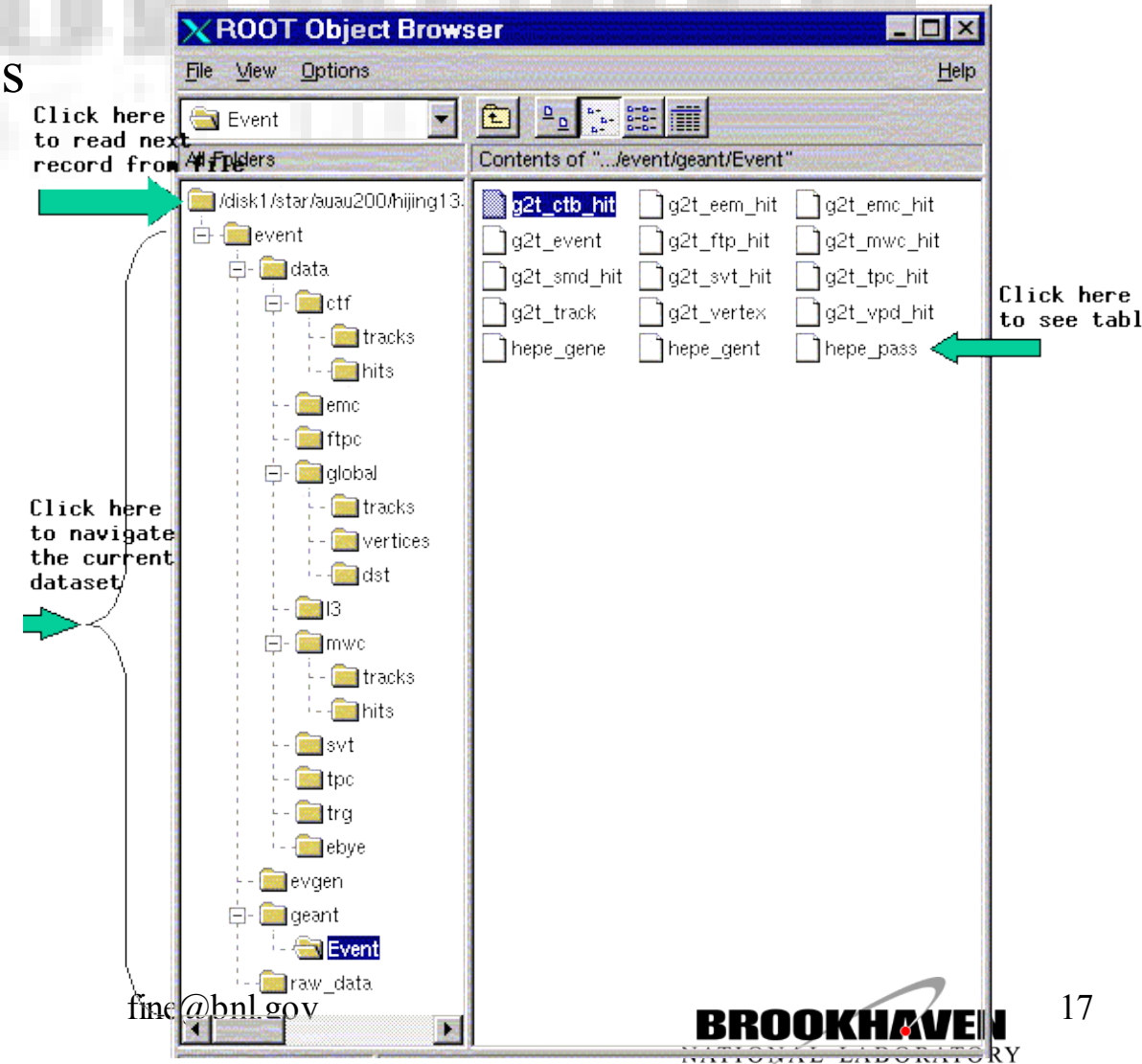
"regular" makers communication

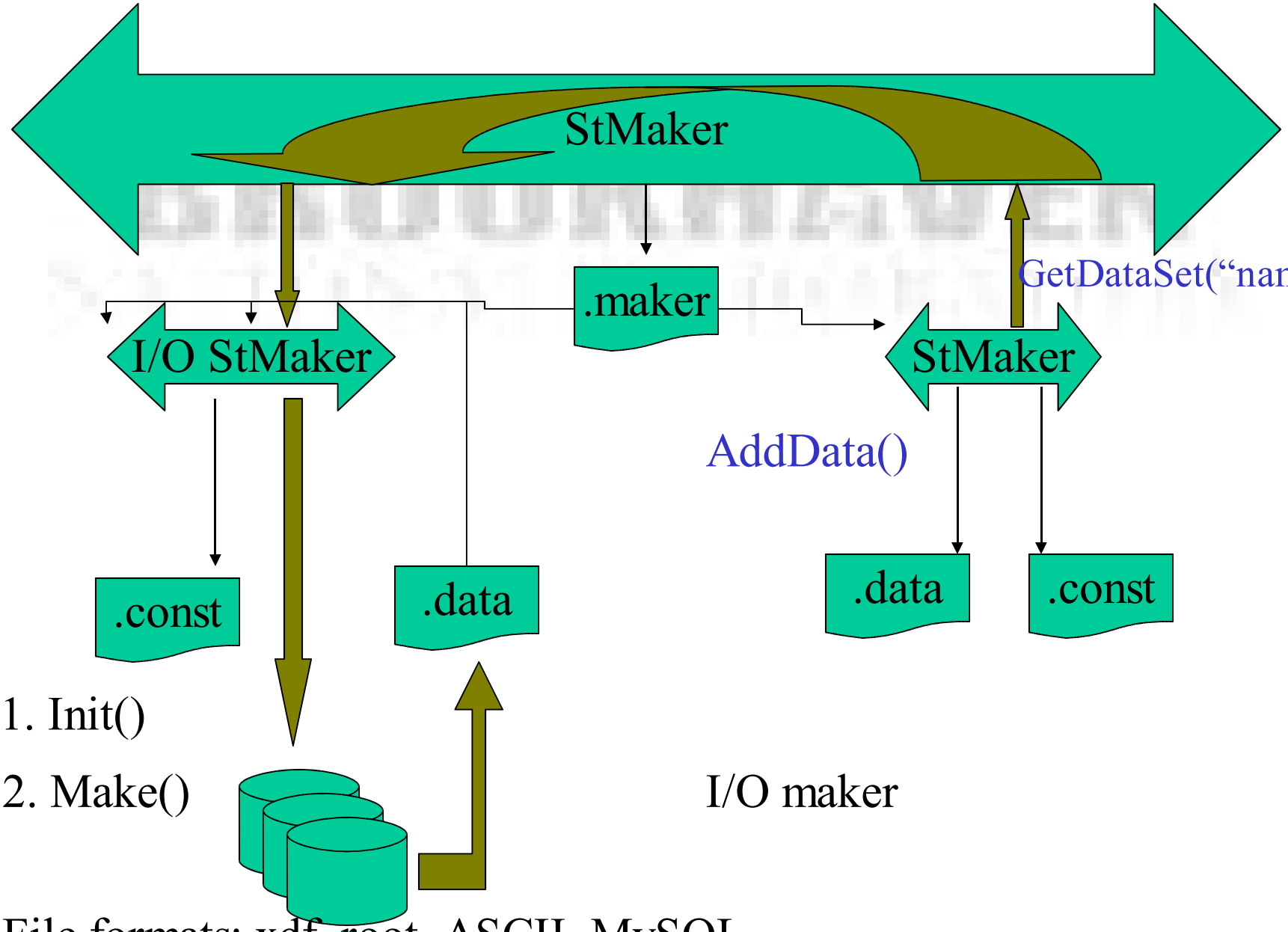

ROOT O...
File View Options Help
bfc
All Folders
bfc
.make
.data
.const
.garb
.aliases
.runco
.hist
7 Obje
ROOT Obj...
File View Options Help
.make
All Folders
bfc
.make
geant
db
tpc
svt
ftpc
emc
l0
rch
match
primary
v0
xi
kink
dst
StEventMaker
analysis
QA
tree
data
19 Obj
ROOT Object Browser
File View Options Help
new
All Folders
Contents of ".../calib/emc/new"
.make
geant
db
calib
.data
calib
emc
bemc
bprs
bsmc
eemc
eprs
esmc
new
old
ebye
.const
.garb
.aliases
.runco
.hist
tpc
.data
bemc
bprs
bsmde
bsmdp
eemc
emsde
emsdp
eprs
8 Objects.
directory

Typical STAR TDataSet/TTable structure

Since TTable's are subclasses of TDataSet it is easy to combine them in the various hierarchical structure.

This way we compensate lacking of pointers within TTable objects





1. Init()

2. Make()

I/O maker

File formats: xdf, root, ASCII, MySQL

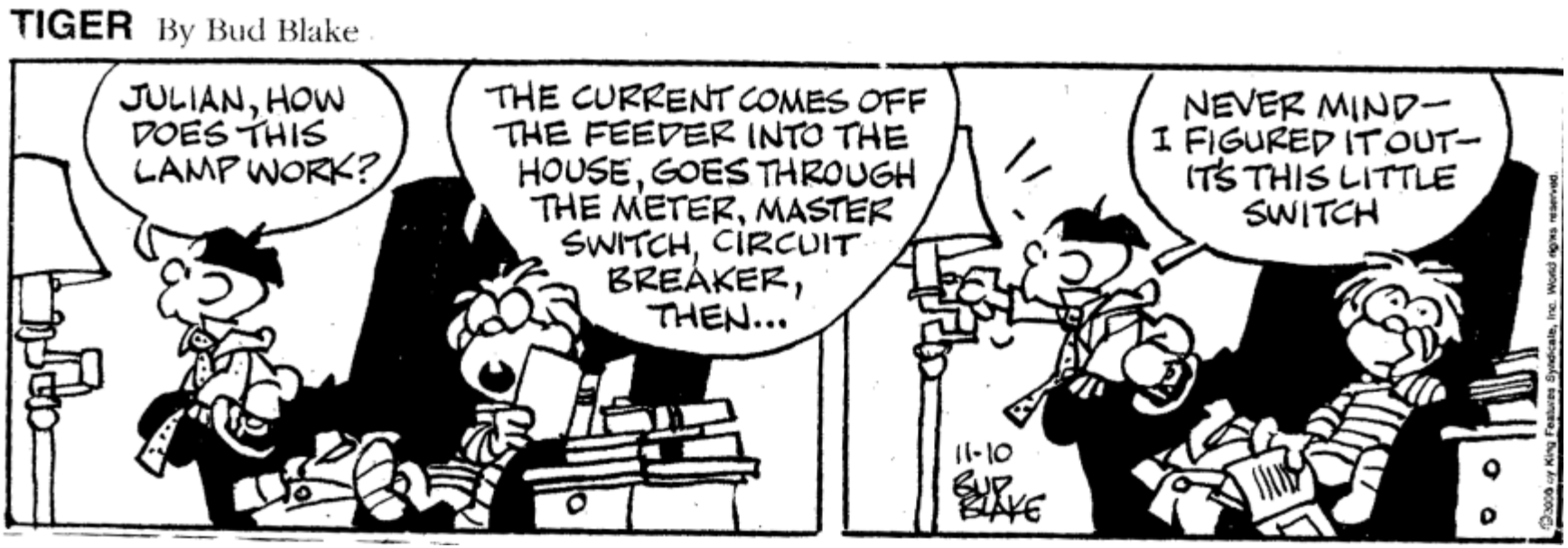
STAR reconstruction OO model:

1. STAR reconstruction code is one single instance of StBFChain class
2. StBFChain is a class derived from StChain class
3. StChain class is a class derived from StMaker class
4. StMaker class is derived from TDataSet
5. TDataSet is a collection of TDataSet or empty.

In other words STAR reconstruction code is a hierarchical collection of STAR Makers.

A Framework

provides utilities and services.



How to create your own maker

- cvs co StRoot/St_TLA_Maker
- **`mv St_TLA_Maker St<MyCustomName>Maker`**
- **`mv St_TLA_Maker.h St<MyCustomName>Maker.h`**
- **`mv St_TLA_Maker.cxx St<MyCustomName>Maker.cxx`**
- **`cons`**
- "Maker"s are called in the order they were being created by the steering object "StBFChain" or by the custom user macro like "doEvents.C"

```
class StMaker : public TDataSet{
public:
enum {kSTAFCV_BAD, kSTAFCV_OK, kSTAFCV_ERR=2, kSTAFCV_FATAL=3}
EModule_return_Status;

                        StMaker(const char *name="",const char *dummy=0);
   virtual              ~StMaker();
   virtual Int_t IsChain() const {return 0;}

   /// User defined functions
   virtual void         Clear(Option_t *option="");
   virtual Int_t        InitRun(int runumber);
   virtual Int_t        Init();
   virtual void         StartMaker();
   virtual Int_t        Make();
   virtual Int_t        Finish();
   virtual Int_t        FinishRun(int oldrunumber);

   // Get methods
   virtual TDataSet  *GetData(const char *name, const char* dir=".data") const;
   virtual TDataSet  *GetDataSet (const char* logInput) const
                     {return FindDataSet(logInput);}
   virtual Int_t        GetEventNumber() const ;
   virtual Int_t        GetRunNumber() const ;
   virtual TDatime      GetDateTime() const;
   virtual Int_t        GetDate()  const ;
   virtual Int_t        GetTime()  const ;
   virtual const Char_t *GetEventType() const ;
```

Real “life”

- There are 4 “offline” layers:
  - Production (bfc)
  - Analysis (muDst)
  - Intermediate (StEvent)
  - Simulation (gstar)