Proposal for new ROOT Fitter

- Proposal for new fitting classes
  - a mini fitting framework (like a simplified RooFit)
  - idea is to have the functionality of the current *TVirtualFitter* with new set of fitting classes
- Require a modular design:
  - easy possibility to extend and add new complex functionality :
    - perform parallel fits (use in multi-threads env.)
    - add new minimization algorithms
    - implement parameter constraints
  - easy maintainability in the long term

TVirtualFitter

- *TVirtualFitter*:
  - current base class for fitting with various implementations which can be instantiated via the plug-in manager:
    - *TFitter (*based on *TMinuit*)
    - *TFumili*
    - *TFitterMinuit (*based on *Minuit2)*
    - *TFitterFumili* (based on Fumili of Minuit2)
    - *TLinearFitter*

Problems with TVirtualFitter

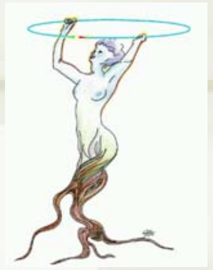
- class designed for *TMinuit*, difficult to adapt for other minimizers
  - i.e. *TVirtualFitter::ExecuteCommand*
- no separation Minimization-Fitting
  - it is more an interface for Minimization
- assume users provides the function to be minimized
  - there is no possibility to pass an object as the function to be minimized (must be a free function)

    ```
    func(Int_t &, Double_t *, Double_t &f, Double_t *par, Int_t iflag)
    ```
- assume exist a TObject representing the fit data (TH1, TGraph, ..) and one representing the fit function (TF1)

Fitting Domain Analysis

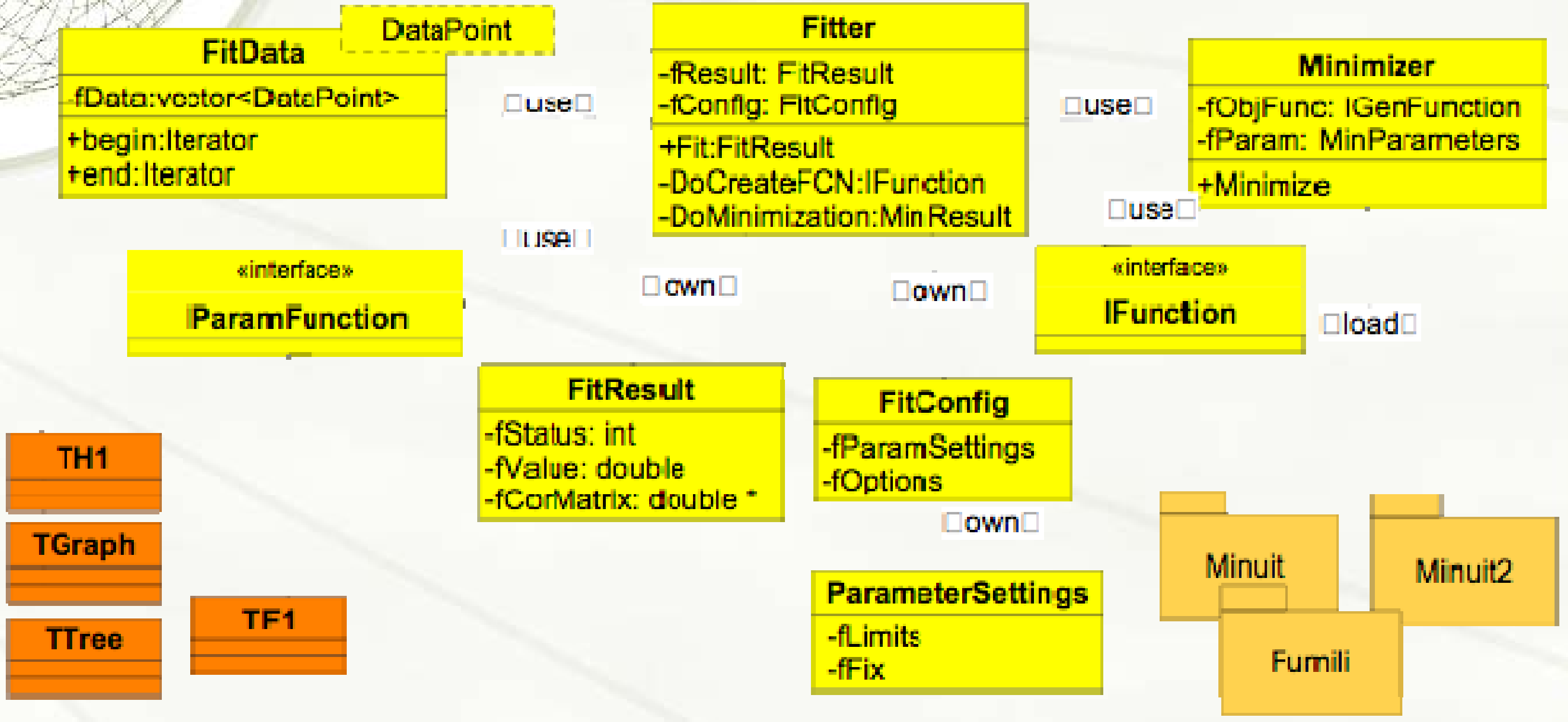
- Major Entities used in fitting:
  - Fit Data
    - binned data (histograms, graphs):
      - coordinates (x[] and value y)
      - poisson errors, gaussian errors, errors in coordinates and values
    - Unbinned data (from TTree)
      - multidim- set of only values (x[])
  - Model Function (Parameteric function)
    - function describing the data: f(x[], parameters)
    - must be normalized (be a pdf) in case of likelihood fits

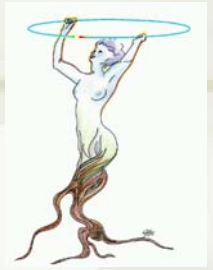
Fitting Entities



- Objective Function (chi square, likelihood, etc..)
  - function of the parameters (parameters are their variables) which must be minimized
  - various standard function:
    - chi square, likelihood (binned/un-binned), extended likelihood, etc…
- Minimizer
  - algorithm to find minimum of the multi-dimensional objective function
  - Exact solution (linear fitter)
  - Numerical solution (MINUIT)
- Constraints
  - conditions on the parameters (e.g. parameter limits)

Fitter Design





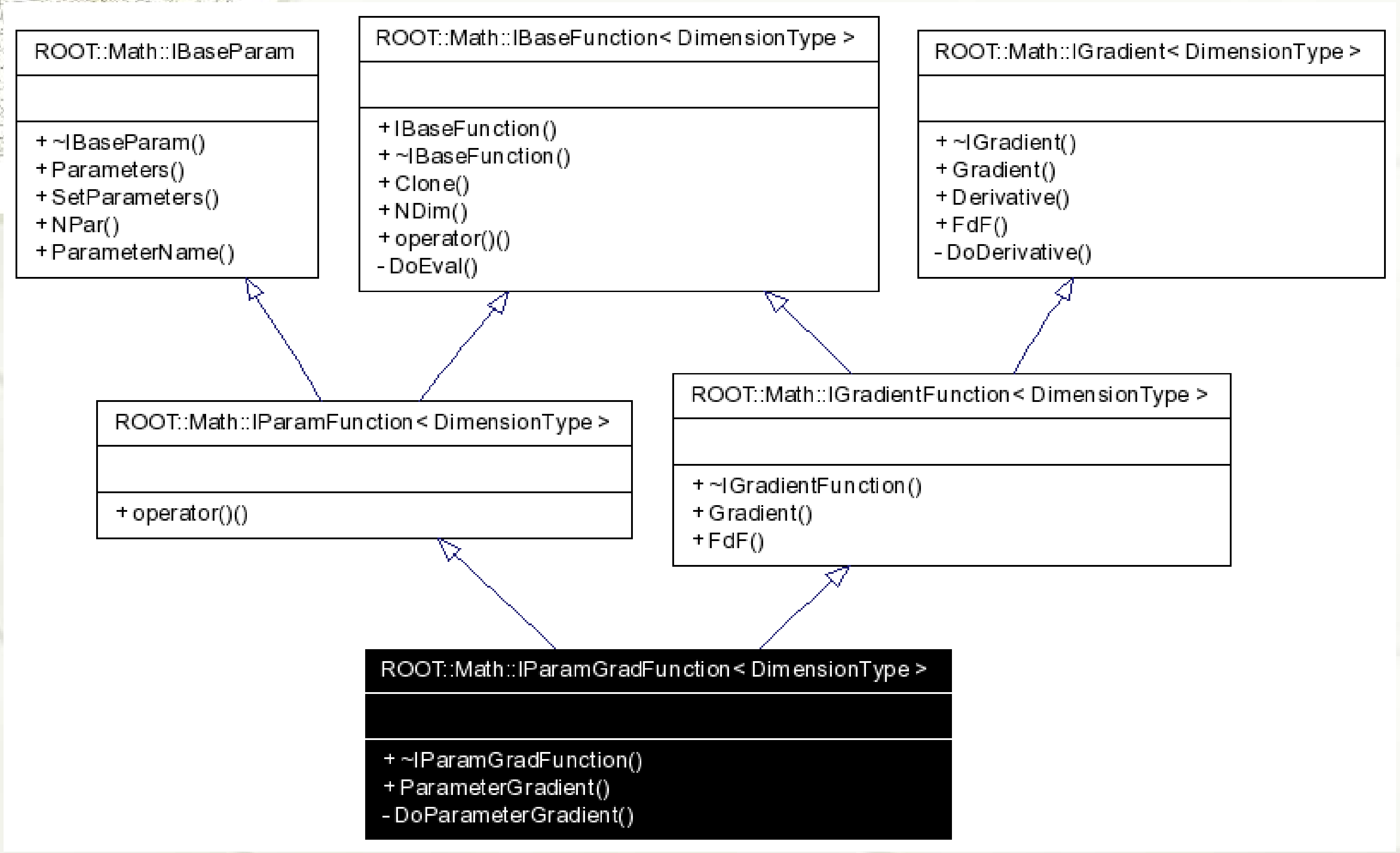
Main Characteristics

- Decoupling of Fitter from the various data sources
  - coupling only at the level of the FitData classes
  - tune the fit data according to the source
    - optimize memory vs CPU performances
- Have an abstract interface for the Minimizer
  - instantiate the Minimizer classes via the plug-in manager
  - user can deal directly with minimizer interface
- Have minimal Function interface
  - describe only the Math functionality
    - evaluation, derivative, possibly integral (for the pdf)
    - state with parameters (for the model functions)
  - decouple Fitter from complex function objects like TF1

Fitter class

- Fitter class glue together data and the model function
  - *Fitter::Fit(IParamFunction & , const FitData &)*
    - create a concrete objective function (like a Chi2)
      - from a *const* reference to the data
      - copying the given parametric function which will be modified during the minimization (allows for paralelization)
    - create the concrete *Minimizer* class according to the chosen implementation type(Minuit, Minuit2,Fumili, etc..) and configuration
    - find the minimum
    - perform optionally error analysis
    - fill and return the *FitResult* class
      - parameter values, errors, error matrix, etc…

Function Interfaces



Function Interfaces

- Minimal interface for classes providing only Math functionality
- Common to other numerical algorithms (in MathMore)
- Distinguish between one and multi-dim functions
  - exist algorithms only for 1D functions
- Provide template *WrappedFunction* classes to wrap in the *IFunction* interfaces:
  - any callable objectfree functions and classes implementing operator()
  - any class member functions with the right signature
  - TF1 objects

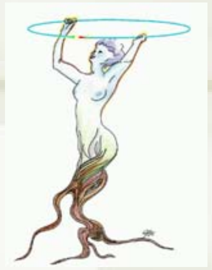
Current Status

- Have a prototype for Least Square fits working with a Minuit and Minuit2 implementations

```
TH1 * h1 = ......
TF1 * func = .....


ROOT::Fit::BinData d;
// fill the data set from the histogram
ROOT::Fit::FillData(d,h1);

// create wrapped parametric function
ROOT::Math::WrappedTF1 f(*func);

ROOT::Fit::Fitter fitter;
// set minimizer type
fitter.Config().SetMinimizer("Minuit2");
// fit
bool ret = fitter.Fit(d, f);
// retrieve optionally fit result
if (ret)  fitter.Result().Print(std::cout);
```

Open questions

- Description of function parameters:
  - prefer to keep separate concept of parameters and the variables (different than RooFit)
  - Have in the function only parameter value (and name)
    - extra parameter properties needed by the Fitter are stored in a different class (ParameterSettings)
  - Have a Parameter class contained in the model Function defining parameter values, limits, etc…
- Fitter is stateless versus Function and Data
  - big advantage (can be independent of the model function type)
- Use simpler layout for function interfaces
  - no virtual inheritance and no *IParamGradFunction*



Open Questions(2)

- Provide set of pre-defined functions (pdf) like in RooFit
  - have a catalog of the most used functions
  - providing analytical implementations for the gradient, integral , etc..
- Provide eventually possibility to compose functions:
  - additions : h(x) = f(x) + g(x)
  - multiplications: h(x) = f(x)g(x)
  - composition:
    - h(x) = g (f(x))
    - h(x,y) = f(x) g(y)
  - convolution

Outlook

- Designed and have first implementation of the major classes
  - Fitter, Minimizer, FitData, etc..
- A first version could be soon available to be committed in CVS (after the production release) with at least the same functionality as the *TVirtualFitter*
  - new package depending only on MathCore
- Could re-implement the *TVirtualFitter* using the new classes (for maintaining backward compatibility)
- Re-implement the FitPanel and methods like *TH1::Fit* using directly the new classes