Object Oriented design of Particle Identification Software for the Instrumented Flux Return subsystem of the BaBar detector

G. De Nardo[1], *L.Lista*[2] *for the BaBar Computing Group*

[1] INFN and Università di Napoli "Federico II "
[2] INFN sezione di Napoli

Abstract

We show how the systematic use of abstract interfaces and polimorphism has allowed a flexible and estensible management of the Particle Identification calibration data samples and their persistency. Moreover the design, originally developed to deal with charged particles, has permitted a natural extention to neutrals without affecting the existing code and with a large reuse of it.

1 Introduction

The Instrumented Flux Return (IFR)[1] subdetector of the BaBar[2] experiment is designed for muons and neutral hadrons identification. The iron yoke of the magnet is segmented in a non-uniform way and its gaps are instrumented with Resistive Plate Chambers (RPC), 19 layers in the barrel and 18 layers in the endcaps. Two additional inner layers of cylindrical RPC are installed inside the coil. Each RPC layer is equipped with two planes of strips, for a digital readout. The strips are grouped in charged and neutral clusters using criteria that are described in another paper[3]. Discriminating variables are extracted from the clusters to perform particle identification.

2 Particle Identification in the IFR

Let **x** be a set of discriminating variables, **p** a set of parameters (momentum and/or polar angle) and h a particle hypotesis. If the probability density functions $f_k^{(h)}(x_k;\mathbf{p})$ in each hypothesis are known, then the likelihood to observe the values **x** of the discriminating variables in the particle hypotesis h is (if the variables are not correlated):

$$L(\mathbf{x};\mathbf{p}\mid h)=\prod_k f_k^{(h)}(x_k;\mathbf{p}) \tag{1}$$

Since these functions are not *a priori* known they have to be estimated from Monte Carlo simulation or high purity unbiased real events samples

Combining likelihoods from more subdectectors is straigthforward:

$$L_{TOT}(h)=\prod_{det} L_{det}(h) \tag{2}$$

At the analysis level, the likelihood from expression (2) has to be further normalized to the number of expected particles for the given hypothesis.

Another frequently used quantity is the significance level defined by:

$$s(\mathbf{x}_{obs}\mid h)=1-\int_{P(\mathbf{x}|h)>P(\mathbf{x}_{obs}|h)} P(\mathbf{x}\mid h)d\mathbf{x}$$

This quantity has the property that its distribution is flat for the correct hypothesis.

3 IFR Particle Identification Software packages

The software consists in the following packages:

- `IfrPidData` contains objects that model the discriminating variables definition and the distributions and parametrization of the discriminating variables and the likelihood.
- `IfrPid` contains classes that manage the summary of the data extracted from calibration event samples and their association with the particle hypotheses. It depends on `IfrPidData`.
- `IfrPidP` implements the persistency of all the objects as described in section in section 6. It depends on both `IfrPid` and `IfrPidData`.

The user retrives the likelihood and the significance level trought an object of type `IfrPidInfo` defined in the `IfrPidData` package. It inherits from `AbsPidInfo`, a common base class of the BaBar software, whose interface let the user access the likelihood and the significance level in a given particle hypothesis. Moreover, it has references to the basic reconstruction objects from which discriminating variables are calculated. Since it takes care to dispatch the likelihood and significance level requests to the objects defined in the `IfrPid` package, the user is unware of the way the likelihood and significance level are estimated. The discriminating variables are calculated in the following way: a base class `IfrAbsDiscrFunct` defines the interface to extract the discriminating variable value from an `IfrPidInfo` object. For each discriminating variable, a concrete class, inheriting the interface from `IfrAbsDiscrFunct`, polimorphically implements the interface.

4 Probability density functions and likelihood parametrized distributions

Probability density functions (pdf) of the discriminating variables and likelihood distributions are parametrized as a function of measured particle variables like the momentum or the traversed IFR sections (barrel, endcaps ...), etc.

For each distribution, at fixed value of the parameters, an `IfrHisto` object samples a pdf as a binned histogram. An `IfrHistoArray` object contains an `IfrHisto` object for each bin of the parameter space. The `strategy` pattern[4] is used to implement a flexible binning of the discriminating variable values and the parameters. The `IfrHisto` class has the responsibility to fill the histogram and to retrive the number of entries, but does not know how its data content is binned. It refers to an external object of class type `IfrPartition`, which is a pure abstract base class whose subclasses polimorfically implement the methods which define the concrete binning. For instance, `IfrHistoFixedBinPartition` implements fixed binning for an `IfrHisto`, leaving room for a different binning solution as variable binning. In order to have an extensible choice for the number of the parameters and their binning, the same design pattern is applied to `IfrHistoArrays`: `IfrHistoArrayPartition` is the base class whose interface is currently implemented by its two subclasses: one defines fixed binning in one parameter (IFR angular section), another one defines fixed binning in two parameters (momentum and IFR angular section). Different choices in the number of parameters and their binning can be added very easily.

In order to provide the significance level the `observer` pattern[4] is used. An observer of the `IfrHisto` object calculates its cumulative distribution using a cache that is invalidated when the histogram is updated.

5 Calibration samples management

The data extracted from an events sample is summarized by an object. The design is divided in three levels of abstraction:

1. An `IfrSampleSummary` class is a pure abstract interface; client objects use this type, decoupling themselves from the concrete implementations. The interface consists of a set of selectors to obtain likelihood and significance levels and modifiers to accumulate the data.
2. The second level of abstraction is provided by the `IfrLikelihoodSampleSummary` class which inherits the interface from the `IfrSampleSummary` class. It implements the selector and modifier methods of the parent `IfrSampleSummary` and owns the following objects:
   - the discriminating variables, their distributions and their partitions;
   - an `IfrHistoArray` and an `IfrPartition` which contain the likelihood distribution;
   - an `IfrHistoArrayPartition` defining the parameters and their binning.

   Note that every reference is a base class and the responsibility to create the concrete objects is delegated to the lower level of abstraction.
3. The objects of concrete type `IfrChargedSampleSummary` and `IfrNeutralSampleSummary` handle the differences between the charged and the neutral identification. Their constructors create the concrete discrimination functions, histograms and partitions, and implement specific methods.

The association between a sample summary and one or more particle hypotheses are managed by a discriminator object of type `IfrLikelihoodDiscr`. It mantains an association map between an object of type `PdtEntry` (from the Particle Data Table package[5]), which uniquely defines a particle hypothesis, and a object of type `IfrSampleSummary`.

In order to choose at run-time the association between one or more particle hypotheses to a data sample summary, the class provides a method to associate an `IfrSampleSummary` object to a `PdtEntry` object. For instance, at the present moment, a muon sample is associated to the muon hypothesis and two samples of positive and negative charged pions are associated to the positive and negative charged hadron hypotheses. The associations are set by the user in a tcl script like in the following example:

```
module talk IfrChargedCalib
  associate muonSample mu+ mu-
  associate pion+Sample pi+ K+ p
  associate pion-Sample pi- K- anti-p
exit
```

As more specific samples are accumulated, more refined associations can be set. A class `IfrSampleManager` has two global istances (one for charged and the other for neutral) which create the appropriate discriminator and provide access to it. The existence of exactly two istances is enforced using a modified `singleton` pattern[4] (“multiton”).

6 Persistency of the objects

Object persistency is currently implemented using Rogue Wave `Tools.h++`[6] which provides a persitency mechanism on flat files. The migration to Objectivity Database[7] is under development. For each inheritance class tree of the previously described model a `visitor` pattern[4] is applied to decouple the transient model from the persistency technogy. Given an inheritance tree

(`ConcreteXYZ1`, `ConcreteXYZ2` subclasses of `AbsXYZ`), an abstract `XYZPersister` class provides the interface to "persist" all the concrete types. The concrete persister subclass that implements –say– Rogue Wave technology creates the persistent object corresponding to a given transient one in the inheritance tree (`ConcreteXYZ1RW`, `ConcreteXYZ2RW` subclasses of `AbsXYZRW`).

Adding a new technology, like Objectivity DB, requires no change in the existing code ("Open/Close principle").

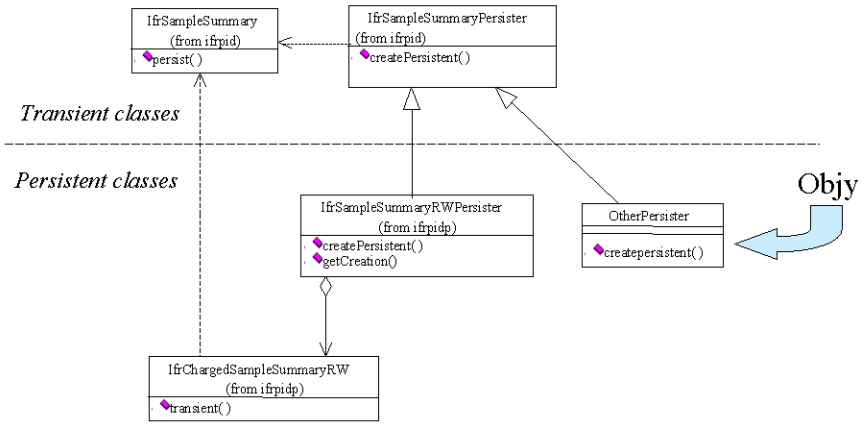

Figure 1: Persistency class diagram

7 Conclutions

The design described reaches the following goals:

- Likelihoods and significance levels for any particle hypotheses are provided. Discrimination variables can be added or removed in a trasparent way.
- Calibrations samples are managed in such a way that the association of the samples to one or more particle hypotheses is possible at run time in a flexible way.
- Commonalities between the charged and neutral particle identification are exploited implementing the differences only at the most concrete level.
- The object persistency is managed to be estensible to other solutions without affecting the design of the transient objects.

References

1 The Muon and K_L Detector for the BABAR Experiment: Physics Requirements, Final Design and Start of Construction, Nucl. Physics B(Proc. Suppl.) 61B(1998) 244-249

2 BABAR Technical Design Report, BABAR Collaboration, SLAC Report SLAC-R-95-4578, March 1995

3 L.Lista, proceeding of CHEP 2000: "Object Oriented Reconstruction for the Instrumented Flux Return of BABAR"

4 Design Patterns, E.Gamma,R.Helm, R.Johnson, J.Vlissides, Addison Wesley 63361, 1995

5 L.Lista, the PDT (Particle Data Table) Package, BaBar Note 414, 15 March 1998

6 Tools.h++, Rogue Wave Software Inc., http://www.RogueWave.com/products/tools

7 Objectivity Database, Objectivity Inc., http://www.objectivity.com