Redesign of the FairRootManager

R. Karabowicz[1], *M. Al-Turany*[1], *D. Kresan*[1], *and F. Uhlig*[1]
[1]GSI, Darmstadt, Germany

Introduction

FairRoot is a data simulation and reconstruction framework developed in the Experiment Software group and used by various experiments for the physics analysis. It has originally been designed both to read the input data and to store the output data from/in the root files. The recent requirements of several experiments made it necessary to develop alternative ways of accessing input data, like the MBS systems or LMD files.

In the meantime, it became imperative to unify and standarize the different approaches.

Source

In the original design, the central data manager, the *FairRootManager*, reads the data from the input ROOT trees stored in (multiple) files. The redesigned *FairRootManager* accesses the data inputs via specific implementations of the *FairSource*. The generic *FairSource* class provides abstract functions to access input data. The concrete derived classes include:

- *FairMbsSource* - for reading data from the MBS event server;
- *FairLmdSource* - for reading data from the LMD files;
- *FairFileSource* - for reading data from the root files, it is possible to add many files (creaing chains of input data) or to add friend files (in case if the event data is spread between different files);
- *FairMixedSource* - for mixing of input data from the background file with signal files. Each added signal requires providing of a value of mixing ratio, which determines the occurence of signal events in respect to the background events.

The reorganization allows now easy creation of another, more experiment-specific, implementations of the *FairSource*, should such be needed.

Manager

With the introduced changes, the *FairRootManager* complexity has been greatly reduced and the source file size scaled down by about 55%. The functions that were responsible for reading the input tree structure or comparing integrity of the trees if more input files were provided, were moved to the *FairFileSource* and *FairMixedSource*.

```
FairRunAna* fra = new FairRunAna();
fra->SetBackgroundFile(‘‘file_bg.root’’);
fra->SetSignalFile(‘‘file_sg.root’’);
fra->AddTask(TTask* task);
fra->Init();
fra->Run(ev_start,ev_end);
```

Figure 1: Standard way of adding input files.

```
FairRunAna* fra = new FairRunAna();
FairSource* source = FairMixedSource(‘‘file_bg.root’’);
source->SetSignalFile(‘‘file_sg.root’’);
fra->SetSource(source);
fra->AddTask(TTask* task);
fra->Init();
fra->Run(ev_start,ev_end);
```

Figure 2: Adding FairSource after introduced modifications.

User macros

With the introduced changes the analysis macros, that steer the reconstruction, will slowly undergo a reorganization process. Previously, when only root files were recognized as the input source, the user was setting file names in the *FairRunAna*, as presented in Figure 1. Currently, one has to create an instance of a wanted source and set its properties, compare Figure 2. Such a source is then given to the *FairRunAna*, which passes it to the instance of the *FairRootManager*.

It has to be mentioned that to preserve the compatibility, the previous scenario is still supported, although with a warning message given to the user about code obsolence.

Summary

The goal of the introduced changes is to unify the reconstruction running scenario with different sources of input and to increase the framework modularity.

It has been achieved by introducing an abstract layer (*FairSource*) between the input source and the data manager.

The *FairRunAna* works with different kinds of sources. The experimantal software developers may now easily implement experiment-specific sources.

DOI:10.15120/GR-2015-1-INFRASTRUCTURE-IT-05