The FINUDA pre-analysis monitor

Diego Faso (faso@to.infn.it)

Last update: Friday, October 7, 2005

Main requirements

- **Trigger, DAQ and DAΦNE status:**
  - Scalers
  - Trigger efficiency (which parameters could we use?)
  - ??? (I need some suggestions…)

- **BEAM status … evaluation of:**
  - Instantaneous Luminosity (online calculation)
  - Φ decay position
  - Center of mass energy

- **Event quality:**
  - Number of reconstructed K^-/K^+ (also used for the luminosity calculation)
  - Percentage of stopped K^-/K^+
  - Number of reconstructed hits per layer (related to physical background/noise)
  - Number of recognized/reconstructed tracks
  - Angular distribution of reconstructed tracks (ϕ angle) (positive/negative tracks)

- **Reconstruction efficiency:**
  - Momentum resolution (μ^+)
  - ParticleIDentification from ISIM/LMD/TOF
  - More???

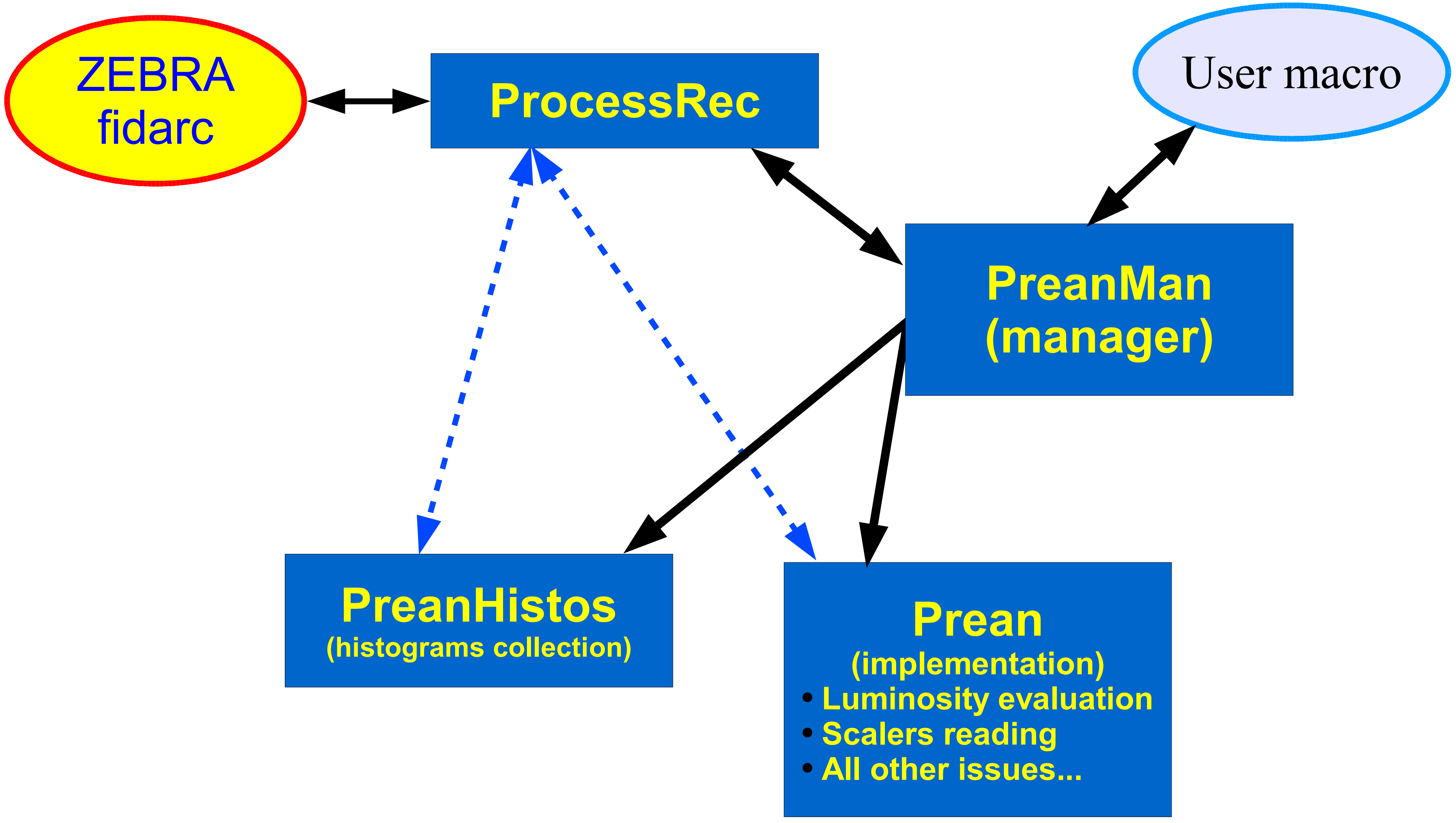
It should run like this:

- **Start New Run (or new online-monitoring)**
  - Open raw file (offline mode)
  - Open UDP socket (online mode)
  - Reset All scalers and currents (optional)
  - Get the run number (from DAQ?)
- **Event Loop**
  - Every event:
    - Read scalers
    - Fill histograms (if needed)
  - Every 3 events (for example):
    - Perform the complete fidarc reconstruction process
    - Fill histograms (if needed)
- **End Of Run**
  - End of run operations to be defined...

Write text files, which will be shared with DAFNE control room and with the FINUDA web server.

The luminosity evaluation refresh depends on time, not on the number of events.

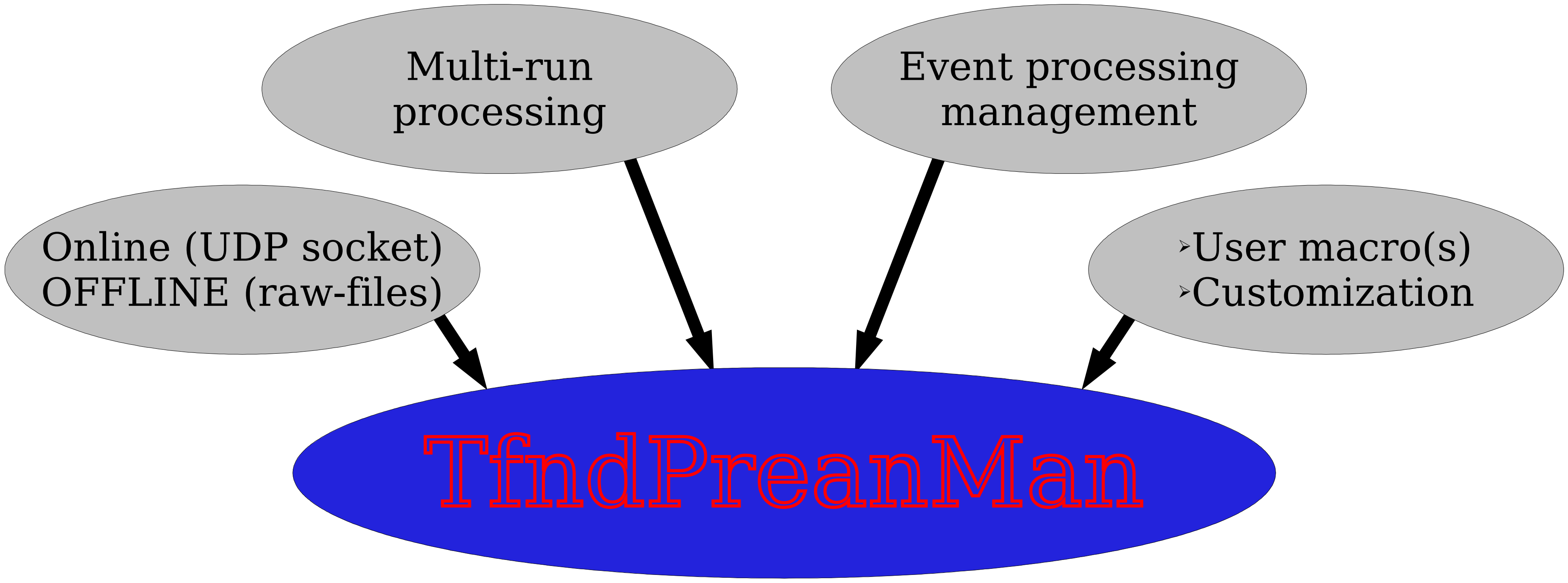
Structure (block diagram)



“Prean Manager”: features

The “TfndPreanMan” has been developed within the froot environment, thus exploiting all advantages provided by the steer class “TFndRun”.

The event processing can now be easily handled by writing a simple **(f)root** macro in which a pointer to TFndPreanMan is used.



Preanalysis steps

The **event-processing** has been **splitted** into several steps, in order to allow a complete **customization** for the running process(es)

1) **Get** the **event** (optionally fill froot HDT structure).
2) **Perform pre-analysis (step 1)**
3) **Perform** a **custom reconstruction** (optional)
4) **Build** standard **ZEBRA-structure** (read-lib is used)
5) **Perform pre-analysis (step 2)**
6) **Perform** a **custom reconstruction** starting from geometrical hits reconstructed by fidarc (optional)
7) **Perform** the FINUDA **standard reconstruction** (fidarc official FORTRAN code is used)
8) **Perform pre-analysis (step 3)**

The event-processing can be stopped at any step: the preanalysis can be customized according to:
cpu-performances, required statistics, DAQ rate, ...

Preanalysis steps

- The access to raw-event is handled by:
  - the TFndRun class:
    a pointer to the raw-event is stored as data member of TfndRun
  - the read-lib (fin_open):
    the fidarc program is controlled via the froot<->fidarc interface (TFndProcessRec class).

- User can chose if the froot HDT structure shall be filled or not
  by passing a boolean parameter to the TfndPreanMan::GetNextEvent() method

- A pointer to the raw-event is stored as data-mamber of TFndRun (parent of TFndPreanMan)

1)Get the event
2)Pre-analysis (step 1)
3)Custom reconstruction
4)Build ZEBRA (FGES)
5)Pre-analysis (step 2)
6)Custom reconstruction
7)Standard reconstruction
8)Pre-analysis (step 3)

C++ steers the whole process

raw-file is opened twice

Preanalysis steps

Some important informations are extracted directly from the pointer to the raw-event:

- Run completed

PreanMan

- DAQ correctly running
- Run number
- Event number
- Trigger bit
- Scalers

Prean

fidarc not involved yet

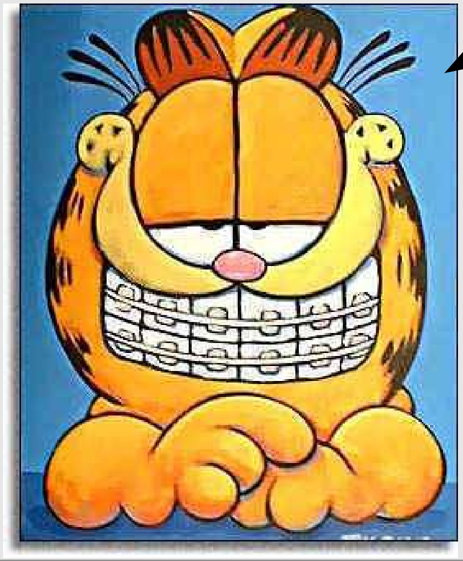
1)Get the event
2)Pre-analysis (step 1)
3)Custom reconstruction
4)Build ZEBRA (FGES)
5)Pre-analysis (step 2)
6)Custom reconstruction
7)Standard reconstruction
8)Pre-analysis (step 3)

Preanalysis steps

(This step is optional)

The raw-event can be directly used for custom checks or analysis:

The pre-analysis manager can return the pointer to the raw-event..... ...this could make happy some users!



1) Get the event
2) Pre-analysis (step 1)
3) Custom reconstruction
4) Build ZEBRA (FGES)
5) Pre-analysis (step 2)
6) Custom reconstruction
7) Standard reconstruction
8) Pre-analysis (step 3)

The pre-analysis manager works as an interface for getting raw-events without the need to write new code!

Preanalysis steps

The read-lib is used to fill ZEBRA FGES structure:

- Geometrical hits are reconstructed starting from physics signals
- ZEBRA-FGES structure is filled with:
- Geometrical reconstructed hits
- Detectors ADCs & TDCs

read-lib

C++ is steering the step, but no C++ code is required to implement these functions

1)Get the event
2)Pre-analysis (step 1)
3)Custom reconstruction
4)Build ZEBRA (FGES)
5)Pre-analysis (step 2)
6)Custom reconstruction
7)Standard reconstruction
8)Pre-analysis (step 3)

updates/changes in the fidarc code do not require any changement in the C++ code!

Preanalysis steps

Some important informations are extracted directly from the content of the ZEBRA FGES structure

- Number of hits/layer (related to noise)
- Multiplicity
- Further ideas?

Prean

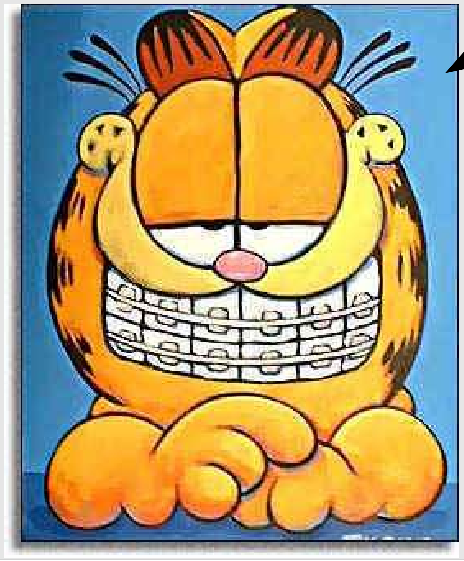
1) Get the event
2) Pre-analysis (step 1)
3) Custom reconstruction
4) Build ZEBRA (FGES)
5) Pre-analysis (step 2)
6) Custom reconstruction
7) Standard reconstruction
8) Pre-analysis (step 3)

Preanalysis steps

(This step is optional)

ZEBRA-FGES structure can be accessed for custom checks or analysis:

...this could make happy some users!



1) Get the event
2) Pre-analysis (step 1)
3) Custom reconstruction
4) Build ZEBRA (FGES)
5) Pre-analysis (step 2)
6) Custom reconstruction
7) Standard reconstruction
8) Pre-analysis (step 3)

Any <u>C++/based reconstruction</u> can be written inside a standard <u>root macro</u> without modifying the compiled code

User macros dependend on the zebra-FGES structure

Preanalysis steps

The fidarc (FORTRAN) complete event reconstruction is performed.

- The content of ZEBRA-FGES is used to reconstruct the event
- The ZEBRA-DST structure is filled

(all hidden into **one** user command)

fidarc

C++ is steering the step, but no C++ code is required to implement these functions

- NOTE:
  The content of ZEBRA-FGES can not be used anymore

1) Get the event
2) Pre-analysis (step 1)
3) Custom reconstruction
4) Build ZEBRA (FGES)
5) Pre-analysis (step 2)
6) Custom reconstruction
7) Standard reconstruction
8) Pre-analysis (step 3)

The C++ code steering this step is completely independent on any update/change in fidarc

Preanalysis steps

Some important informations are extracted directly from the content of the ZEBRA FDST structure

- Trigger flag: **Bhabha**
  - Fit succesfull?
    (increment counter for luminosity evaluation)
  - Φangle of reconstructed tracks
  - Position of the interaction point
  - Center of Mass energy
  - […]
- Trigger flag: **HYPE**
  - Fit succesfull?
    (increment counters for luminosity evaluation)
  - Number of reconstructed tracks / event
  - Momentum (μ^+ ; π^-)
  - […]

1) Get the event
2) Pre-analysis (step 1)
3) Custom reconstruction
4) Build ZEBRA (FGES)
5) Pre-analysis (step 2)
6) Custom reconstruction
7) Standard reconstruction
8) Pre-analysis (step 3)

- No need to store luminosity parameters into a huge database table
- Possibility of evaluating several parameters in a single step!

C++ code dependends on the zebra-FDST structure

Preanalysis customization

- The event-processing can be stopped at any step.
- The user can select the “stop-step” event by event.
- Users' compiled code can be used (ACLiC).
- Thinking about a configuration file
  (or better... a configuration class) for:
  - Shared files refresh rate
  - Default event processing customization
  - more?...

Only to use HDT froot structure	1)Get the event
Very fast / no geometrical info / no reconstruction	2)Pre-analysis (step 1)
Up to the user...no official code implemented	3)Custom reconstruction
Necessary to continue	4)Build ZEBRA (FGES)
Fast / no reconstruction info	5)Pre-analysis (step 2)
Up to the user...no official code implemented	6)Custom reconstruction
Necessary to continue (time consuming)	7)Standard reconstruction
Complete process performed	8)Pre-analysis (step 3)

Preanalysis customization

Event-processing example:

Low statistics on reconstructed informations / Fast scalers-reading / Fast processing

1) **Get the event**
2) **Pre-analysis (step 1)**

Every event

3) Custom reconstruction
4) **Build ZEBRA (FGES)**
5) **Pre-analysis (step 2)**

Every 3 event

6) Custom reconstruction
7) **Standard reconstruction**
8) **Pre-analysis (step 3)**

Every 9 event

Status of the art (Oct 10, 2005)

- **ProcessRec** (from the TProcess of Filippini/Panzarasa):
  - Completed and tested (multi-run processing available)

- **PreanMan:**
  - Implemented and tested (also online)

- **PreanHistos:**
  - Structure of the class implemented
  - First test histogram added
  - Histograms collection...to be defined (are histograms needed)

- **Prean:**
  - Class structure designed
  - Implementation:
    - Read scalers: OK
    - Luminosity evaluation (from Bhabha): In progress
    - Luminosity evaluation (from K^+K^-/K_LK_S): to be implemented

- **Performances and benchmarks ... still missing**

Open questions

- **Shall we provide a GUI for the pre-analysis monitor?**
  (histograms, GUI windows, frames and buttons...)

- **How many** (and which) **informations/parameters** shall be available on the **Online-Status web** page?

- **Should be the pre-analysis customization available to standard users?**
  this would require a dedicated GUI or a tutorial for
  the set-up of personal customizations via configuration file(s)

- What do you think about the web page and the documentation of:
  - **froot**
  - **pre-analysis**
  - **slow-control**