Implementation of on-line analysis library in NARVAL: the PRISMA case

E. Calore\textsuperscript{1}, E. Farnea\textsuperscript{2}, D. Mengoni\textsuperscript{3}

\textsuperscript{1} INFN Laboratori Nazionali di Legnaro, \textsuperscript{2} INFN Sezione di Padova, \textsuperscript{3} Università e Sezione INFN di Padova

AGATA Week 2008
Outline

1. Basic ingredients
   - The NARVAL data acquisition system
   - The libPRISMA library

2. NARVAL’s basic concepts
   - Overview
   - The actors

3. PRISMA data acquisition
   - The first Prototype
   - Integration in the AGATA DAQ
Outline

1. Basic ingredients
   - The NARVAL data acquisition system
   - The libPRISMA library

2. NARVAL’s basic concepts
   - Overview
   - The actors

3. PRISMA data acquisition
   - The first Prototype
   - Integration in the AGATA DAQ
NARVAL

Nouvelle Acquisition temps-Réel Version 1.2 Avec Linux

- It is a distributed data acquisition system, written in Ada95
- It is currently being developed in Orsay
- It will be used to manage the AGATA DAQ
- It is needed to distribute the calculations of the on-line analysis among different computers
PRISMA is a large acceptance magnetic spectrometer for heavy ions.

It will be coupled with the AGATA demonstrator in Legnaro.
libPRISMA

also known as the **PRISMA preprocessing library** is a **C++** library providing tools to analyze PRISMA data.

It is being developed in Legnaro by Enrico Farnea

It extract recoil information such as:

- the (vector) velocity
- the atomic number
- the mass number
Outline

1. Basic ingredients
   - The NARVAL data acquisition system
   - The libPRISMA library

2. NARVAL’s basic concepts
   - Overview
   - The actors

3. PRISMA data acquisition
   - The first Prototype
   - Integration in the AGATA DAQ
Why using NARVAL?

NARVAL is needed to split the calculations in various “blocks”
An actor can be provided with the Ada code to elaborate data or it can be a generic one:

A generic actor can “link to” a C++ library
Three main kinds of actors

- **Producer**: Only output buffer/s
- **Filter**: Both Input and Output buffer/s
- **Consumer**: Only Input buffer/s
Outline

1. Basic ingredients
   - The NARVAL data acquisition system
   - The libPRISMA library

2. NARVAL’s basic concepts
   - Overview
   - The actors

3. PRISMA data acquisition
   - The first Prototype
   - Integration in the AGATA DAQ
The running prototype

- **Producer**
- **File reader**
- **Filter**
- **PRISMA Filter Actor Library**
- **libPRISMA**
- **Consumer**
- **File writer**

**Reading PRISMA events from file**

**Preprocessing of read events**

**Writing data to file**
Basic ingredients

NARVAL's basic concepts

PRISMA data acquisition

**NARVAL System**

- **Producer**
- **File reader**

**Filter**

- **PRISMA Filter Actor Library**
- **libPRISMA**
- **GRU Server**

- **Consumer**
- **File writer**

**TCP/IP**

**Vigru histograms visualizer**

<table>
<thead>
<tr>
<th>Mass</th>
<th>Entries</th>
<th>Mean</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>55404</td>
<td>48.12</td>
<td>0.745</td>
</tr>
</tbody>
</table>
<configuration>
  <producer>
    <name>producer</name>
    <hostname>narval01</hostname>
    <binary_code>generic_producer</binary_code>
    <output_buffer_name>datal1</output_buffer_name>
    <size output_buffer="datal">1000000</size>
    <port output_buffer="datal">eth1</port>
    <debug>info</debug>
  </producer>
  <intermediary input_buffers="1" output_buffers="1">
    <name>filter</name>
    <hostname>narval02</hostname>
    <binary_code>generic_filter</binary_code>
    <data_source source_port="eth1" source_buffer="datal">producer</data_source>
    <output_buffer_name>data2</output_buffer_name>
    <size output_buffer="data2">1000000</size>
    <port output_buffer="data2">eth1</port>
    <debug>info</debug>
  </intermediary>
  <consumer>
    <name>consumer</name>
    <hostname>narval03</hostname>
    <binary_code>generic_consumer</binary_code>
    <data_source source_port="eth1" source_buffer="data2">filter</data_source>
    <debug>info</debug>
  </consumer>
</configuration>
#ifndef _PRISMA_CLASS_H_
define _PRISMA_CLASS_H_
#include "base_class.h"

class prisma_class : public intermediary
{
  public:
    prisma_class ();
    void process_config (char *directory_path, unsigned int *error_code);
    void process_block (void *input_buffer,
                        unsigned int size_of_input_buffer,
                        void *output_buffer,
                        unsigned int size_of_output_buffer,
                        unsigned int *used_size_of_output_buffer,
                        unsigned int *error_code);
    bool processEvent( unsigned short int* event, int size );
};
The future integration with AGATA DAQ
Thank you for your attention