A Framework for PreSPEC-AGATA Data Analysis*

M. Reese[1], *N. Pietralla*[1] *for the PreSPEC-AGATA-collaboration*
[1]Technische Universität Darmstadt, Germany

Introduction

The PreSPEC-AGATA experiment [1] at GSI investigates the structure of radioactive nuclei by means of in-flight γ-spectroscopy. This report describes a new software framework, which is a supportive tool to create data analysis programs for complex experiments like this. In order to reduce the workload of the user, the framework facilitates repetitive tasks, such as visualizing data in histograms, defining conditions and gates, and passing data to different stages of the analysis process. The user has only to write the actual data processing algorithms in C++, where a strict interface for data input and output ensures the re-usability of the analysis components. Examples of such algorithms are e.g. analysis of a specific detector or Doppler-correction of detected γ-rays.

The Framework

Every data analysis program is passing information through different stages of a data reduction process, until the final, high-level data is available. The layout of the information flow can be described as *directed graph*. The internal design of the described framework follows this graph structure as closely as possible. A user of the framework has to provide the description of the graph by means of two concepts:

1. Node types: Graph nodes process data. The definition of the node's algorithm, such as unpacking raw data or performing higher-level analysis tasks, is done in terms of C++ classes. These classes implement abstract interfaces: Processor and Unpacker. Every Processor implementation has to specify what output it provides and what input it requires. This is done in a way that strongly decouples the actual processing of data from passing it between different processing nodes. There are special nodes in the graph that are composed of Unpackers. These nodes are called Crates and get raw data, unpack it, and inject it into the graph.

2. The graph layout: A set of nodes and their connections define the graph. This information is provided by the user in a configuration script that is written in terms of a simple and specialized graph description language. It allows a compact description with loops and array assignments. Every node has a name and a type, where the type refers to an implementation of a Processor class. The node is internally represented as an instance of that Processor class. It has parameter and calibration files associated with its name, i.e. there is a unique place for every parameter.

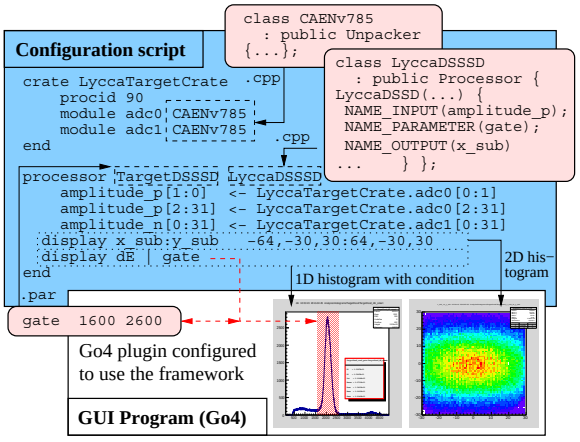

Figure 1: Complete analysis of the hitpattern of a DSSSD: A configuration script (big box) defines the graph structure. Dashed boxes indicate the use of a parameter file or Processor class (rounded boxes). Statements in dotted boxes create graphical representations in the GUI program (white box). The graphical cut (red area in the 1D-histogram) is defined by the parameter `gate`.

A simple example of the interplay between the components is shown in Fig. 1. The configuration script defines the graph (with 2 nodes) and provides, together with the parameter and calibration files, a complete description of the analysis for a given raw-event. The user can specify data or graphical cuts to be visualized during the analysis. All changes inside the configuration script have immediate effect after restarting the analysis, reducing development and debug time significantly.

The analysis framework is a C++ library, using STL and Boost. It does not provide any data-replay or visualization capability, but needs to be used by a host program that accesses the raw data and provides a graphical user interface. Inside the host program, the raw event data is passed to the analysis framework. Afterwards, the host program can read high level information from the framework and, for example, write it into a ROOT tree. Go4 (`http://go4.gsi.de`) is an example of such a host program, but any program capable of data replay and visualization can be adapted to use the framework. It is not restricted to the PreSPEC-AGATA setup but could as well be used for future experiments, like HISPEC/DESPEC at the FAIR facility.

References

[1] P. Boutachkov *et al.*, this report.

*Work supported by BMBF NuSTAR.DA - TP 6, FKZ: BMBF 05P12RDFN8 (TP 6)