PACMAN - THE MEDAUSTRON MEASUREMENT DATA ANALYSIS FRAMEWORK

Alexander Wastl[*], Tobias Kulenkampff, Sebastian Nowak, Adriano Garonna
EBG MedAustron, Wiener Neustadt, Austria

Abstract

During the commissioning of the synchrotron-based MedAustron accelerator facility, the analysis and interpretation of data of various sources was required. A dedicated framework was developed to analyze the raw data provided by the accelerator control system (ACS). A tested and documented software core with a simple and standardized interface allows also non-programming professionals to easily base their applications on this framework which is essential to efficiently make progress in the dynamic environment of commissioning. This document presents the structure of the framework, the interface between the software core and higher level applications and gives an example using all framework levels.

INTRODUCTION

Although the commissioning activities for proton treatments in a horizontal irradiation room [1] of the MedAustron accelerator have been completed, commissioning will continue in the next years to provide beams of protons and carbon ions to all irradiation rooms (in total three horizontal beam line, one vertical and a gantry) for clinical treatment and non-clinical research.

For the commissioning of the beam and characterization of the accelerator, digital signals of various origins have to be acquired and analyzed. In the injector, beam measurement data is available from wire scanners, profile grid monitors, Faraday cups, current transformers and scintillating plates. The synchrotron hosts shoebox pick-ups as well as a fast and a slow current transformer. Scintillating fiber monitors provide beam profile data in the high energy beam transfer line (HEBT) whereas a scintillating plate can be used to analyze the extracted particle flux.

Furthermore, signals of the synchrotron RF-cavity and of power supplies have been used during the commissioning phase. In the future, the acquisition of data from the dose delivery monitors (in-room position and intensity) and a Schottky monitor is foreseen.

Control system tools [2] have been developed and set up to perform measurements of these signals. These applications are also capable of moving devices, changing of magnet settings or varying timing events during the cycle. Therefore, single measurements and measurement series can be performed easily.

Due to the need of more advanced analysis, a software providing the computation of the center of gravity and FWHM (full width at half maximum), fast-fourier transformation with different filters or the option to correlation of signals of different measurement devices was required. Furthermore, the typical cycle length of less than 10 seconds allows for statistics of reasonable cycle numbers which has to be included in the analysis as well.

For this purpose a dedicated analysis framework has been developed at MedAustron: *Python Algorithms Coded for Measurement data ANalysis* - PACMAN. It represents the interface between the raw data and the user processing the data providing data-source-dependent analysis options.

[*] alexander.wastl@medaustron.at

FRAMEWORK

Python [3] was chosen as programming language for a variety of reasons: immanently object oriented, fast learning, widely used in the scientific community and therefore vast number of scientific libraries already available and existing code could be included.

All modules are structured in four levels and a common support modules library within the PACMAN framework (see Figure 1). The support modules provide commonly used functionality like a logger, specific errors and flags and helper functions for file filtering or decryption of control system specific codes.

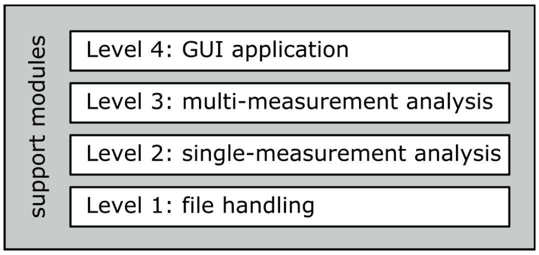
support modules
Level 4: GUI application
Level 3: multi-measurement analysis
Level 2: single-measurement analysis
Level 1: file handling

Figure 1: PACMAN framework concept of levels and supporting modules.

Although the PACMAN core consisting of the support library, level one and two as well as the GUI (level four) is developed and maintained by a few experts only, all interfaces are universal and simple and coding constraints are low in order to enable all members of the beam commissioning team to base their codes on this PACMAN core. Lowering the inhibition level of the user by providing an easy-to-access way to the data (see 'Level two') was a clear goal of the design, even though it meant having to go without advanced programming language features.

LEVEL ONE

The first level handles the reading of different input file formats. For measurement data, a dedicated MedAustron Exchange File Format (MEFF) was defined and is used by all measurement devices (MD). It determines the file structure including the header, meta data and the measurement data to be written in a comma-separated, human readable style. This simple and clear format also allows other tools to easily import the data.

Additionally a Measurement Data Storage Specification (MDSS) defines which data and where it shall be stored within the file structure. These specifications are MD specific and determine names, data types and units of information on e.g. the accelerator setup, machine operation mode, MD-settings, trigger events and the measurement raw data. As the ACS is continuously being extended and new information becomes available, MDSS is adapted regularly and the MDSS version can be used to trigger new analysis options.

Other sources of input data are devices not generating MEFF output like oscilloscopes or particle tracking applications like MAD-X (e.g. to verify analysis algorithms).

LEVEL TWO

Modules of the second level are used to analyze the data of a single measurement. There is one module per MD type each containing at least two classes: the MD class and the results class. The MD classes of all second level modules inherit from a common parent class 'SIMON' (SIngle Monitor data) while all result classes inherit from 'ResultsBase'. The results class itself represents a container class holding results data gathered during the analysis process. From ResultsBase all level two results classes inherit a method ('scalarresults') that returns a dictionary of all scalar results - an easy to use interface for higher level applications - and a parameter 'validdata'. This parameter by default is set to (boolean) False and level two MD classes have to set it to True in case the read data is valid.

SIMON focuses on providing methods for two purposes: a) extracting information from MEFF file names and b) reading the MD-specific measurement data file header defined in MDSS. A dedicated 'header' class is instanced in SIMON providing the option to access all header and meta data via 'instance.header' from within the framework.

All MD classes have to contain the following methods required for a common data processing work flow valid for all level two modules. Furthermore, the module developer has to make sure that calling the methods in the given order without handing over parameters results in a standard analysis of the data.

constructor

- initializes a new logger if none is handed over from higher level tools
- creates an instance of the results class and saves it as 'results' attribute. Therefore users can access result data via 'instance.results'.
- resets all flags and temporary attributes

read

- reads a single measurement data file using level one. The raw data is stored in the attribute 'instance.data'
- imports all the header and meta data via SIMON
- raises a data acquisition error ('DAQError') if indicated by the data
- save MD settings in a 'measurementsettings' attribute
- finalizes with setting the 'readsuccess'-flag to True

plotraw

- creates a plot of the raw data. In case the succeeding analysis fails, at least this plot is available for evaluation.

analyze

- verifies that the 'readsuccess'-flag is True
- performs the MD specific data analysis
- raises a 'NoBeamError' if no beam was distinguishable
- raises an 'AnalysisError' if the analysis was not performed correctly
- stores all analysis results in of the 'results' attribute
- sets the 'instance.results.valid' attribute to True in case the analysis worked properly
- finalizes with setting the 'analysissuccess'-flag to True

plot

- verifies that the 'analysissuccess'-flag is True
- creates a MD-specific plot containing the raw data, analysis results, filter settings and header and meta data
- saves the plot if the 'save' parameter was set to True

For level two modules, unit tests have been implemented which are used to verify the PACMAN core functionality with MD specific data for different scenarios (data with and without beam). Level two unit tests are set up to intrinsically test the key functionality of SIMON and level one. Therefore, no specific unit tests are available for these modules.

The following interface to higher level modules is common to all level two modules :

- method structure (read, analyze, plot)
- errors (DAQ, NoBeam, Analysis)
- raw data (instance.data)
- MD settings (instance.measurementsettings)
- results (instance.results)
- scalar results (instance.results.scalarresults)
- header data (instance.header)
- flags (readsuccess, analysissuccess)

LEVEL THREE

Modules of this level process data of multiple MDs using the (results) data of level two modules. The focus therefore moves from the technical raw data analysis to the interpretation of beam properties. The PACMAN core contains level three modules for trajectory analysis, component parameter variation scans (see paragraph 'Example') and emittance measurements. As the work flow is similar to the one of level two, also level three modules follow the structure of read, analyze and plot methods (plotraw is usually not possible as there is no raw data to be plotted).

Users performing measurement series create dedicated level three modules to analyze their data. As PACMAN is not part of the ACS and level three user modules are not part of the PACMAN core, these modules are not included in the complex release process for software of a machine for medical purpose and therefore can be implemented and adapted fast to provide new features and data treatment methods to the beam commissioning team.

LEVEL FOUR

Level four presents a graphical user interface of selected PACMAN core modules to the users (see Figure 2). The GUI was designed using the library Qt [4] and acts as an application launcher for PACMAN modules. The options are limited to 'standard' analysis as the GUI is used by less experiences users rather than beam commissioning experts.

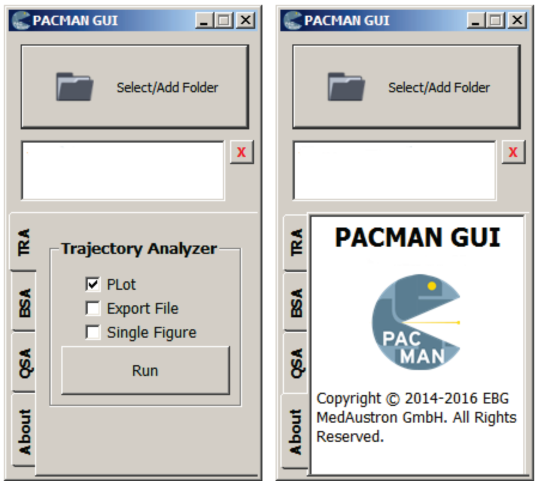

Figure 2: Two tabs of the PACMAN GUI.

EXAMPLE

Figure 3 shows a 2D result map of a level three PACMAN analysis. The strengths of two corrector magnets were varied with the aim to optimize the trajectory in a beam transfer line. A beam position monitor downstream performed the absolute measurements (result not shown). Additionally the same measurement series was performed with a different setting of the quadrupole magnets between the corrector magnets and the monitor. The following plot shows the beam position differences introduced by the different quadrupole settings for each corrector setting combination (SID).

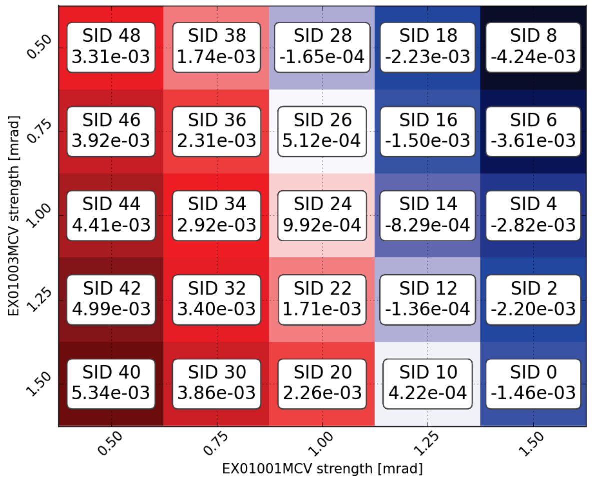

Figure 3: Level three example: map of beam position deltas (bottom value in cells [m]) for different corrector and quadrupole settings (indicated by SID)(in color online).

The PACMAN level three module calls the level two module corresponding to the beam position monitor in order to read the input data files using level one modules. After the level two analysis, a plot of each measurement is saved and the level three application extracts the beam position from the level two results. Furthermore level three directly uses level one to read a 'setup' file containing the corrector and quadrupole settings for each SID.

The level three itself then performs an analysis of the 'setup' file to identify corresponding corrector settings and computes the beam position deltas plotted in Figure 3. The darker red and blue the background of the map cell, the larger is the beam position difference. As the analysis of measurement series like the example presented here can be automatized and executed by any user, the application is available in the GUI (Level four, BSA tab).

CONCLUSION AND OUTLOOK

The PACMAN framework has become an indispensable tool during the beam commissioning of the MedAustron accelerator facility [5] [6]. The robust and reliable analysis applications allowed the beam commissioning team to focus on the physics of the accelerator and technical challenges rather than on raw data interpretation. With the upcoming commissioning of additional beam modalities (particle types, energies, ...) and integration of new beam monitors into the ACS, the maintenance and development of PACMAN is and will continue to be in constant evolution.

ACKNOWLEDGMENT

The authors would like to acknowledge the important contributions of the beam commissioning team, the ACS group and Virgile Letellier from the medical physics department.

REFERENCES

[1] Ch. Kurfuerst et al., presented at IPAC'16, paper TUPMR035, this conference.

[2] A. Wastl et al., IPAC2015 (2015).

[3] `www.python.org`

[4] `www.qt.io`

[5] A. Garonna et al., presented at IPAC'16, paper THOAB01, this conference.

[6] T. Kulenkampff et al., presented at IPAC'16, paper TUPMR036, this conference.