COMMISSIONING THE MEDAUSTRON ACCELERATOR WITH PROSHELL

R. Moser [1], A. Brett[1], U. Dorda[1], M. Eichinger[1], J. Gutleber[2], M. Hager[1], M. Janulis[1], J. Junuzovic[1], M. Junuzovic[1], M. Marchhart[1], H. Pavetits[1], C. Torcato de Matos[1]
[1]EBG MedAustron, Wr.Neustadt, Austria
[2]CERN, Geneva, Switzerland

Abstract

MedAustron is a synchrotron-based centre for light ion therapy under construction in Austria. The accelerator and its control system entered the on-site commissioning phase in January 2013. This contribution presents the current status of the accelerator operation and commissioning procedure framework called ProShell. It is used to model measurement procedures for commissioning and operation with Petri-Nets. Beam diagnostics device adapters are implemented in C#. To illustrate its use for beam commissioning, procedures currently in use are presented including their integration with existing devices such as ion source, power converters, slits, wire scanners and profile grid monitors. The beam spectrum procedure measures distribution of particle species generated by the ion source. The phase space distribution procedure performs an emittance measurement in beam transfer lines. The trajectory steering procedure measures the beam position in each part of the machine and aids in correcting the beam positions by integrating MAD-X optics calculations. Additional procedures and (beam diagnostic) devices are defined, implemented and integrated with ProShell on demand as commissioning progresses.

INTRODUCTION

MedAustron [1] [2] is an ion therapy and research centre presently under construction in Wiener Neustadt, Austria. The facility features a synchrotron-based accelerator (Figure 1) with up to 5 ion sources for protons, carbon ions and possibly other light ions. It will provide ion beams with energies up to 800MeV to 5 beam lines, one of which is a rotating proton gantry.

The Procedure Shell Execution Framework (ProShell) is a C# application to automate high-level control and analysis tasks for commissioning and operation [3]. Each task called a procedure implements a standardized procedure interface and is deployed as .NET assembly (shared objects). Key features of the ProShell are:

- Allocating resources on behalf of a procedure.
- Uniform access to system, software and physical devices independent of communication protocols for monitoring and control purposes.
- Reception and visualization of device measurements.
- Management of generic procedure lifecycle and custom procedure workflow.
- Parallel execution of multiple procedures.
- Automatic procedure execution without user intervention.
- Manual procedure execution to step through the procedure specific workflow.
- Provide access to control system services hiding implementation specific interfaces and communication protocols.

ARCHITECTURE

Overview

ProShell is a framework to dynamically load and execute procedures implemented as C# classes. As outlined in Figure 2 it provides access to system, software and physical devices for monitoring and control purposes. These services are accessible from ProShell through service-specific *Driver* objects that are used internally and are not directly accessible from the loaded procedures:

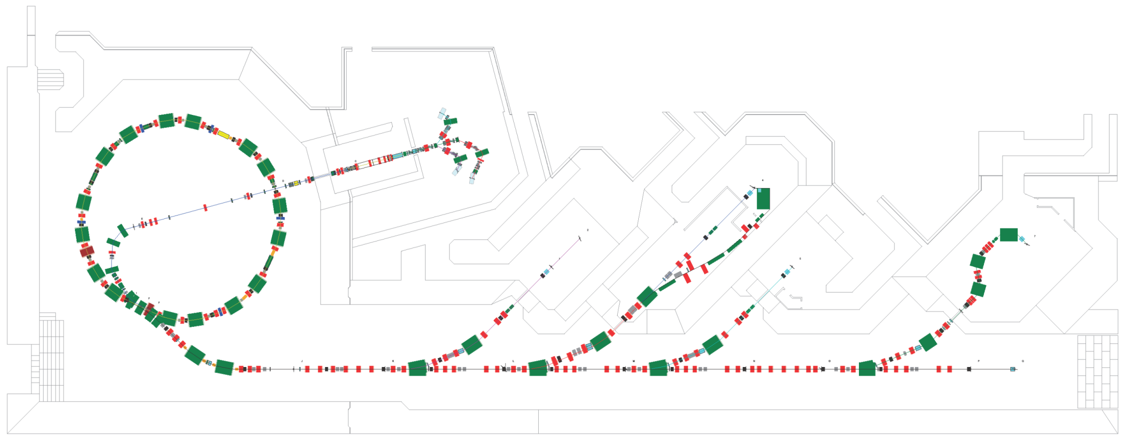


Figure 1: MedAustron accelerator layout.

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7

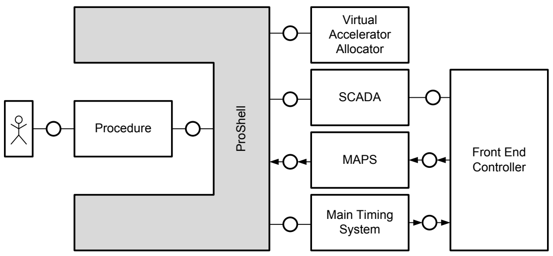


Figure 2: General ProShell architecture and main communication partners.

- *Virtual Accelerator Allocator* (VAA) is the scheduler of the system that allocates resources for exclusive usage on behalf of a user application.
- *WinCC OA* is a Supervisory Control and Data Acquisition (SCADA) tool from Siemens [4]. It acts as the main communication backbone between user interfaces and procedures on tier 1 and frontend controllers and devices on tier 3 [5].
- *MAPS services* are a set of data servers implementing a publisher subscriber protocol. It forwards measurements and main timing information from front-end controllers to user applications in non-real-time.
- *Main Timing System* (MTS) generates events for beam generation that are delivered to the frontend controllers with a precision of 100ns [6].

Resources

All device instances and types are represented in WinCC OA as data points (DP) and data point types (DPT). In addition each DPT contains a set of data point elements (DPE) that are name-value pairs with a defined value type. Each device implements one of the following interfaces:

- *BasicDevice* is a state-less front-end device interface that only provides a minimal set of DPEs for monitoring.
- *StateDrivenDevice (SDD)* is a state-driven front-end device interface that extends the BasicDevice with additional DPEs to provide an interface for commanding and login to guarantee exclusive access to a device.

Two special resource types are defined in WinCC OA to control a set of devices concurrently through a single virtual device:

- *Working Set* (WS) is a virtual device that implements the SDD interface and controls a set of SDDs.
- *Virtual Accelerator* (VAcc) controls a set of Working Sets and subsequently a set of devices. It also implements the SDD interface. In addition a VAcc also contains a dynamically assigned Main Timing Generator that allows a procedure to emit timing information for beam generation with an accuracy of 100ns.

ProShell encapsulates communication over a number of different communication technologies, shielding

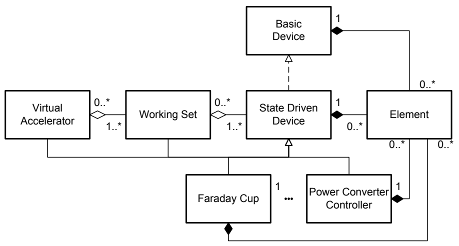


Figure 3: Class diagram of resources adapters for devices, Working Sets and Virtual Accelerators.

procedures from the underlying addressing and communication specifics by providing devices as object that follow the adapter pattern [7]. Adapter objects provide an object-oriented API that implement BasicDevice or StateDrivenDevice interfaces as shown in Figure 3.

Procedure

A *Procedure* encapsulates specific repetitive control and processing tasks for operation and commissioning written in C#. All procedures implement a unified interface and the life cycle defined in Figure 4. Each procedure is running a separate ProcedureContext that acts as a container and provides coordinated access to devices and control system services and manages the procedure lifecycle. Thus it provides the capability to execute procedures in parallel. In case of resource conflicts the VAA will delay the allocation of one procedure and subsequently ProShell will delay the execution of this procedure.

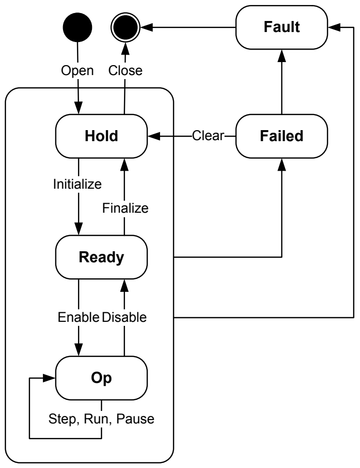


Figure 4: Procedure context lifecycle.

While the procedure is in *Hold* state no resources are allocated. When the user issues an *Initialize* transition the procedure allocates the specified resources from the VAA. The procedure lifecycle state moves to *Ready* when all resources have been allocated. Thereafter the user may emit an *Enable* transition that parameterizes the procedure specific Petri net and moves the state machine into *Op* state. While in *Op* state the procedure specific workflow defined in a Petri Net Modelling Language (PNML) file [8] can be executed in steps or run until termination.

ISBN 978-3-95450-139-7

Copyright © 2014 CC-BY-3.0 and by the respective authors

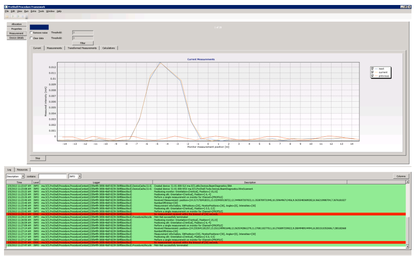


Figure 5: Single beam intensity measurement performed by a wire scanner.

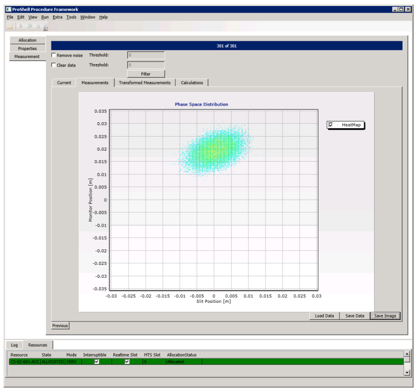


Figure 6: Combined phase space distribution measurement plotted over slit gap and wire position.

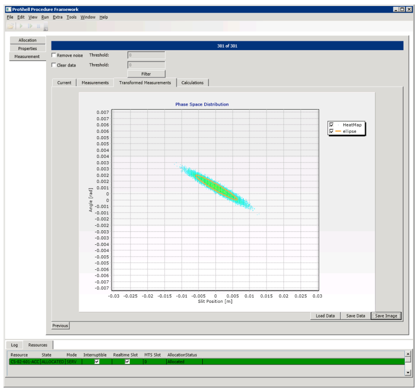
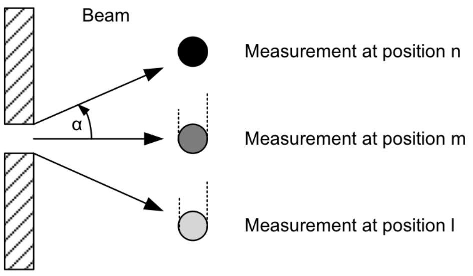


Figure 7: Result of the phase space distribution measurement procedure.

PHASE SPACE DISTRIBUTION MEASUREMENT

The phase space distribution measurement procedure measures the horizontal or vertical beam distribution in the low energy beam transfer line (LEBT) or the medium energy beam transfer line (MEBT). The basic principle of this procedure is to position two horizontal or vertical slit plates to form a gap through which only the selected part of the beam can pass. Further downstream, a wire scanner beam diagnostic device performs a beam profile measurement as depicted in Figure 8.


Beam
α
Measurement at position n
Measurement at position m
Measurement at position l
Figure 8: Measuring a beam profile for a single slit position.

These beam profiles measurements depicted in Figure 5 are repeated for gap-positions all across the beam pipe aperture based on the workflow defined in Figure 9.

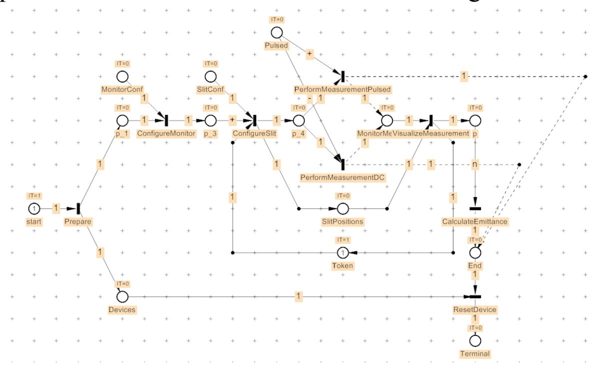


Figure 9: Petri net implemented by the phase space distribution measurement procedure.

The results for all slit-gap position specific measurements are combined to a single heat-map with slit position on x-axis, monitor position on y-axis and beam intensity as heat as shown in Figure 6. Based on slit and monitor positions the plot is transformed to a heat-map with slit position on the x-axis, beam angle on y-axis and beam intensity as heat as reproduced in Figure 7. Finally an emittance calculation is performed on the transformed measurement data.

The measurement can also be performed with a profile grid monitor (as used in the MedAustron MEBT) instead of the wire scanner but with a lower resolution due to the number of physical wires.

TRAJECTORY STEERING

The trajectory steering procedure is a commissioning procedure to compute correction values for steering magnet power converter set-points in transfer lines.

After a beam line is selected, steering magnets and beam diagnostics devices (wire scanners and profile grid monitors) are visualized based on their optical position (s-position) within the beam line as shown in Figure 10.

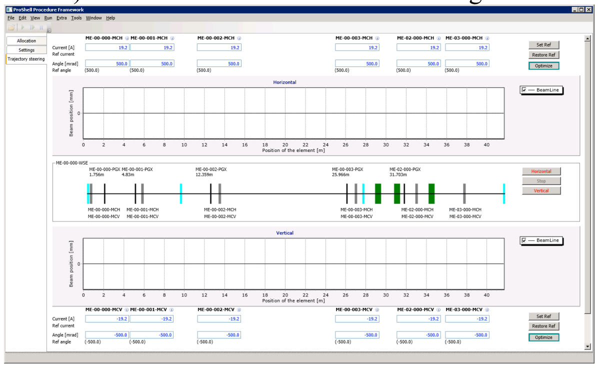


Figure 10: Trajectory steering procedure user interface.

The operator may manually adapt the steering angle Θ in mrad based on initial values stored in the element configuration database [9]. The power converter current to be applied is calculated in two steps. First, the angle Θ is transformed to the integrated magnetic field Bl that has to be generated in the steering magnet using $Bl = \Theta * B\rho$ where Bρ is a beam characteristics value constant for each

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7

transfer line. In a second step the magnetic field Bl is converted to the power converter current through by applying a previously measured magnet specific B(I) transformation curve. Alternatively the operator may apply a current directly.

When the operator starts to execute the steering procedure, currents for all steering power converters are applied and destructive measurements are performed with wire scanners and profile grid monitors in the order of their optical position from the ion source (see Figure 11).

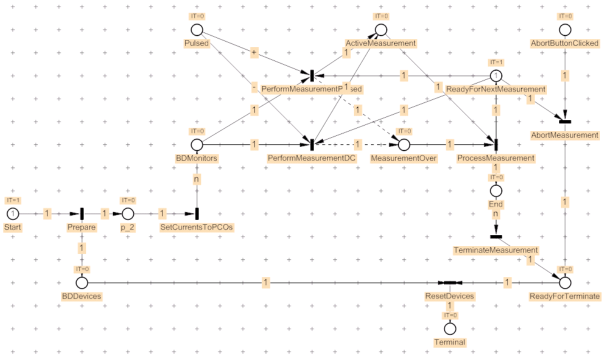


Figure 11: Petri net implemented by the trajectory steering procedure.

Subsequently the operator may decide to export a revised machine model with corrector settings for specific beam properties. These files can be imported into the element configuration database and be used in subsequent commissioning activities as default values as shown in Figure 12.

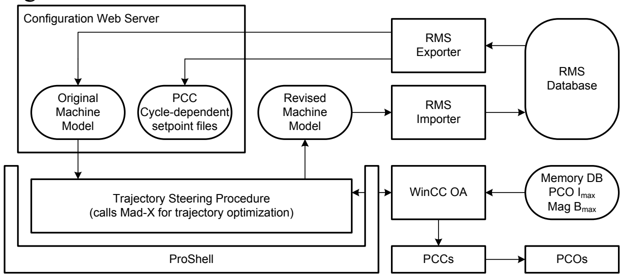

Configuration Web Server
Original Machine Model
PCC Cycle-dependent setpoint files
Revised Machine Model
RMS Exporter
RMS Importer
RMS Database
Trajectory Steering Procedure (calls Mad-X for trajectory optimization)
ProShell
WinCC OA
Memory DB PCO I_{max} Mag B_{max}
PCCs
PCOs

Figure 12: Basic data flow for trajectory steering.

Additionally the measured beam positions can be exported to MAD-X in order to compute the optimal corrector settings [10].

SUMMARY

This article gave an overview of the architecture for the MedAustron Procedure Shell Execution Framework. The framework follows a container-based approach where each procedure is executed in its own sandbox. Accelerator devices are monitored and controlled through resource adapter objects that provide an object-oriented interface hides protocol specific details.

Integration with the main systems (WinCC OA, MTS, VAA and MAPS) has been concluded and been tested in the production system. Resource adapters for power converter controllers and several beam diagnostics devices such as wire scanners, slits and profile grid monitors have been implemented. Subsequently the design for two procedures namely the phase space distribution measurement and the trajectory steering procedures have been elaborated.

Implementation and testing of the phase space distribution measurement procedure has concluded and tests have been successfully carried out in the production system in April 2013 using the operational wire scanner and slit devices. Since then the procedure is used for beam commissioning in the low energy beam transfer line.

Implementation of the trajectory steering procedure has concluded for LEBT and MEBT. Tests have been performed with operational power converters and wire scanners. Final tests with profile grid monitors are planned for October 2013.

These two procedures are used for beam commissioning since May 2013 and show that the chosen approach fits the needs of the commissioning team. Future work consists primarily of two activities: First, integrating more devices into the accelerator control system and implementing resource adapters. Secondly, gathering requirements, elaborating designs and implementing additional procedures required for commissioning and operation.

REFERENCES

[1] M. Benedikt, A. Wrulich, “MedAustron—Project overview and status”, Eur. Phys. J. Plus (2011) 126: 69.

[2] M. Benedikt, A. Fabich, “MedAustron—Austrian hadron therapy centre”, Nuclear Science Symposium Conference Record 2008. NSS '08. IEEE, pp.5597-5599, 19-25 Oct. 2008.

[3] R. Moser et al., “ProShell – The MedAustron Accelerator Control Procedure Framework”, in Proc. ICALEPCS 2011.

[4] P. Golonka, M. Gonzales-Berges, “Integrated Access Control for PVSS-Based SCADA Systems at CERN”, in Proc. ICALEPCS 2009.

[5] J. Gutleber et al., “The MedAustron Accelerator Control System”, Proc. ICALEPCS, 2011.

[6] J. Dedic et al., “Timing System for MedAustron Based on Off-The-Shelf MRF Transport Layer”, in Proc. IPAC 2011.

[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides, “Design Patterns—Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995.

[8] J. Billington et al., “The Petri Net Markup Language: Concepts, Technology, and Tools”, Proc. 24th Int. Conf. Application and Theory of Petri Nets (ICATPN’2003), Eindhoven, The Netherlands, June 2003.

[9] R. Moser et al., “The MedAustron Control System Configuration Management Database”, poster presented at ICALEPCS 2011.

[10] L. Deniau, “MAD-X progress and future plans”, CERN-ATS-Note-2012-078 MD, 13 Sep 2012.

ISBN 978-3-95450-139-7

Copyright © 2014 CC-BY-3.0 and by the respective authors