ProShell - The MedAustron Accelerator Control Procedure Framework

R. Moser[1], A. B. Brett[1], J. Dedič[3], J. Gutleber[2], M. Marchhart[1], C. Torcato De Matos[1], S. Sah[3]
[1]EBG MedAustron, Wiener Neustadt, Austria
[2]CERN, Geneva, Switzerland
[3]Cosylab, Ljubljana, Slovenia

ABSTRACT

MedAustron is a centre for ion-therapy and research in currently under construction in Austria. It features a synchrotron particle accelerator for proton and carbon-ion beams. This paper presents the architecture and concepts for implementing a procedure framework called ProShell. Procedures to automate high level control and analysis tasks for commissioning and during operation are modelled with Petri-Nets and user code is implemented with C#. It must be possible to execute procedures and monitor their execution progress remotely. Procedures include starting up devices and subsystems in a controlled manner, configuring, operating O(1000) devices and tuning their operational settings using iterative optimization algorithms. Device interfaces must be extensible to accommodate yet unanticipated functionalities. The framework implements a template for procedure specific graphical interfaces to access device specific information such as monitoring data. Procedures interact with physical devices through proxy software components that implement one of the following interfaces: (1) state-less or (2) state-driven device interface. Components can extend these device interfaces following an object-oriented single inheritance scheme to provide augmented, device-specific interfaces. As only two basic device interfaces need to be defined at an early project stage, devices can be integrated gradually as commissioning progresses. We present the architecture and design of ProShell and explain the programming model by giving the simple example of the ion source spectrum analysis procedure.

CONTACTS

Name: Roland Moser
Organization: EBG MedAustron
Email: roland.moser@cern.ch
Phone: +41 22 76 78609
Website: http://www.ebgmedaustron.at

INTRODUCTION

MedAustron is an ion therapy and research centre presently under construction in Wiener Neustadt, Austria. The facility features a synchrotron-based accelerator 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 during operation. Each task called a procedure implements a standardized procedure interface and is deployed as .NET assembly (shared objects).

ARCHITECTURE

ProShell is a framework to dynamically load and execute procedures implemented as C# classes. 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 procedures:

- **Virtual Accelerator Allocator** (VAA) allocates resources for exclusive usage on behalf of user applications.
- **WinCC OA** is a SCADA system that acts as the main communication backbone between user interfaces and frontend controllers (FEC).
- **MAPS** is a publisher subscriber to forward measurements from FECs to user interfaces.
- **Main Timing System** (MTS) generates events for beam generation that are delivered to the frontend controllers with a precision of 100ns.

PROCEDURE CONTEXT

Each procedure is executed in a separate **Procedure Context** that acts as a **container** to provide coordinated access to devices and control system services and manages the procedure lifecycle. Each context manages the following object separately (Figure 2):

- **Procedure** implements a specific control or processing task and provides a standardized interface to be controlled by the Procedure Context.
- **Procedure Lifecycle** handles the general lifecycle of the procedure that is common to all procedures (Figure 4).
- **Petri Net Engine** is a workflow engine that executes the procedure specific workflow defined in a Petri Net Modelling Language (PNML) file.
- **DeviceCache** allocates resources on using the VAA driver and keeps a cache of C# resource adapter objects.

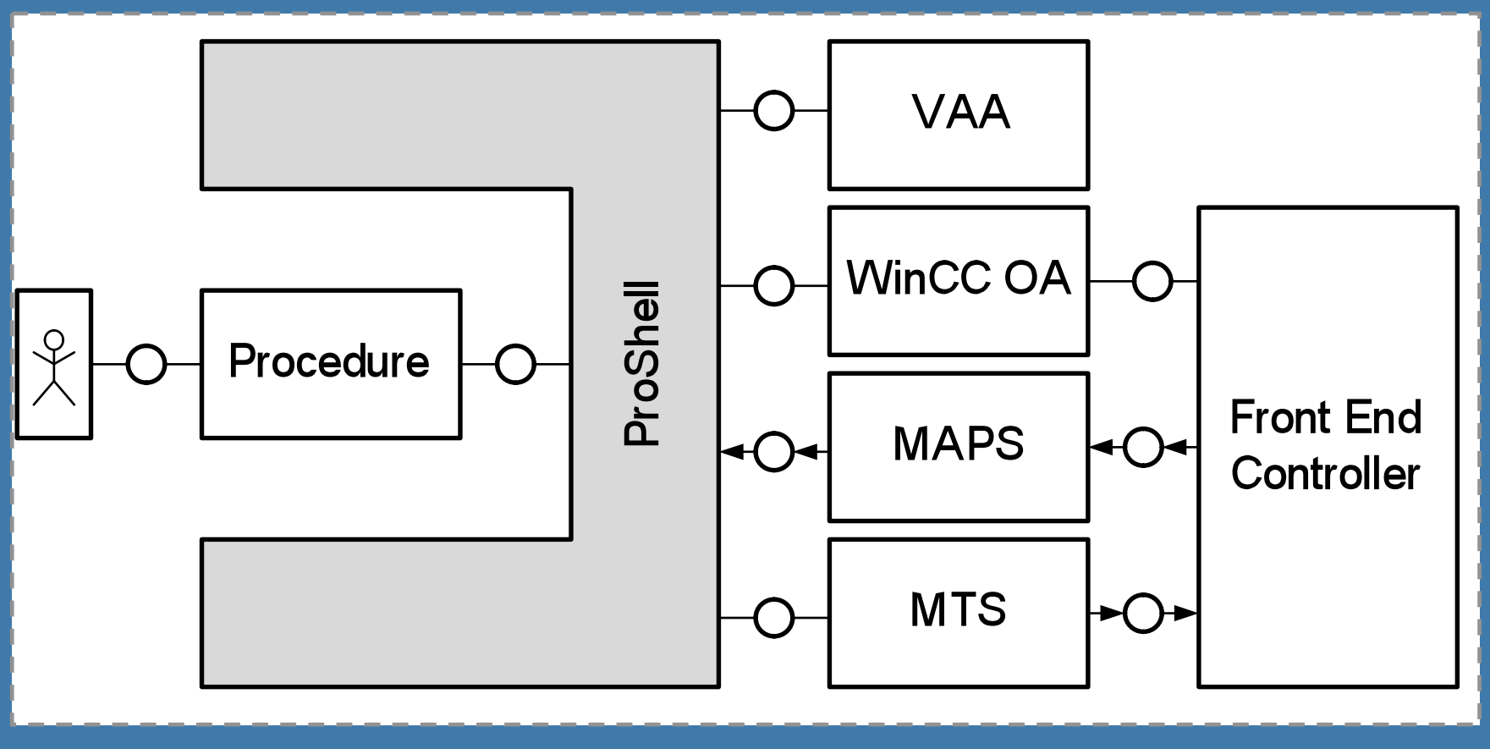

Figure 1. ProShell Architecture

RESOURCES

Resources are made available through adapter objects that shield procedures from the underlying addressing and communication specifics. Each adapter owns a set of fields that implement the IElement interface. Classes implementing the IElement interface interact with drivers to communicate with the front-end controllers and (1) implement data type conversion and (2) perform client-side validity checks. They implement one of the following interfaces:

- **BasicDevice** is a state-less front-end device interface that provides a minimal set of DPEs for monitoring.
- **StateDrivenDevice** (SDD) is a state-driven front-end device interface that extends the BasicDevice to provide mechanisms for sending commands to front-end controllers and for login to gain exclusive access to a device.

ProShell also provides resource adapters to control a set of SDDs 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. 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.

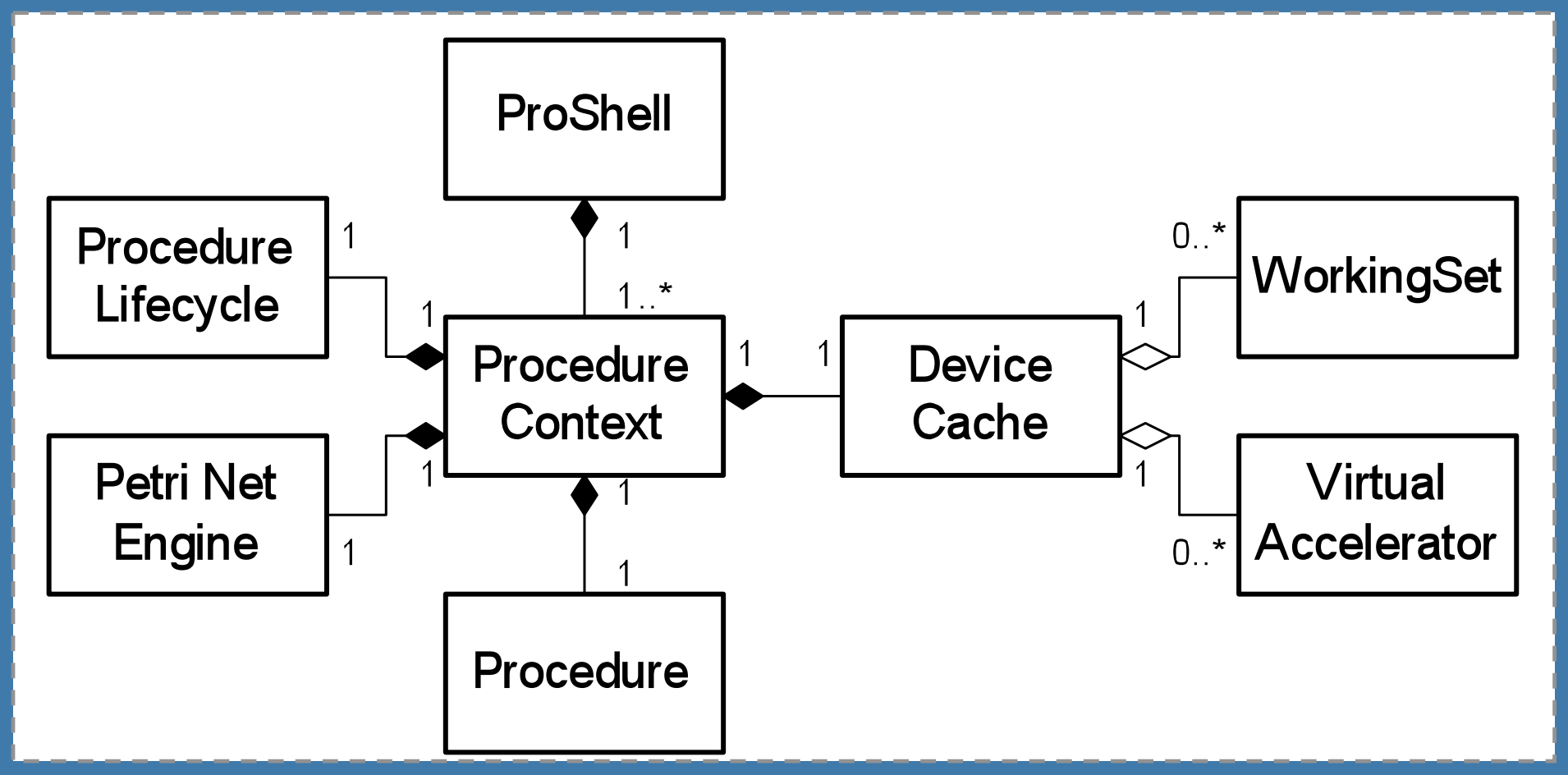

Figure 2. ProcedureContext managing the procedure and related objects

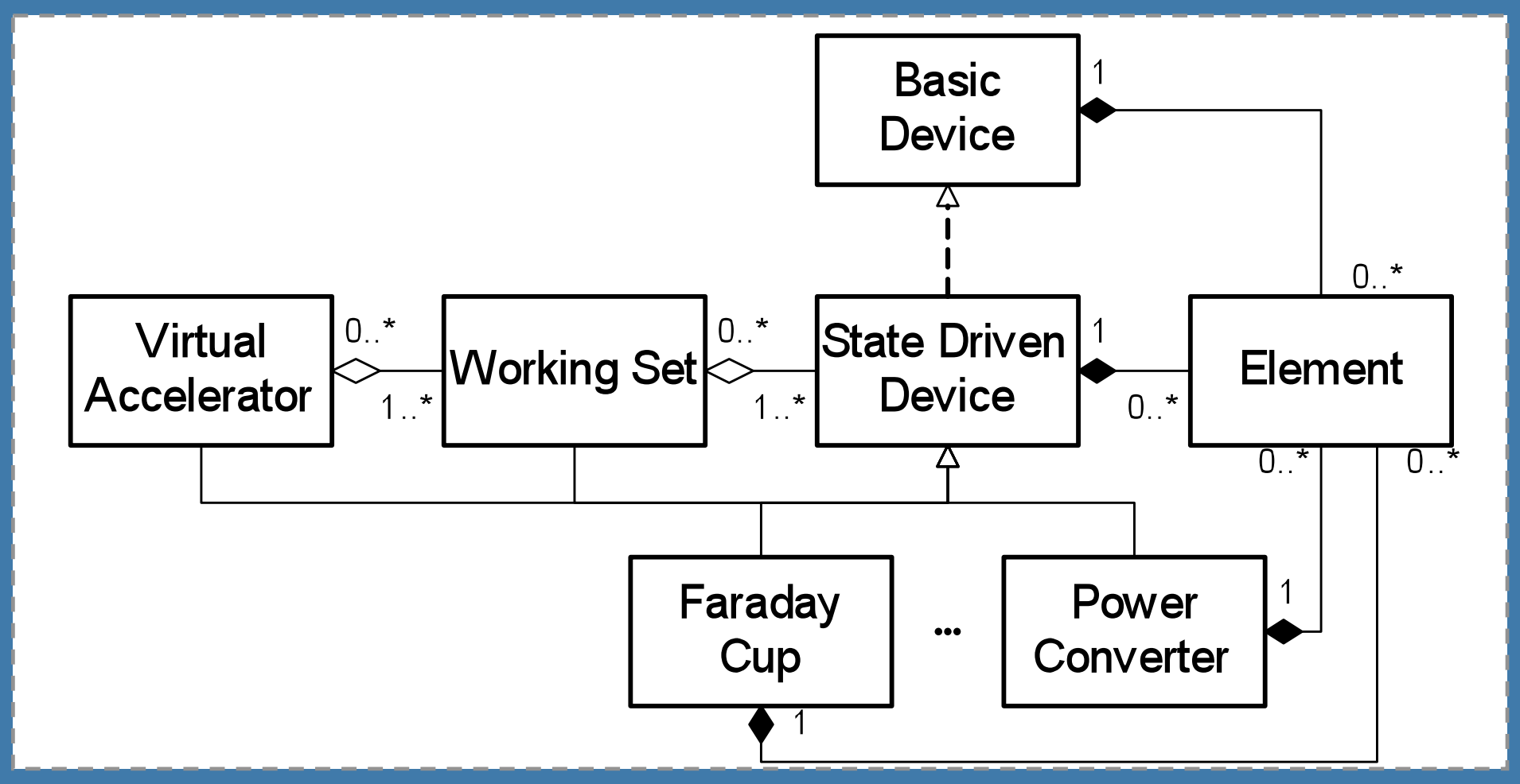

Figure 3. Class Hierarchy for resources adapters

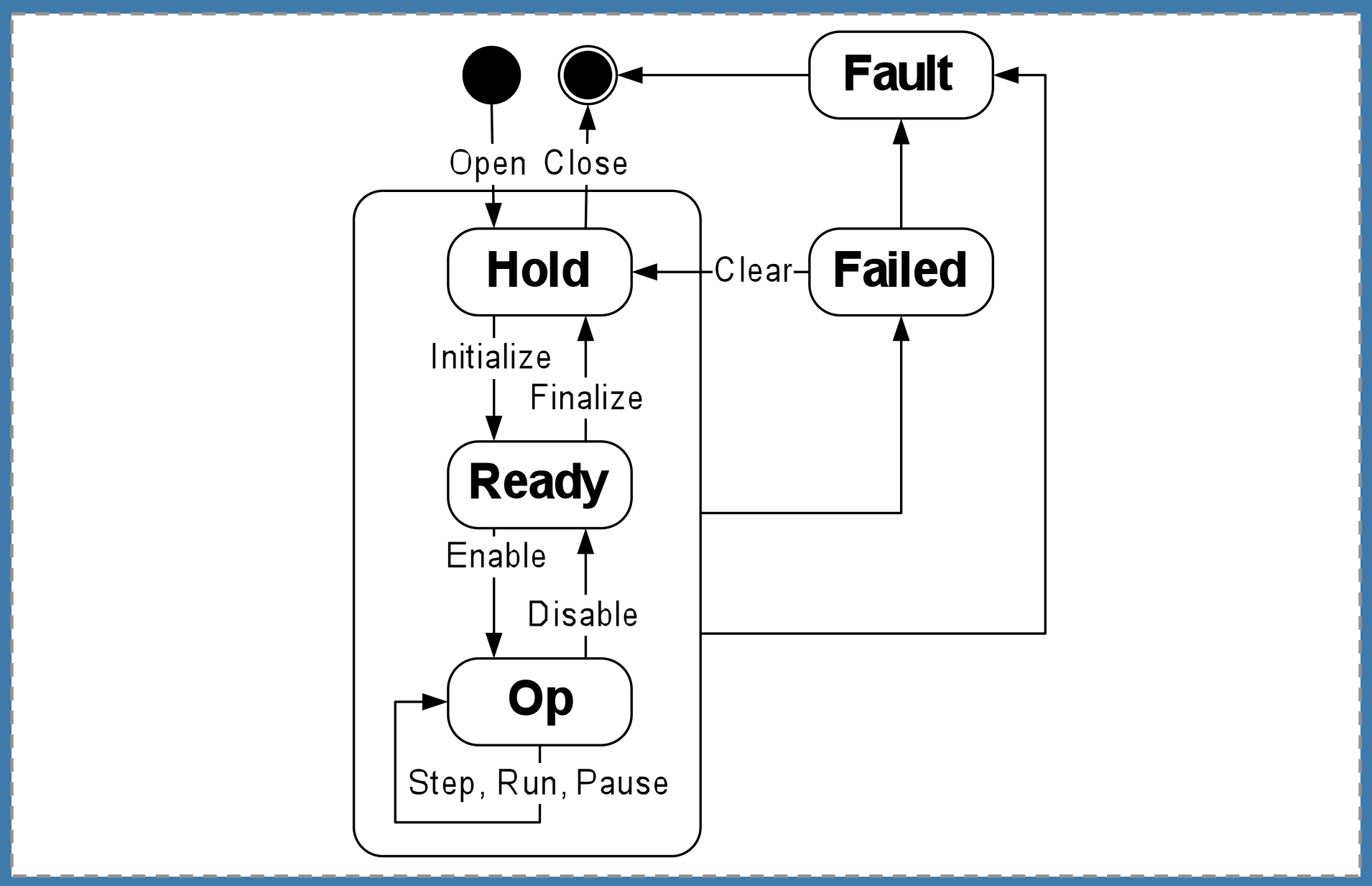

Figure 4. Procedure Context Lifecycle

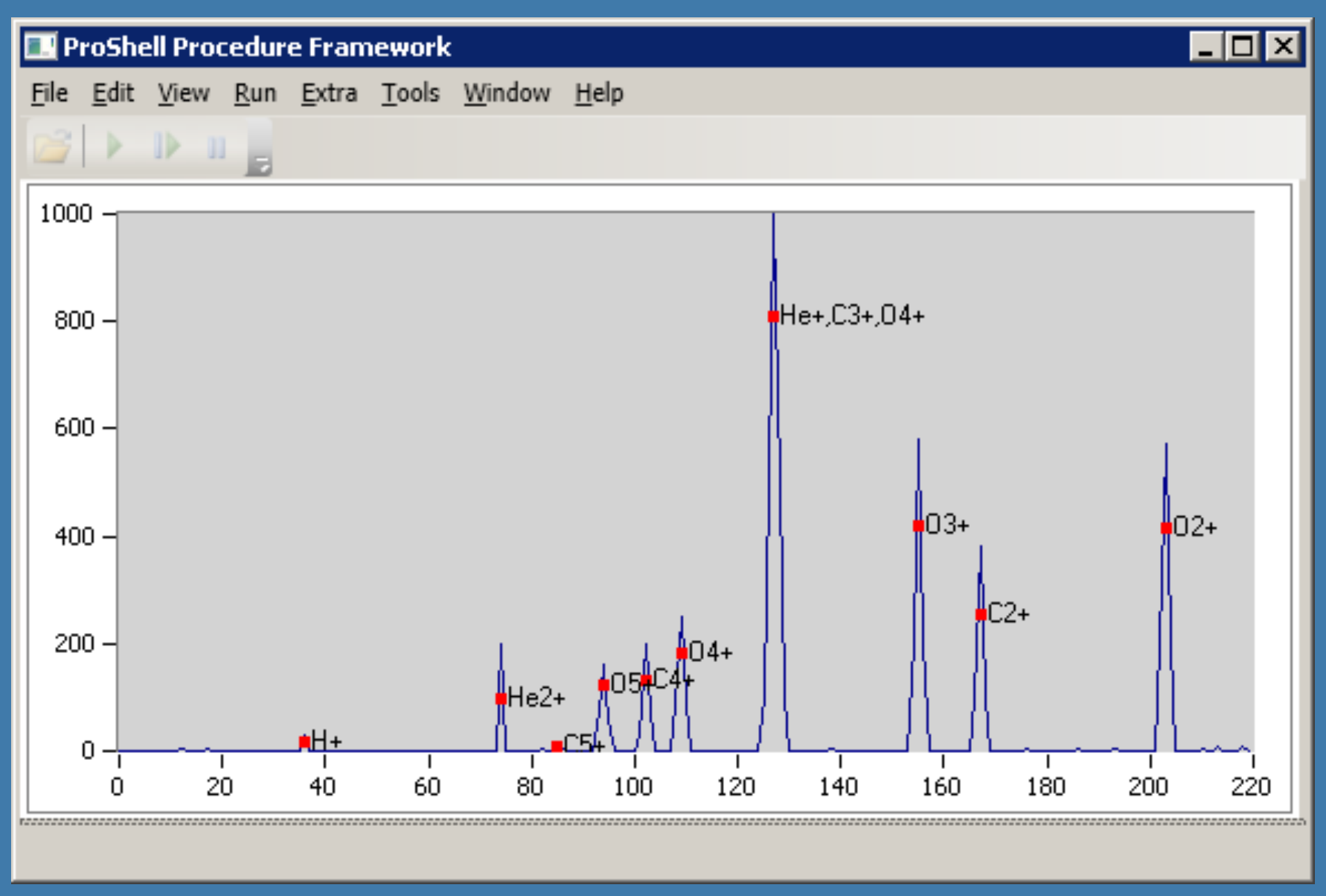

Figure 5. Beam Spectrum Measurement Output

BEAM SPECTRUM ANALYSIS

The ion source beam spectrum analysis procedure **detects the particles types generated by** a specific **ion source**. Therefore a current is applied to the bending magnet. The generated field deflects the particles in an angle depending on the particle mass and the applied field. The energy of the particles that hit the following faraday cup is measured. Due to correlation of current to particle type, the generated particles can be detected with peak detection algorithm on the energy over current plot (Figure 5).

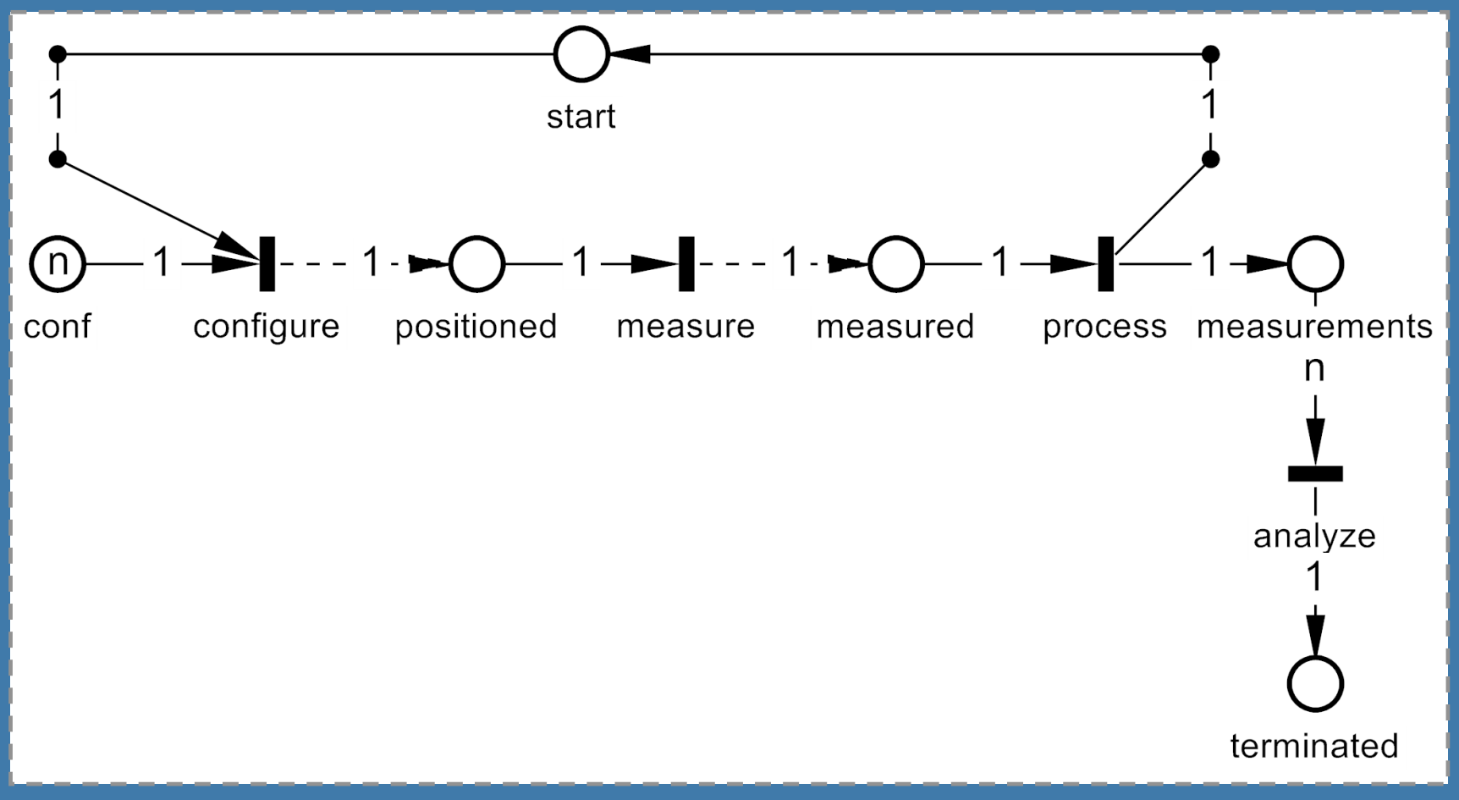

Figure 6. Ion source beam spectrum analysis procedure workflow

```
public override void OnEnable()
{
 _faradayCup.Move(true, 20);
 PetriNet net = Context.PetriNet;
 net.BindParameter("n",
  (uint)Currents.Cur.Count);
 IPlace place = net.GetPlace("conf");
 foreach (Tuple<double> current in Currents.Cur)
 {
   CurrentToken token = new CurrentToken()
   { Current = current.Item1 };
   place.AddInitialToken(token);
 }
 Context.PetriNet.Trigger();
}
```

Figure 7. Enable transition to move the faraday cup in and to configure the Petri net.

```
private void Configure(PetriNet petrinet,
  ITransition transition,
  Dictionary<IArc, Queue<IToken>> input,
  Dictionary<IArc, Queue<IToken>> output)
{
 List<CurrentToken> tokens =
  petrinet.GetTokens<CurrentToken>(input);
 if (tokens.Count == 1)
 {
  var cycleToken = (CurrentToken) token;
  _pcc.CurrentAqn.Subscribe(
   petrinet.GetPlace("positioned"),
   token[0].Current);
  _pcc.CurrentCcv.Value = token[0].Current;
  petrinet.GenerateTokens(output);
 }
}
```

Figure 8. *conf* transition that applies a setpoint and waits for it to be reached by the power supply.

SUMMARY

Integration with the main systems (WinCC OA, MTS, VAA and MAPS) has been concluded. First tests have been carried out in the MedAustron test column with the presented procedure and generic procedures that execute beam cycles emulating not-available devices with WinCC OA scripts. These tests have shown that generic tasks such as allocation, data conversion and validity checks can be encapsulated in ProShell to provide a simplified interface for procedures.

REFERENCES

[1] M. Benedikt, A. Wrulich, “MedAustron—Project overview and status”, Eur. Phys. J. Plus (2011) 126: 69.
[2] J. Billington et al., “The Petri Net Markup Language: Concepts, Technology, and Tools”, Proc. ICATPN, 2003.
[3] J. Dedic et al., “Timing System for MedAustron Based on Off-The-Shelf MRF Transport Layer”, Proc. IPAC 2011.
[4] P. Golonka, M. Gonzales-Berges, “Integrated Access Control for PVSS-Based SCADA Systems at CERN”, Proc. ICALEPCS, 2009.
[5] E. Gamma et al., “Design Patterns—Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995.
[6] J. Gutleber et al., “The MedAustron Accelerator Control System”, Proc. ICALEPCS, 2011.