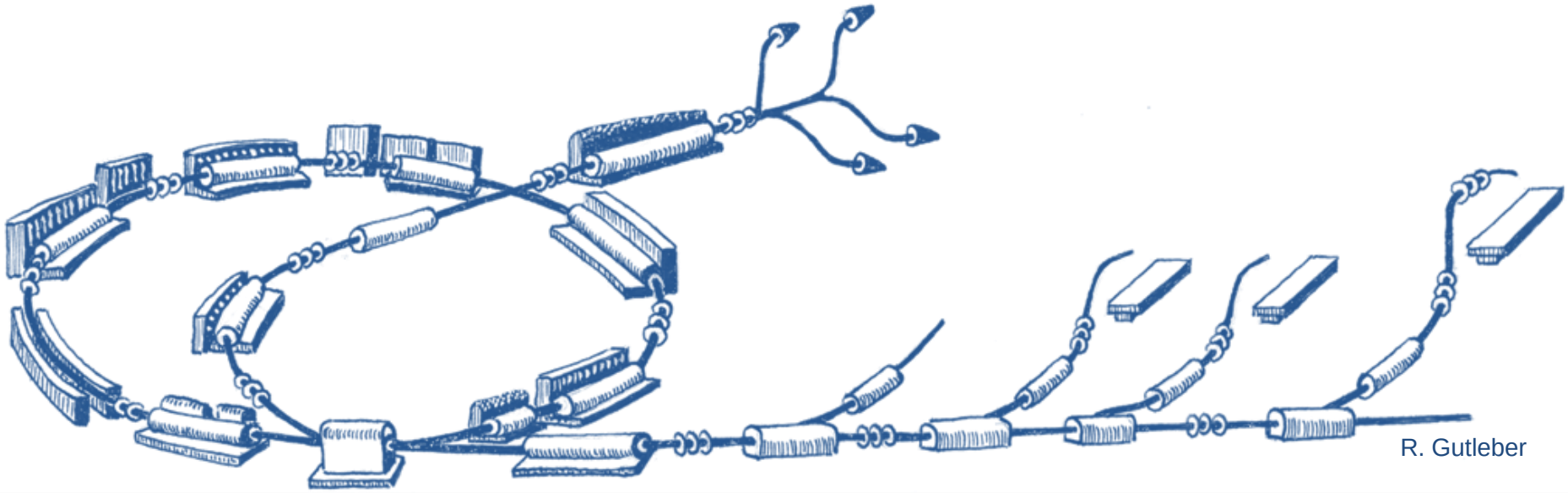
ProShell Procedure Framework Status

MedAustron Control System Week 2

October 7th, 2010

Roland Moser



R. Gutleber

Overview

- Scope
- Concept
- Architecture
- Status

Scope

- Presents an **overview** of the planned **architecture** of the ProShell Procedure Framework
- Shows the **current status** and **plan** till December

CONCEPTS

General Terms

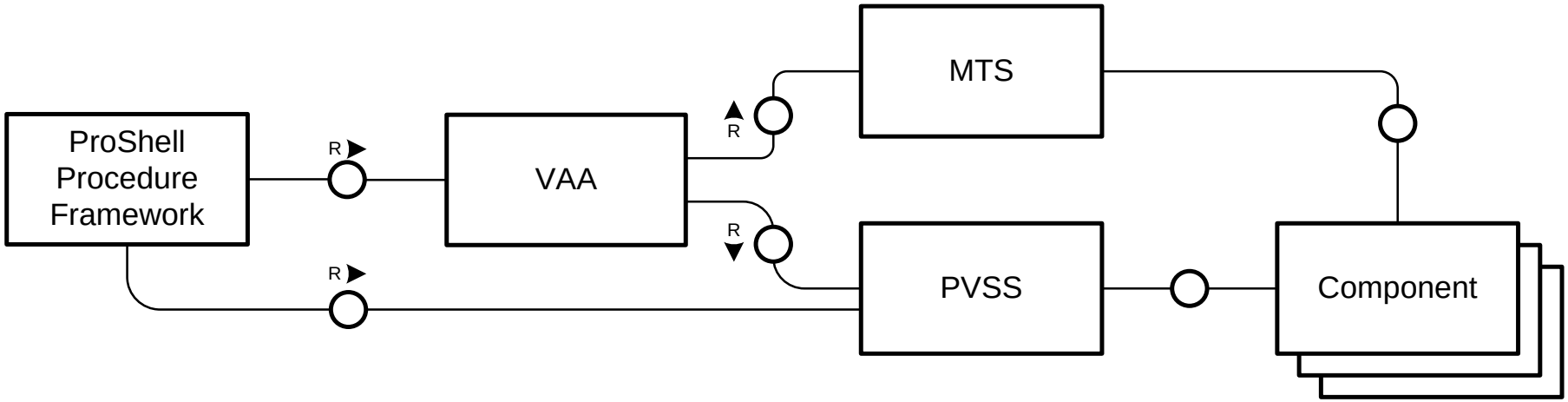
- **ProShell** Procedure Framework
  - Windows C# Application managing procedures
- **Procedure**
  - a module that performs a task
  - E.g. Emittance measurement, Procedure to change mode to clinical, Quality assurance procedures, etc.
- **Resource**
  - Device, Working Set, Virtual Accelerator
- **Driver**
  - Provides a high-level interface to a component
  - E.g. PVSS

ProShell Procedure Framework

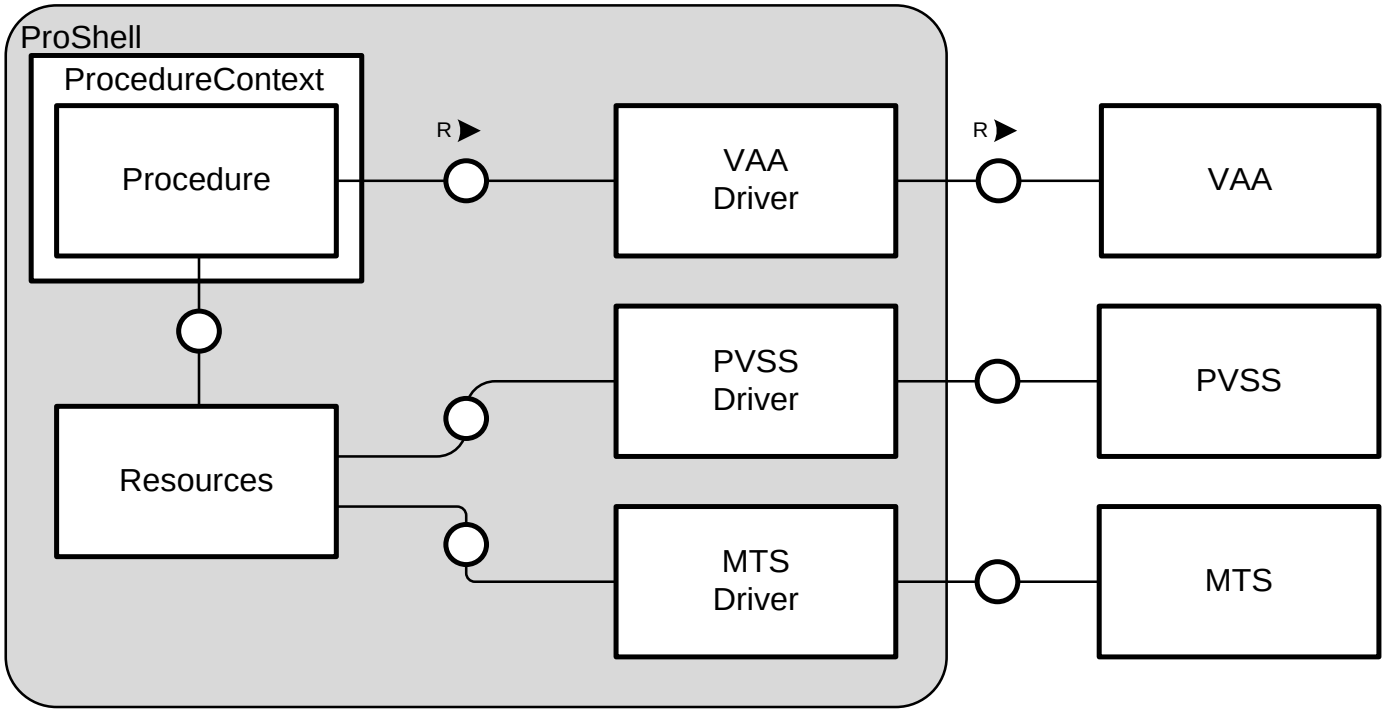
- proides a graphical user interface that
- dynamically **loads** Procedures,
- **manages** Procedures,
- provides **APIs** to interact with control system components
  - Allocate resources through VAA
  - Communicate with resources

ARCHITECTURE

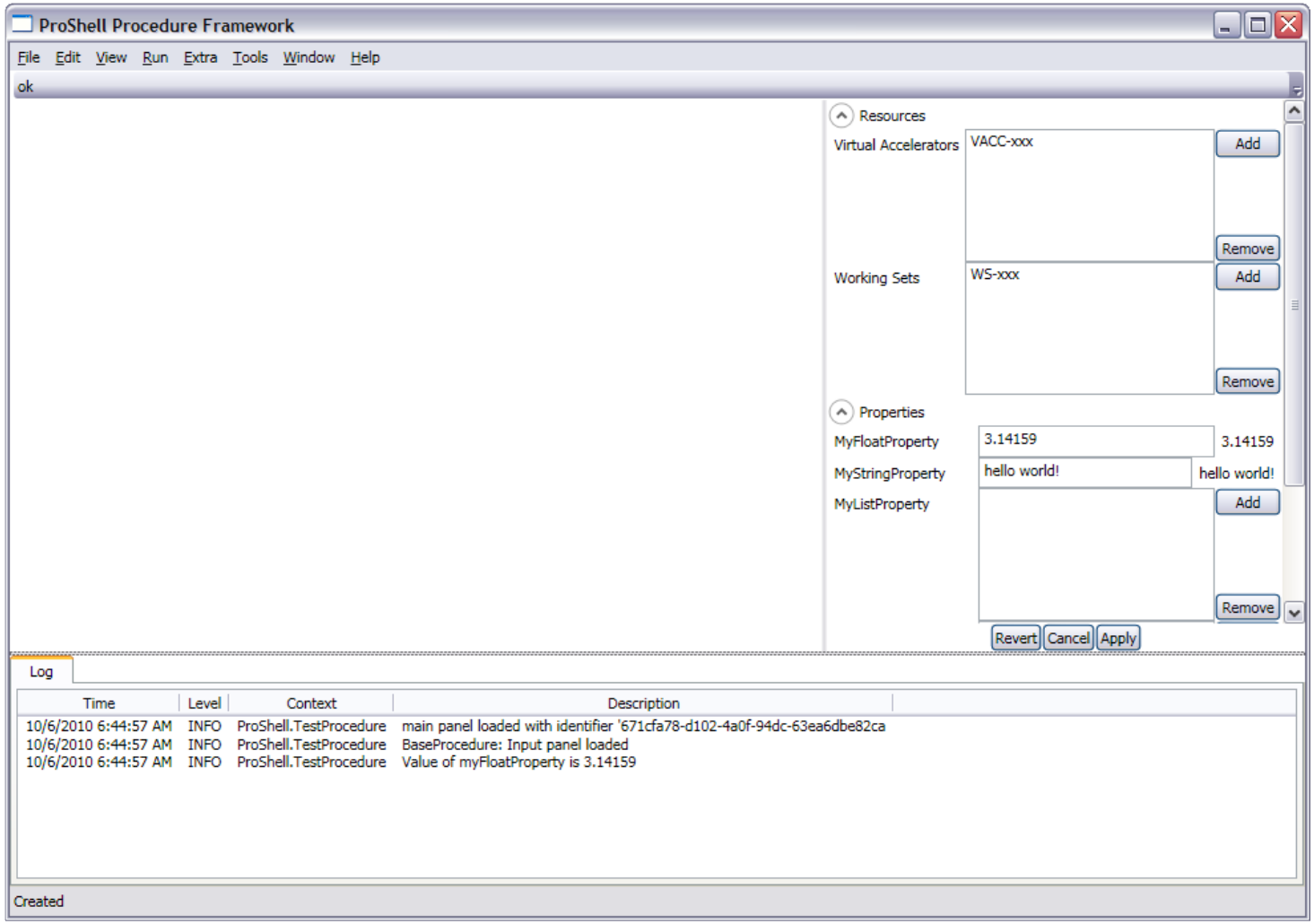
Overview



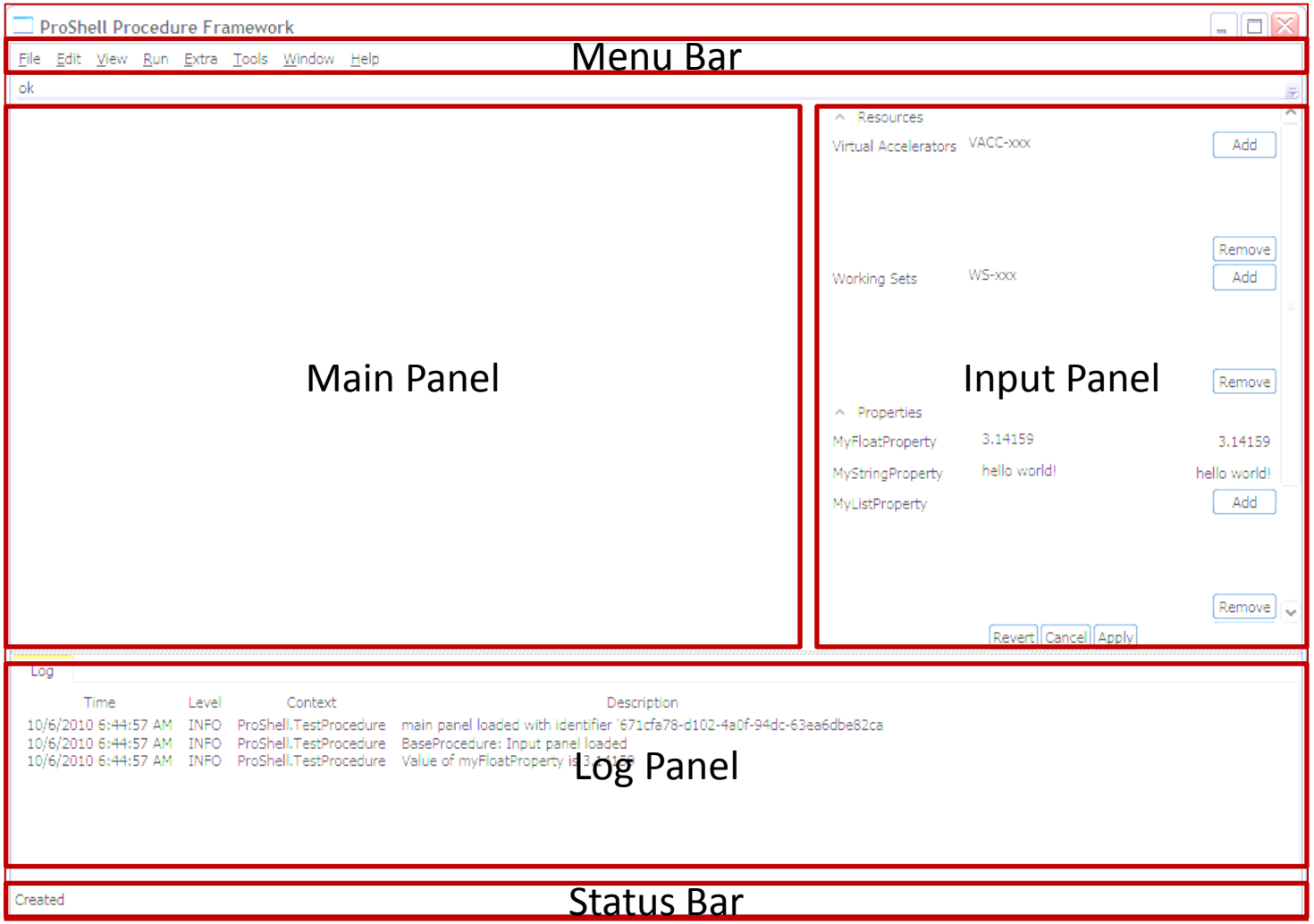
Architecture



Screenshot


ProShell Procedure Framework
File Edit View Run Extra Tools Window Help
ok
Resources
Virtual Accelerators
VACC-xxx
Add
Remove
Working Sets
WS-xxx
Add
Remove
Properties
MyFloatProperty
3.14159
3.14159
MyStringProperty
hello world!
hello world!
MyListProperty
Add
Remove
Revert
Cancel
Apply
Log
Time
Level
Context
Description
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure main panel loaded with identifier '671cfa78-d102-4a0f-94dc-63ea6dbe82ca
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure BaseProcedure: Input panel loaded
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure Value of myFloatProperty is 3.14159
Created

Screenshot

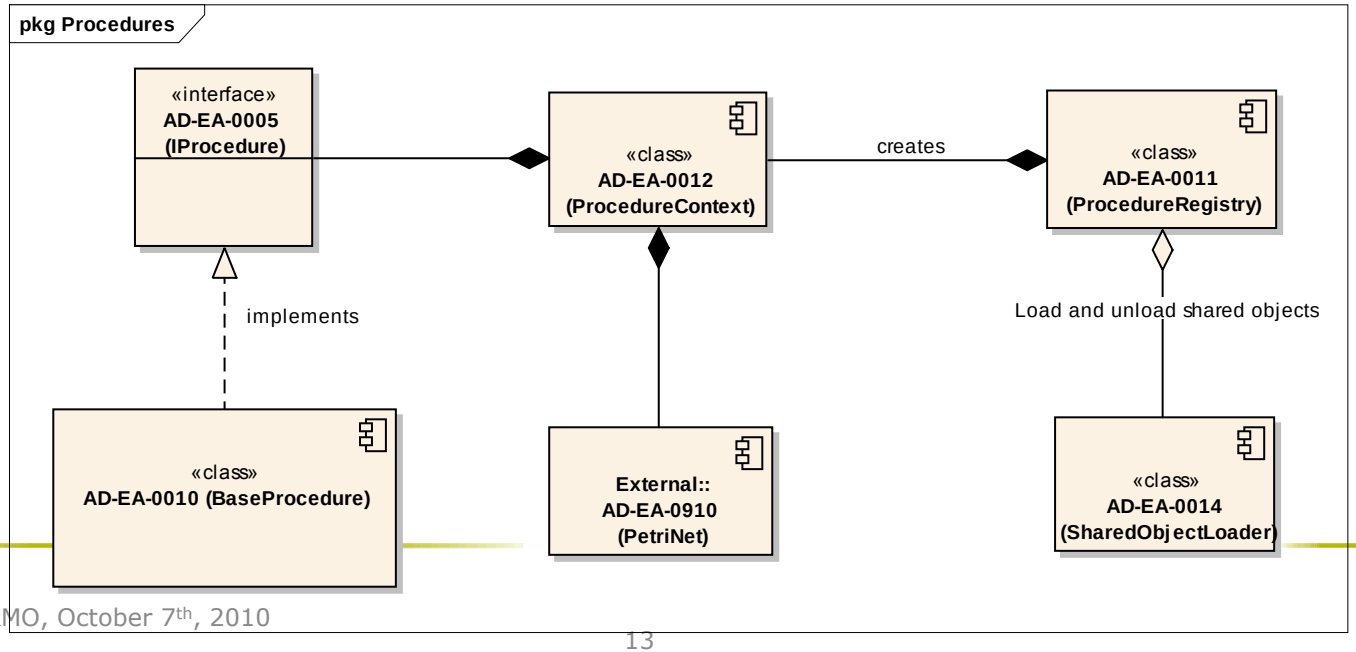

ProShell Procedure Framework
File Edit View Run Extra Tools Window Help
Menu Bar
ok
Main Panel
Resources
Virtual Accelerators
VACC-xxx
Add
Remove
Working Sets
WS-xxx
Add
Remove
Input Panel
Properties
MyFloatProperty
3.14159
3.14159
MyStringProperty
hello world!
hello world!
MyListProperty
Add
Remove
Revert
Cancel
Apply
Log
Time
Level
Context
Description
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure main panel loaded with identifier '671cfa78-d102-4a0f-94dc-63ea6dbe82ca
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure BaseProcedure: Input panel loaded
10/6/2010 6:44:57 AM INFO ProShell.TestProcedure Value of myFloatProperty is
Log Panel
Created
Status Bar

Procedure

- implements a **well-defined API** to perform a task
  - Example: „Emittance measurement“
- operates on **resources** (devices, WS, VAccs)
- inherits from BaseProcedure that provides default implementations for each function
  - Loading **configuration** parameters
  - Generating the **graphical user interface** and handling on changes
  - **Allocating** of VAccs and Working Sets
- may override default implementations to customize functions
- Handling of user events (button pressed)

ProcedureContext

- Manages a minimal set of data required by every procedure
  - User Interface Elements
  - Reading Configuration
  - Provide logging capabilities
  - A coloured PetriNet used for executing a procedure

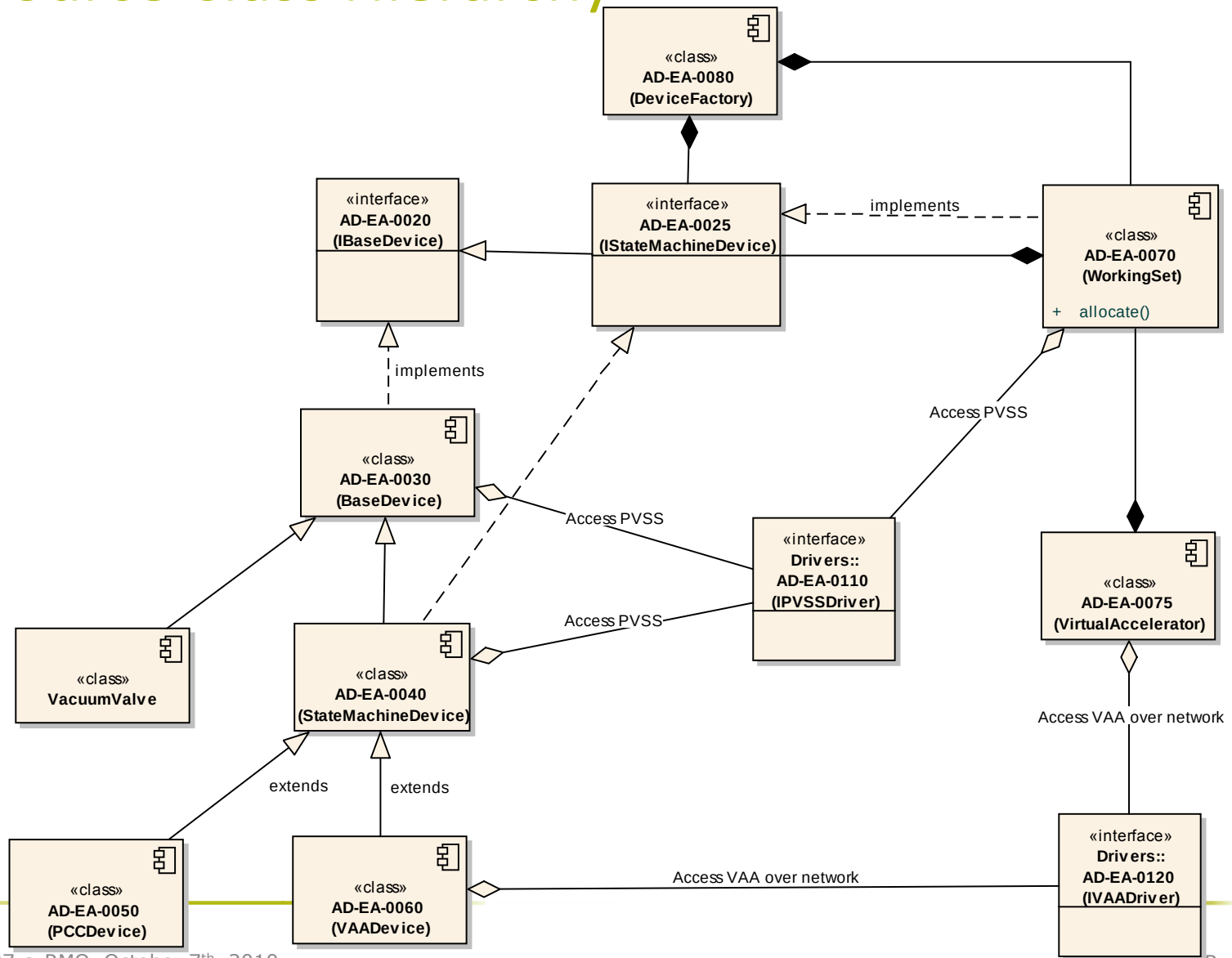


Resources

Procedures operate on the following types of resources

- Virtual Accelerators
  - List of working sets
- Working Sets
  - List of state machine devices
- **State Machine Devices**
  - May contain a list of base devices
  - E.g. Vacuum Control System for a sector
- **Base Devices**
  - E.g. Vacuum Valve

Resource Class Hierarchy

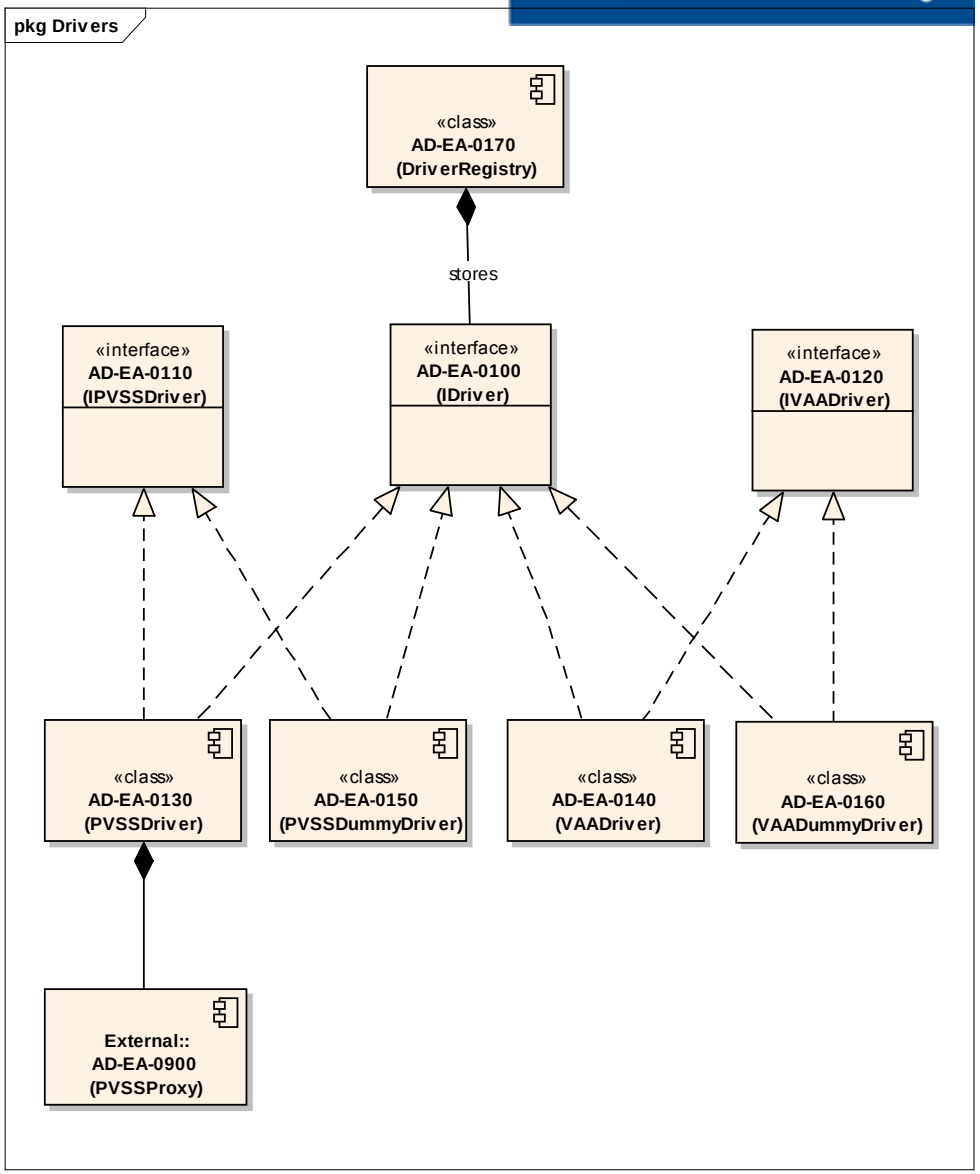


Devices

- **Base devices** provide an **API** to
  - Read default properties (e.g. Name)
- **State machine devices** provide an **API** to
  - control the state machine
  - change the mode
  - etc.
- **Additional device** types can be added with a **custom interface** that extends one of the previous APIs
  - API simplifies the source code in the procedures by not using PVSS dpGet/dpSet directly
  - Power converter: ground(), unground() functions
  - Beam stopper: moveIn() and MoveOut() functions

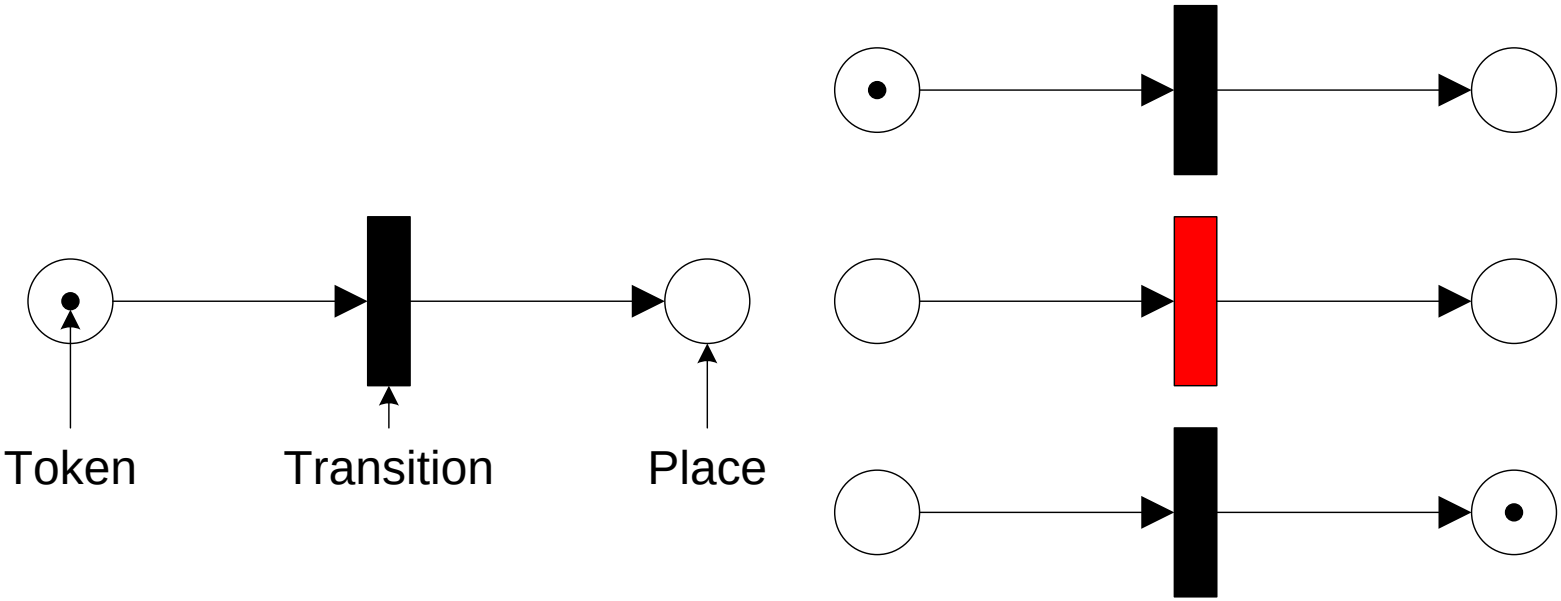
Drivers

- **Encapsulate** communication
  - E.g. PVSS, MTS, VAA, ...
- Two **types** of drivers
  - **Real drivers** communicate to the real system
  - **Dummy drivers** emulate the functionality (for testing without real system)



Petri-Nets

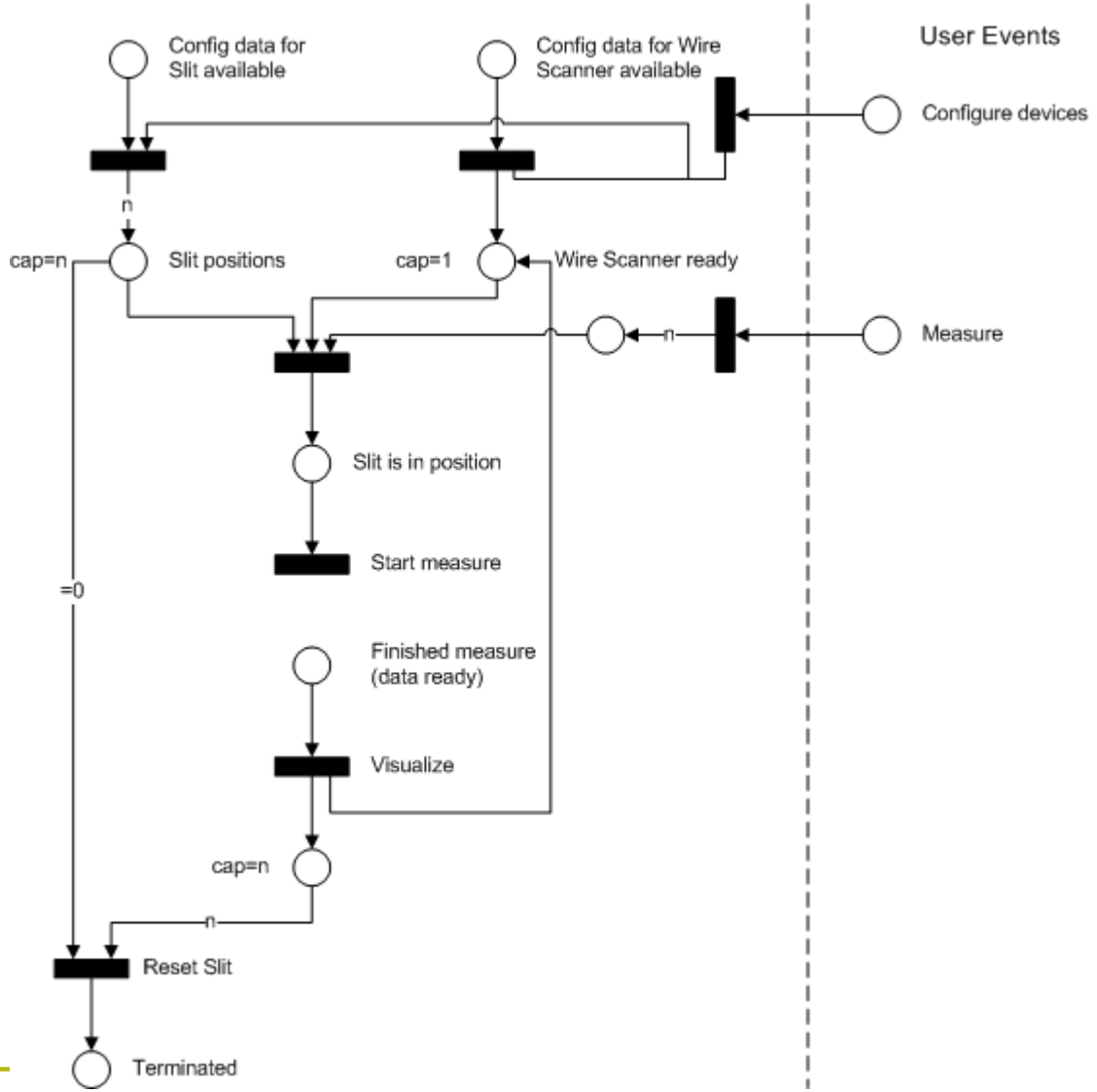
- A Petri net is a **mathematical model** language to describe a **distributed system**. It is a graph where nodes are either
  - **Transitions** represent events that may occur
  - **Places** represent conditions to fire transitions
  - Allows **parallel execution**



Coloured Petri-Nets

- Procedures use **Coloured-Petri Nets** where
  - Tokens may carry additional information (**coloured**)
  - **Callbacks** are attached to **transitions** and are executed synchronously
  - **Places** can be filled on **user action** (pressing a button) or **programmatically** (e.g. Asynchronous measurement finished)
  - Petri nets can be executed step-by-step (e.g. Debugging) or started where the net runs until its terminal condition is satisfied
  - **Parameterization** of the petri net

Example: Emittance Measurement



STATUS

Current Status

- First draft of ProShell **Enterprise Architect Model** contains
  - Initial Requirements gathering
  - Initial Architecture and Design document
- **Current implementation** of ProShell contains
  - A first draft of the **procedure interface** including
  - Loading of **configurations**
  - Automatic **creation input panel** based on configuration
  - Handling **multiple procedures** concurrently
  - Initial **Petri Net** support (programmatic creation and execution)
  - Driver interface provided with **dummy drivers**
  - Resource hierarchy available for the **generic devices**
    - Custom device interfaces to be provided on demand

Plan till December 2010

- Working on the **Enterprise Architect Model**
  - **requirements**
  - **architecture** and design
- Provide an initial ProShell **skeleton**
  - **Procedure interface** finalized
  - Eventually provide **integration** with **PVSS**
  - Dynamic loading of procedures (Cosylab)
  - Editor for Coloured PetriNets (Cosylab)

Summary

- Ahead of Time
  - None
- In-Time
  - ProShell implementation
- Behind schedule
  - ProShell requirements
  - ProShell architecture and design

Additional Information

- can be found on SVN
  - ES-100722-a-RMO ProShell Enterprise Architect Model
  - Source code can be found at /trunk/SCS/ProShell

Questions?

ADDITIONAL SLIDES