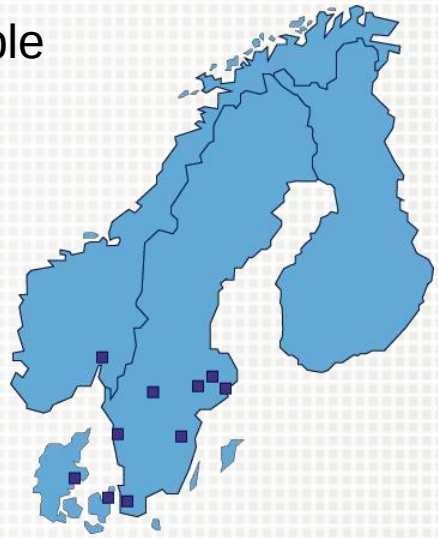
Andreas Kihlberg
Andreas.Kihlberg@Prevas.se
Prevas

Committed to your success

- Founded in 1985, >20 years experience of delivering profitable solutions to clients.
- Approximately 550 employees
- Listed on the Stockholm Stock Exchange since 1998.

- Certified according to ISO 9001:2000. Using well documented and tested project and development models.



World leading customers

ABB
ALK-Abelló
Arla
AstraZeneca
Atlas Copco
Autoliv
Biacore
Biovitrum
Bombardier
Danfoss
Domstein-Enghav
Ericsson
Findus
Gambro
GE Healthcare
Haldex
ICA
Kanthal
Maquet
Nokia
Novo Nordisk
Nycomed
Oticon
Pfizer
SAAB
Sandvik
Sanmina-SCI
SCA
Scania
Sectra
Siemens
SSAB
Stoneridge
Texas Instruments
Vattenfall
Vestas
Viasat
Volvo
Westinghouse

Prevas

NATIONAL INSTRUMENTS
Select Alliance Partner

Plugin Architectures

Customizable application with plug-in architecture

Jonas Mellroth
Jonas.Mellroth@Prevas.se
Prevas

What is a plugin?

From Wikipedia:

In computing, a plug-in (also called plugin, addin, add-in, addon, add-on, snap-in or snapin, but see also extension) consists of a computer program that interacts with a host application (a web browser or an email client, for example) to provide a certain, usually very specific, function "on demand". Add-on is often considered the general term comprising plug-ins, extensions, and themes as subcategories.

- A plugin is a set of methods that are dynamically loaded into a (binary) application.
- The plugins expands the functionallity and/or customizies the user interface.

Why use Plugins

- Easier to maintain.
  - Many applications can use the same base.
  - The basics can be more or less static for a long time, since the specific functionallity sits in the plugins.
- Reduces cost
  - The main difference between projects will be the contents of the plugins, and many plugins might be reused between projects.
- Easy to customize
  - Customers can customize the application to meet their specific needs
- Protecting IPs
  - Company IPs are protected in a binary application, while still allowing plugins to benfit from them through a well defined interface.

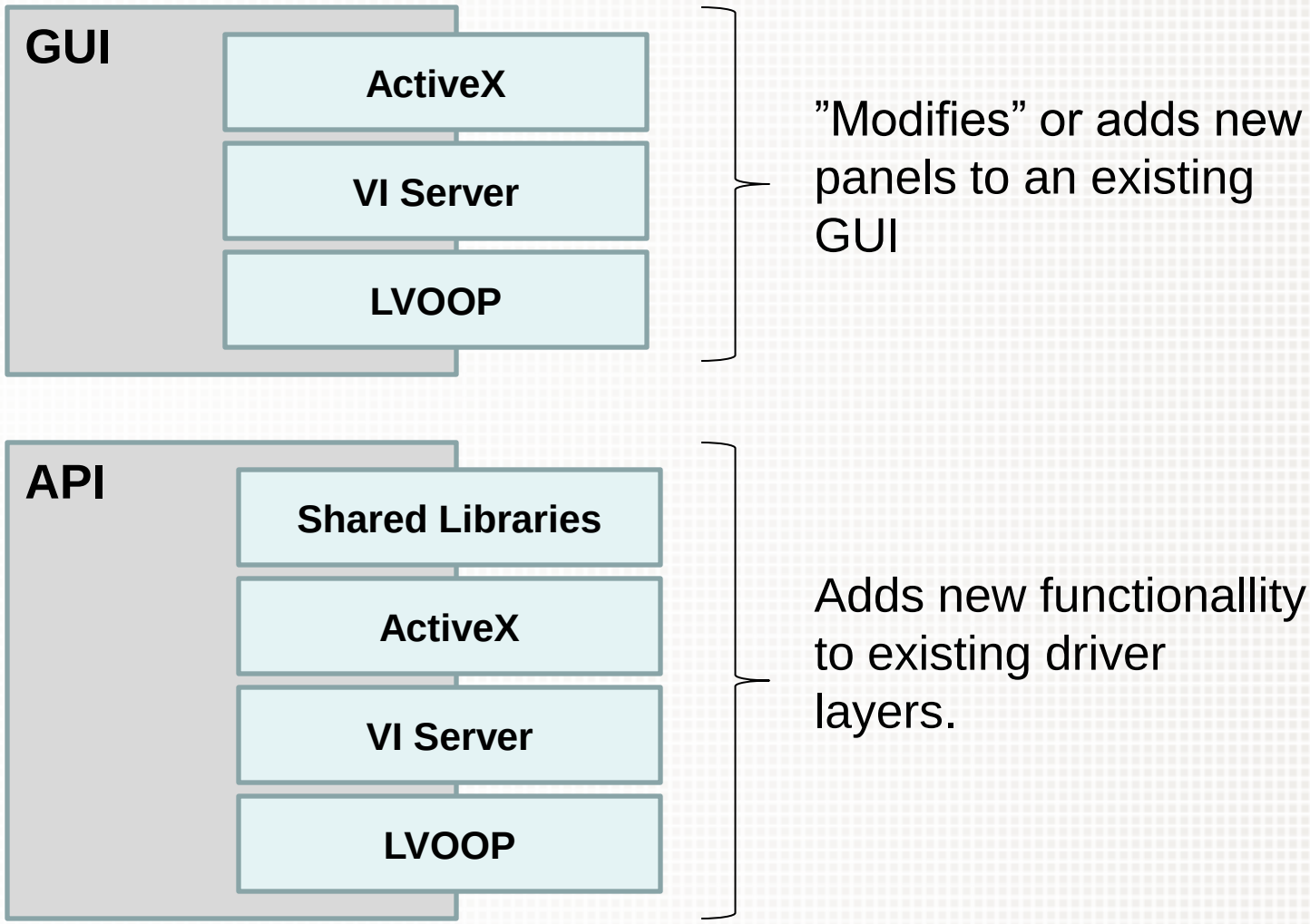
Prevas

NATIONAL INSTRUMENTS
Select Alliance Partner

Plugin types for LabVIEW

- Shared Libraries
- ActiveX
- VI Server
- LVOOP

LabVIEW Plugins



Shared Libraries

Pros

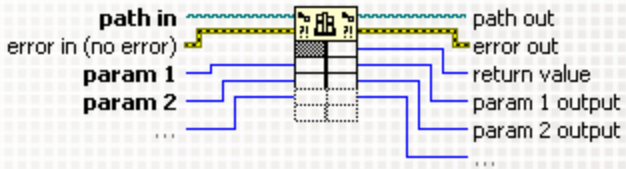
- Support plugins from many other program languages
- Many instrument drivers are delivered as DLL's

Cons

- LabVIEW Dll's are hard-linked to a specific LabVIEW version
- Dll's written in other languages require other environments.
- OS specific
- No Plugin Panel support
- No type checking if the function prototype changes.
  (only runtime error)

Note:

In LabVIEW < 8.5 the DLL had to be loaded from start, and could not be unloaded



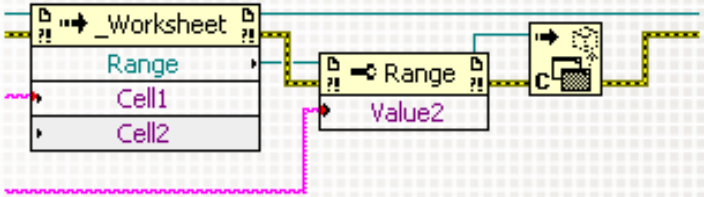
ActiveX

Pros

- Support plugins from other program languages
- Can add panel views from the plugin using ActiveX containers
- Supported by ”most” major Windows applications.

Cons

- Require other environments.
- Not full control of memory loading/unloading.
- Can be difficult to know what methods are needed, and in what order.



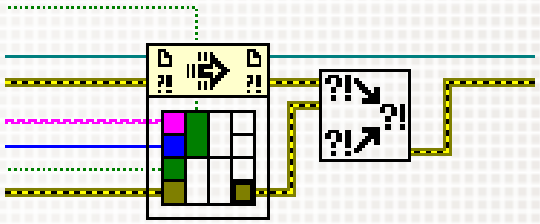
VI Server

Pros

- Custom UI Panels can easily be loaded into the application.
- Available on more than the Windows platform.
- Works on LabVIEW RT, even between targets.

Cons

- More difficult to debug
- Inline operations are difficult. i.e. difficult to perform operations on data without making copies.



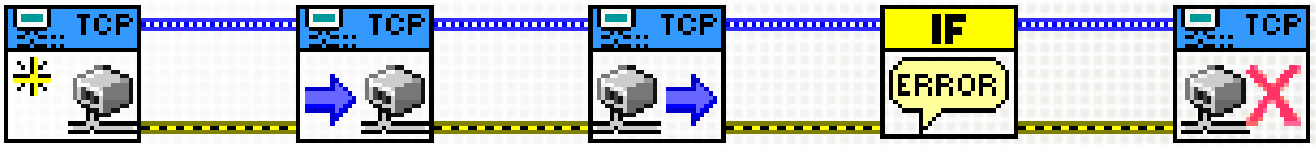
LVOOP

Pros

- Methods can overlay existing methods easily. i.e. the calls to the interface methods are automatically “replaced” with the plugin methods
- Easy to make inline operations on data
- True parallel execution
- Edit time type checking
- Available on more than the Windows platform, from LV2009 even on LV-RT
- Good protection of private items, (attribute and methods)

Cons

- Can be hard to unload a lvclass plugin from memory.
- Not always easy to debug, e.g. reentrant methods

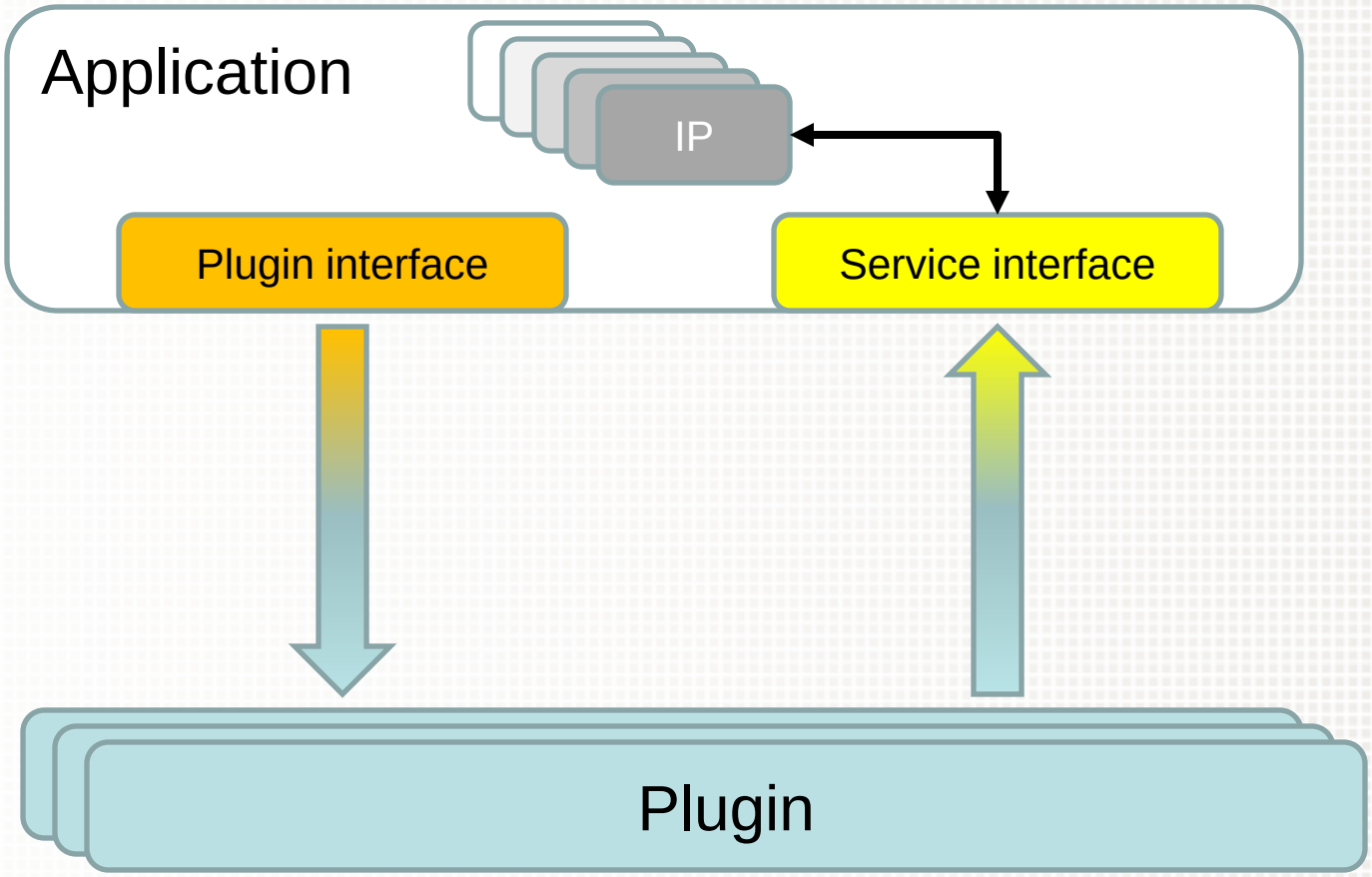


LVOOP vs. VI Server

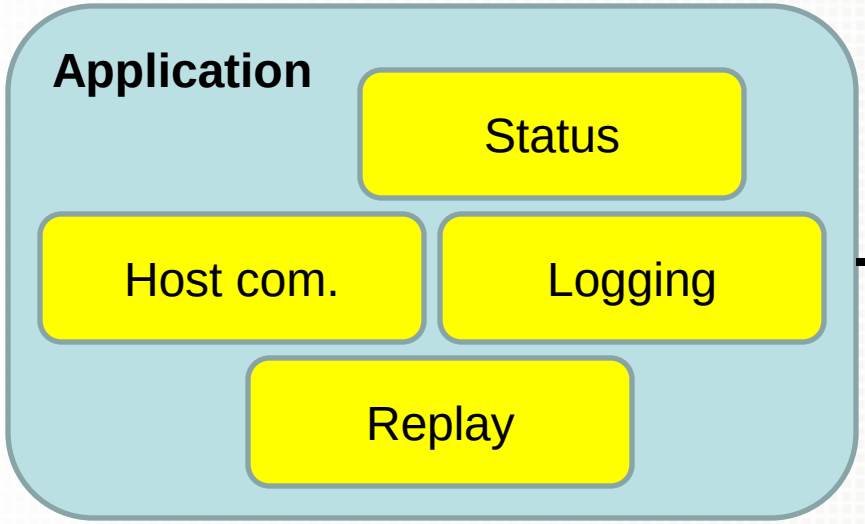
some differences

Action	LVOOP	VI-Server (call by reference node)
Skipped interface method	Load the plugin as normal, once called it will use the parent implementation instead	Must be handled as the plugin is loaded.
Missing required plugin method	Tracked at edit time by using a property in the lvclass.	This is handled at run-time, and should return a File not exist error.
Reentrant method	All plugins must have the same setting (reentrant or not) for a method, but the actual calls are reentrant as with any other VI call.	To get true parallel execution we would have to load the method N-times to allow N parallel calls. Plus we will have to handle which reference to be used.
VI prototype checking	If the plugin implementation differs from the interface implementation, the VI is not executable (at edit time).	Will be detected when the VI is loaded.
Loading the plugin into the application	Only loads the .lvclass file with the method ”Get LV Class Default Value”	Each method must be loaded with a separate call to Open-VI-Reference, and multiple times for reentrant methods.

Plugin architecture



Creating an application



Plugin interface

Develop the plugins needed for the final application.

- DAQmx
- ScanEngine
- CAN
- RS232
- RS485

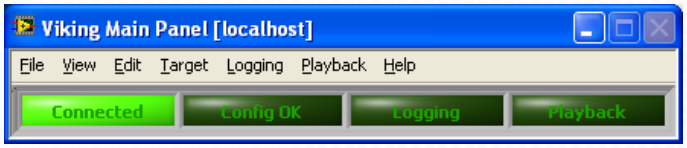
First decide what should be part of the core application, e.g. Logging, Replay etc.

Then define the plugin interfaces needed to add plugins to the application.

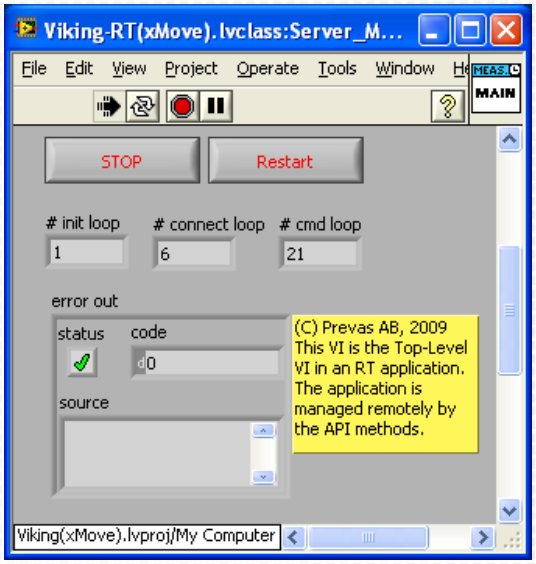
Example

RT control and monitor software

Host application



Measurement server



Out-of the box

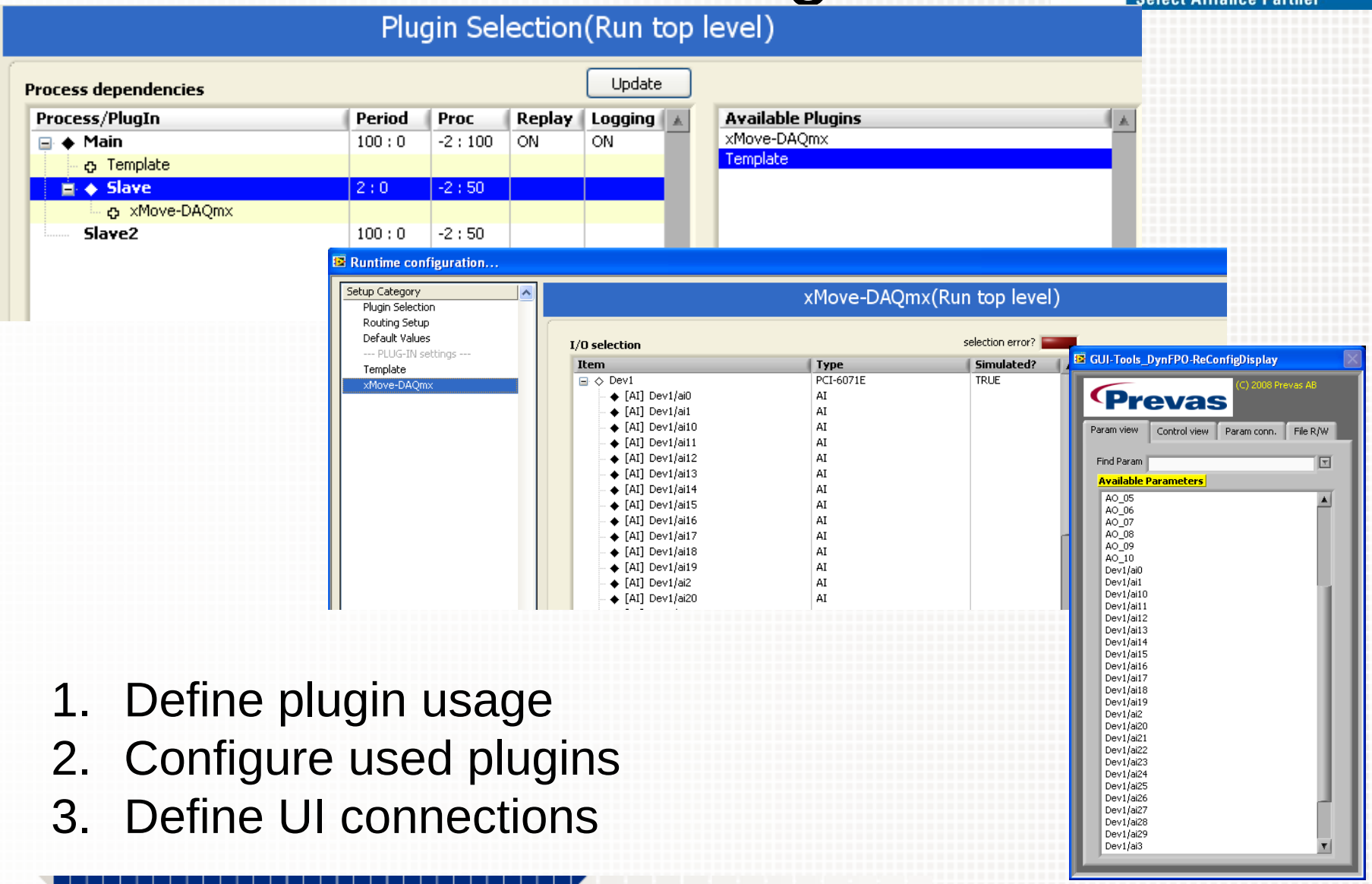
- Host-Server communication
- Logging & Playback
- Plugin configuration
- Possible to save configuration to disk.

Example cont.

Plugins

- The platform does not support any HW in the basic application. Instead it is supposed to load plugins for all HW necessary for the current application.
- Plugins are realized as a set of LabVIEW classes, and have one server part and one GUI part.

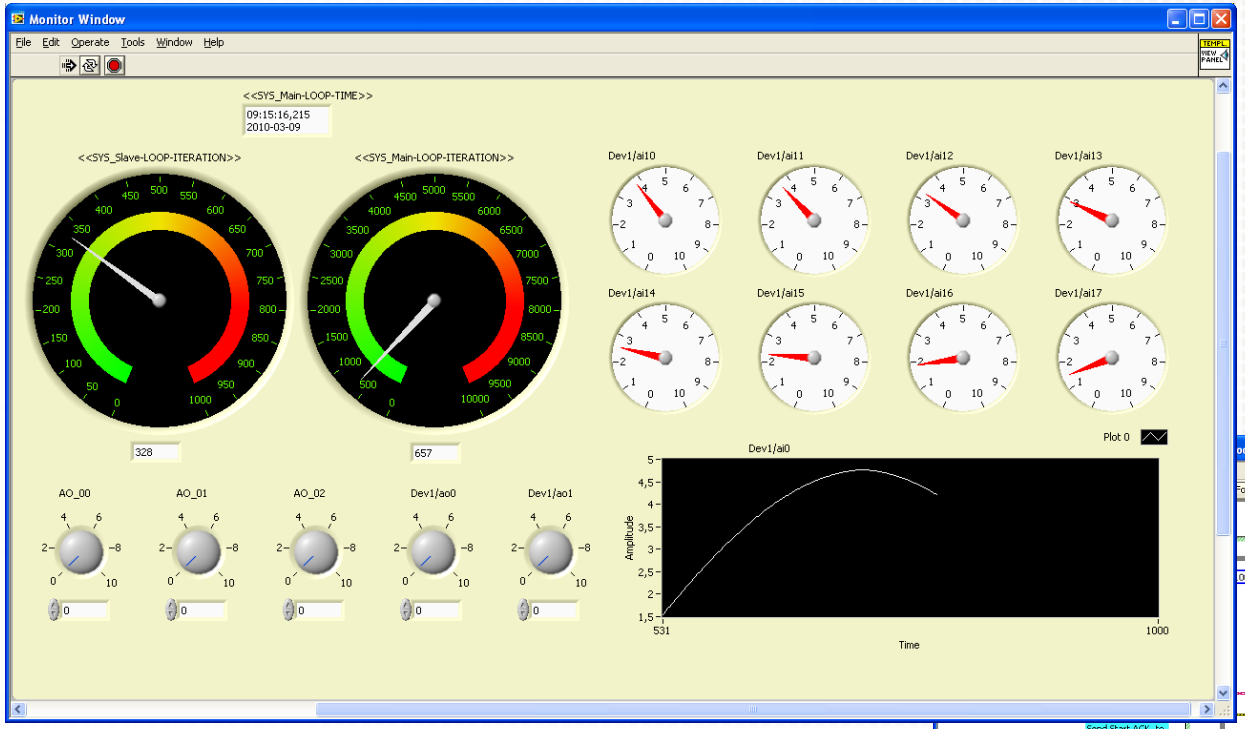
Runtime configuration

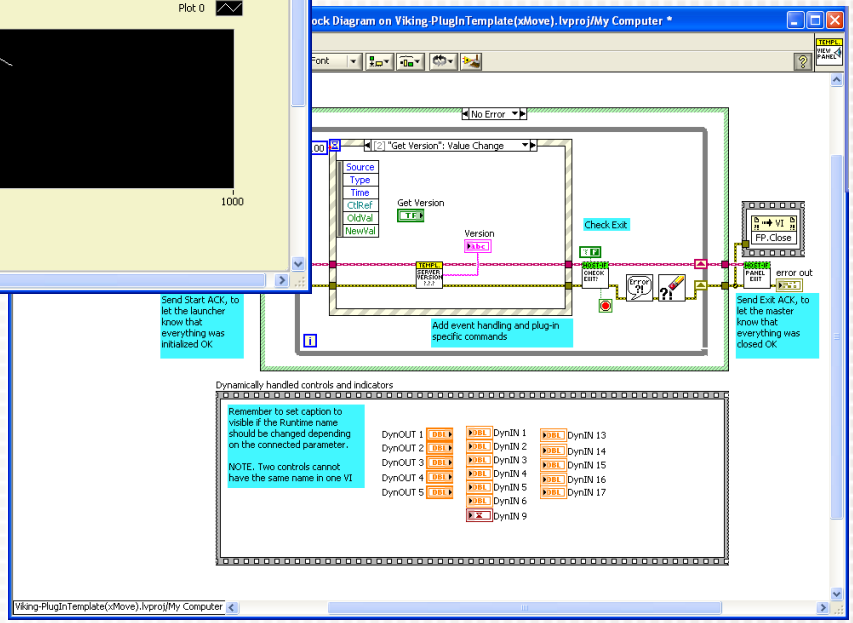


1. Define plugin usage
2. Configure used plugins
3. Define UI connections

Prevas Runtime-ViewPanel

NATIONAL INSTRUMENTS™
Select Alliance Partner





Questions

Thanks

Jonas Mellroth
Jonas.Mellroth@prevas.se
Prevas