Prevas

Plugin Arcitectures

Customizable application with plug-in architecture

Hans Nyström
Hans.nystrom@prevas.se
Prevas

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.

Prevas

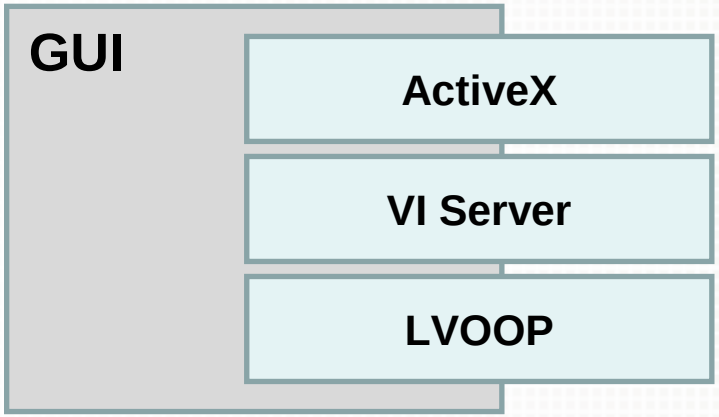
Why use Plugins

- 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.

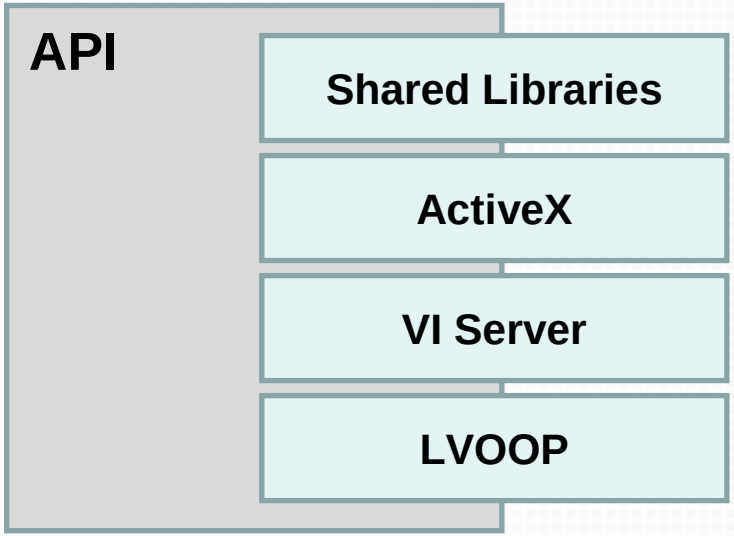
Plugin types for LabVIEW

- Shared Libraries
- ActiveX
- VI Server
- LVOOP

LabVIEW Plugins



”Modifies” or adds new panels to an existing GUI



Adds new functionallity to existing driver layers.

Shared Libraries

Pros

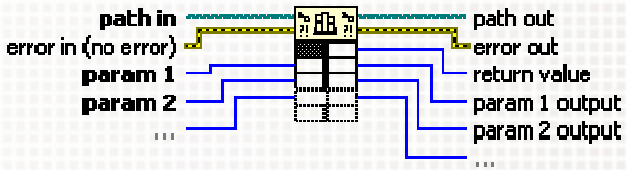
- Support plugins from many other program languages
- Many instrument drivers are delivered as DLL's

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



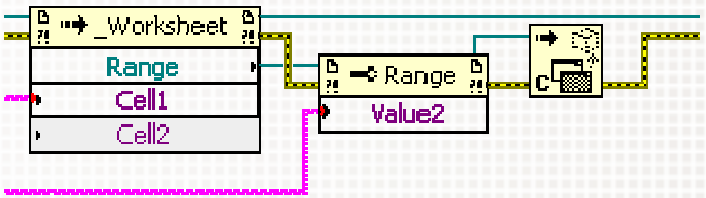
ActiveX

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.



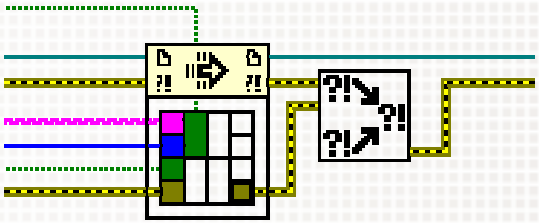
VI Server

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.



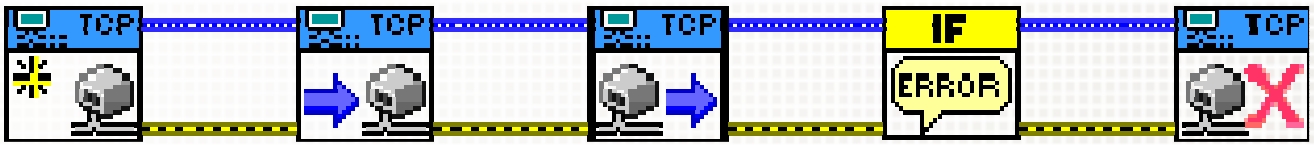
LVOOP

Pros

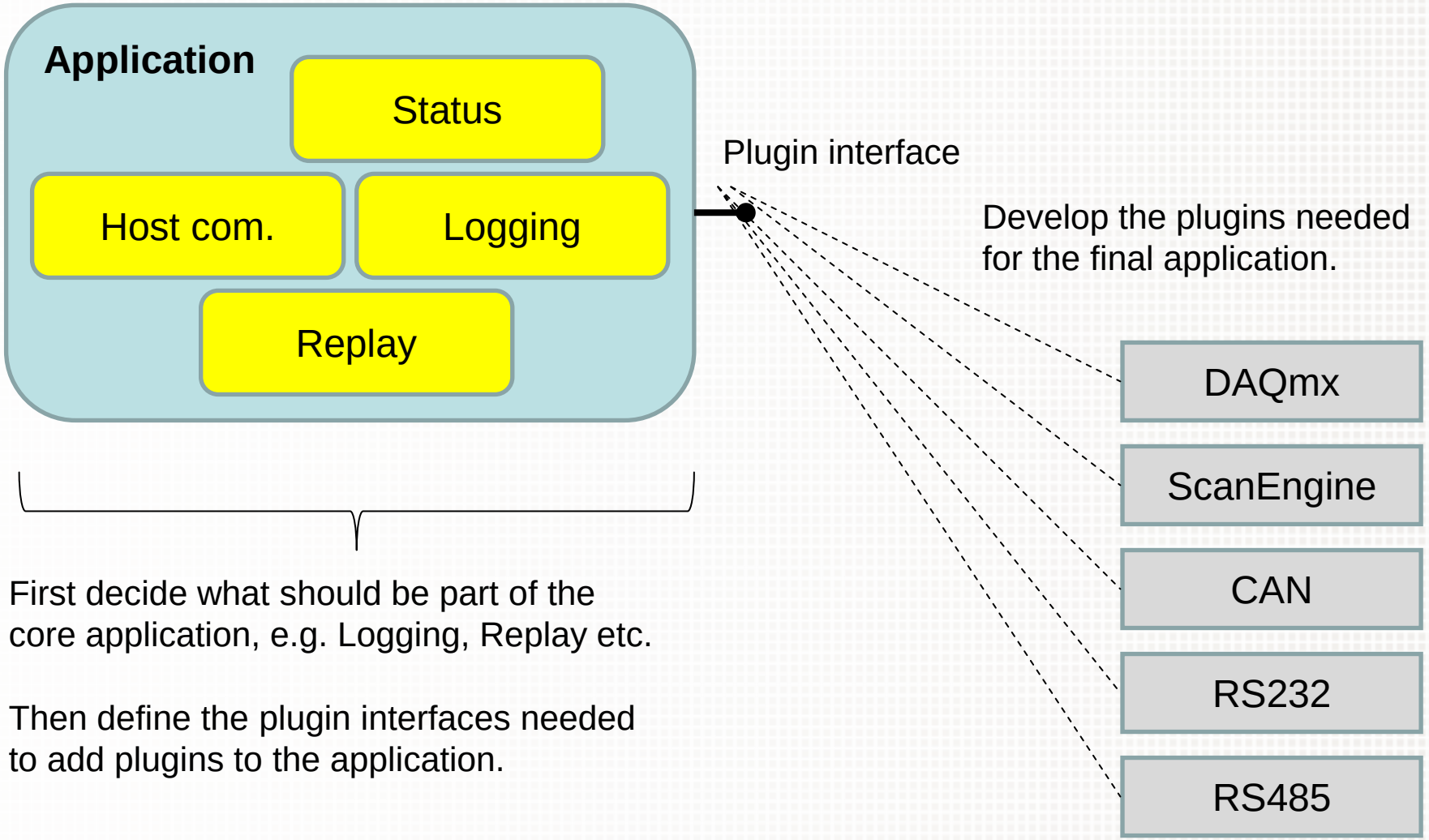
- Methods can overlay existing methods easily. i.e. the calls to the interface methods are automatically “replaced” with the plugin methods
- Easy to make inline operations on data
- True parallel execution
- Edit time type checking
- Available on more than the Windows platform, from LV2009 even on LV-RT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



Creating an application



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

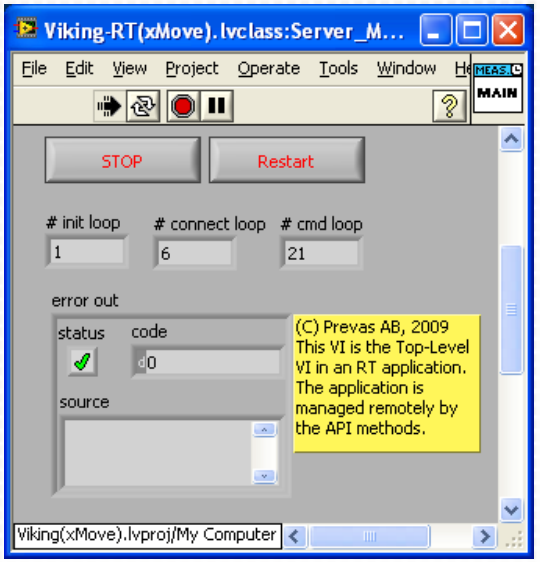
Then define the plugin interfaces needed to add plugins to the application.

Example

RT control and monitor software

Host application

Meas server



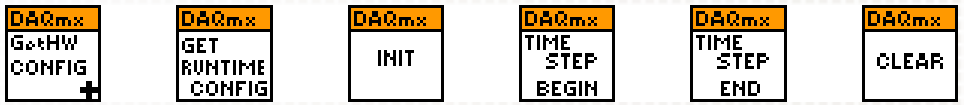
Out-of the box

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

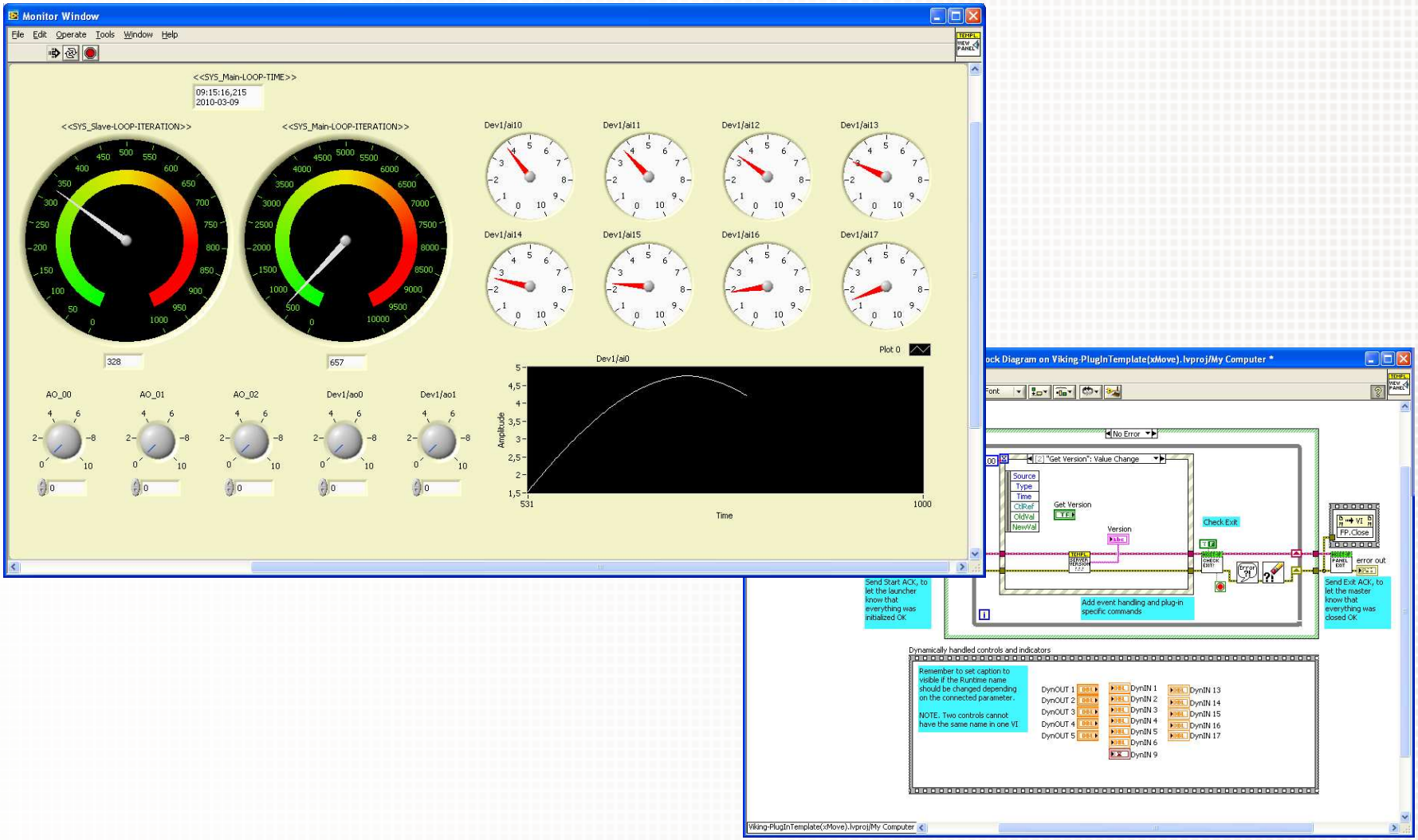
Example cont.

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-ViewPanel



Questions

Thanks

Hans Nyström

Hans.nystrom@prevas.se

Prevas