

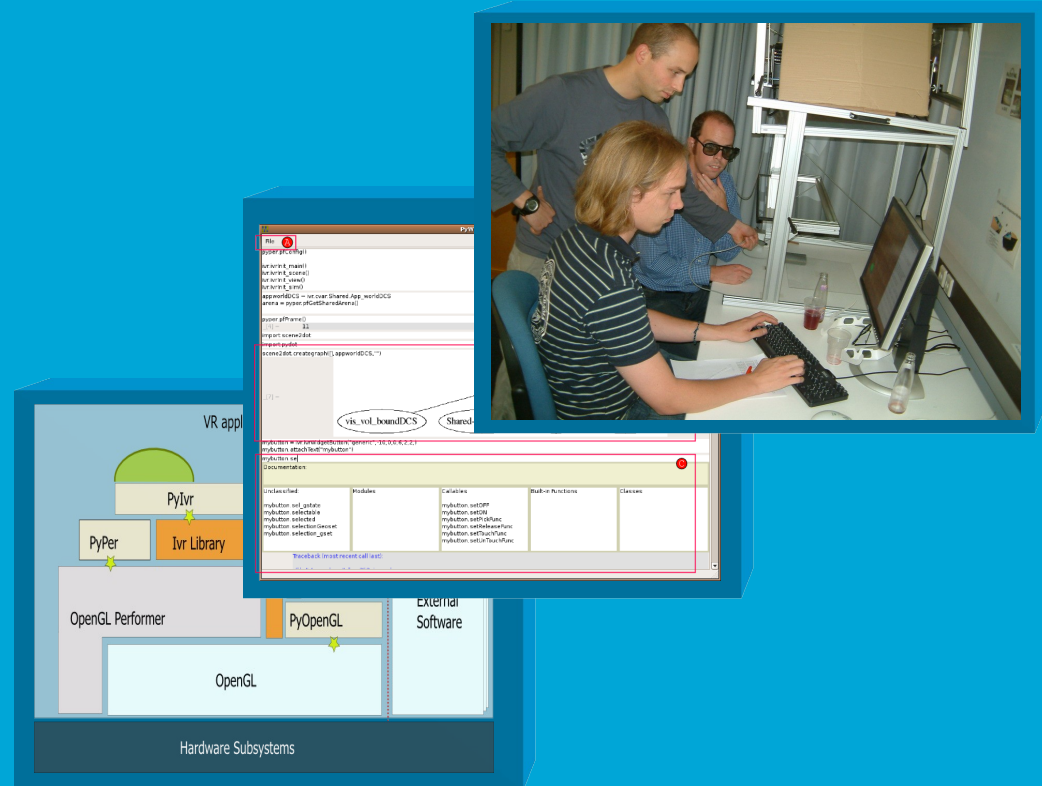
# Flexible Abstraction Layers for VR Application Development

G. de Haan,  
M. Koutek,  
F.H. Post

IEEEVR 2007, Charlotte NC, USA

Gerwin de Haan

July 27, 2007



# VR Applications development, context

- VR for Visualization: Interactive Data Exploration



July 27, 2007

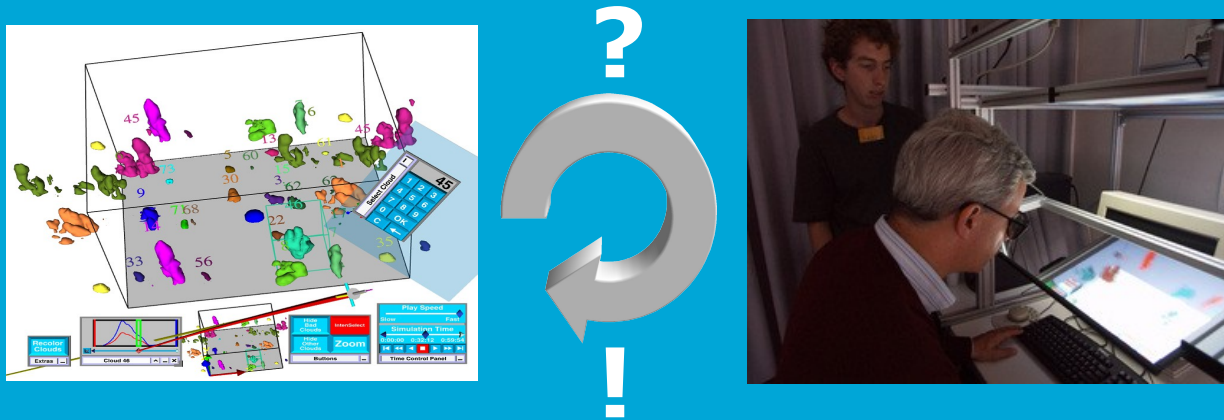
2

G. de Haan, M. Koutek, F.H. Post

“Flexible Abstraction Layers for VR Application Development”

# VR Application development, problem

- **Custom product** (niche market, special purpose)
- **Highly complex** (real-time, interactive, 3D, large data)
- **End-user involvement** (continuous, on-line sessions)



**process should match problem:  
be agile, prototype often**

July 27, 2007

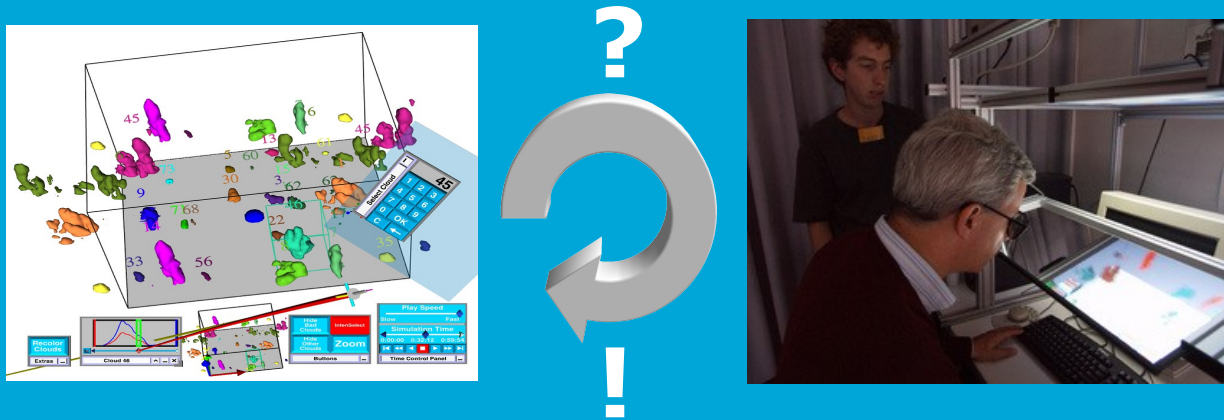
3

G. de Haan, M. Koutek, F.H. Post

“Flexible Abstraction Layers for VR Application Development”

# VR Application development, issues

- lack of **standards** and structure, common **toolchain**
- complex, high perf. **C++** where it's not needed



**Progress and maintenance ?  
Just Another Demo**

July 27, 2007

4

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# VR Application development, related

- Related disciplines (Math, Distr. Comp., Visualization)
  - Frameworks and APIs
  - Scripting support
  - Domain Specific Languages
  - Problem Solving Environments

**Flexibility vs. Rigor**  
**General vs. Special Purpose**

# VR Application development, goal

## Flexibility for VR software life-cycle

- integration, configuration of VR components
- integration of external software libraries
- continuous, iterative development
- evolution from prototype to "end-user" app
- "end-user" ease-of-use, programming

**"End-user is... you"**

# VR Application development, paradigm

“flexible abstraction layers”

build many simple layers on top of complex layers

- abstract often -> **be flexible**
- learn by trying -> **prototype**
- don't reinvent -> **integrate**
- single abstraction language -> **unify**

**prototyping environment**

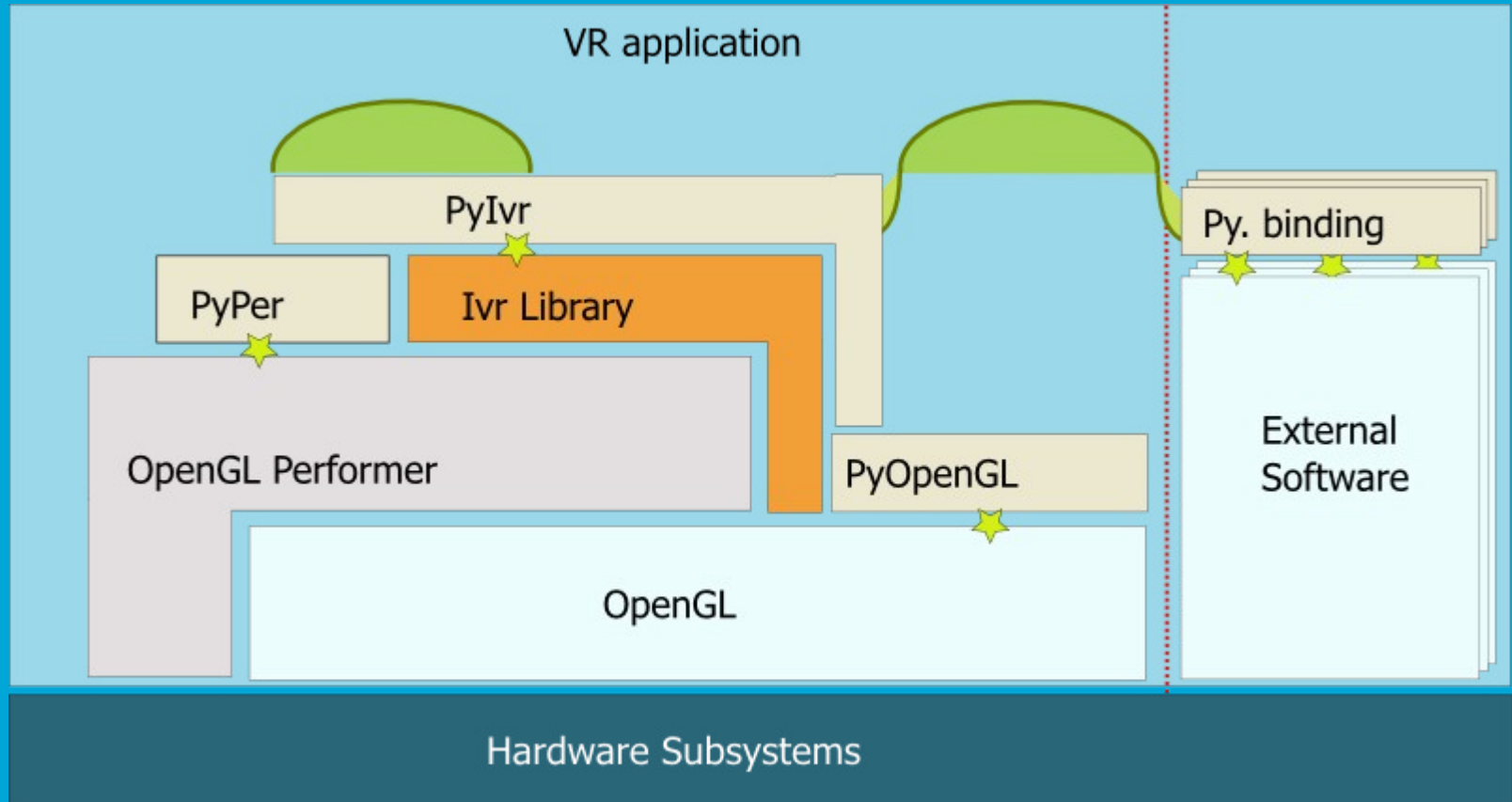
July 27, 2007

7

G. de Haan, M. Koutek, F.H. Post

“Flexible Abstraction Layers for VR Application Development”

# Prototype description



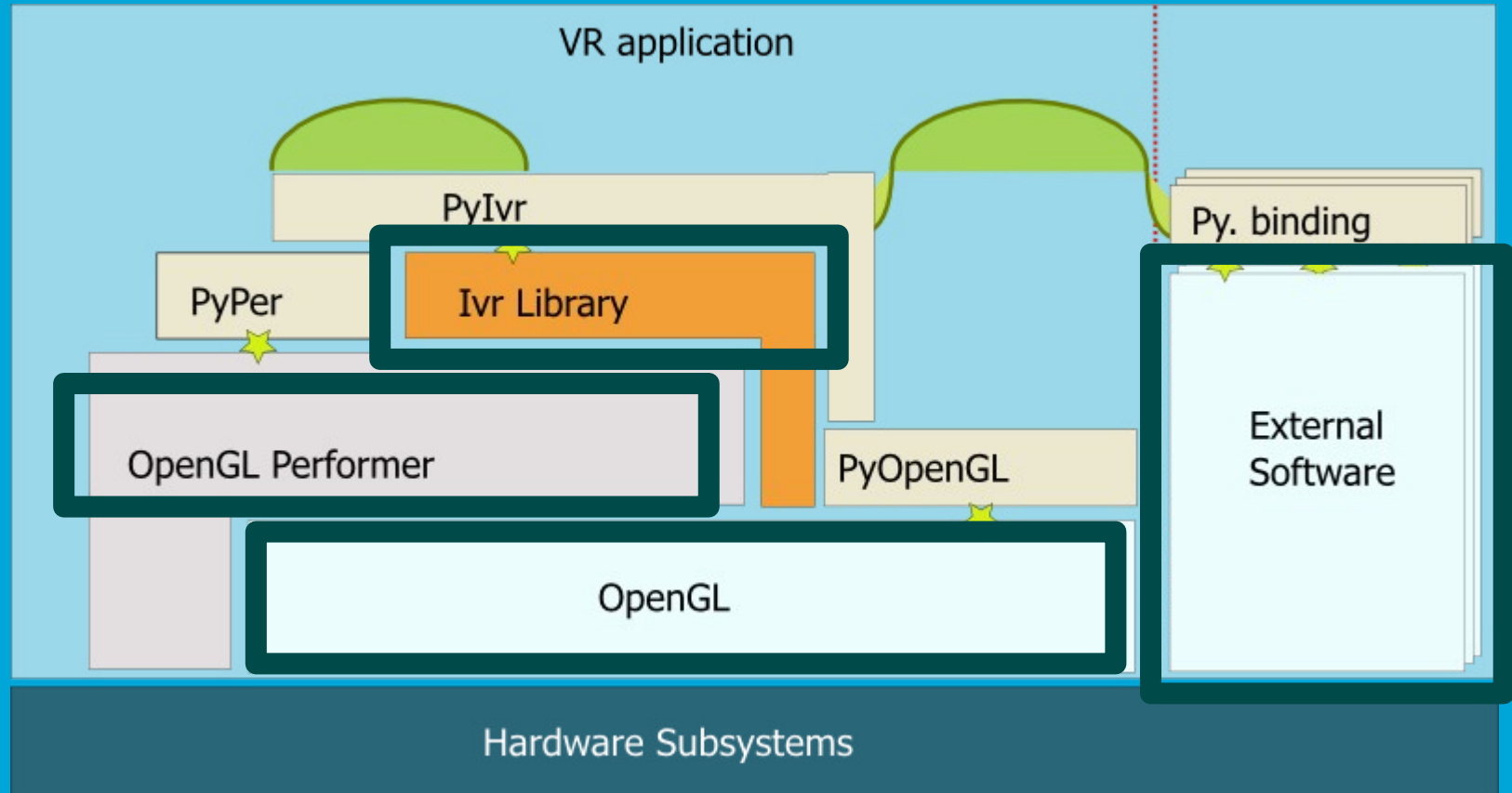
July 27, 2007

8

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Prototype description: C++ libraries



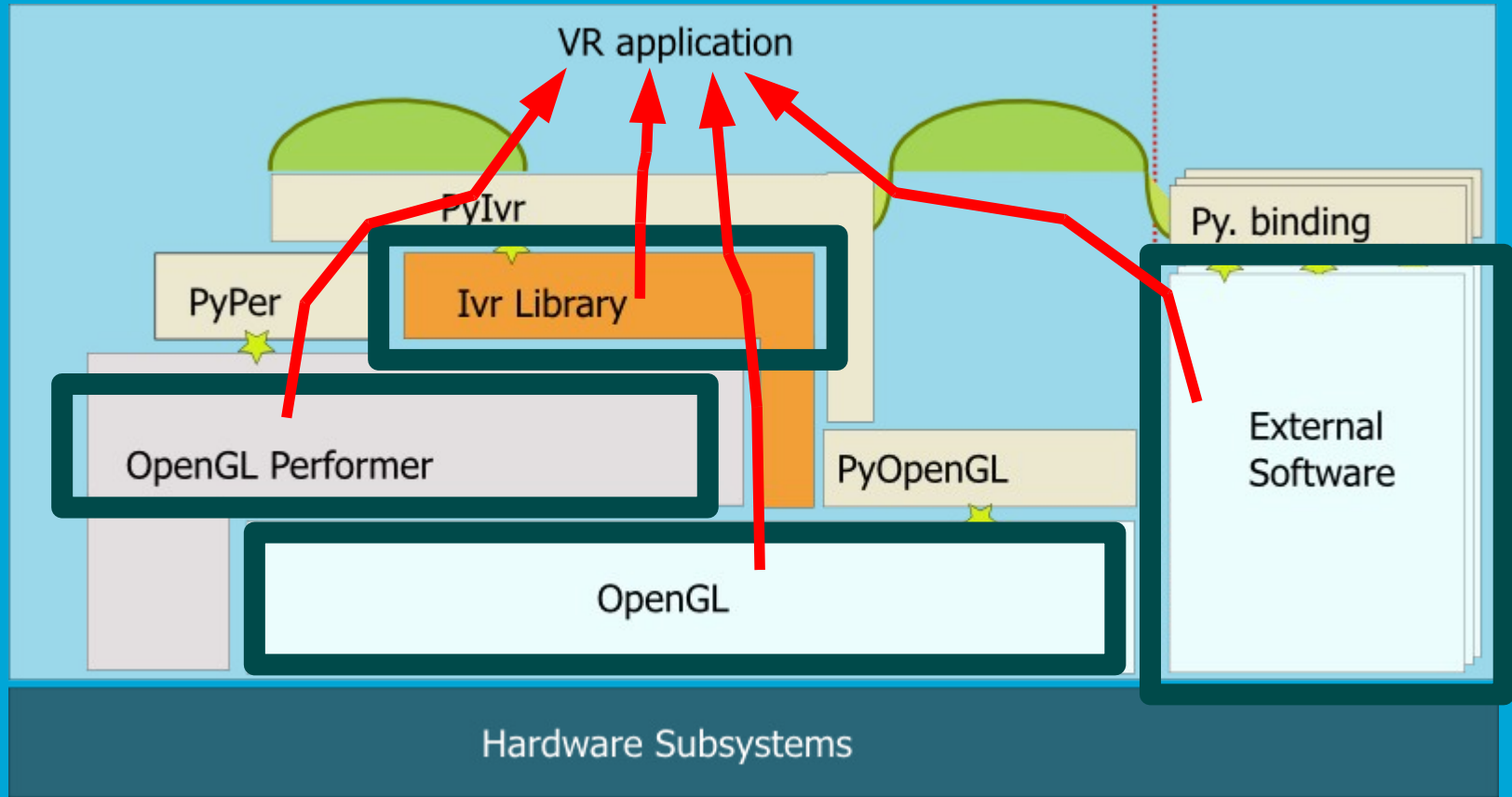
July 27, 2007

9

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Prototype description: C++ App



July 27, 2007

10

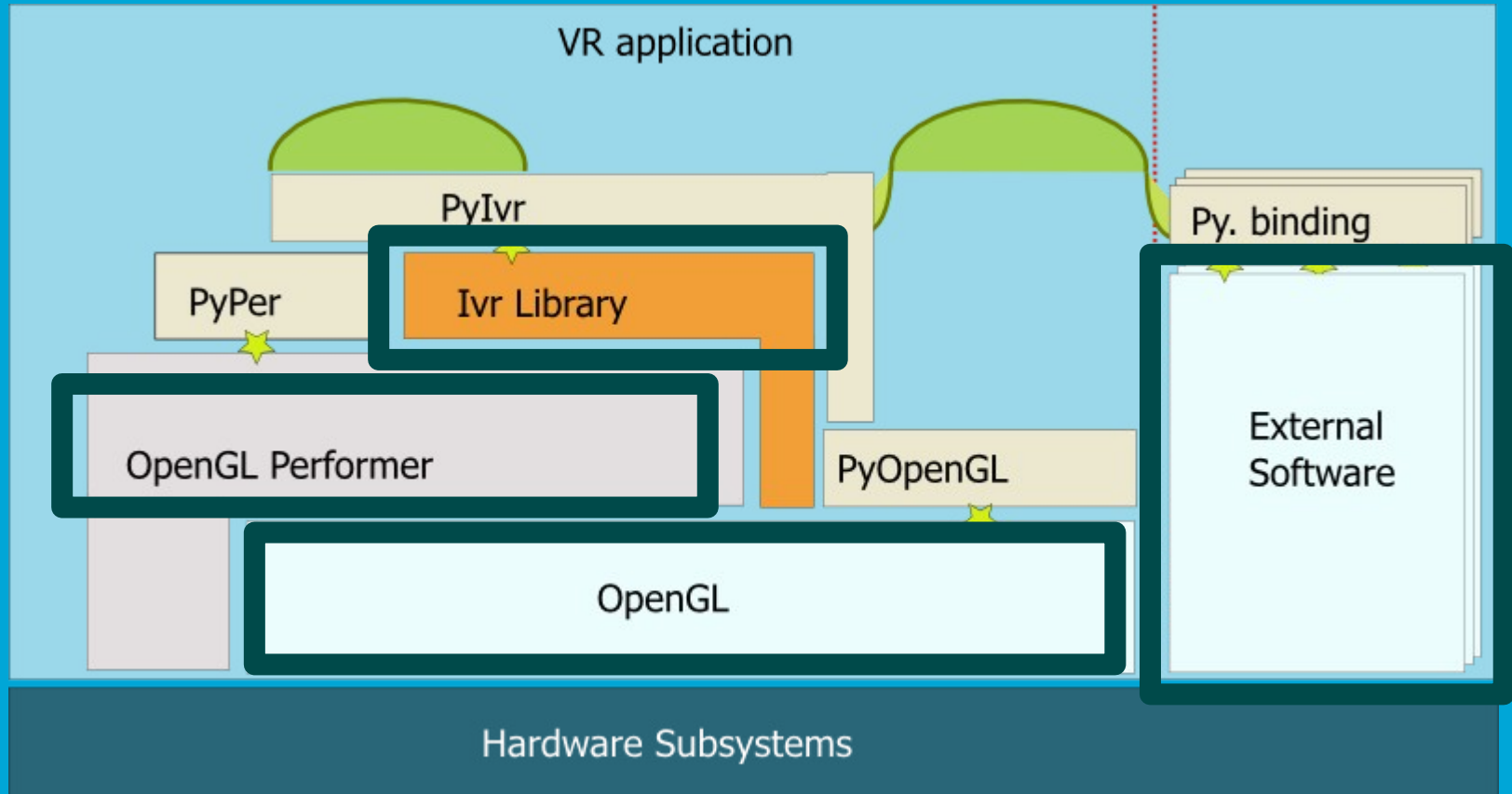
G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Single Abstraction Language, Python

- **Technical point of view**
  - object-oriented, high-level data structures
  - dynamic typing, dynamic binding
  - plays nice with C++
- **User's point of view**
  - around since 1991, active user community
  - popular language, even in science
  - readable “pseudo-” code, easy to use

# Prototype description: C++



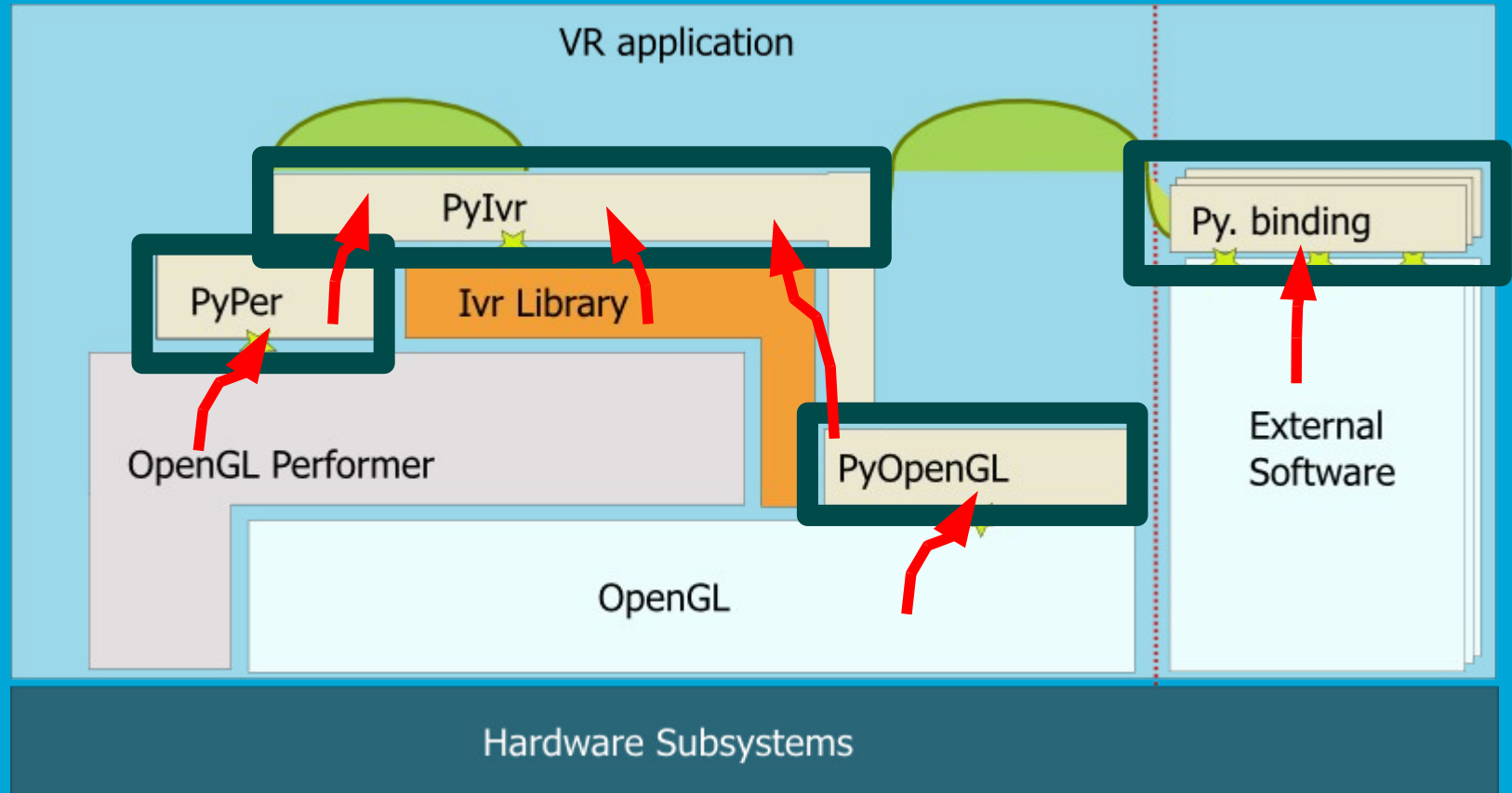
July 27, 2007

12

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Prototype description: to Python



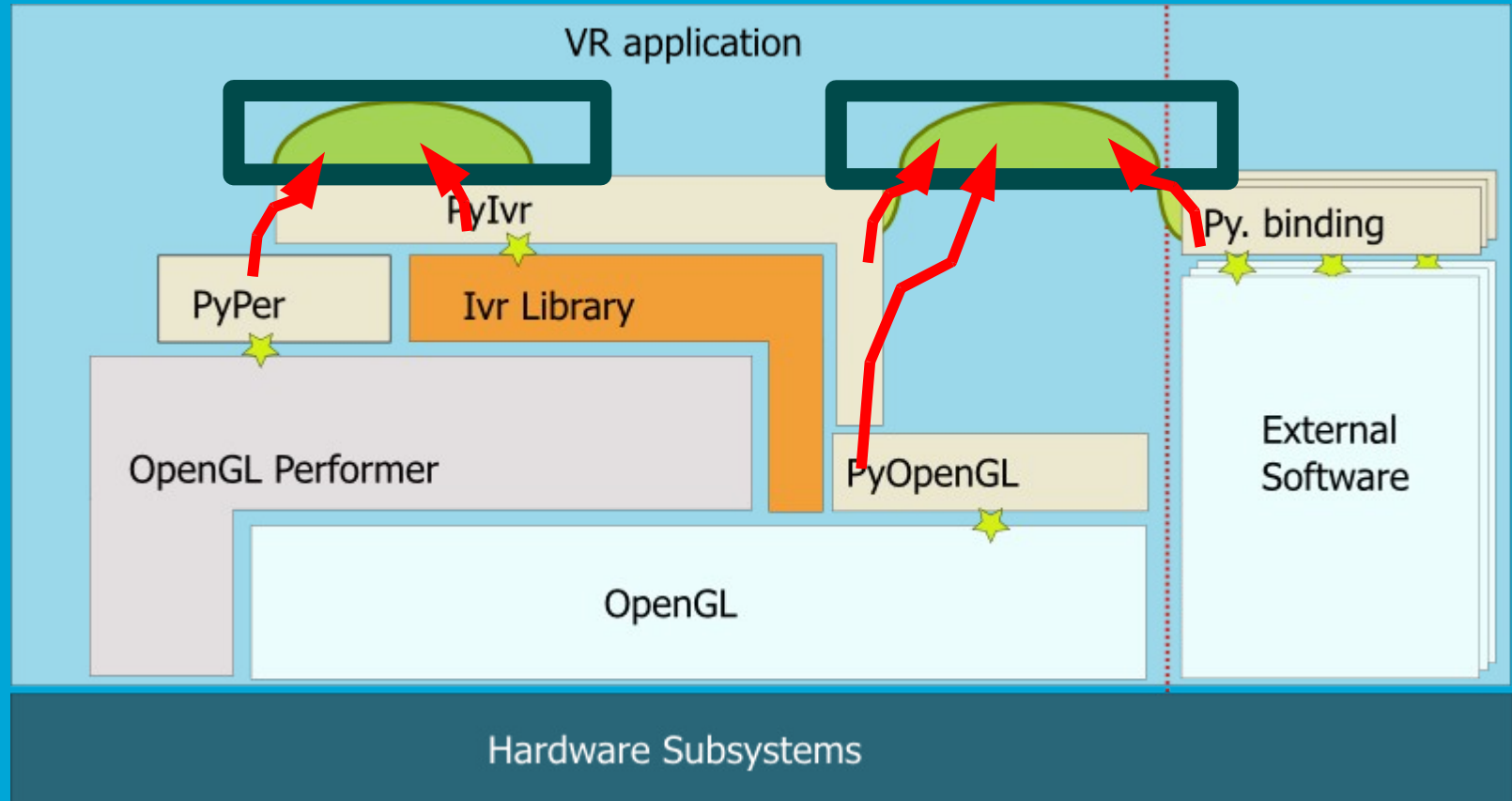
July 27, 2007

13

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Prototype description: more abstraction



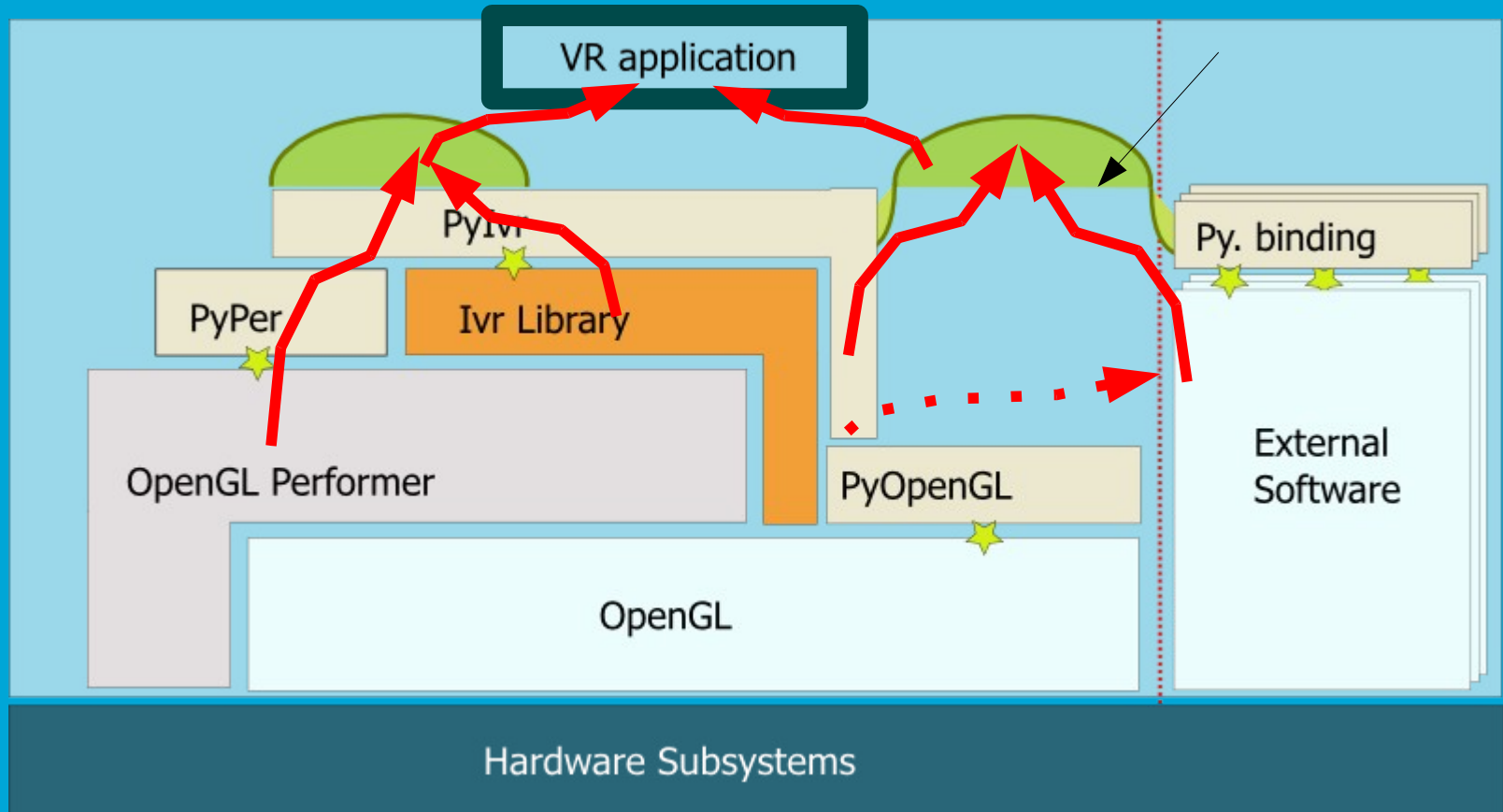
July 27, 2007

14

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Prototype description: Python app



July 27, 2007

15

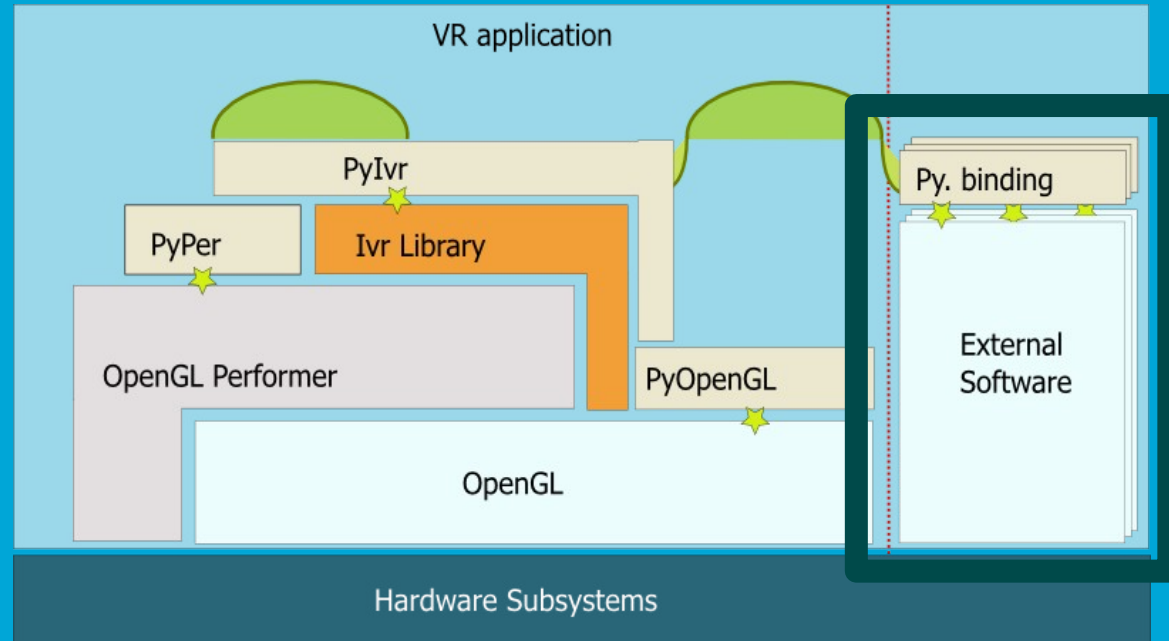
G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# App. Example

- Python layers,
- Python script

```
import vtk
import vtk2vr
import vr
```



```
dataset = vtk.makeIsoActor("dataset.dat")
3dobj = vtk2vr.makeModel(dataset)
slider = vr.makeSlider(0,1,3dobj.setIso)

vr.mainloop()
```

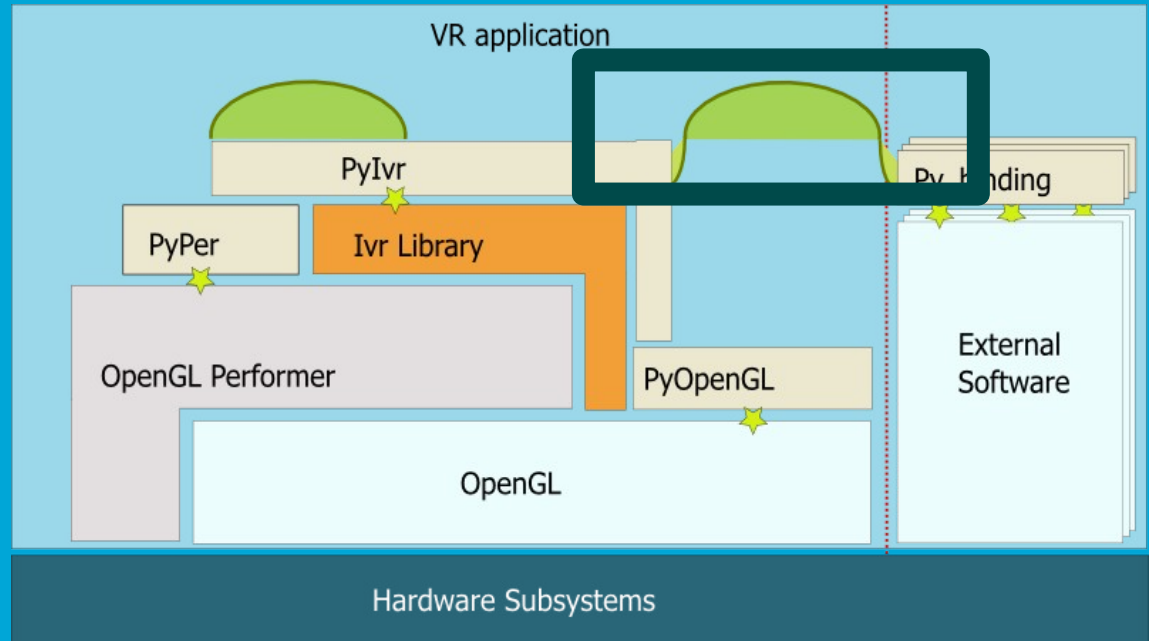
# App. Example

- Python layers,
- Python script

```
import vtk
import vtk2vr
import vr
```

```
dataset = vtk.makeIsoActor("dataset.dat")
3dobj = vtk2vr.makeModel(dataset)
slider = vr.makeSlider(0,1,3dobj.setIso)

vr.mainloop()
```



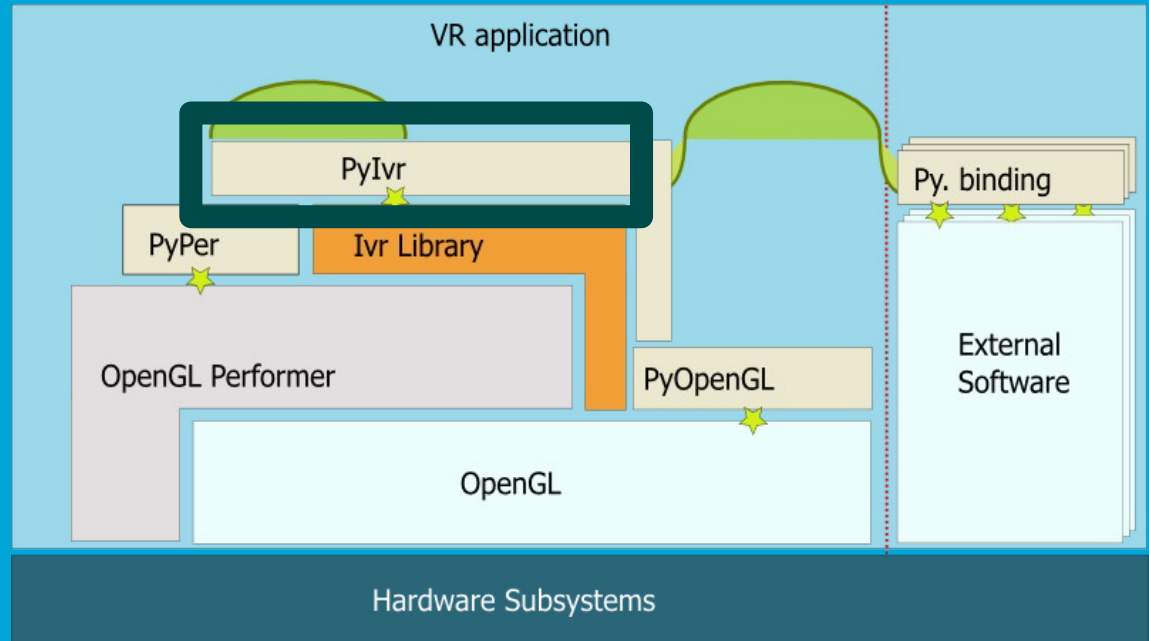
# App. Example

- Python layers,
- Python script

```
import vtk
import vtk2vr
import vr
```

```
dataset = vtk.makeIsoActor("dataset.dat")
3dobj = vtk2vr.makeModel(dataset)
slider = vr.makeSlider(0, 1, 3dobj.setIso)
```

```
vr.mainloop()
```



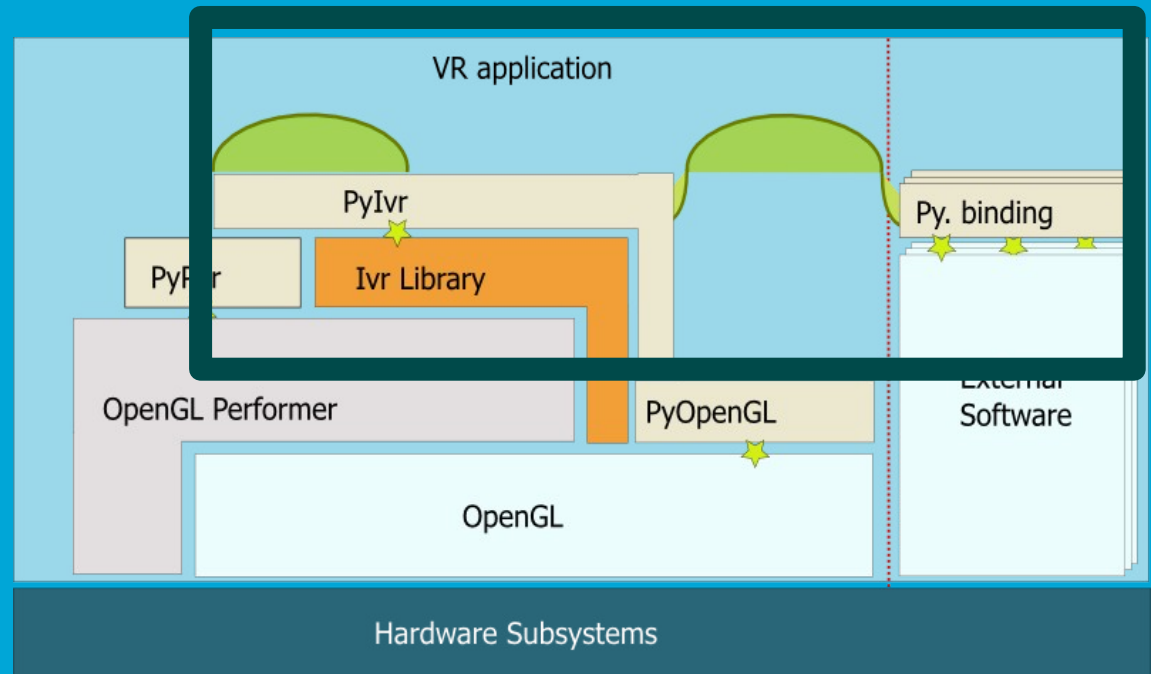
# App. Example

- Python layers,
- Python script

```
import vtk
import vtk2vr
import vr
```

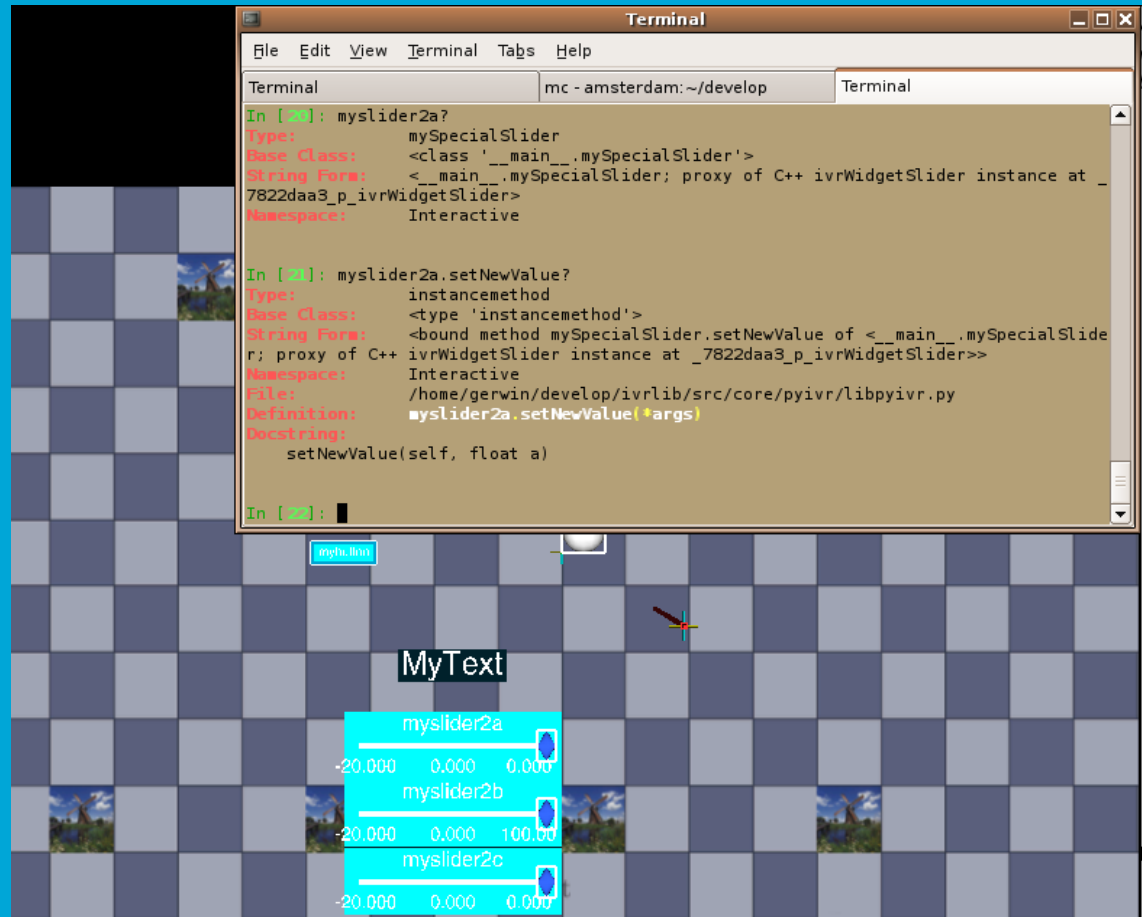
```
dataset = vtk.makeIsoActor("dataset.dat")
3dobj = vtk2vr.makeModel(dataset)
slider = vr.makeSlider(0,1,3dobj.setIso)
```

```
vr.mainloop()
```



# Environment: Run-time Shell

- CLI : **iPython**
- runs concurrent
- debugging
- documentation
- comm. completion
- comm. history
- configurable



July 27, 2007

20

G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Results, extending VR library internals

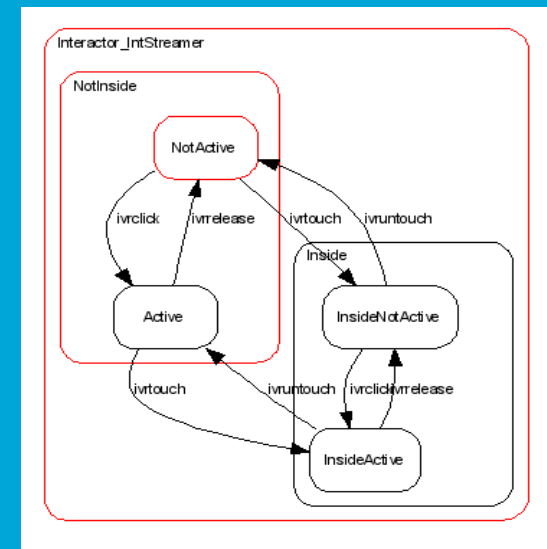
- **Customize and extend C++**

- graphical widgets
- interaction settings

```
class mySlider(ivr.ivrWidgetSlider):  
    def SliderExtensionFunc(self):  
        print "Value=", self.getValue()  
  
myslidero = mySlider("Iso-value", worldDCS)
```

- **Interaction mechanism redesign**

- override current VR library behavior
- prototype before/during design
- **gradual** transition



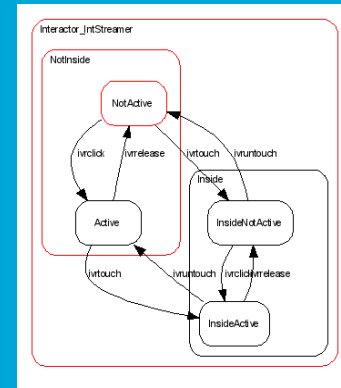
# Results, mixing with external software

## Our most used

- VTK, ODE, Graphviz, Matplotlib, R, pySerial

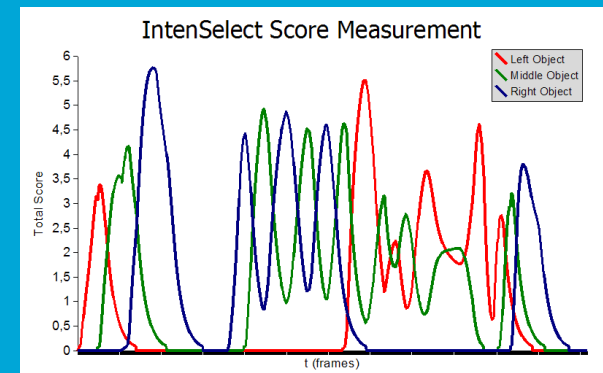
## Others of (y)our interest

- Serialization (e.g. Blender, Cal3d, XML), networking, databases, image handling, (GUIs)



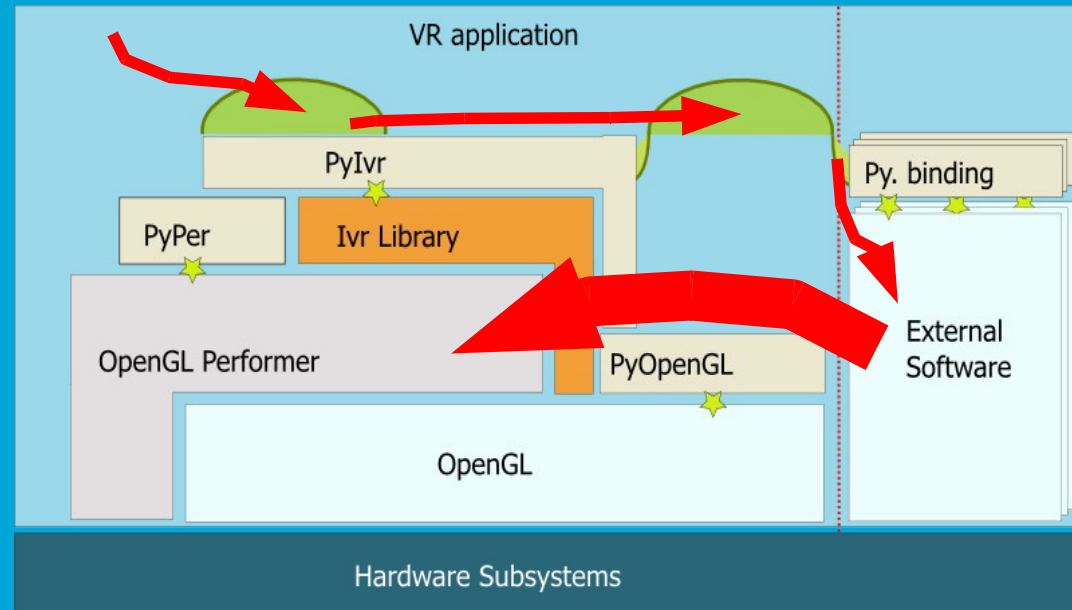
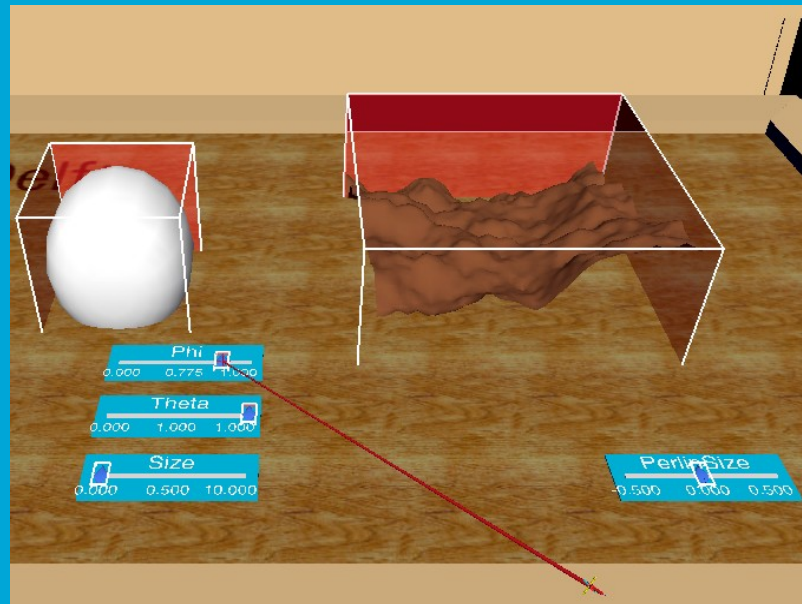
## Communication

- uni-directional
- bi-directional
- performance?



# Results, integrating VTK

- Visualisation ToolKit, dataflow
- bi-directional transfer
- Performance: wrapped C++ level glue



July 27, 2007

23

G. de Haan, M. Koutek, F.H. Post

“Flexible Abstraction Layers for VR Application Development”

# Results, recent use

- **Prototyping:** Students build interaction techniques
- **Interaction study:** from technique, tasks to analysis
- **Ease of Use:** Workshop, demonstration and hands-on VR



July 27, 2007

24

G. de Haan, M. Koutek, F.H. Post

“Flexible Abstraction Layers for VR Application Development”

# Technical Issues: wrapping layers

- **Mix-and-Match software components**
  - display: scene graph
  - interaction mechanisms
  - external libraries
- **Abstraction layer generation:**
  - Wrapping: manual, *SWIG*, *Boost.Python*, *ctypes*
  - data types, incompatibilities
  - cross-language functionality
  - see <http://visualisation.tudelft.nl/VRdev>

# Beyond wrapping: abstracting more

- **Technical point of view**
  - embedding vs. extending (who's in charge?)
  - mix and match data types
  - performance, where necessary
  - threading, memory
- **“End User's” point of view**
  - clarity and robustness
  - rethink use: handles, debug information
  - documentation

# Conclusions

- **Flexible Abstraction Layers:**
  - Many, “simple” code layers to unify in one language
  - Run-time development environment
- **Interactivity** for interactive graphics
- **Flexibility** in development cycles
- **Shift** in development effort (developers -> users)
- Call for more **robust** and **flexible** layers for libraries

Promising strategy for larger, “real” applications

# Ongoing and future work

## Prototype is basis for:

- interaction scenario development
- remote, distributed VR development
- “real” application development

## Prototype extensions:

- More and new VR components
- More simple, abstracted functionality
- Run-time development environments
- usability -> integration CLI with VR, GUI

# Run-time environment: Notebook

- *Mathematica*
- concept

The screenshot shows a PyWorksheet notebook with the following content:

```
File
pyper.pfConfig()
ivr.IvrInit_Main()
ivr.IvrInit_Scene()
ivr.IvrInit_View()
ivr.IvrInit_Sim()
appWorldDCS = Ivr.Cvar.Shared.App_worldDCS
arena = pyper.pfGetSharedArena()

pyper.pfFrame()
_[4] = 11
import scene2dot
import pydot
scene2dot.creategraph([], appWorldDCS, "")

_[7] =

mybutton = Ivr.IvrWidgetButton("generic",-10,0,0,6,2,2.)
mybutton.attachText("mybutton")
mybutton.se|
```

The tree diagram shows a root node **App\_worldDCS** with four children: **vis\_vol\_boundDCS**, **Shared->localDCS**, **None**, and **ground\_objDCS**.

The documentation table is as follows:

Unclassified:	Modules	Callables	Built-in Functions	Classes
mybutton.sel_gstate mybutton.selectable mybutton.selected mybutton.selectionGeoset mybutton.selection_gset		mybutton.setOFF mybutton.setON mybutton.setPickFunc mybutton.setReleaseFunc mybutton.setTouchFunc mybutton.setUnTouchFunc		

Traceback (most recent call last):

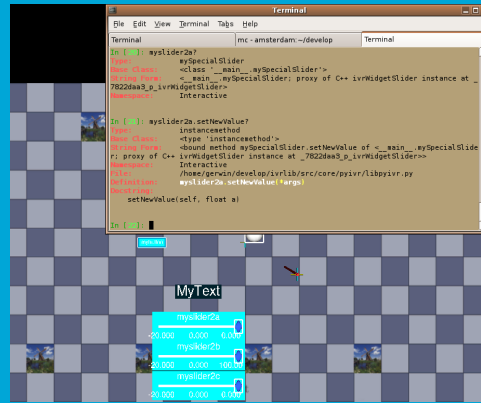
July 27, 2007

29

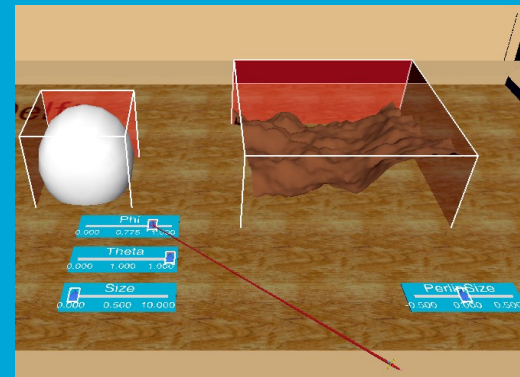
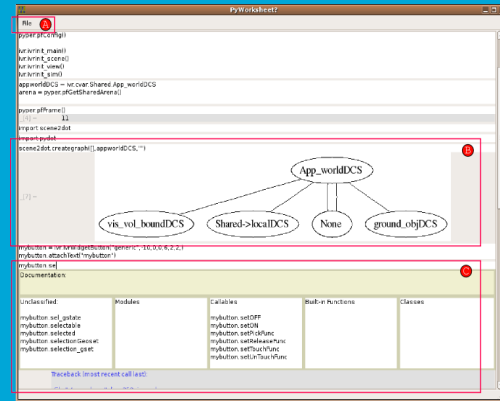
G. de Haan, M. Koutek, F.H. Post

"Flexible Abstraction Layers for VR Application Development"

# Questions ?



```
Qt [1] myslider?
Qt [1] mySpecialSlider
Qt [1] mySlider2a
Qt [1] mySlider2a.setValue?
Qt [1] mySlider2a
```



# Remarks !

g.dehaan@ewi.tudelft.nl - <http://visualisation.tudelft.nl/VRdev>