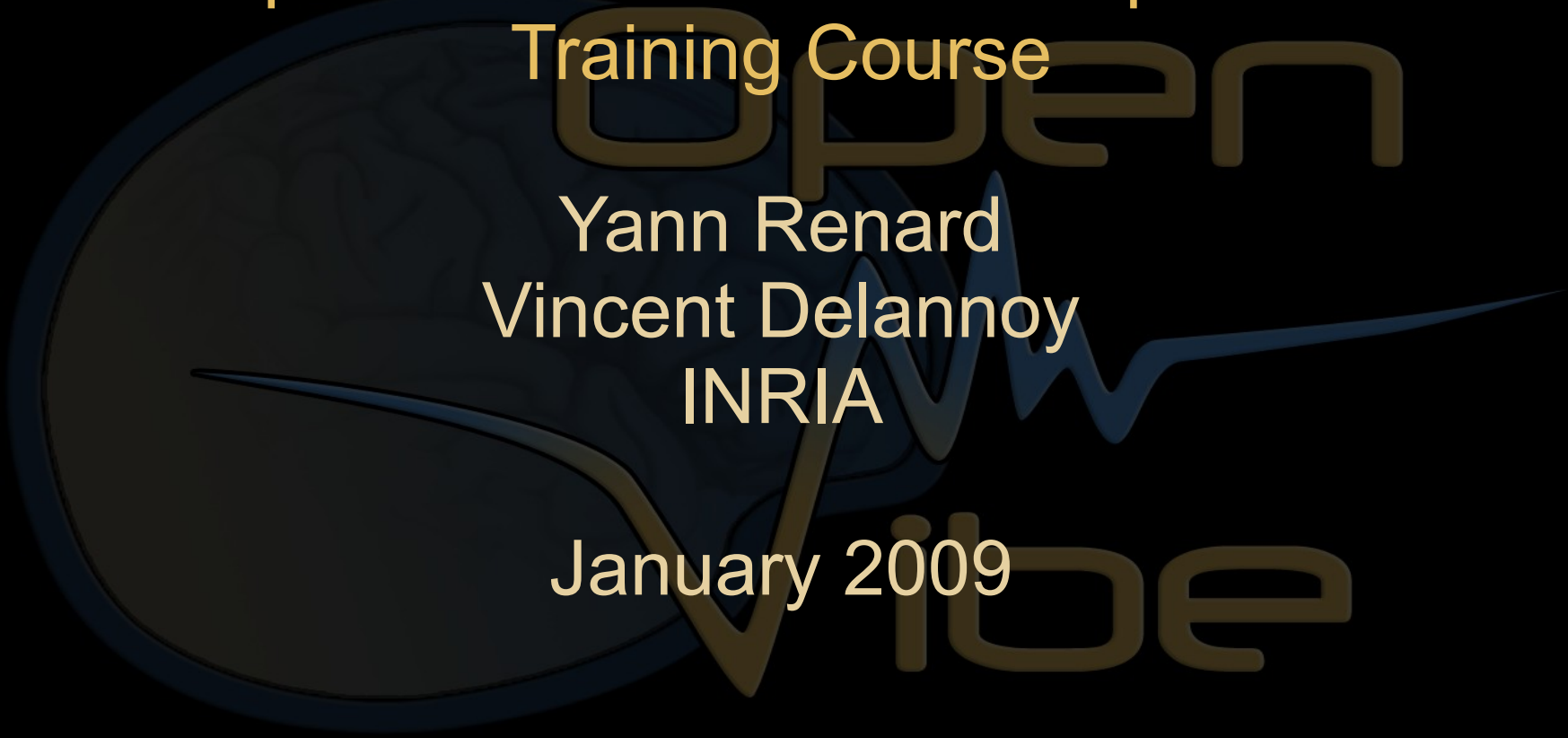
OpenViBE Platform Development Training Course

Yann Renard
Vincent Delannoy
INRIA

January 2009



OpenViBE Platform Development Training Course
Contents (1/2)

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- OpenViBE tools and resources
- OpenViBE concepts
- Data acquisition
  - Overview of the main software components and their relationships
  - The Acquisition Server : an introduction
  - The different approaches to design a driver
  - The driver creation API
  - Exercise 1 : create a driver (sinusoidal signal + stimulations at each packet)
  - Exercise 2 : create a configuration interface for this driver
-

OpenViBE Platform Development Training Course

Contents (2/2)

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The OpenViBE Algorithm
  - Overview of the kernel/plugin architecture
  - Algorithm concepts overview
  - A simple signal processing algorithm
  - Exercise 1 : create an algorithm setting all samples to 0
  - Exercise 2 : create an algorithm computing the Graz Band Powers
- The OpenViBE BoxAlgorithm
  - Box algorithm concepts overview
  - A look at parameter handlers
  - A simple signal processing box algorithm
  - Exercise 1 : add a parameter to the algorithm and initialize it in the box
  - Exercise 2 : add a setting to the box and use it to modify the algorithm parameter

OpenViBE Platform Development Training Course

OpenViBE Tools & Resources

OpenViBE Tools & Resources

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The OpenViBE software is hosted on the INRIA gForge
- The forge offers a number of services :
  - Project website
  - Source code management (concurrent versions) : CVS or SVN
  - Bug tracker
  - Mailing lists
  - Statistics
  - Automated builds (not used yet)
  - ...

OpenViBE Tools & Resources

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Bug tracking
  - Can (should) be done on the forge.
  - Enables to :
    - Keep a trace of previous bugs
    - Centralise all feature requests
  - As the number of developers grows, this tool will become essential
- Mailing lists (registration on the forge)
  - Several lists were created on the forge for different types of users
    - openvibe-bugs : to discuss bug-related issues
    - openvibe-devel : to discuss platform development
    - openvibe-info : to get news about the platform
    - openvibe-commits : to be notified every time a contributor « *commits* » code on the forge

OpenViBE Tools & Resources

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Other communication tools
  - Several other means of communication are available
    - The forum (http://www.irisa.fr/bunraku/OpenViBE/forum)
    - The IRC channel for live chat (#openvibe on irc.freenode.net)
  - Users should choose the most relevant means of communication, depending on their preferences and the subject to be discussed
- Coding rules
  - To improve code readability and ensure style is consistant across source files, it is important to use coding rules
  - A number of rules were defined and should be followed by developers. They can be found in the OpenViBE documentation.

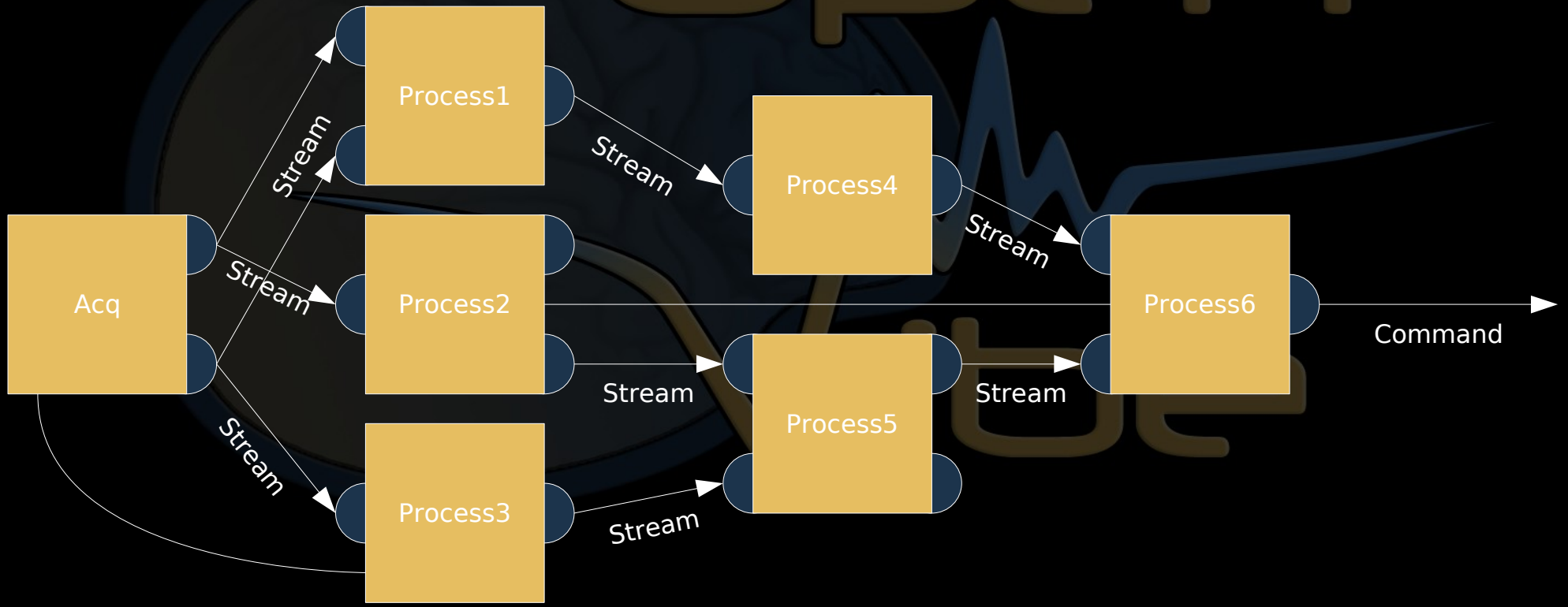
OpenViBE Platform Development Training Course

OpenViBE Concepts

OpenViBE Concepts
Introduction

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

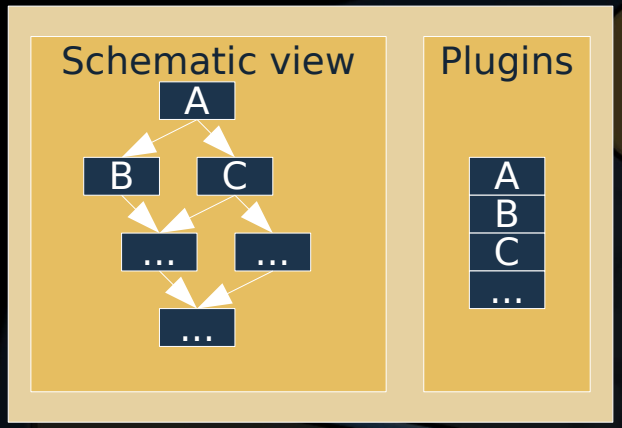
- Basic idea
  - A modular system allowing to chain unit processings quickly to come up with a high level processing or a complete processing chain

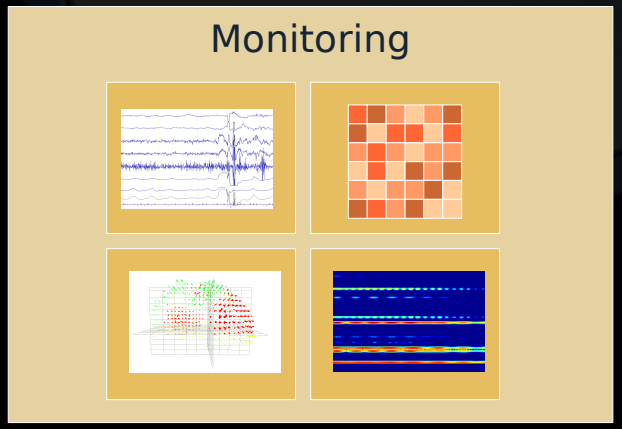


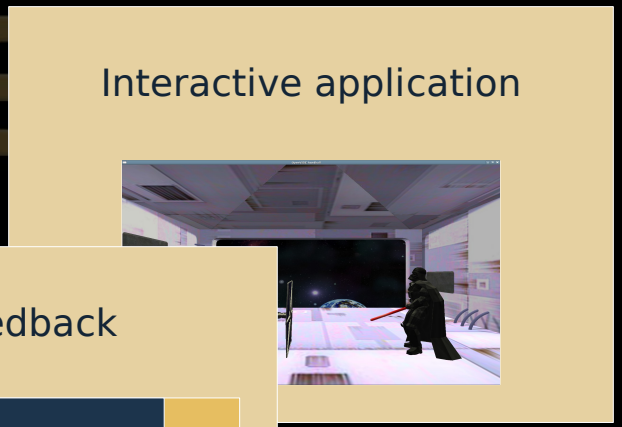
OpenViBE Concepts
Introduction

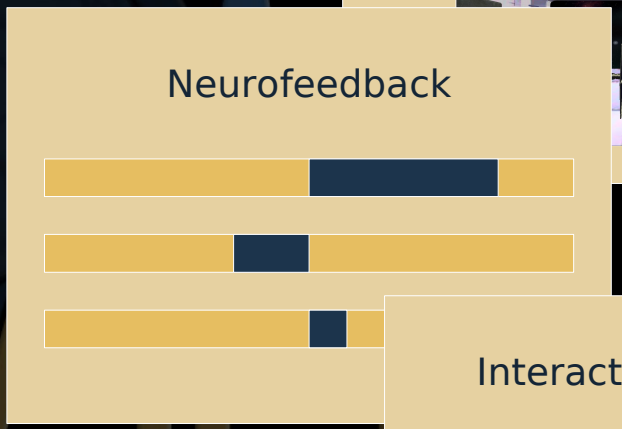
Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

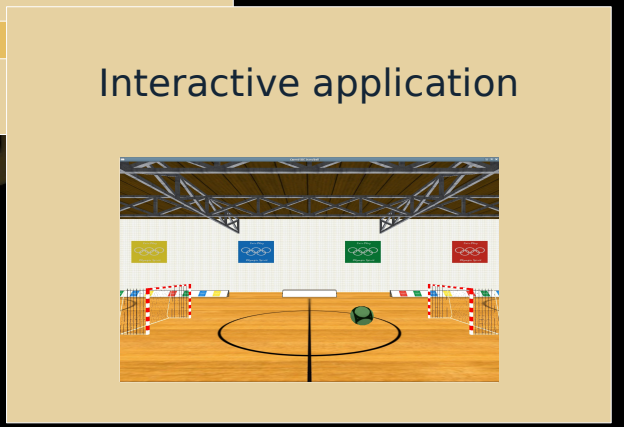
- Several user levels are considered
  - Author view
  - Operator view
  - Subject view







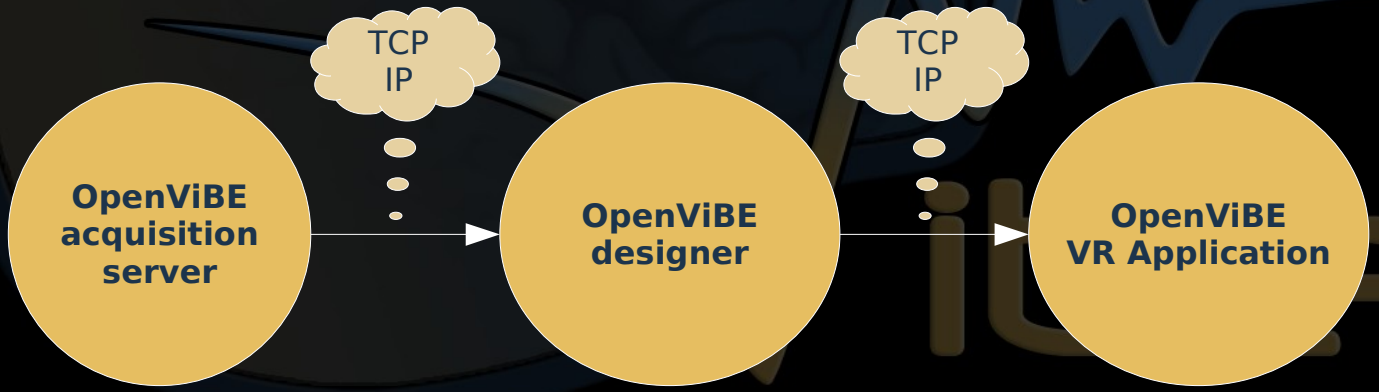




OpenViBE Concepts
Software architecture

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

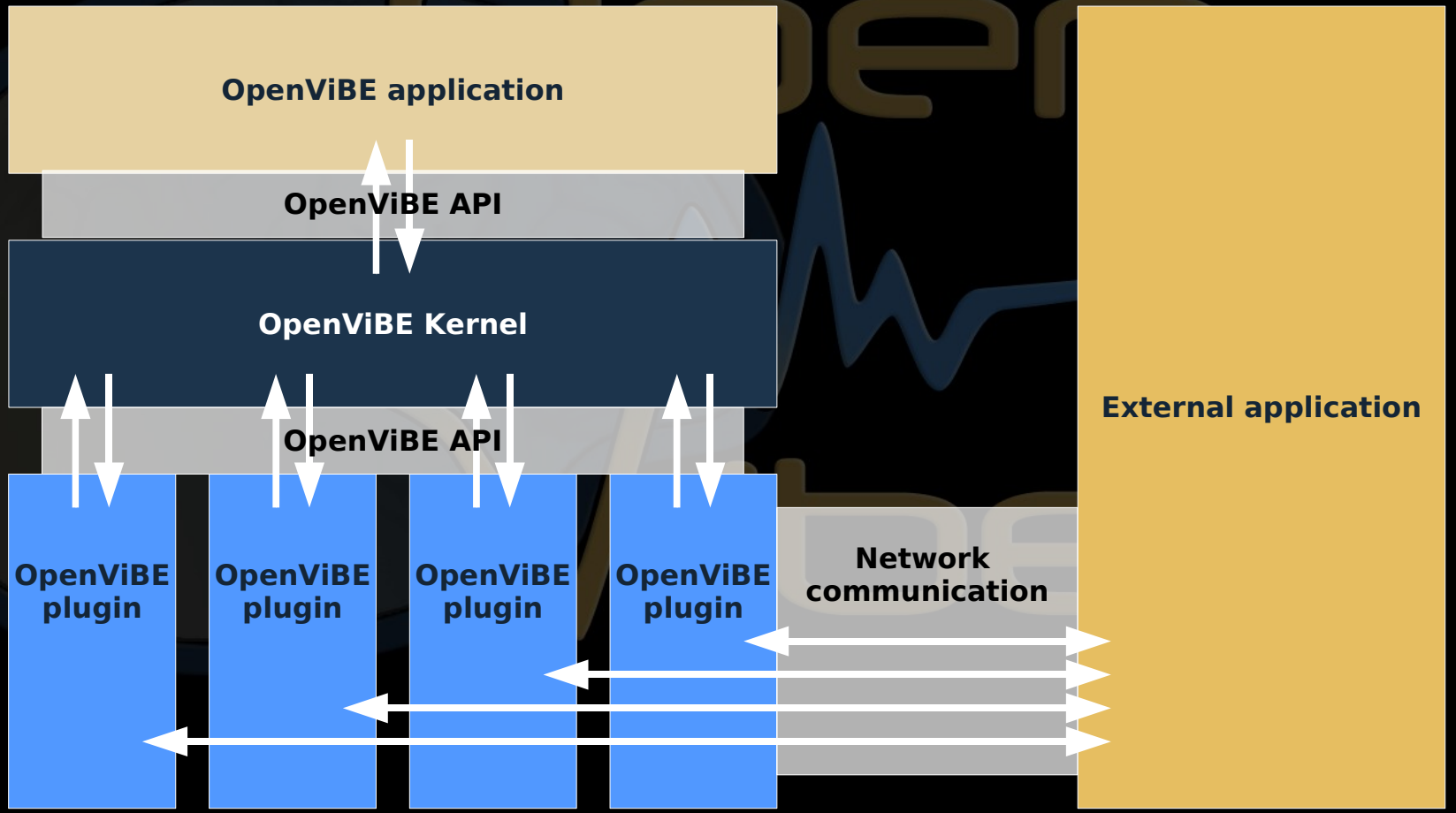
- Three applications communicating across a network :
  - OpenViBE acquisition server
  - OpenViBE designer
  - OpenViBE virtual reality application



OpenViBE Concepts
Software architecture

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Structural view of an OpenViBE application and communication with the outside world



OpenViBE Concepts
Source tree structure

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Sources are divided into subprojects (openvibe, openvibe-applications, openvibe-kernel-omk, openvibe-toolkit, openvibe-modules, openvibe-plugins)
- An overall tree structure is common to all subprojects (branches, trunc, tags...)
- However, each subproject has its own tree structure (branch names, tagged versions, source files...)

OpenViBE Platform Development Training Course

Data Acquisition

Data Acquisition
Introduction

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The OpenViBE Acquisition Server employs a driver abstraction
- New peripherals can thus be added to the platform by creating a new driver for the Acquisition Server
- The role of a driver consists in retrieving signals and events and put them at the disposal of the Acquisition Server
- Several approaches may be followed to retrieve such data

Data Acquisition
Data acquisition methods

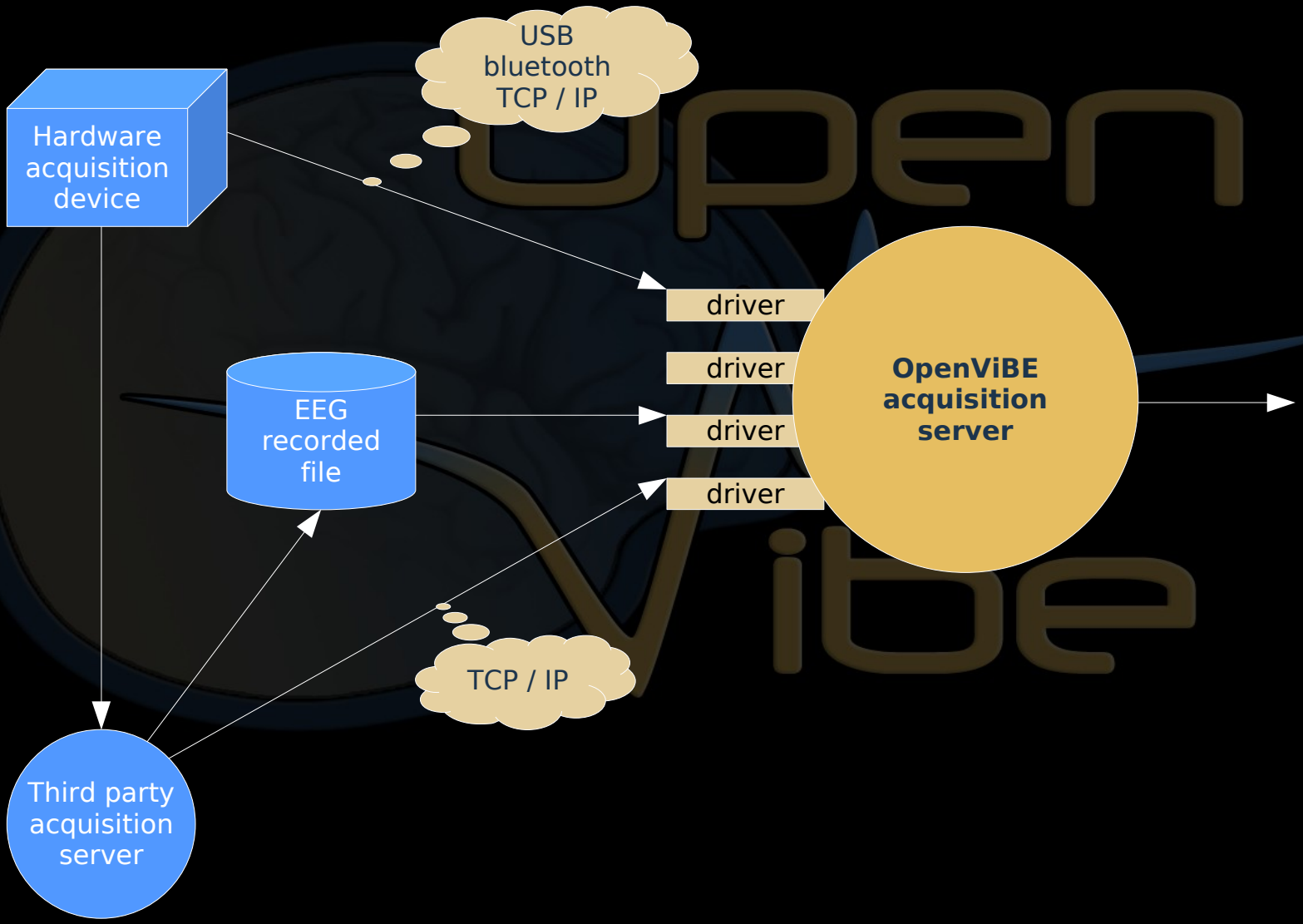
Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Signals and events retrieval :
  - Direct communication with peripheral
    - through an API (ideal case)
    - through a hardware interface (serial port, parallel port, USB, TCP / IP...)
  - Communication with a proprietary acquisition server
    - across a network
    - through a file
    - through a system component such as COM
- Advantages / drawbacks to consider :
  - Performance (latency, latency variability...)
  - Code portability
  - Amount of code to produce
  - Code maintainability

Data Acquisition
Data acquisition methods

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

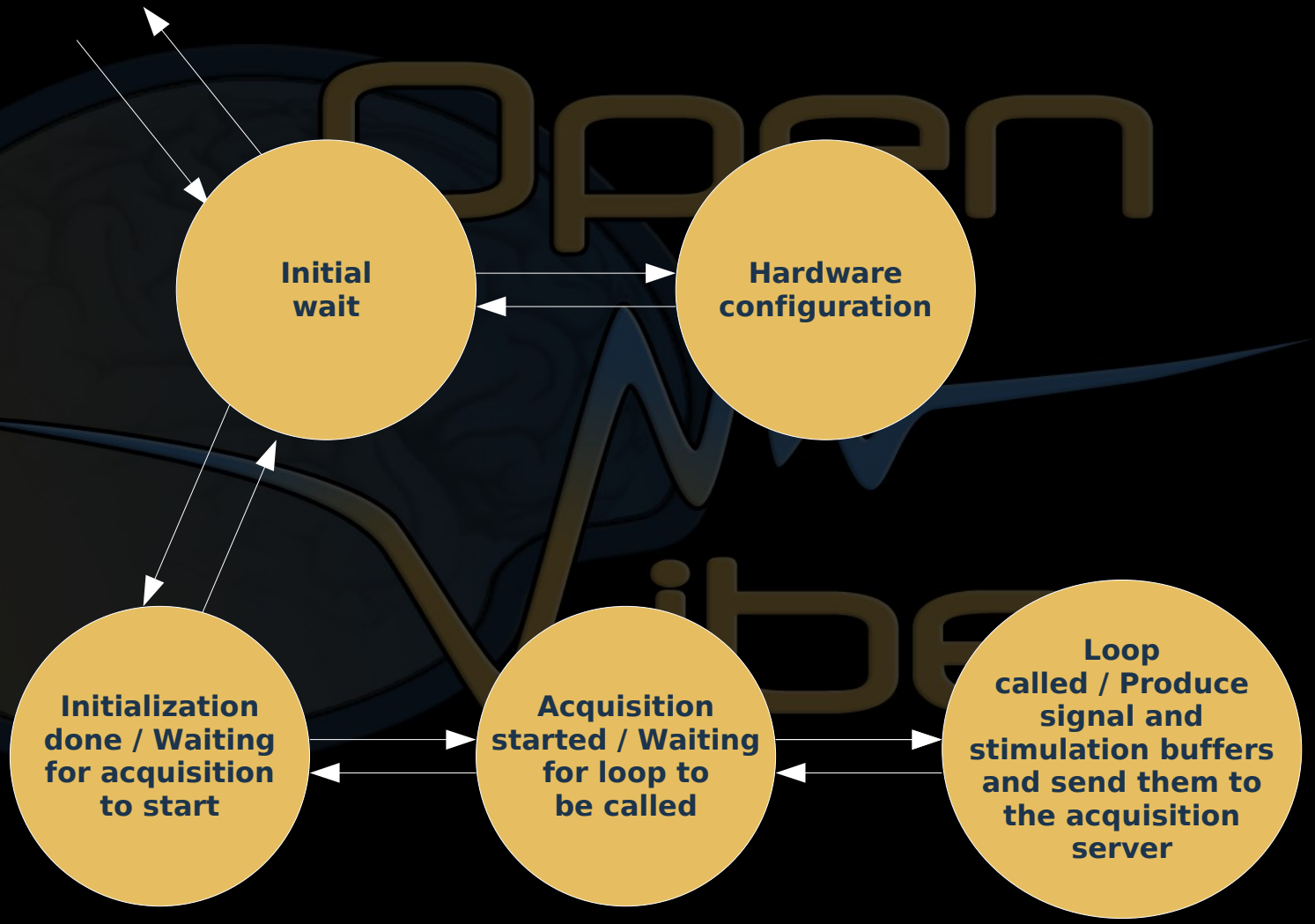
- Driver types for the acquisition server :



Data Acquisition
Driver behaviour

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Driver state automaton :



Data Acquisition
Driver development

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Steps to follow to develop a driver :
  - Inherit from the OpenViBEAcquisitionServer::IDriver interface
  - Implement the following methods (declared in ovasIDriver.h) :
    - getName (retrieve driver name)
    - initialize (initialise driver)
    - uninitialize (uninitialise driver)
    - getHeader (retrieve information about the peripheral : channel count, channel names, gains, etc...)
    - start (start acquisition)
    - stop (stop acquisition)
    - loop (provide a data block)
  - Declare the driver in the acquisition server :
    - Source file : ovasCAquisitionServer.cpp
    - Add this line to the constructor : m_vDriver.push_back(new CMyDriver());

Data Acquisition
Exercise 1

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Exercise :
  - Create a simple driver simulating a peripheral with the following features :
    - 4 channels
    - 100 Hz sampling frequency
    - Sinusoidal signals
    - Events at each data block
- Hints :
  - Keep track of generated samples count, compare it to elapsed time to determine when to send data
  - Time can be computed (in ms) using System::Time::getTime()
  - Samples and events are sent to the acquisition server using the rCallback object provided in initialize()

Data Acquisition
Solution (1/3)

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution :
  - Header file - driver class scope :
    - OpenViBEAcquisitionServer::CHeader m_oHeader; //channel count, names, frequency
    - OpenViBE::float32* m_pSample; //array of generated samples
    - OpenViBE::uint32 m_ui32TotalSampleCount;
    - OpenViBE::uint32 m_ui32StartTime;
  - Implementation file - driver constructor :
    - m_pSample=NULL;
    - m_ui32TotalSampleCount=0;
    - m_ui32StartTime=0;
    - m_oHeader.setChannelCount(4);
    - m_oHeader.setChannelName(0, "Pz");
    - m_oHeader.setChannelName(1, "Cz");
    - m_oHeader.setChannelName(2, "C3");
    - m_oHeader.setChannelName(3, "C4");
    - m_oHeader.setSamplingFrequency(100);

Data Acquisition
Solution (2/3)

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Implementation file - driver initialize() method :
  - m_pSample=new float32[4*ui32SampleCountPerSentBlock]; //allocate one block of data
  - m_ui32TotalSampleCount=0;
  - m_ui32StartTime=System::Time::getTime();
- Implementation file - driver uninitialize() method :
  - delete [] m_pSample;
  - m_pSample=NULL;
- Implementation file - driver getHeader() method :
  - return &m_oHeader;

Data Acquisition
Solution (3/3)

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Implementation file - driver loop() method :

```
uint32 l_ui32CurrentTime=System::Time::getTime();
if(l_ui32CurrentTime-m_ui32StartTime >
        (1000*m_ui32TotalSampleCount)/m_oHeader.getSamplingFrequency()) {
        CStimulationSet l_oStimulationSet;
        l_oStimulationSet.setStimulationCount(1);
        l_oStimulationSet.setStimulationIdentifier(0, 0);
        l_oStimulationSet.setStimulationDate(0, 0);
        l_oStimulationSet.setStimulationDuration(0, 0);
        for(uint32 j=0; j<4; j++)       {
                for(uint32 i=0; i<m_ui32SampleCountPerSentBlock; i++) {
                        m_pSample[j*m_ui32SampleCountPerSentBlock+i]=
                                sinf(((i+m_ui32TotalSampleCount)*2.3)/100+j);
                }
        }
        m_ui32TotalSampleCount+=m_ui32SampleCountPerSentBlock;
        m_pCallback->setSamples(m_pSample);
        m_pCallback->setStimulationSet(l_oStimulationSet);
}
```

Data Acquisition
Driver configuration

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Sometimes, a driver can have different options :
  - Channel count, if it can't determine it itself
  - Channel names, if it can't name them itself
  - Sampling frequency if it can be chosen freely
  - Port used to communicate with the peripheral
  - Proprietary acquisition server to connect to...
- Therefore, it may be necessary to make the driver configurable
- Glade can be used to create the configuration dialog box
- A helper class can help you manipulate this dialog box from your driver

Data Acquisition
Exercise 2

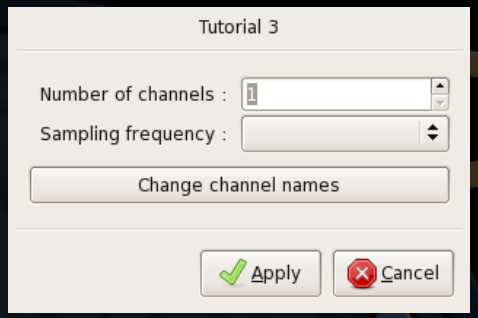
Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

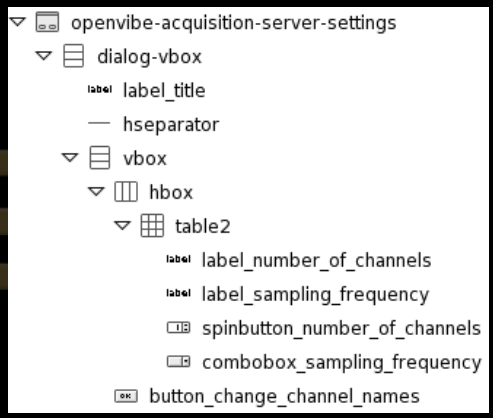
- Exercise :
  - Create a simple graphical interface for our peripheral
- Hints :
  - The CConfigurationGlade helper class allows to handle most parameters of a driver configuration dialog box
  - To use this class, some widgets must be of a given type and use a predefined name :
    - spinbutton_number_of_channels for the spin button specifying the channel count
    - combobox_sampling_frequency for the combo box listing sampling frequencies
    - button_change_channel_names for the button that pops up a dialog box allowing to assign names to channels (from a text file)

Data Acquisition
Solution

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution :
  - Dialog box design in Glade :





  - In .cpp file :
    - #include "../ovasCConfigurationGlade.h" //include helper class header
  - In isConfigurable() :
    - return true; //tell server this driver uses a configuration dialog box
  - In configure() :
    - CConfigurationGlade m_oConfiguration("../share/openvibe-applications/acquisition-server/interface-tutorial-3.glade"); //load configuration dialog box
    - return m_oConfiguration.configure(m_oHeader); //configure header using dialog box

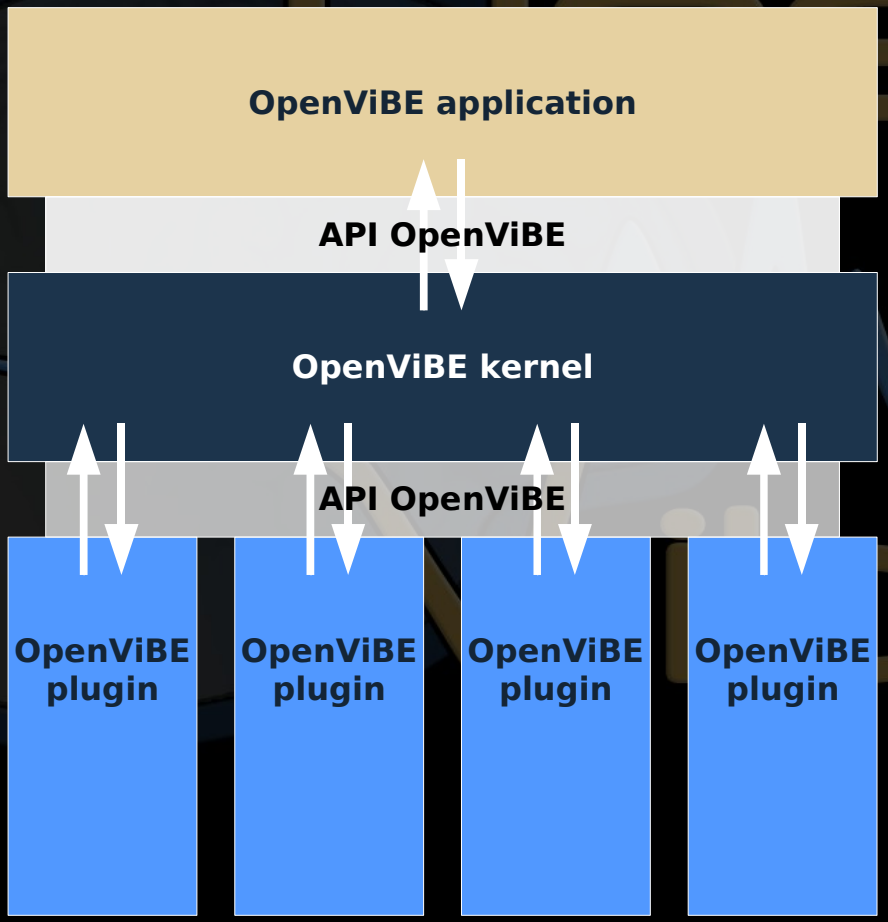
OpenViBE Platform Development Training Course

Kernel / Plugins Architecture

Kernel / Plugins Architecture
Introduction

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- OpenViBE applications (e.g. Designer) relie on a kernel
- The kernel itself relies on plugins to perform its tasks



Kernel / Plugins Architecture
The kernel

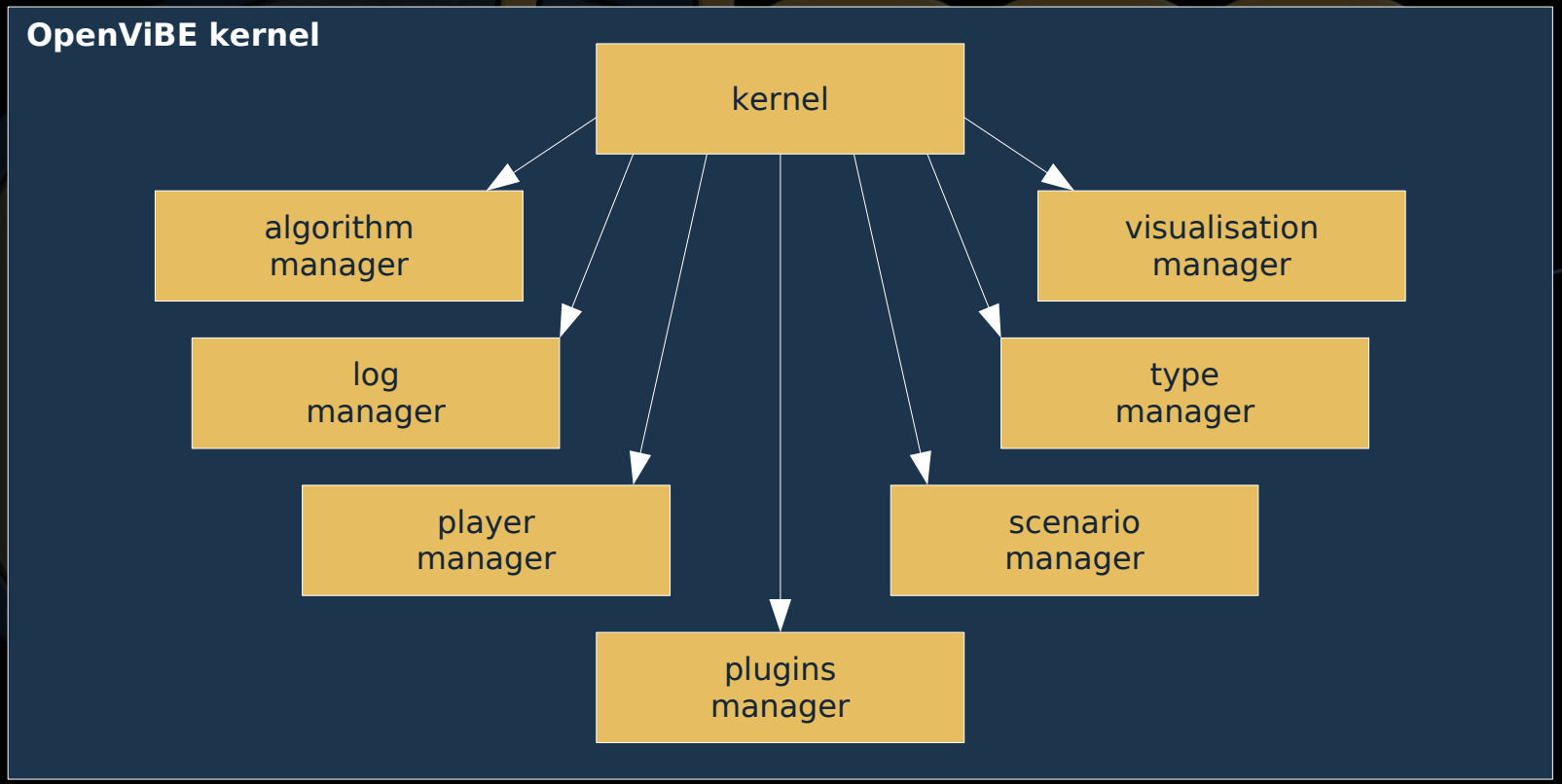
Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The OpenViBE kernel is made up of several managers, each of which being responsible for handling a certain type of objects / tasks
  - the *algorithm manager* handles algorithms,
  - the *log manager* handles traces,
  - the *player manager* handles scenario execution,
  - the *plugin manager* handles the plugin mechanism,
  - the *scenario manager* handles scenario edition,
  - the *type manager* handles data types,
  - the *visualisation manager* handles window layout and graphics requests
- Plugins are called from various places in kernel components, depending on the task at hand

Kernel / Plugins Architecture
The kernel

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Schematic view of the OpenViBE kernel :



Kernel / Plugins Architecture
Plugins

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- OpenViBE plugins
  - They come in two flavors :
    - algorithms
    - boxes (or 'box algorithms')
  - Regardless of the type, to create a new plugin, one must fill a plugin descriptor to allow the kernel or an application to use it
  - Thus, there always are two classes to implement :
    - the plugin descriptor
    - the plugin itself
  - The descriptor allows to :
    - provide textual information about the plugin (author, date, version, concise documentation, etc...)
    - inform the kernel about the class of plugin which can be created
    - create actual plugin instances

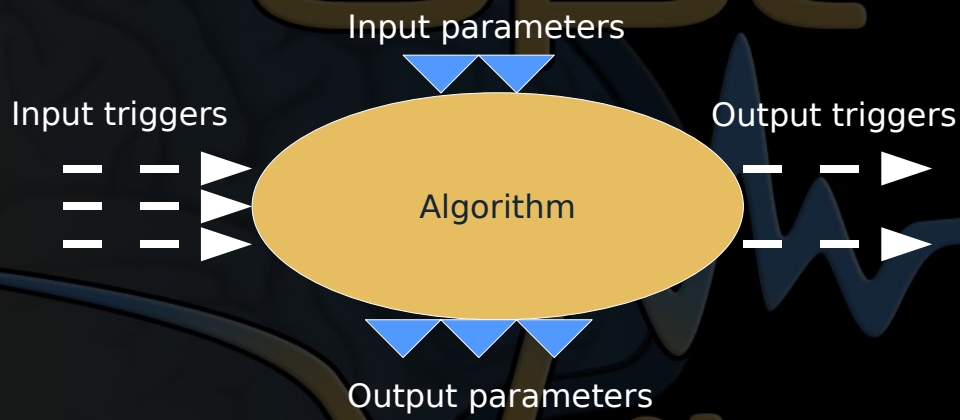
OpenViBE Platform Development Training Course

The OpenViBE Algorithm

The OpenViBE Algorithm
Introduction

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The algorithm is a very generic, low level component which can easily communicate with other algorithms
- It comprises input / output parameters and input / output « triggers »

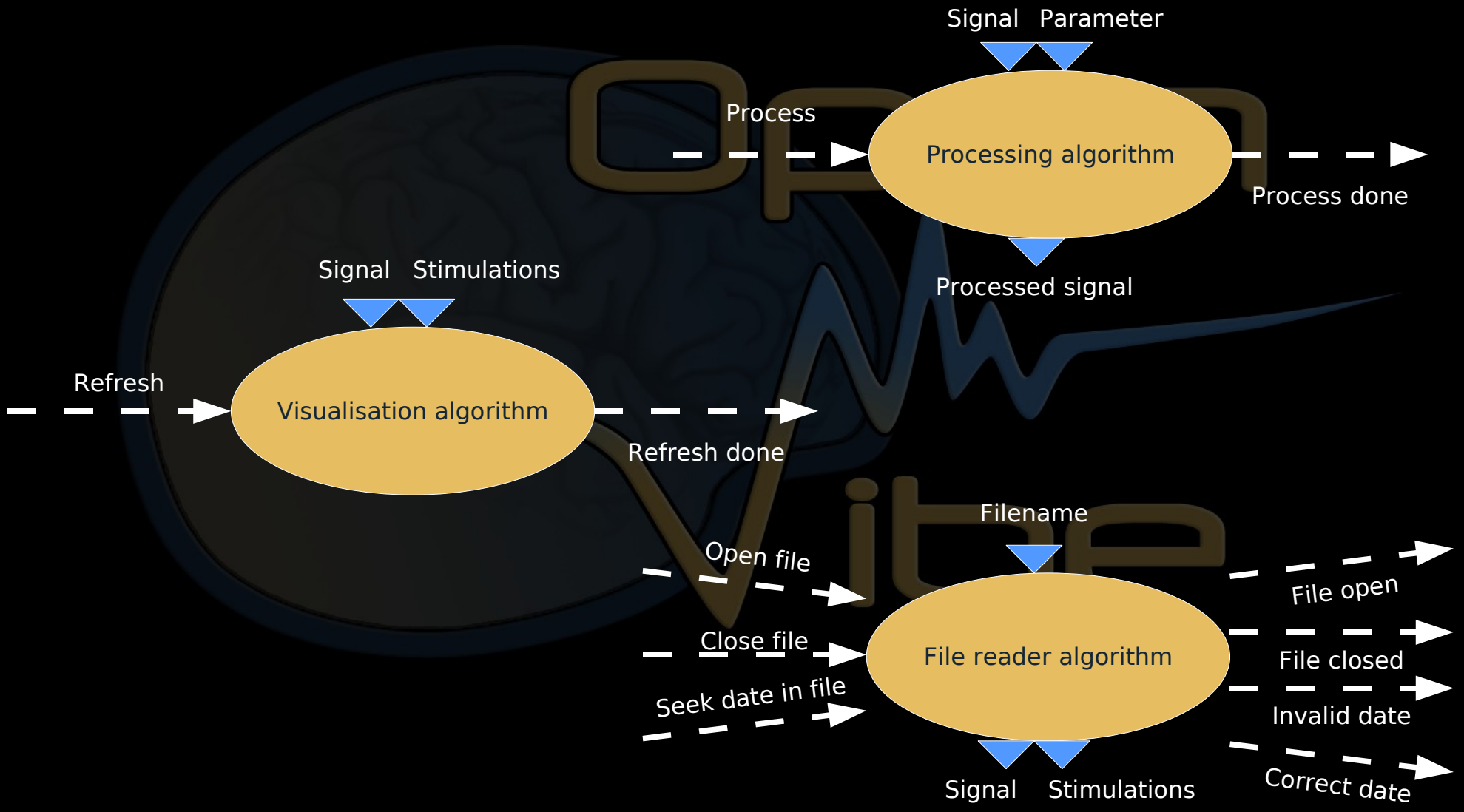


- One can organise several algorithms together to form a more complex algorithm

The OpenViBE Algorithm
Examples

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

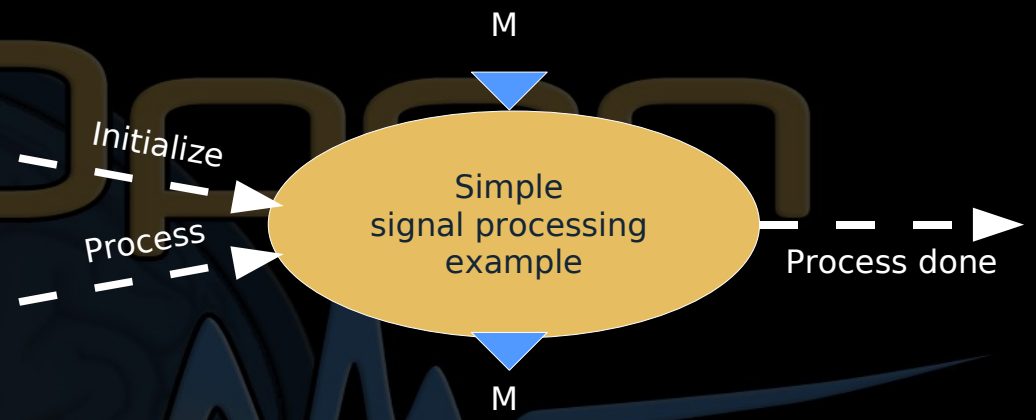
- Algorithm examples :



The OpenViBE Algorithm
Signal processing algorithm example

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Let's take a simple signal processing algorithm
  - 2 input triggers
  - 1 output trigger
  - 1 input matrix
  - 1 output matrix



- Sources contain one such algorithm
  - ovpCAlgorithmSignalProcessingAlgorithm.h
  - ovpCAlgorithmSignalProcessingAlgorithm.cpp

The OpenViBE Algorithm
Signal processing algorithm example

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- The descriptor implements several methods providing information about the algorithm, particularly :
  - getAlgorithmPrototype to describe the aspect of the algorithm
  - getCreatedClass to get the algorithm class
  - create to build one instance of the algorithm
- The algorithm implements 3 methods :
  - initialize
  - uninitialize
  - process
- The algorithm manipulates its input / output parameters using « parameter handlers »

The OpenViBE Algorithm
Exercise 1

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Exercise :
  - Develop an algorithm which sets all samples to 0, using the simple signal processing algorithm as a basis
- Hints :
  - Contents of a signal matrix, as received by the algorithm :
    - 1st dimension (index 0) : channel
    - 2nd dimension (index 1) : samples of a given channel
  - Accessing the matrix buffer (samples) :
    - Use getBuffer()
    - Example : to affect 'x' to M[i][j], do :
    - M->getBuffer() [i * sampleCountPerChannel + j] = x;
    - where sampleCountPerChannel is the size of the 2nd dimension (index 1)
  - Modify the process() method to reset all samples

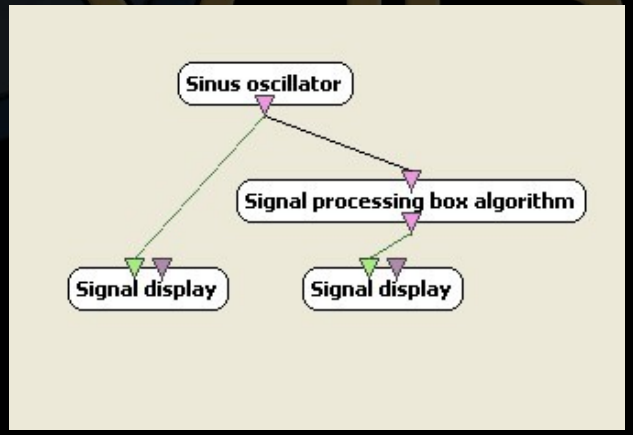
The OpenViBE Algorithm
Solution 1

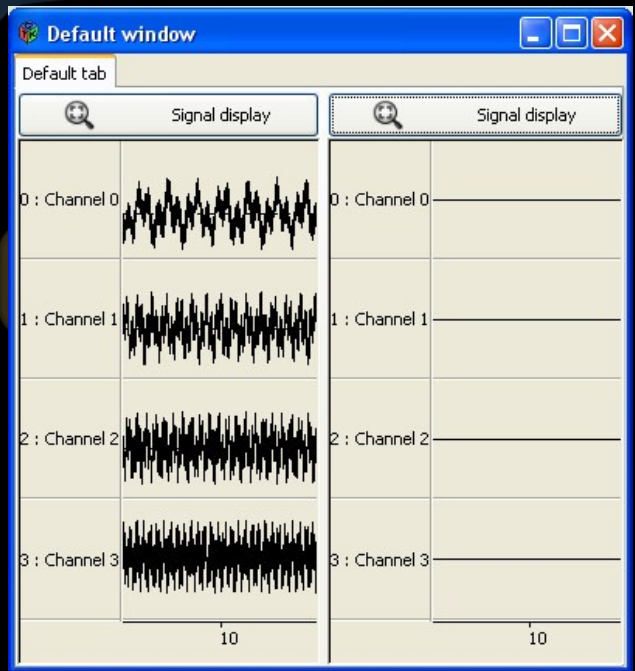
Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution :
  - In process() :

```
for(uint32 i=0; i<l_pInputMatrix->getDimensionSize(0); i++)
{
        for(uint32 j=0; j<l_pInputMatrix->getDimensionSize(1); j++)
        {
                l_pOutputMatrix->getBuffer()
                        [i*l_pInputMatrix->getDimensionSize(1)+j]=0;
        }
}
```

Visualising sinusoidal signals as generated by the sinus oscillator, and after modification by the signal processing box :





The OpenViBE Algorithm
Exercise 2

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Exercise :
  - Modify the algorithm used in the previous exercise to compute the Graz Band Powers of the signal
- Hints :
  - For each channel
    - Compute $\log(1+(\Sigma S_i^2) / N)$, where S_i is the ith sample and N the sample count per channel
    - Store this value in the output matrix (one value per channel only)
  - Resize the output matrix I_pOutputMatrix
    - The first dimension (corresponding to channels) may be left unchanged
    - The second dimension should be resized to 1
    - Use setDimensionSize(i, j) where i is the dimension index and j its size
    - This can be done when the « initialize » trigger is set

The OpenViBE Algorithm
Solution 2

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution :
  - In process(), when the Initialize trigger is set:
    - l_pOutputMatrix->setDimensionSize(1, 1); //resize dimension 1 to 1
  - In process(), when the Process trigger is set:

```
uint32 l_ui32SamplesCount = l_pInputMatrix->getDimensionSize(1);
for(uint32 i=0; i<l_pInputMatrix->getDimensionSize(0); i++) {
        float64 l_f64SquaresSum = 0;
        float64 l_f64CurrentSample = 0;
        for(uint32 j=0; j<l_pInputMatrix->getDimensionSize(1); j++) {
                l_f64CurrentSample =
                        l_pInputMatrix->getBuffer()[i*l_ui32SamplesCount + j];
                l_f64SquaresSum += l_f64CurrentSample * l_f64CurrentSample;
        }
        l_pOutputMatrix->getBuffer()[i] =
                log(1+l_f64SquaresSum/l_ui32SamplesCount);
}
```

OpenViBE Platform Development Training Course

The OpenViBE BoxAlgorithm

The OpenViBE BoxAlgorithm
Introduction

Yann Renard
Vincent Delannoy
INRIA

- A box has inputs / outputs (streams) and settings
- A box can be notified on several types of events
  - At clock ticks, at its own frequency
  - Upon data arrival
  - Upon message arrival (not implemented yet)
- A file reading or acquisition box will want to be notified at clock ticks
- A signal processing, recording or visualisation box would rather be notified upon data arrival
- Box processing is triggered when the box is ready
- The typical way this is done is :
  - input reading
  - processing internal to the box or call(s) to processing algorithm(s)
  - output production

The OpenViBE BoxAlgorithm
Introduction

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- A box can use one or more algorithms internally to delegate tasks such as :
  - Input streams decoding
  - Data processing
  - Output streams encoding
- Boxes can be regarded as algorithm schedulers
- When notified by the kernel, they react by triggering algorithms

The OpenViBE BoxAlgorithm
Parameter handlers

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Parameter handlers allow to easily manipulate input and output algorithm parameters

Algorithme side:

```
float64 l_f64Value;
TparameterHandler < float64 > ip_f64Value;

ip_f64Value.initialize(
        this->getInputParameter(
                ParameterId));

ip_f64Value=1.0;

// ...
// ...

l_f64Value=ip_f64Value; // parameter value is now 2
ip_f64Value=3.0; //set value to 3
```

Box side :

```
float64 l_f64Value;
TparameterHandler < float64 > ip_f64Value;

ip_f64Value.initialize(
        m_pAlgorithm->getInputParameter(
                ParameterId));

// ...

l_f64Value=ip_f64Value; // parameter value is 1
ip_f64Value=2.0;

// ...
// ...

l_f64Value=ip_f64Value; // value is now 3
```

The OpenViBE BoxAlgorithm
Parameter handlers

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Parameter handlers allow to connect the output of an algorithm to the input of another algorithm using setReferenceTarget. Once this is done, the box doesn't have to pass data from one algorithm to the next anymore.

Box :

```
TparameterHandler < float64 > op_f64Value;
TparameterHandler < float64 > ip_f64Value;

op_f64Value.initialize(m_pAlgorithm1->getOutputParameter(OutputParameterId));
ip_f64Value.initialize(m_pAlgorithm2->getInputParameter(InputParameterId));

ip_f64Value.setReferenceTarget(op_f64Value);
```

Algorithm A :

```
TparameterHandler < float64 > op_f64Value;

op_f64Value.initialize(
        this->getOutputParameter(
                OutputParameterId));
// ...
op_f64Value=1.0;
// ...
op_f64Value=2.0;
// ...
```

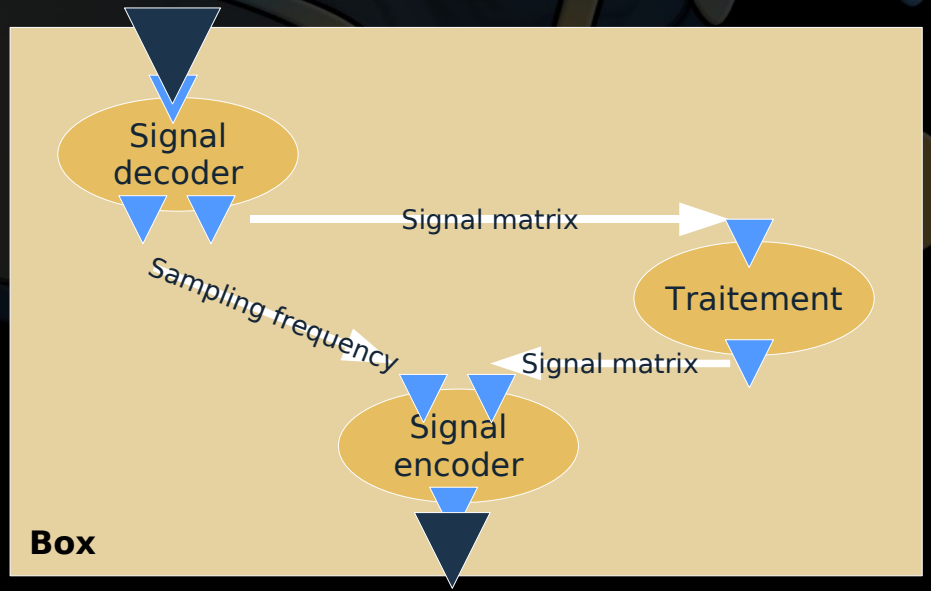
Algorithm B :

```
float64 l_f64Value;
TparameterHandler < float64 > ip_f64Value;

ip_f64Value.initialize(
        this->getOutputParameter(
                InputParameterId));

// ...
l_f64Value=ip_f64Value; // value is 1
// ...
l_f64Value=ip_f64Value; // value is now 2
```

The OpenViBE BoxAlgorithm
Signal processing box example

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- A box using the aforementioned algorithm can be found in
  - ovpCBoxAlgorithmSignalProcessingBoxAlgorithm.h
  - ovpCBoxAlgorithmSignalProcessingBoxAlgorithm.cpp
- Its descriptor implements several methods providing information about the box, particularly :
  - getBoxPrototype to describe the box aspect
  - getCreatedClass to give the box class
  - create to create an instance of the box



The OpenViBE BoxAlgorithm
Signal processing box example

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Boxes implement 4 methods :
  - initialize
  - uninitialize
  - process
  - processInput
- The box described in this tutorial uses 3 algorithms (m_pSignalDecoder, m_pSignalProcessingAlgorithm and m_pSignalEncoder)
- For each algorithm, it declares parameter handlers to manipulate data

The OpenViBE BoxAlgorithm
Exercise 1

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Algorithms can use parameters to adapt their behaviour to the situation
- Exercise :
  - The box must control the value of a parameter (e.g. a float)
  - Send this value to the algorithm, and use it so that the algorithm multiplies each sample with this parameter
- Hints :
  - The algorithm descriptor must declare a new « input parameter »
  - The algorithm must have a new parameter handler, whose value will be multiplied to samples in process()
  - The box must declare a new parameter handler, initialising it with the algorithm parameter
  - It must give an initial value to the encapsulated type using the '=' operator (subsequently, the box will be able to modify the value of this parameter at any time using this operator)

The OpenViBE BoxAlgorithm
Solution 1

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution - algorithm modifications :
  - Header file - general scope:
    - #define OVP_Algorithm_SignalProcessingAlgorithm_InputParameterId_MultiplicationFactor
    - OpenViBE::CIdentifier(0x1E6940B9, 0x27017EB3)
  - Header file - algorithm class :
    - OpenViBE::Kernel::TParameterHandler < OpenViBE::float64 > ip_f64MultiplicationFactor;
  - Header file - algorithm descriptor getAlgorithmPrototype() method :
    - rAlgorithmPrototype.addInputParameter(
    - OVP_Algorithm_SignalProcessingAlgorithm_InputParameterId_MultiplicationFactor,
    - "MultiplicationFactor", OpenViBE::Kernel::ParameterType_Float);
  - Implementation file - algorithm initialize() method :
    - ip_f64MultiplicationFactor.initialize(
    - this->getInputParameter(
    - OVP_Algorithm_SignalProcessingAlgorithm_InputParameterId_MultiplicationFactor));
  - Implementation file - algorithm uninitialize() method :
    - ip_f64MultiplicationFactor.uninitialize();
  - Implementation file - algorithm process() method :
    - for(uint32 i=0; i<l_pInputMatrix->getBufferElementCount(); i++) {
    - l_pOutputMatrix->getBuffer()[i] = l_pInputMatrix->getBuffer()[i] * ip_f64MultiplicationFactor; }

The OpenViBE BoxAlgorithm
Solution 1

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution - box modifications :
  - Header file - general scope :
    - OpenViBE::Kernel::TParameterHandler < OpenViBE::float64 > ip_f64MultiplicationFactor;
  - Implementation file - general scope :
    - #include "ovpCAlgorithmSignalProcessingAlgorithm.h"
  - Implementation file - box initialize() method :
    - ip_f64MultiplicationFactor.initialize(
    - m_pSignalProcessingAlgorithm->getInputParameter(
    - OVP_Algorithm_SignalProcessingAlgorithm_InputParameterId_MultiplicationFactor));
    - ip_f64MultiplicationFactor = 0;
  - Implementation file - box uninitialize() method :
    - ip_f64MultiplicationFactor.uninitialize();

The OpenViBE BoxAlgorithm
Exercise 2

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Exercise :
  - The box must declare a new setting (which will be exposed in the Designer when double clicking the box)
  - The setting should have different possible values : it will be defined as an enumeration
  - At box initialisation time, the value of the setting will be retrieved and used to initialize the algorithm multiplication factor parameter
- Hints :
  - An enumeration and its values are declared using the type manager methods registerEnumerationType & registerEnumerationEntry
  - Methods addSetting (box prototype) and getSettingValue (box) allow to manipulate the new setting
  - The getEnumerationEntryValueFromName method (type manager) converts an enumeration name to a numerical value

The OpenViBE BoxAlgorithm
Solution 2

Yann Renard
Vincent Delannoy
INRIA
14-15 / 01 / 2009

- Solution :
  - Header file - general scope:
    - #define OVP_TypeId_SignalProcessingMultiplicationFactor
    - OpenViBE::CIdentifier(0x12345678, 0x00000000)
  - Header file - box descriptor getBoxPrototype() method :
    - rBoxAlgorithmPrototype.addSetting("Multiplication factor",
    - OVP_TypeId_SignalProcessingMultiplicationFactor, "1");
  - ovp_main.cpp :
    - rPluginModuleContext.getTypeManager().registerEnumerationType(
    - OVP_TypeId_SignalProcessingMultiplicationFactor, "Multiplication factor");
    - rPluginModuleContext.getTypeManager().registerEnumerationEntry(
    - OVP_TypeId_SignalProcessingMultiplicationFactor, "0", 0);
    - rPluginModuleContext.getTypeManager().registerEnumerationEntry(
    - OVP_TypeId_SignalProcessingMultiplicationFactor, "1", 1);
  - Implementation file - box initialize() method :
    - CString l_sMultiplicationFactor;
    - getStaticBoxContext().getSettingValue(1, l_sMultiplicationFactor);
    - ip_f64MultiplicationFactor =
    - (float64)getTypeManager().getEnumerationEntryValueFromName(

OpenViBE Platform Development Training Course

Thank you for reading this!

Find out more about OpenViBE from the wiki :

http://www.irisa.fr/bunraku/OpenViBE/wiki