NEO : base and integration in OpenElectrophy

Samuel Garcia

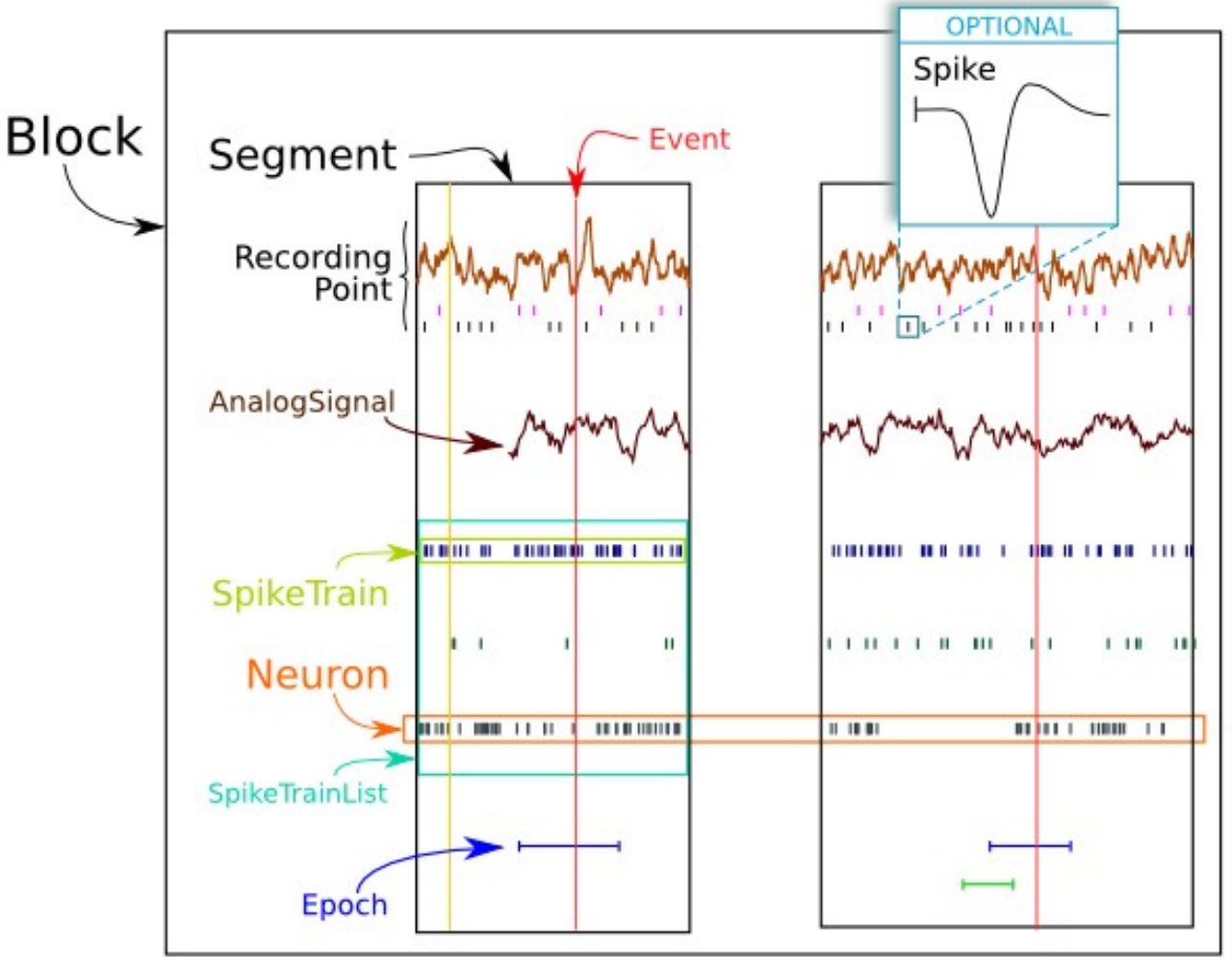
neo : Neural Ensemble Objects

History :
• convergence between NeuroTools and OpenElectrophy (and more we hope..)
• It was started in last code jam in Freiburg : 3 hours for calling a spike train : "SpikeTrain"

What is neo ? A different zoom:
• neo = nomenclature
• neo = straightforward implementation
• neo = core for other projects

What are main interests :
• Compatibility between projects
• A common layer for file reading/writing datasets.

Class tour



Neo : Neurotools/OpenElectrophy shared base architecture

Class tour : definition

Block: main container gathering all the data discrete or continous for a given setup. It can be view as a list of Segment. A block is not necessary a homogeneous recorging contrary to Segment

Segment: Heterogeneous container of several data sharing a common time base. A Segment is a container gathering discrete or continous data acquired during the same time lapse. In short, a Segment may contain AnalogSignal, SpikeTrain, Event and Epoch that share the same time base.

AnalogSignal: a continuous data signal acquired at time t_start at a certain sampling rate.

RecordingPoint: A RecordingPoint is a physical location identifying the recorded data. It can for example be the position of the Eletrode. It is util for spikesorting when you want to detect and sort spikes coming from of many dicountinoued segment of signal coming from the same recordingpoint.

SpikeTrain: an array of Spike emitted by the same Neuron in a time lap.

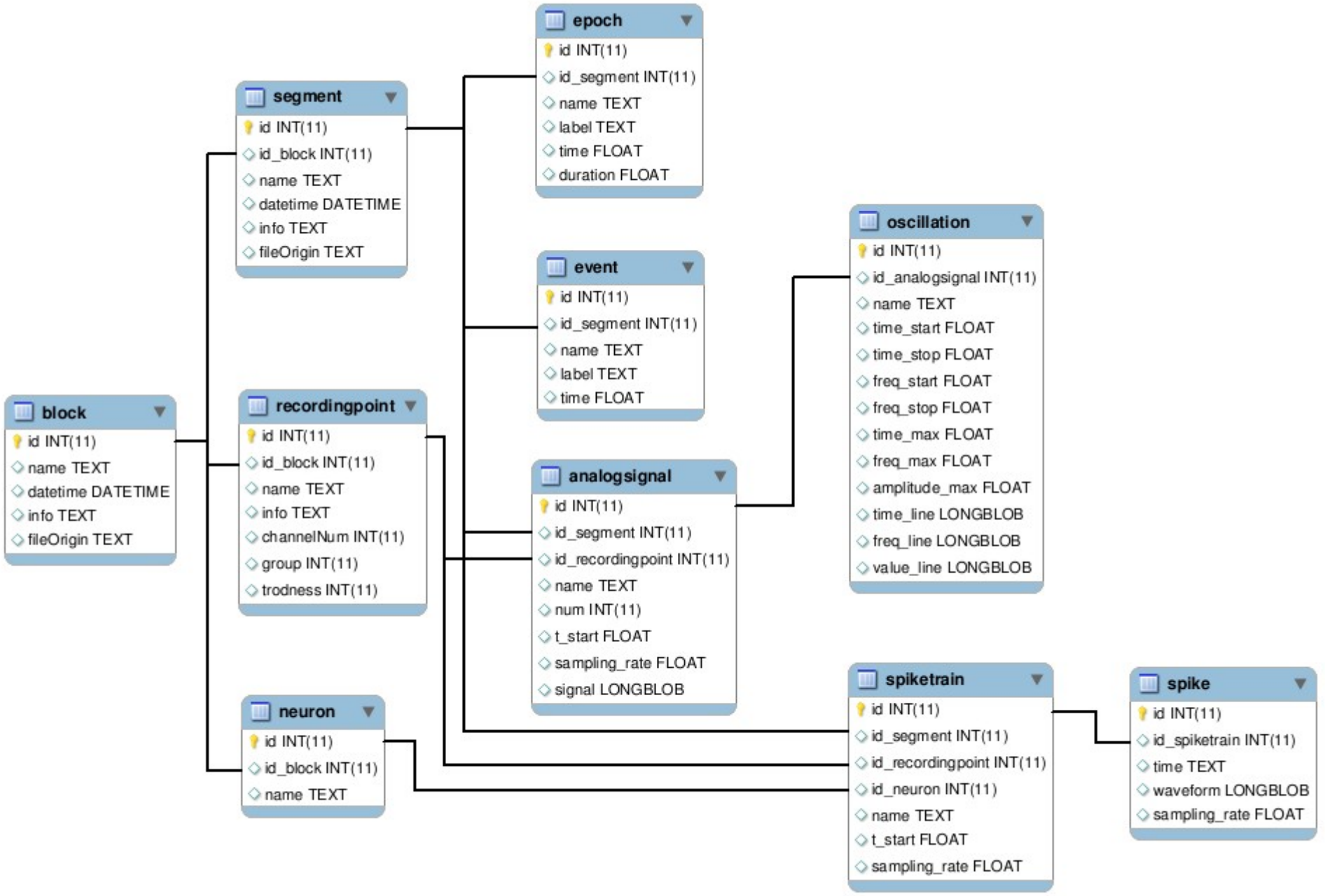
Neuron: A Neuron regroups all the SpikeTrain objects within a common Block, gathered accross several Segment, that has been emitted by the same cell.

Spike: one action potential carracterized by its tims and waveform.

Event: Object to represent ponctual time event. Util for managing trigger, stimulus, ...

Epoch: Similar than Event but with a duration. Util for dicribing a period, a state of a subject, ...

Class tour: schema



IO tour

First interest to have same classes :
Same API to read/write data files.

All formats are really different so we need a flexible API:
- ABF = Block+Segment+AnalogSignal+Event
- Plexon = Segment+SpikeTrain+Spike+AnalogSignal
- PyNN = SpikeTrain+AnalogSignal
- RAW = AnalogSignal

What is this API ?
- For each format you have an IO class
- The IO class can read or write one or several neo objects.

IO : tour

Pure python implementation of :

- PlexonIO
- Spike2IO
- NexIO
- AxonIO
- MicromedIO
- AsciiSignalIO
- EegLabIO
- ExampleIO
- RawIO
- WinWcpIO
- ElanIO
- PyNNIO
- PyNNBinaryIO
- AsciiSpikeIO
- ElphyDacIO
- NeuroshareIO (win32 only...)

Coming soon:
- EDF
- OpenElectrophy DB
- TDT
- New Elphy format
- G-node DB

IO tour : example

Examples of use

The basic syntax is as follow. If you want to load a file format which is implemented in a MyFormatIO class

```
>>> from neo.io import MyFormatIO
>>> file = MyFormatIO("myfile.dat")
```

To know what types of objects are supported by this io interface:

```
>>> file.supported_objects
[Segment , AnalogSignal , SpikeTrain, Event, Spike ]
```

Supported objects, do not meens objects that you can read directly. For instance, many formats supports AnalogSignal but you can't acces them directly : you must read a Segment and acces your AnalogSignal like that :

```
>>> seg = file.read_segment()
>>> seg.get_analogsignals()
```

To have the list of directly readeable objects :

```
>>> file.readable_objects
[Segment]
```

The first element of the previous list is the highest level for reading the file.

To read all the file :

```
>>> result = file.read()
>>> type(result)
neo.core.Segment
```

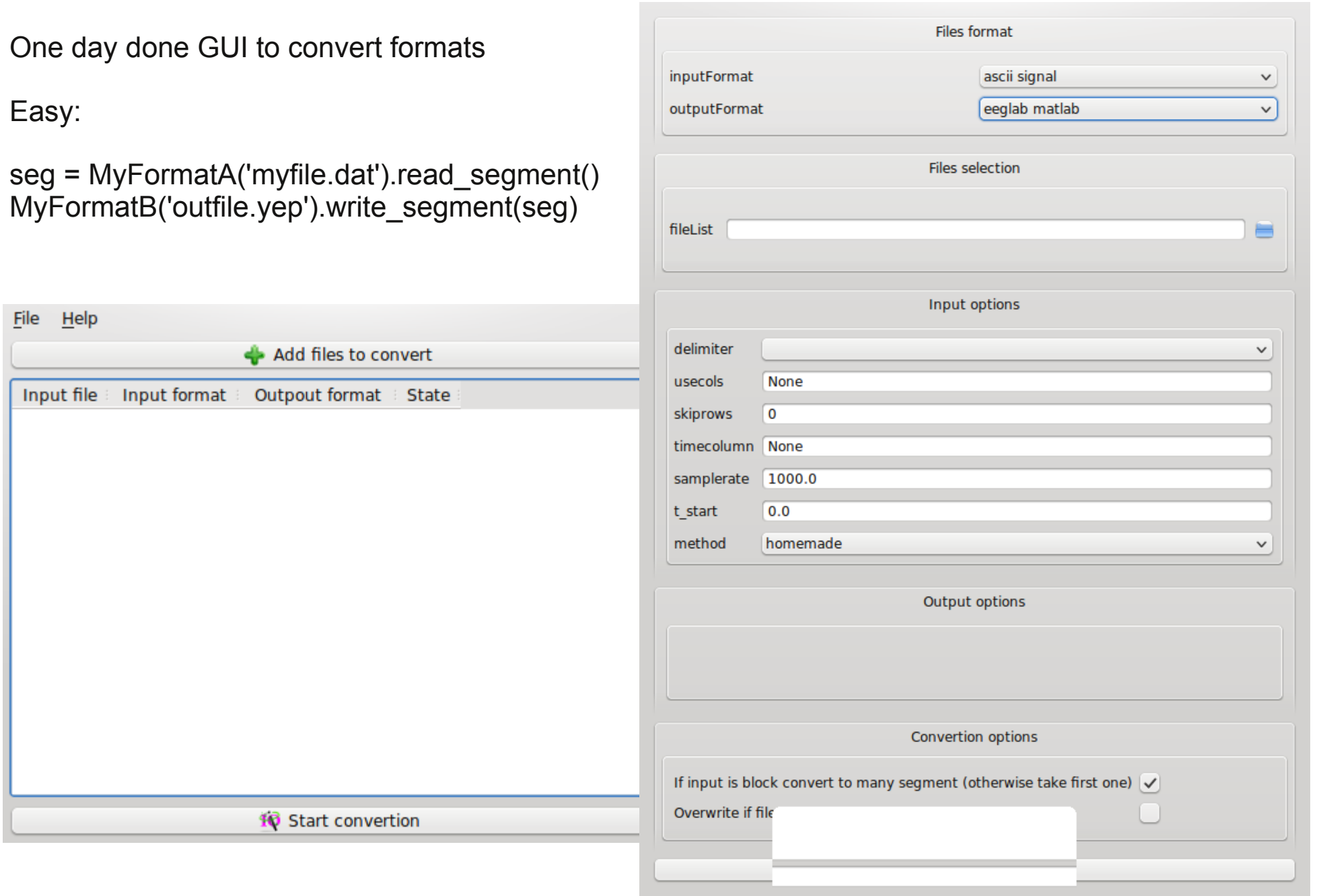
In this case, this is equivalent to : >>> seg = file.read_segment()

First application : NeuroConvert

One day done GUI to convert formats

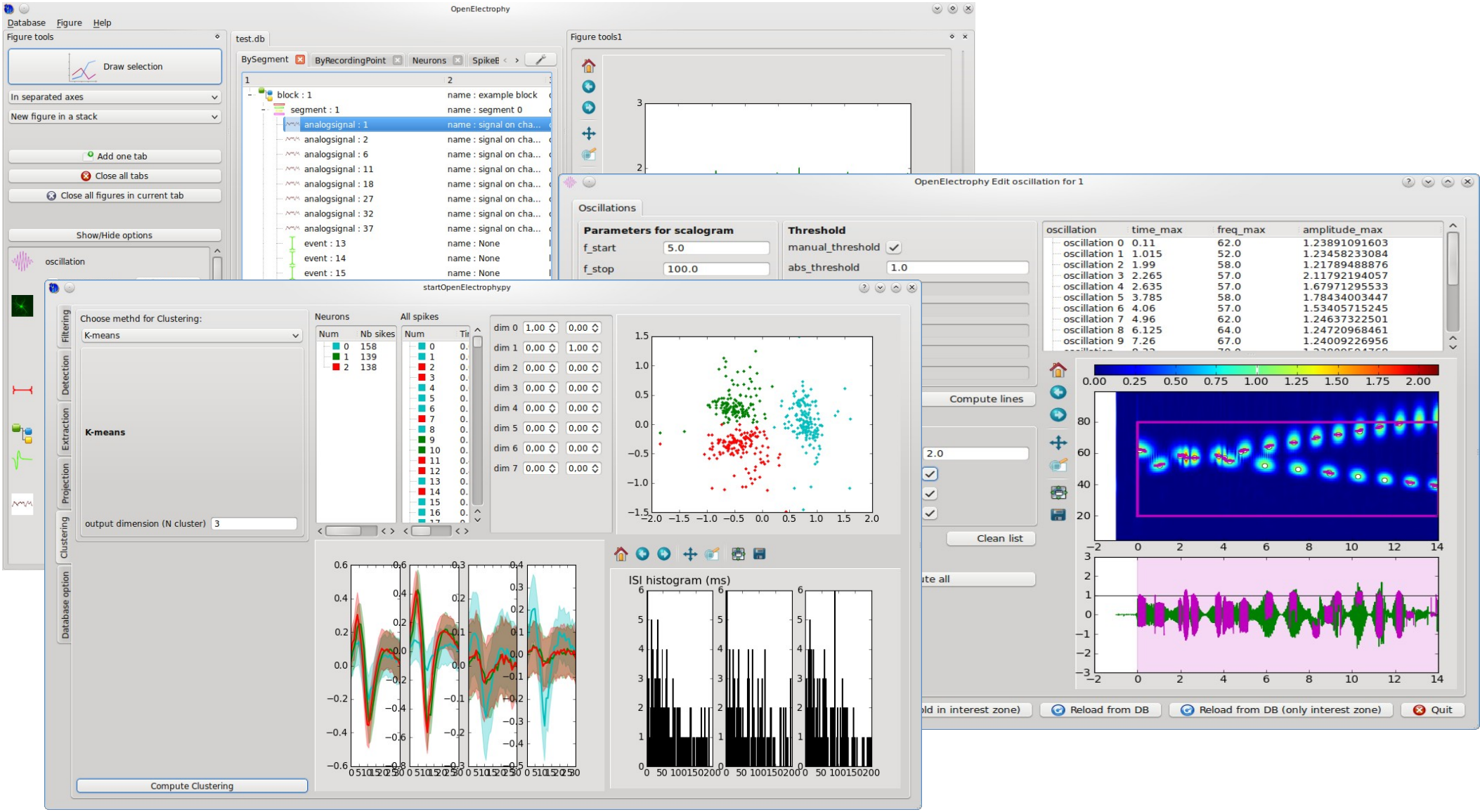
Easy:

```
seg = MyFormatA('myfile.dat').read_segment()
MyFormatB('outfile.yep').write_segment(seg)
```



Second work : re write from scratch OpenElectrophy (2 months)

= GUI + global DB storage + scripting module + spike sorting + time frequency



Goal

from neo import SpikeTrain , AnalogSignal , Segment, ...
from neo.io import PlexonIO, AxonIO , PynnIO, ...

from OpenElectrophy import SpikeTrain , AnalogSignal , Segment, ...
from OpenElectrophy.io import PlexonIO, AxonIO , PynnIO, ...

from Neurotools import SpikeTrain , AnalogSignal , Segment, ...
from Neurotools.io import PlexonIO, AxonIO , PynnIO, ...

from GoodFriendlyProject import SpikeTrain , AnalogSignal , Segment, ...
from GoodFriendlyProject.io import PlexonIO, AxonIO , PynnIO, ...

Conclusion

Support neo by:
- Using neo classes in yours projects
- Injecting new object (and definition!) in neo
- Implementing an IO for data file format used in your lab