Integrating data storage and annotation in the data workflow using the NIX format and libraries

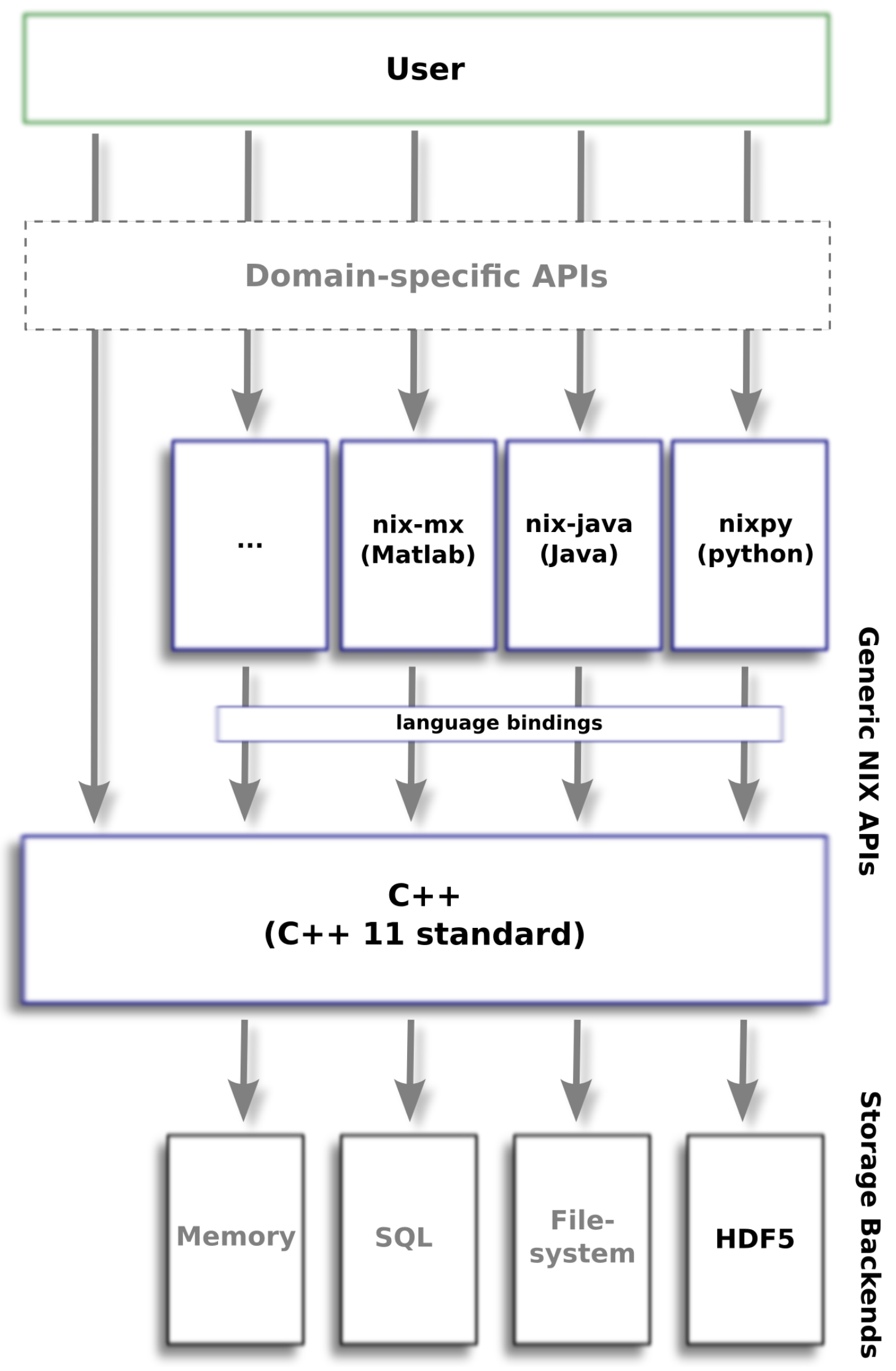
Adrian Stoewer [1], Christian Kellner [1], Jan Benda [2], Michael Sonntag [1]
Andrey Sobolev [1], Thomas Wachtler [1], and Jan Grewe [1,2]

[1]*German Neuroinformatics Node, Department Biologie II,*
Ludwig-Maximilians-Universität München, Germany;
[2]*Institut für Neurobiologie, Universität Tübingen, Germany*

Introduction

Increasing complexity of experimental approaches in neurosciences challenges methods for managing recorded data and metadata. Storing such information consistently is an essential part of experimental research and depends crucially on available file formats. Currently existing file formats are subject to several restrictions: some formats are vendor specific or only accessible via proprietary software. Others are highly domain specific, designed with respect to efficiency for certain kinds of data and therefore not versatile enough to be used in a wide variety of use cases. Moreover, many existing formats provide only limited support for storing metadata along with the data. The **NIX** project specifies a versatile format for neuroscientific data. It provides libraries for accessing these files from different platforms. NIX is based on a well defined data model which can be used to represent both **data** and related **metadata**. In particular, it provides **generic entities** designed to store a wide variety of data types like **continuous signals**, **spike events**, **image stacks**, or other **multi-dimensional data**. Metadata storage is supported via adaption of the **odML** data model[1].

The architecture



User

Domain-specific APIs

...

nix-mx (Matlab)

nix-java (Java)

nixpy (python)

language bindings

C++ (C++ 11 standard)

Generic NIX APIs

Memory

SQL

File-system

HDF5

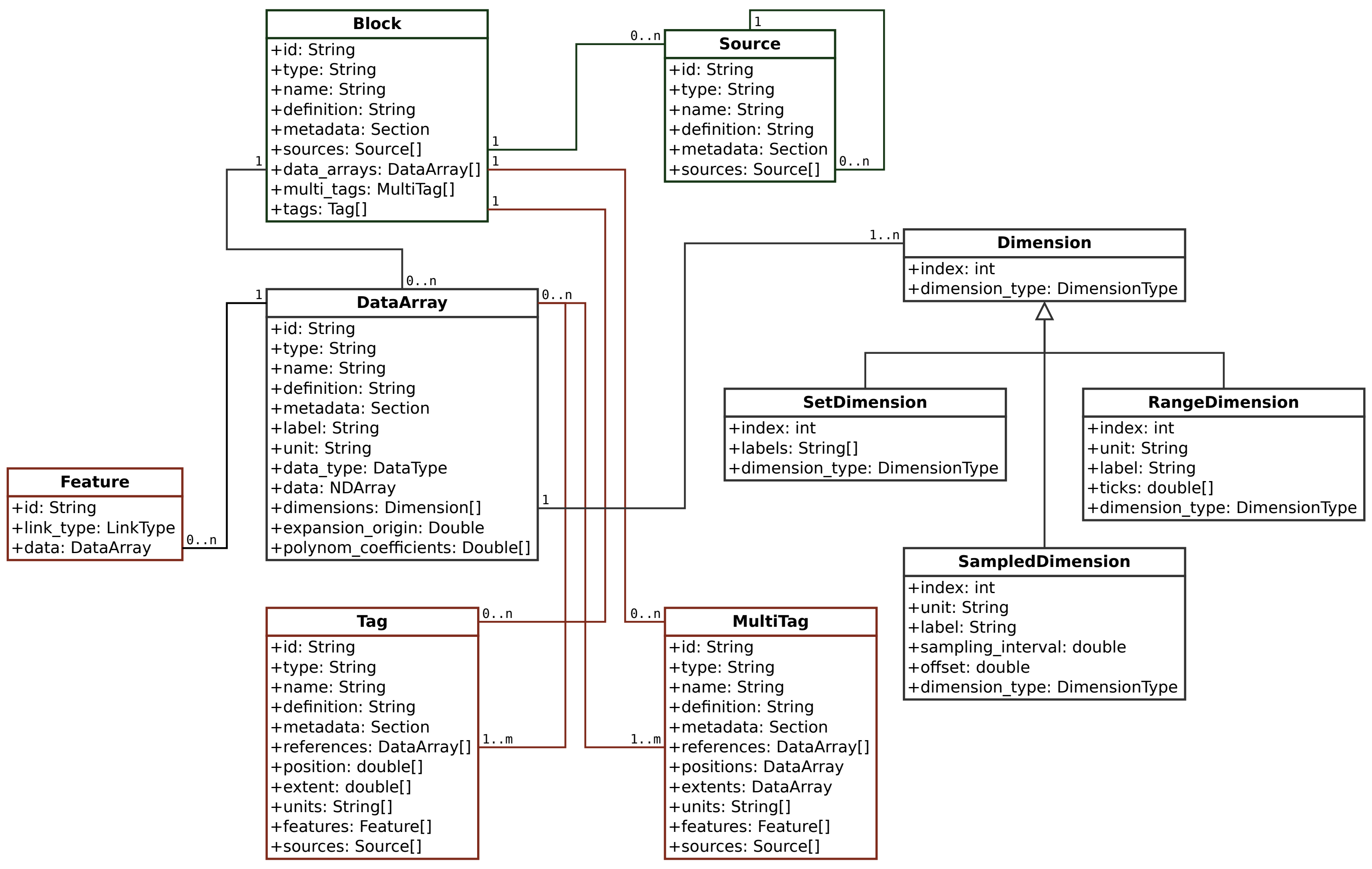
Storage Backends

- A generic data model as foundation of a flexible, yet concise file format specification
- NIX I/O libraries expose a generic programming model based on the original data model
- Domain-specific APIs can be implemented based on the NIX libraries

The data model

General features

- All entities have a unique **id**, a **name** and a **type**.
- The **id** allows synchronization and identification of entities accross files.
- The **name** serves as a human readable identifier.
- The **type** provides semantic context. It introduces domain-specificity.



Block
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+sources: Source[]
+data_arrays: DataArray[]
+multi_tags: MultiTag[]
+tags: Tag[]

Source
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+sources: Source[]

Dimension
+index: int
+dimension_type: DimensionType

DataArray
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+label: String
+unit: String
+data_type: DataType
+data: NDArray
+dimensions: Dimension[]
+expansion_origin: Double
+polynom_coefficients: Double[]

Feature
+id: String
+link_type: LinkType
+data: DataArray

SetDimension
+index: int
+labels: String[]
+dimension_type: DimensionType

RangeDimension
+index: int
+unit: String
+label: String
+ticks: double[]
+dimension_type: DimensionType

SampledDimension
+index: int
+unit: String
+label: String
+sampling_interval: double
+offset: double
+dimension_type: DimensionType

Tag
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+references: DataArray[]
+position: double[]
+extent: double[]
+units: String[]
+features: Feature[]
+sources: Source[]

MultiTag
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+references: DataArray[]
+positions: DataArray
+extents: DataArray
+units: String[]
+features: Feature[]
+sources: Source[]

DataArray

- Stores data in an n-dimensional array.
- Provides information about the data type and units of its values.
- Each dimension of a DataArray is defined using a Dimension entity, supporting both regularly and irregularly sampled data.

Block

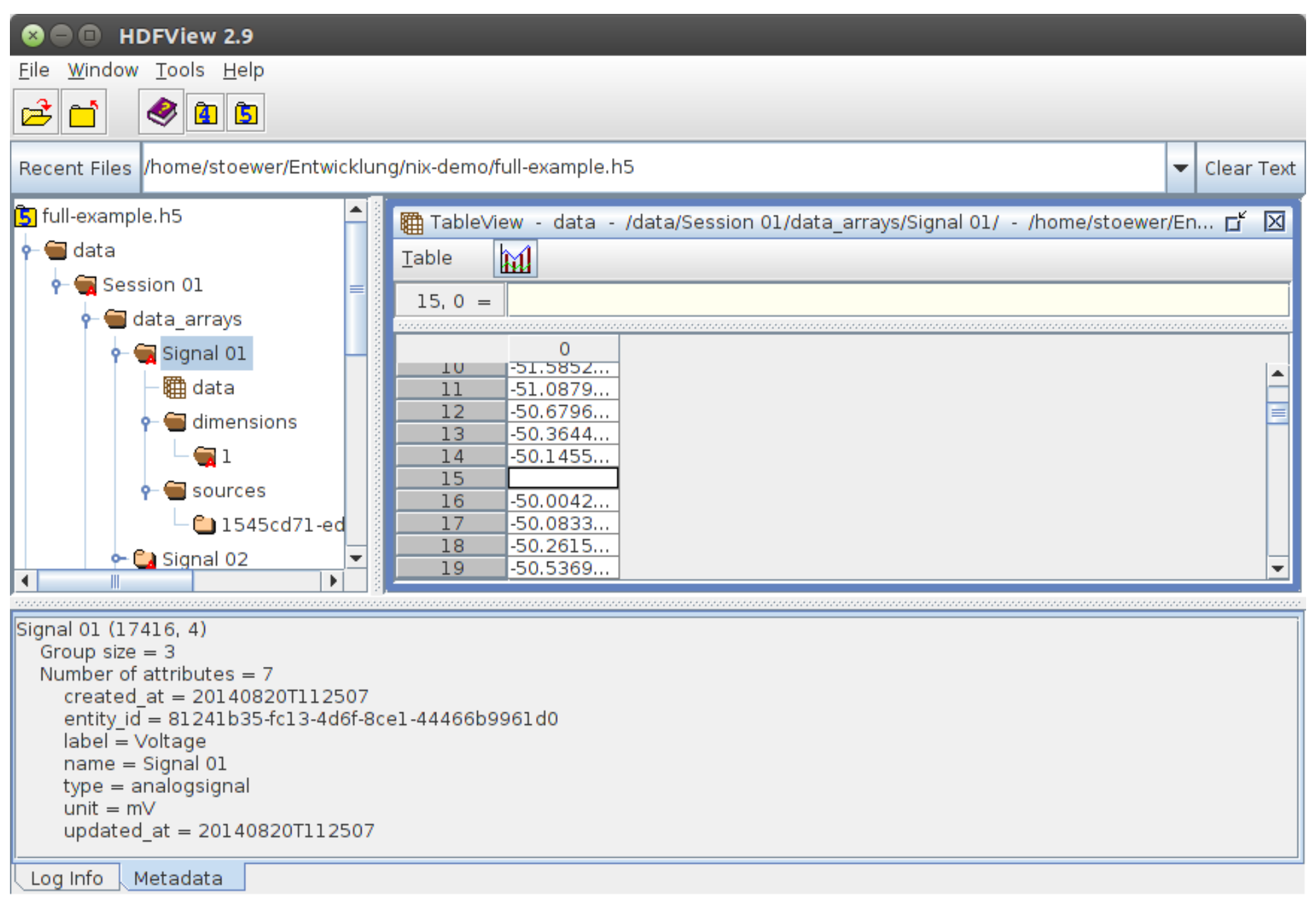
- Block entities group data elements that belong to a certain recording.
- Every other entity of the model has to be associated with exactly one Block.
- Each file may contain any number of Block entities.

Tag

- Defines regions or points of interest: segments, spike times, events ...
- Tags point into referenced DataArrays using position and extent vectors.
- A tag can be associated with data that is a feature of the tag.

HDF5 file schema

- The schema definition for HDF5 [2] represents all entities of the data model in a flat hierarchy.
- It was designed to be easily readable even without a special library.



/home/stoewer/Entwicklung/nix-demo/full-example.h5

full-example.h5
data
Session 01
data_arrays
Signal 01
data
dimensions
1
sources
1545cd71-ed
Signal 02

TableView - data - /data/Session 01/data_arrays/Signal 01/ - /home/stoewer/En...

15, 0 =

	0
10	-51.5852...
11	-51.0879...
12	-50.6796...
13	-50.3644...
14	-50.1455...
15	
16	-50.0042...
17	-50.0833...
18	-50.2615...
19	-50.5369...

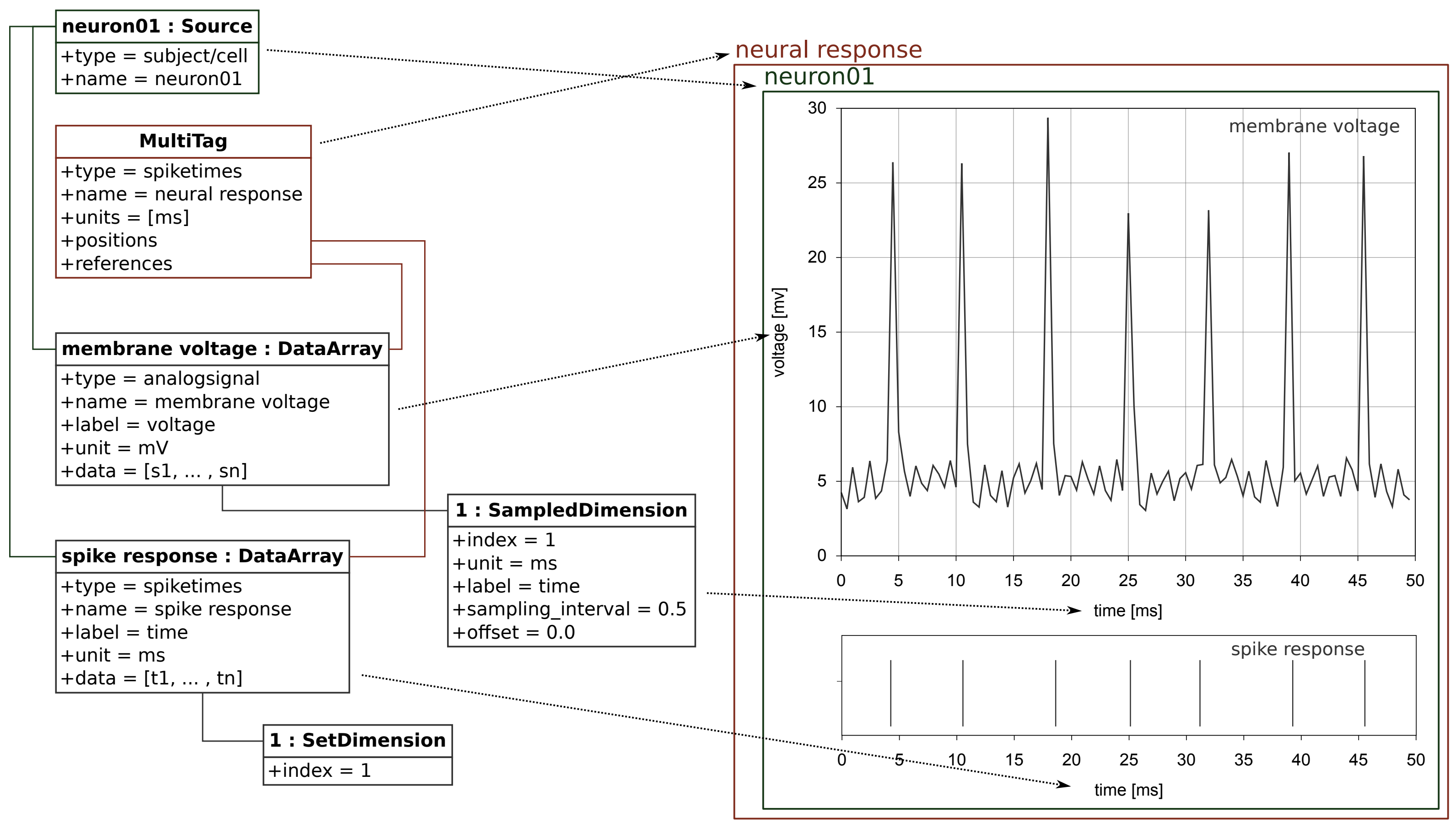
```
Signal 01 (17416, 4)
  Group size = 3
  Number of attributes = 7
    created_at = 20140820T112507
    entity_id = 81241b35-fc13-4d6f-8ce1-44466b9961d0
    label = Voltage
    name = Signal 01
    type = analogsignal
    unit = mV
    updated_at = 20140820T112507
```

Example: Using the model to represent electrophysiology data

This example shows how the model is used to represent the recording of a continuous voltage signal along with a derived series of spike times.

- The voltage signal, including units, labels, etc. is stored in a DataArray.
- The spike times are stored in a second DataArray.
- The spike train and neural response are linked together using a MultiTag.

Note: the model provides all information to interpret the data correctly



neuron01 : Source
+type = subject/cell
+name = neuron01

MultiTag
+type = spiketimes
+name = neural response
+units = [ms]
+positions
+references

membrane voltage : DataArray
+type = analogsignal
+name = membrane voltage
+label = voltage
+unit = mV
+data = [s1, ... , sn]

1 : SampledDimension
+index = 1
+unit = ms
+label = time
+sampling_interval = 0.5
+offset = 0.0

spike response : DataArray
+type = spiketimes
+name = spike response
+label = time
+unit = ms
+data = [t1, ... , tn]

1 : SetDimension
+index = 1

Libraries and language bindings

To read and write data from and to the NIX file format, even without deep knowledge about the exact format specification, the NIX project provides an **IO-library** written in **C++**[3]. The library supports major controllers and operating systems such as **Linux**, **OSX** and **Windows**. Other programming languages and platforms are supported via bindings to the C++ library for Python[4], Matlab[5] and Java[6].

Resources

This work was performed in connection with the activities of the HDF5 working group of the INCF Electrophysiology Data Sharing Task Force
Supported by BMBF grants 01GQ0801 and 01GQ1302

Contact: ***adrian.stoewer@rz.ifi.lmu.de***

[1] Grewe et al (2011), Frontiers in Neuroinformatics 5:16
[2] https://github.com/G-Node/nix/wiki/Implementation-in-HDF5
[3] https://github.com/G-Node/nix
[4] https://github.com/G-Node/nixpy
[5]https://github.com/G-Node/nix-mx
[6] https://github.com/G-Node/nix-java