NIX - Comprehensive Storage of Neuroscience Data and Metadata

Adrian Stoewer

12. Jan. 2015

Overview

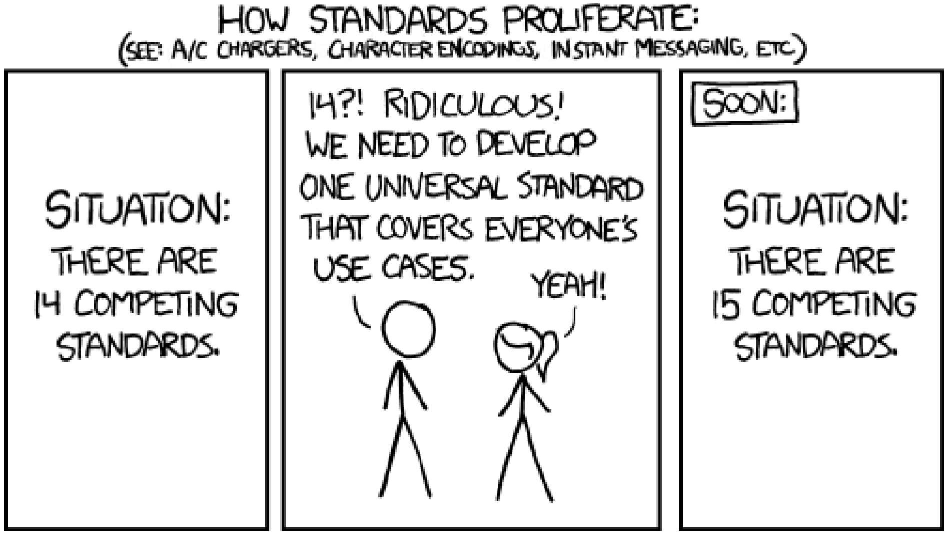
- About the NIX project
- The data model
- Format specification
- The C++ API
- Language bindings
- Tools for NIX
- Outlook

About the NIX Project

- Flexible file format for neuroscience data and metadata
- Started after 2012 INCF Congress
  - INCF Ephys Data Sharing Task Force
  - Hackathon after the congress
- Project goals:
  - A flexible format based on HDF5
  - Development of a data model that can also be used with other back-ends
  - Development of a reference implementation in C++

Why another file format?

- Many formats are proprietary and not open source
- Formats are often poorly documented
- Existing open formats do not support all kinds of data
- Existing open formats do not support complex metadata

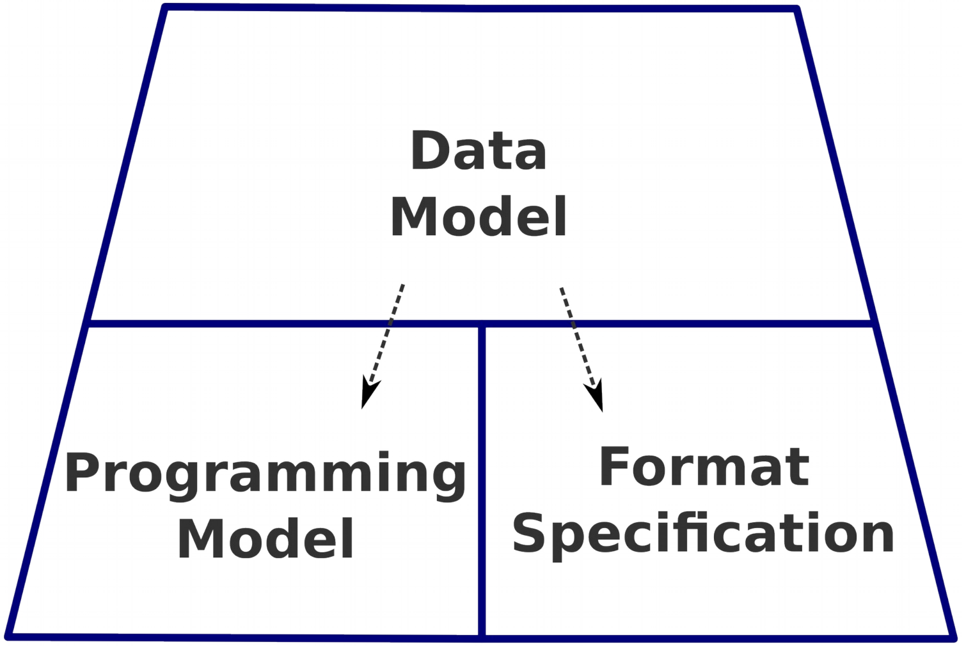

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

What do we need to store?

- Time series data
  - Regularly and irregularly sampled data
- Event data
  - Neural events, behavioral events
- Spatial data
  - Gaze directions, trajectories
- Image data
- Movie data

The NIX Approach

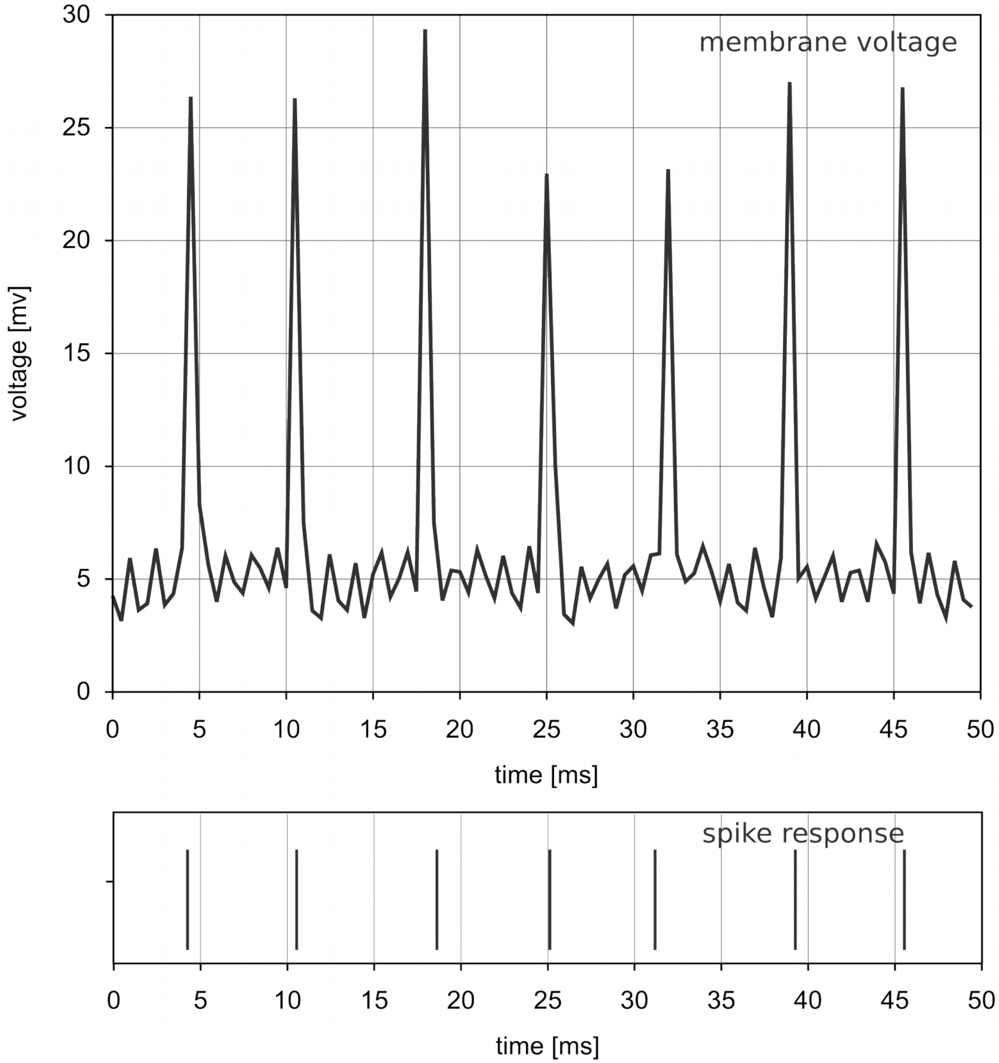
- Define a data model
- Derive from the model …
  - The format specification
  - The programming model / API



The NIX Data Model

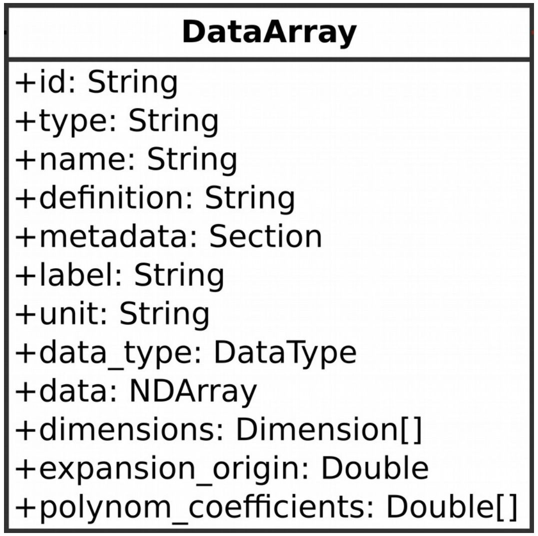
- Flexible data model
  - Generic enough to support various kinds of data
  - Explicit enough to create a meaningful visualization
  - Provides entities to describe the stored data
- Full metadata integration using the odML metadata format
- Fulfils INCF Ephys Data Sharing Task Force requirements for storing electrophysiology data

The NIX Data Model



Data Model: Data Arrays and Dimensions

- Main entity for storing data
- Stores data in n-dimensional array
- Provides unit, data type and label for the stored values
- Has a dimension descriptor for each dimension of the data array
- Additional metadata via link to an odML section
- Value scaling

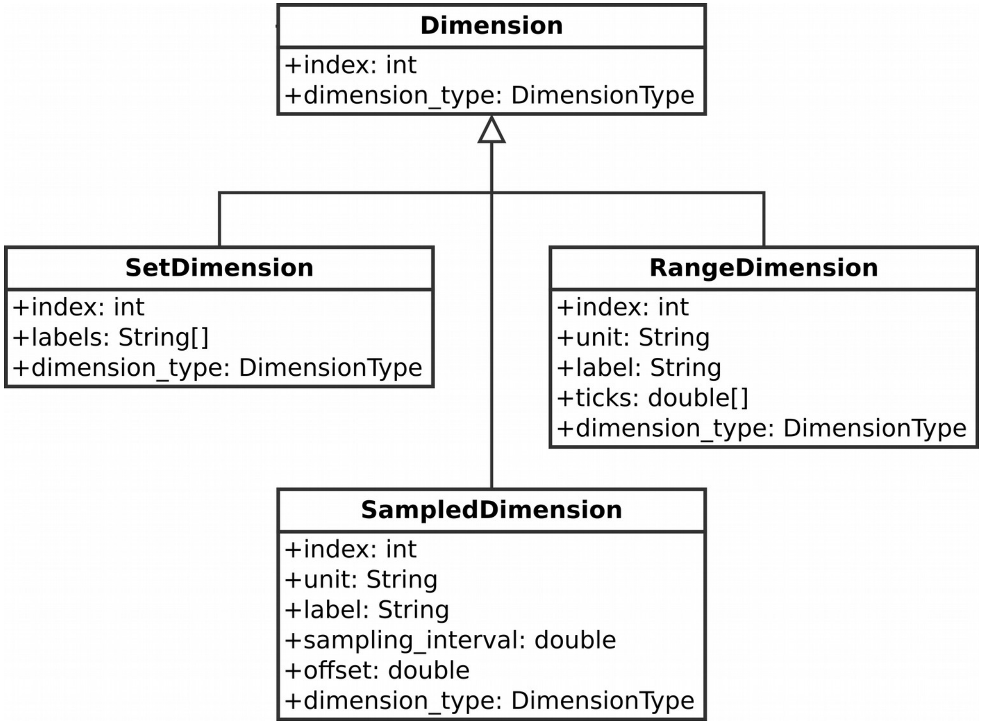


DataArray

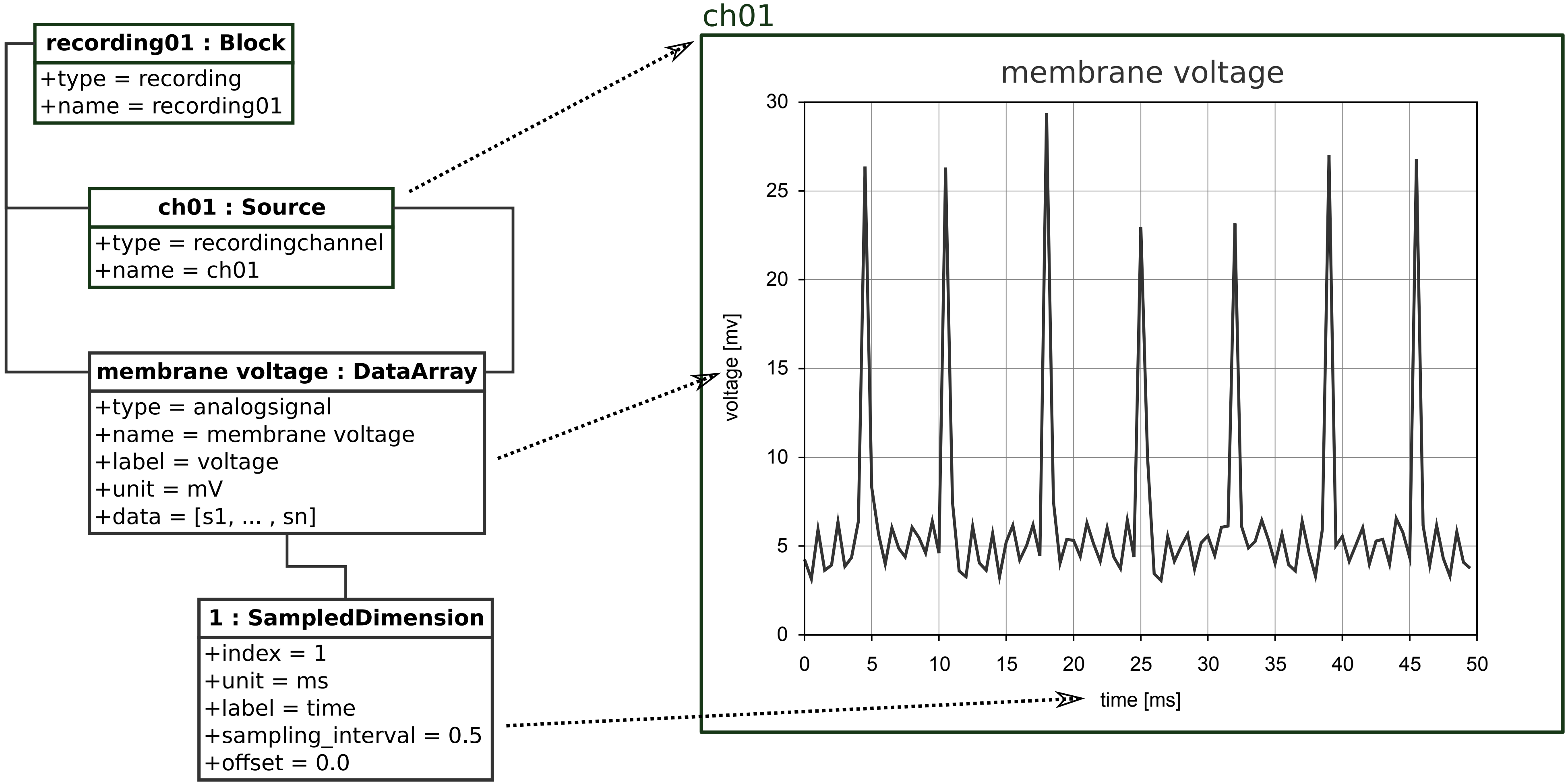
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+label: String
+unit: String
+data_type: DataType
+data: NDArray
+dimensions: Dimension[]
+expansion_origin: Double
+polynom_coefficients: Double[]

Data Model: Data Arrays and Dimensions

- Describe the dimensions of data in a DataArray entity
- Three different kinds of dimensions
    - Sampled
    - Range
    - Set



Data Model: Data Arrays and Dimensions



recording01 : Block
+type = recording
+name = recording01

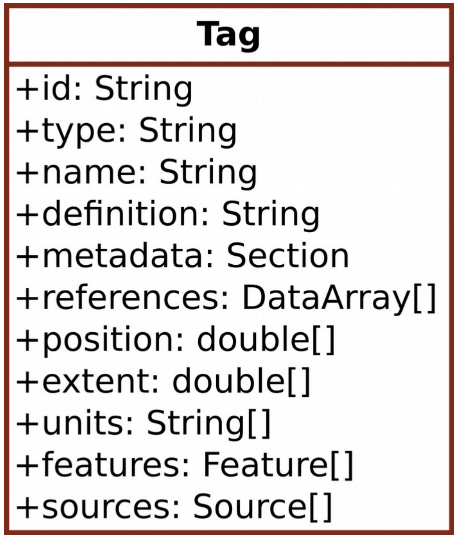
ch01 : Source
+type = recordingchannel
+name = ch01

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

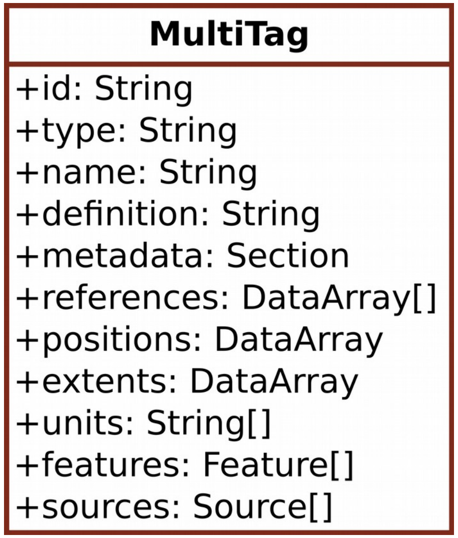
Data Model: Tags

- Can define points or regions of interest
- Examples
  - Events
  - Spikes
  - Epochs
- MultiTag can use DataArray entities to define positions and extents



Tag

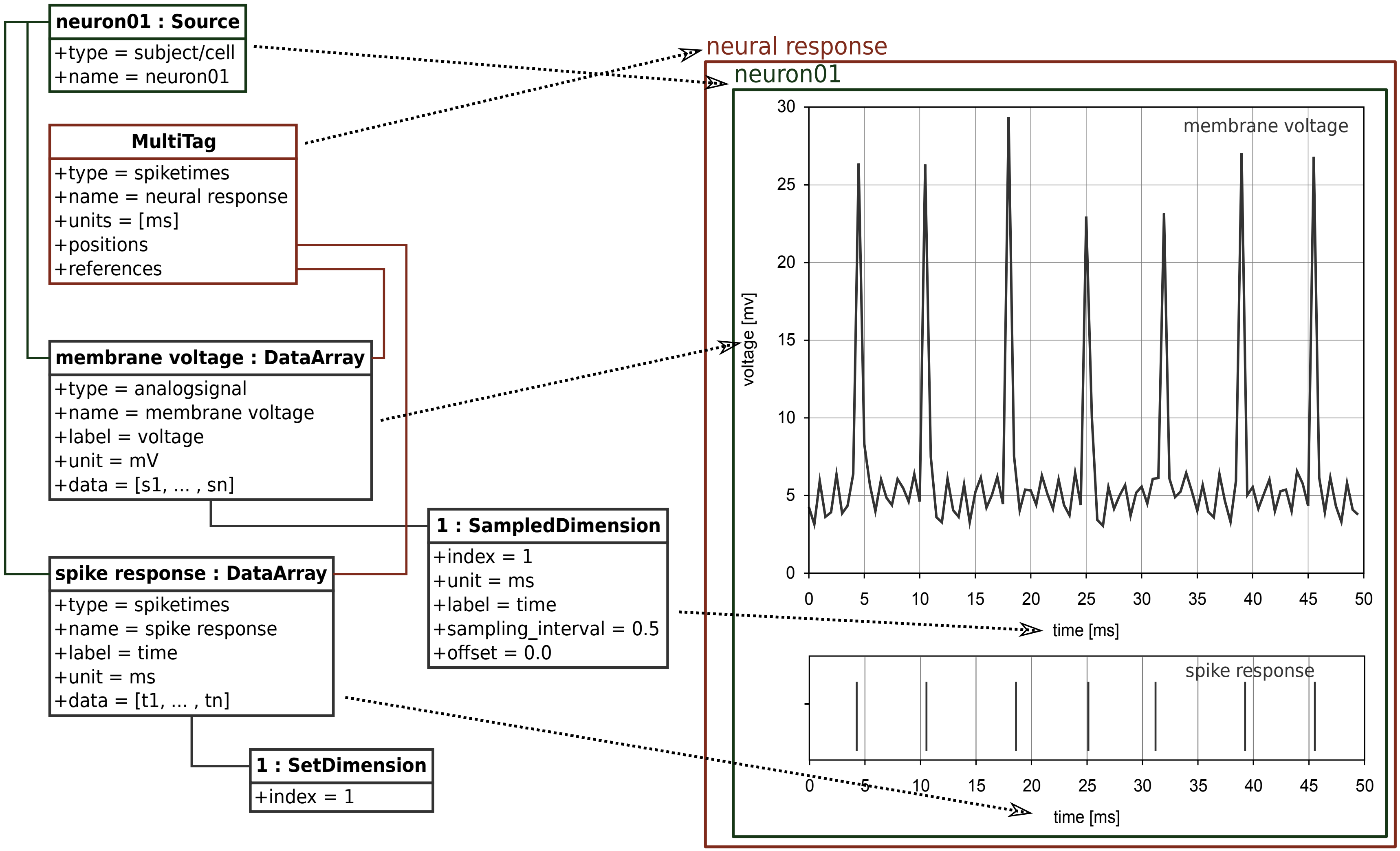
+id: String
+type: String
+name: String
+definition: String
+metadata: Section
+references: DataArray[]
+position: double[]
+extent: double[]
+units: String[]
+features: Feature[]
+sources: Source[]



MultiTag

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

Data Model: Tags



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

Data Model: Other Entities

- Blocks
  - Top level entity
  - Groups all other entities
- Sources
  - Hierarchical
  - Can be used to describe the provenance of other entities

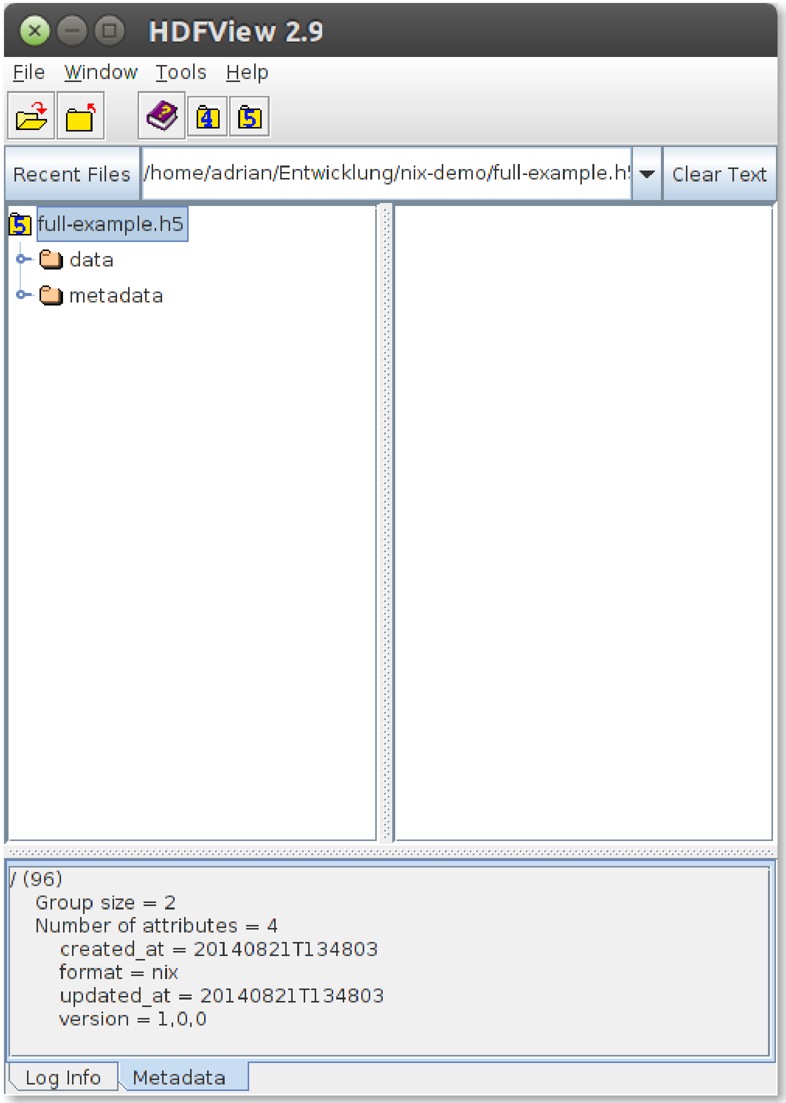
Short demonstration …

NIX Format

- Specification based on HDF5
- File structure reflects the data model
- Easy to understand and read
  - Entities are just HDF5 groups
  - Human readable identifiers
  - HDF5 links
- Each entity type resides at a specific location in the file
- Entity properties are HDF5 attributes

NIX Format: File Root

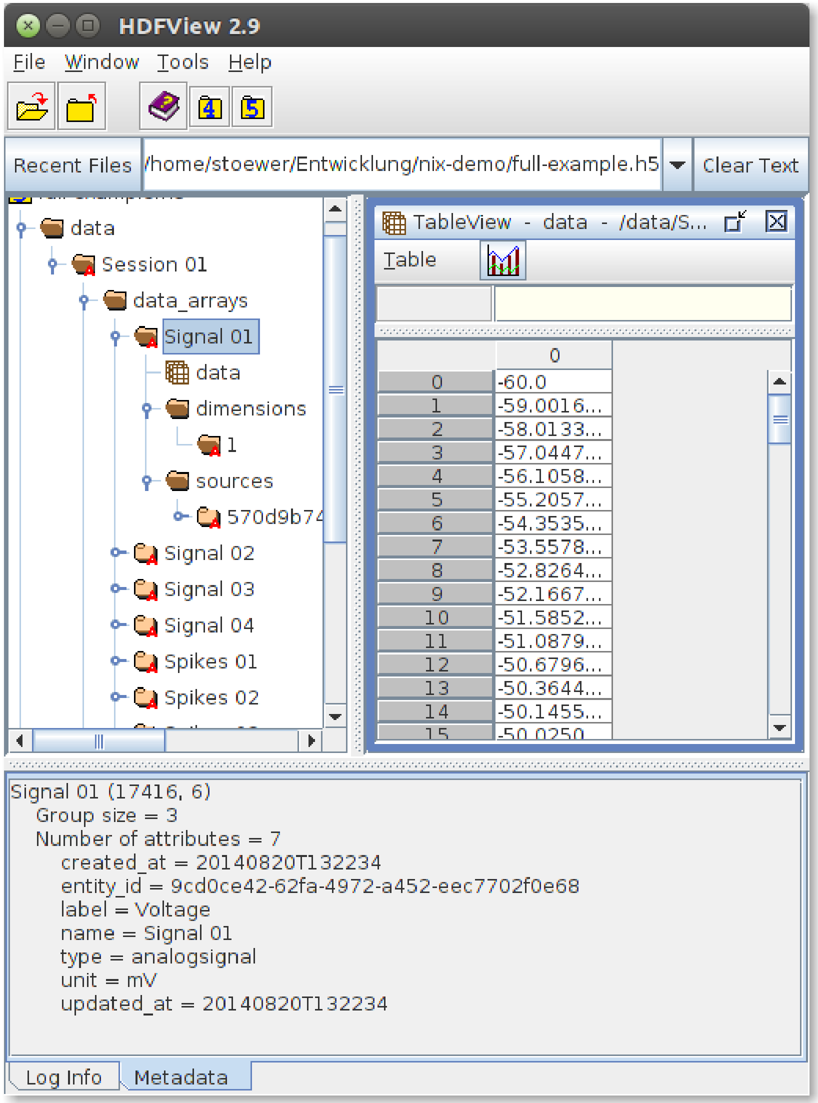
- File root attributes
  - Format hint
  - Format version
  - Update and creation time
- Subgroup *metadata*
  - Contains all odML root sections
- Subgroup *data*
  - Contains block entities



```
/ (96)
  Group size = 2
  Number of attributes = 4
    created_at = 20140821T134803
    format = nix
    updated_at = 20140821T134803
    version = 1,0,0
```

NIX Format: DataArray

- HDF5 dataset for the actual data
- Subgroup for dimension descriptors
- Subgroup for links to Source entities



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

	0
0	-60.0
1	-59.0016...
2	-58.0133...
3	-57.0447...
4	-56.1058...
5	-55.2057...
6	-54.3535...
7	-53.5578...
8	-52.8264...
9	-52.1667...
10	-51.5852...
11	-51.0879...
12	-50.6796...
13	-50.3644...
14	-50.1455...
15	-50.0250

```
Signal 01 (17416, 6)
    Group size = 3
    Number of attributes = 7
        created_at = 20140820T132234
        entity_id = 9cd0ce42-62fa-4972-a452-eec7702f0e68
        label = Voltage
        name = Signal 01
        type = analogsignal
        unit = mV
        updated_at = 20140820T132234
```

The NIX API

- Convenient IO API, utility functions
- Implemented in C++11
  - 1,854 commits / 17,069 lines of actual code
- Well documented
- Linux, MacOS X & Windows support
- Debian packages in a PPA, Windows binaries
- Hosted on GitHub
  - Open source (BSD license)
  - https://github.com/G-Node/nix

Language Bindings

- Python
  - Complete and stable

    https://github.com/G-Node/nixpy
- Matlab
  - Early stage
  - High priority

    https://github.com/G-Node/nix-mx
- Java
  - Proof-of-Concept stage

    https://github.com/G-Node/nix-java

Tools

- Command line tool
  - File browsing
  - Validation of files
  - Benchmark tool
- Recording
  - RELACS (http://relacs.sourceforge.net)

Outlook

- High-level API for electrophysiology
- RDF and ontology integration for metadata
- Better provenance tracking
- Better command line tools (plotting)
- More language bindings (Julia)

NIX Developers

- Adrian Stoewer, Programmer, G-Node

  Design and implementation of the data model, format, API and python bindings

- Jan Grewe, Neuroscientist, Uni Tübingen

  Design and implementation of the data model, format and API

- Christian Kellner, Neuroscientist & Programmer, LMU Munich

  Design and implementation of the API, python and matlab bindings

- Balint Morvai & Andrey Sobolev, Programmers, G-Node

  Implementation of the API