University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Bachelor's thesis

Software tool for standardization of electrophysiological data

Pilsen 2019

Jan Rychlík

Místo této strany bude zadání práce.

Declaration

I hereby declare that this bachelor's thesis is completely my own work and that I used only the cited sources.

Pilsen, 30th April 2019

Jan Rychlík

Acknowledgment

I would like to thank my supervisor Ing. Roman Mouček Ph.D. for his advice, willingness and patience and also I would like to thank G-Node employees for their support.

Abstract

The standardization of electrophysiological data and the choice of the format for storing them is still a relevant problem. Such a standard will allow the sharing of data and results through research centers, thereby increasing the efficiency of research itself. At the university portal, EEG-Base, the data is stored in an outdated format. The aim of this bachelor thesis was to explore possibilities and choose a newer data format. The products of the German node for neuroinformatics and their NIX and odML formats have proved to be suitable. Then EEGBaseToNIX was created to transform outdated formats into new ones. The tool was tested and then used to convert data.

Abstrakt

Standardizace elektrofyziologických dat a volba fomátu pro jejich ukládání je v současné době stále aktuálním problémem. Takový standard umožní sdílení dat a výsledků mezi výzkumnými centry, a tím zvýšení efektivity samotného výzkumu. Na univerzitním portálu, EEG-Base, jsou data uložena v zastaralém formátu. Cílem bakalářské práce bylo prozkoumat možnosti a zvolit novější datový formát. Jako vhodné se ukázaly produkty německého uzlu pro neuroinformatiku a jejich formáty NIX a odML. Poté byl vytvořen nástroj EEGBaseToNIX k transformaci zastaralých formátů na nové. Nástroj byl řádně otestován a následně byl použit k převedení dat.

Contents

1 Introduction

Studying the human brain is a complex, important and demanding task. To understand at least some processes ongoing in the brain takes a lot of time. The human brain is complex in its neuronal connectivity and move-speed of neurological signals that underlie brain activity. Non-invasive and relatively cheap approaches used for the study of the human brain also include the methods and techniques of electroencephalography (EEG) and event-related potentials (ERPs).

The research into the human brain using these methods and techniques is slowed down by many factors. They include low–quality technical devices to acquire data. Also, the device itself used for measuring brain activity is expensive. Another important factor that slows down the research in the field is the lack of well-annotated experimental data. It has its historical context because a well known and broadly accepted standard has never existed, and best practices to describe and store data have not been known or accepted by the scientific community. It has led to the situation that nearly every institution which has collected, described and stored data from brain activity has come with its own structure of data and metadata and software tools to work with them. In general, it has cost a lot of money.

One of the promising solutions of this issue comes from the German Node for Neuroinformatics (G-Node) which has proposed and developed the markup language and structures for storage of electrophysiological data called NIX and for metadata called odML (open metadata markup language). OdML has become considered as a promising standardization effort in electrophysiology, electroencephalography and event-related potentials. G-Node develops and maintains a number of libraries written in Python for working with odML.

Currently increases the number of institutions interested in developing more unified data and metadata description in neuroscience/electrophysiology. All of them have the same goal to enable sharing data in a standard format that is recognised worldwide. Except for odML and NIX, which have been recently introduced, there is a NWB (Neuro data Without Borders) initiative in the US and the format called BIDS (Brain Imaging Data Structure). This is developed at McGill University in Montreal, Canada, and enriched by its eeg extension. More information about standardization initiatives and efforts in electrophysiology is given in chapter 2.

The data stored at the EEG/EPR portal (EEG-Base, University of West

Bohemia) has been continuously saved not using a stable structure. The markup language odML has started to be used a short time ago. In the past, the data was saved using a structure based on the experience of the research group at the University of West Bohemia. The work with data and their visualization were too complex to be easily used. The future maintenance and extension of the EEG/ERP portal is a time-consuming work and existence of third-party libraries is broadly developed in Python, so it is necessary to develop a new tool for transformation of saved data into the odML/NIX structure.

Chapter „State of the Art“ handles institutes, organizations, groups, and tools, which are involved in the works with the electrophysiology data. In the third chapter – „Software requirement specification“ - is exactly defined what the developed tool has to do. Chapter „Architecture and design“ analyses the task and describes packages, modules and scripts. The next part of the document describes the overall implementation and introduces important methods. The sixth part is about testing possibilities and subsequent results.

2 State of the Art

This chapter deals with the state of the art in the field and provides information about organizations that are active in sharing open scientific/electrophysiology data. International Neuroinformatics Coordinating Facility (INCF) provides support for neuroscience data and describes the best informatics practices for neuroscience. The second important international organization is FAIR. This organization provides interested users with the principles on how to create appropriate open data. G-Node is a Germany institute which develops tools for work with scientific/electrophysiology data. NIX is the product of G-Node and represents a general structure for time series data. The following product of G-Node is odML. OdML saves metadata and connects them to the data saved in NIX.

2.1 INCF

The mission of the International Neuroinformatics Coordinating Facility (INCF) is to be an independent international facilitator catalyzing and coordinating the global development of neuroinformatics, and advancing training in the field. The goal of neuroinformatics is to integrate and analyze diverse data across scales, techniques, and species to understand the brain, and positively impact the health, and well being of society. INCF promotes the implementation of neuroinformatics and advances data reuse and reproducibility in brain research through[11]:

The INCF rulers [11]:

- the development and endorsement of global community standards and best practices
- leading the development and provision of training and educational resources in Neuroinformatics
- promoting open science and the sharing of data and other resources to the international research community
- partnering with international stakeholders to promote and prioritize neuroinformatics at global, national and local levels
- engaging scientific, clinical, technical, industry, and funding partners in collaborative, community-driven projects

2.2 FAIR

One of the grand challenges of data-intensive science is to facilitate knowledge discovery by assisting humans and machines in their discovery of, access to, integration and analysis of, task-appropriate scientific data and their associated algorithms and workflows. Here, we describe FAIR - a set of guiding principles to make data Findable, Accessible, Interoperable, and Reusable. The term FAIR was launched at a Lorentz workshop in 2014. The resulting FAIR principles were published in 2016. [2]

Based on 15 principles under, a set of 14 metrics have been defined to quantify levels of FAIRness. The latest developments on FAIR are available at GO-FAIR. [2]

FAIR principles[2]

To be Findable:

- F1. (meta)data are assigned a globally unique and persistent identifier
- F2. data are described with rich metadata (defined by R1 below)
- F3. metadata clearly and explicitly include the identifier of the data it describes
- F4. (meta)data are registered or indexed in a searchable resource

To be Accessible:

- A1. (meta)data are retrievable by their identifier using a standardized communications protocol
  - A1.1 the protocol is open, free, and universally implementable
  - A1.2 the protocol allows for an authentication and authorization procedure, where necessary
- A2. metadata are accessible, even when the data are no longer available

To be Interoperable:

- I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
- I2. (meta)data use vocabularies that follow FAIR principles
- I3. (meta)data include qualified references to other (meta)data

To be Reusable:

- R1. meta(data) are richly described with a plurality of accurate and relevant attributes
  - R1.1. (meta)data are released with a clear and accessible data usage license
  - R1.2. (meta)data are associated with detailed provenance
  - R1.3. (meta)data meet domain-relevant community standards

2.3 NIX

2.3.1 About

The basic idea of the NIX project is to come up with a generic data model that defines as few structures/entities as possible while being able to represent a the NIX project started as an initiative of the Electrophysiology Task Force which is part of the INCF Data sharing Program. As such the project aims to develop standardized methods and models for storing electrophysiology and other neuroscience data together with their metadata in one common file format based on HDF5.

In order to achieve this, NIX uses highly generic models for data as well as for metadata and defines standard schemata for HDF5 files which can represent those models. Last but not least NIX aims to provide a convenient C++ library to simplify the access to the defined format.[6]

Requirements for declaration as INCF standard [6]:

- Such a standard should define a clear data model that exists outside the concrete implementation of a file format.
- It should be able to represent any data and metadata that are used by now or will be used in the near future. Thus, it should allow to work with more than analog signals, but also other derived or related data like histograms and gures etc.
- It does not exclude any feature that is already available in existing models or formats.
- It is easily convertible and compatible to other formats.

- All available formats have no or only limited support for metadata, therefore a new standard will allow the integration of arbitrary metadata and should provide means to annotate data with them.
- It has to be as exible as possible but still suited for automated processing and evaluation of data.

Linking data and metadata

Linking data and metadata is necessary because neurophysiological data ever lives in their metadata context. Annotating data by metadata must be possible. On the other hand, the metadata does not have to be accessible to everyone and data has to make some sense.

2.3.2 Data Model

The data model used by NIX represents time series data stored without any loss of information. The NIX was also designed to mark up data with metadata in the odML format.

The model for data marked up with metadata is specific for electrophysiology, but both models can be linked to predefined or custom terminologies which enable the user to give elements of the models a domain-specific, semantic context. [5]

Model of Data

The NIX model for data consists of six main elements: Block, DataArray, Tag, MultiTag, Source, and Group.

- **Block** -A Block is the top level grouping element for data objects that somehow are related to each other. It is a requirement that every data object has to be associated with one Block object. This way a block can be seen as something that represents a dataset or the combined results of an experiment.[3]
- **DataArray** - The core of the data model is the so called DataArray. Its main purpose is to store arbitrary raw data inside an n-dimensional array. Furthermore the DataArray (see Table) provides means to describe the physical nature of the stored data.[3]

- **Tag** - Tag entities are mainly used to define regions of interest in data inside one or more DataArrays. The Tag defining more than one entity is called **MultiTag**.
- **Source** - Source entities can define the provenance of a Tag or DataArray.[3]
- **Dimension** - The Dimension entities are used to define what the dimensions of the data represent. All realizations of Dimension have a label field providing a textual feature for the axis.
- **Group** - The Group acts as an element which, in its current form, just expresses, that the members of the group (dataArrays, tags, and multi tags) somehow belong together. A Group can link to Sources and can have metadata attached to it.

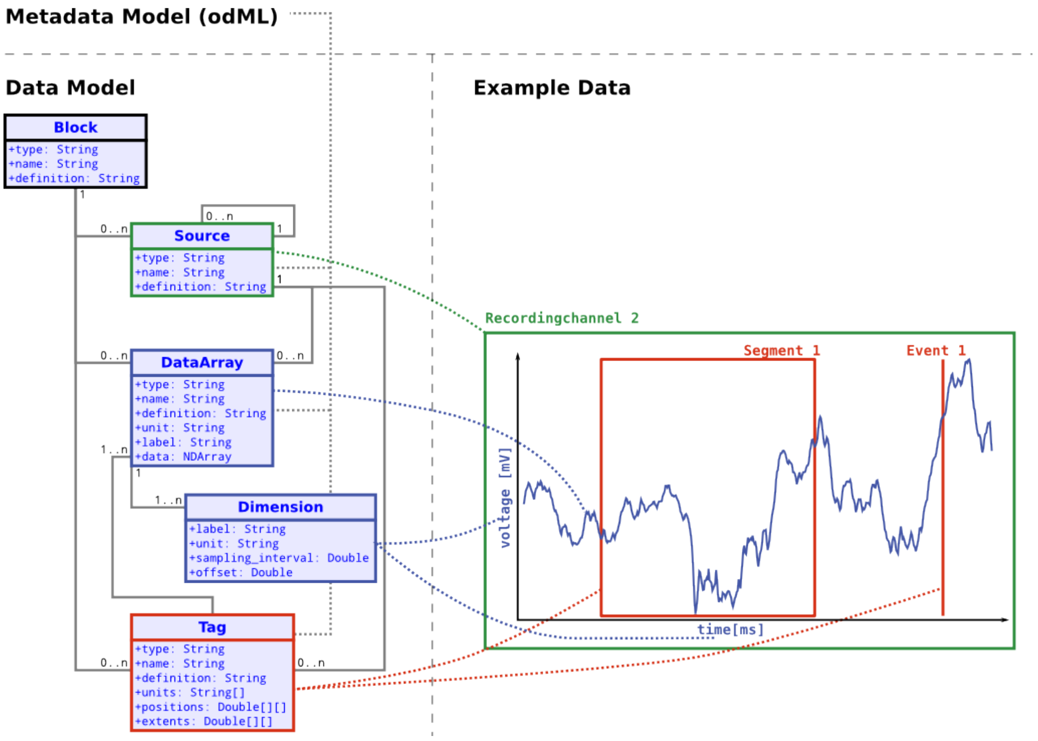


Figure 2.1: Example of the data model with odML and NIX connection. [3]

2.3.3 NIXpy

The NIXpy is a python G-Node project that provides python libraries to work with nix files. The package to download and import to python is called *nixio*. Including the version 1.4 the NIXpy package offers an interface for python.This version still needs NIX libraries implemented in C++ for proper functionality. From version 1.5 it is fully implemented in python including all NIX features. [8]

2.3.4 Hierarchical Data Format Version 5

Hierarchical Data Format version 5 (HDF5) is a key format for NIX files. HDF5 is an open source data format similar to XML files. HDF5 was developed for storing extremely large data collections. The principal of storing data is using folders like a structure. This allows us to work with data in many different ways. The HDF5 data format enables the making of metadata by *seld-describing* function. HDF5 is supported by many programming languages like C, C ++, Python, Java and many others. HDF5 Container contains two stand-alone modules, which allows compressing data with lossless compression.

2.4 OdML

Open metadata Markup Language (odML) is an XML based file format, proposed by Jan Greve, in order to provide metadata in an organized, human- and machine-readable way. Well organized metadata management is a key component to guarantee reproducibility of experiments and to track provenance of performed analyses. [16]

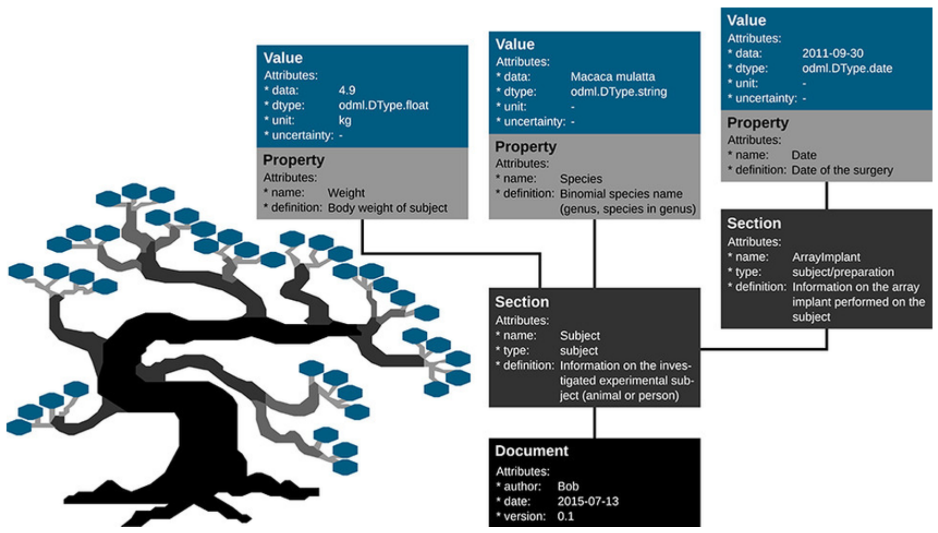


Figure 2.2: odML structure [17]

The key features of odML are

- The odML is able to collect, store and share metadata.
- The odML is an open source.
- The odML is easy to read by human or machine.
- The odML is based on XML language.
- The odML is develop in Python.
- For odML was develop an interactive editor.

2.4.1 OdML 1.0 and OdML 1.4

There are two LTS versions of odml. Version 1.0 and the newest 1.4 and metadata on EEG-Base is in version 1.0. The main changes introduced with python-odml 1.4 was the (breaking) change from odml format version 1.0 to 1.1 in which was mainly removed unused fields and changed the way, how the values are stored. With this transition, was achieved compatibility with the odml structure implemented in nix. All differences are on GitHub [9].

2.4.2 odML Editor

OdML editor includes GUI for working with documents in the odML structure. It can also transform documents from odML 1.0 to odML 1.4.

2.4.3 OdML Converter

This tool reads odML/NIX files and writes the metadata odML structure to newly created NIX files. When running as a script from the command line, it prints information regarding the number of Sections and Properties that were read, written, or skipped for various reasons.[10] In our case, we could use it to update our metadata and then links them to the NIX.

The NIX metadata is a part of NIX file. The odML metadata is an independent file. The odML metadata file has more option, how to represent data. The differences are under the paragraph. For compatibility with the NIX metadata format, which slightly differs from the odML format, the following modifications occur when connecting odML to NIX.

- If a Section has a reference create a property called reference
- If a Property has a reference put the reference in the Property's values

- Values of type URL, person, and text are treated as strings
- Values of type DateTime, date, and time are converted to string representations
- Values of type binary are discarded

[10]

Integration NIX Data and OdML Matadata

The major entities of the data model (Block, Tag, DataArray and Source) contain the metadata field which is used to link between data and metadata. The link is established by referencing the id of the corresponding metadata section. There can be only one link from a data model entity to a section in the metadata. An Information that can be found in the linked section and all its children is assumed to relate to the linking data entity. There is one specificity: if the type of the section and data object match, the connection between them are considered to be of the extends type that is, the information of the data object directly extends the information provided in the metadata and vice versa. If they do not match, metadata object and data object are related in a has a kind of connection. They belong to each other, but the data object is not a further specification of the metadata object. For example, we can relate a section describing a stimulus (type stimulus) to a DataArray of the same type indicating that the contained data is the realization of the stimulus. On the other hand, the same section can be linked to a MultiTag object of the type “analogSignal” which would indicate that the analog signal in the specified range was evoked using that stimulus but is not the stimulus time-course itself.[3]

Example of integration is on the picture 2.3.

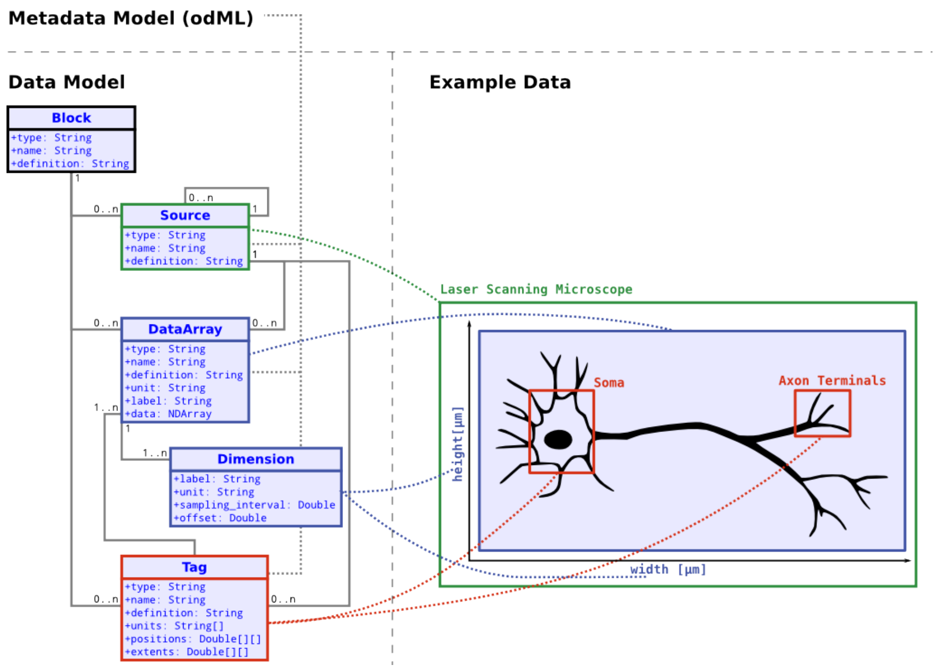
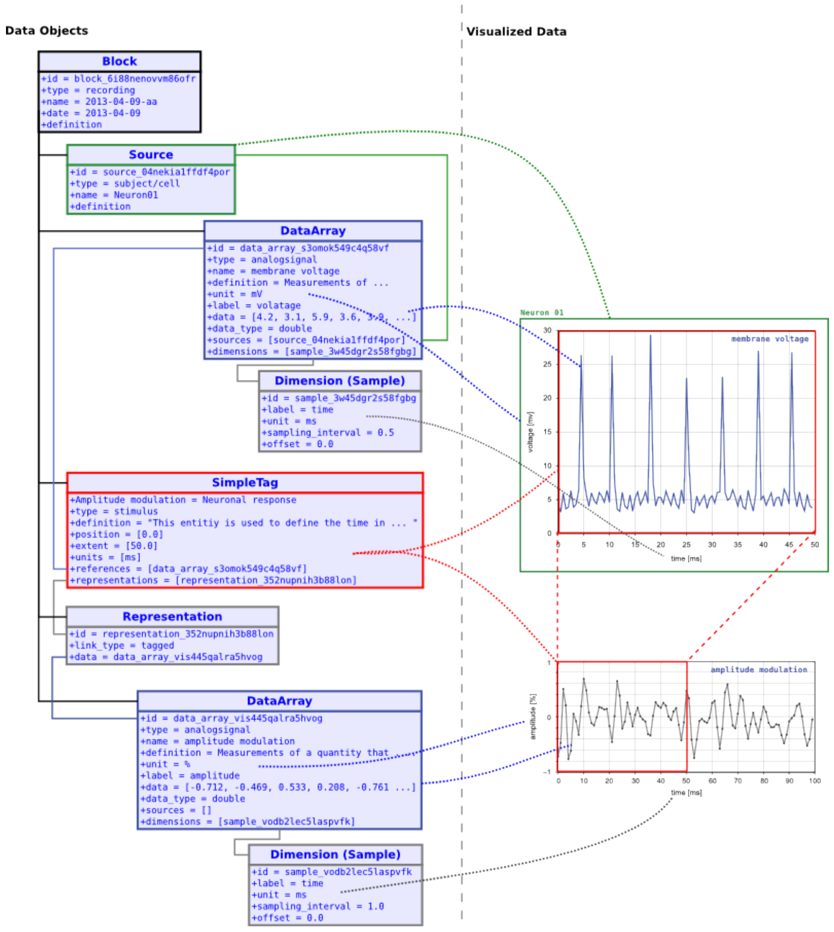


Figure 2.3: Example of integration data and metadata[3]

2.4.4 Storing of Electrophysical Data

Because the brain data are too complex and various (e.g. signals, neural events or spike trains). Often the stimulus waveform needs to stored along with the data. In this example, some white noise amplitude modulation has been presented and the cellular response has been measured. Representing this in the pandora data model can be achieved by storing recorded response and stimulus in a DataArrays and using a Tag to create the link between both. The Tag is of the type "stimulus". It references the DataArray containing the recorded singal and marks the starting position and temporal extent for the segment in which the stimulus was presented. The stimulus waveform itself is considered beeing a Feature of the described stimulus, that mean the features field of the tag contains the id of the Feature. This links to the DataArray using the data field and notes that position and extent of the Tag should be applied to the stimulus as well. The following figure shows the file layout for such an example. [7] It is shown in figure 2.4



Data Objects
Visualized Data
Block
+id = block_6i88nenovvm86ofr
+type = recording
+name = 2013-04-09-aa
+date = 2013-04-09
+definition
Source
+id = source_04nekia1ffdf4por
+type = subject/cell
+name = Neuron01
+definition
DataArray
+id = data_array_s3omok549c4q58vf
+type = analogsignal
+name = membrane voltage
+definition = Measurements of ...
+unit = mV
+label = volatage
+data = [4.2, 3.1, 5.9, 3.6, 3.9, ...]
+data_type = double
+sources = [source_04nekia1ffdf4por]
+dimensions = [sample_3w45dgr2s58fgbg]
Dimension (Sample)
+id = sample_3w45dgr2s58fgbg
+label = time
+unit = ms
+sampling_interval = 0.5
+offset = 0.0
SimpleTag
+Amplitude modulation = Neuronal response
+type = stimulus
+definition = "This entitiy is used to define the time in ... "
+position = [0.0]
+extent = [50.0]
+units = [ms]
+references = [data_array_s3omok549c4q58vf]
+representations = [representation_352nupnih3b88lon]
Representation
+id = representation_352nupnih3b88lon
+link_type = tagged
+data = data_array_vis445qalra5hvog
DataArray
+id = data_array_vis445qalra5hvog
+type = analogsignal
+name = amplitude modulation
+definition = Measurements of a quantity that ...
+unit = %
+label = amplitude
+data = [-0.712, -0.469, 0.533, 0.208, -0.761 ...]
+data_type = double
+sources = []
+dimensions = [sample_vodb2lec5laspvfk]
Dimension (Sample)
+id = sample_vodb2lec5laspvfk
+label = time
+unit = ms
+sampling_interval = 1.0
+offset = 0.0
Neuron 01
membrane voltage
voltage [mV]
time [ms]
amplitude modulation
amplitude [%]
time [ms]

Figure 2.4: Model of sorting electrophysical Data[7]

Other examples of storing data in NIX are here: [7]

2.5 EEG-Base

EEG-Base is a contemporary repository for data collected at the University of West Bohemia. The data are stored as experiments; each experiment contains one or more packages with Brain Vision data and metadata in odML version 1.0. Metadata are visible by using the interface of EEG-Base. Then there is a folder *Data* where are all Brain Vision files. The last two files,*Licence* and *Scenario* ,contain information about the data license and the description of the measurement.

2.6 Brain Vision Data Format

The Brain Vision data format was develop by the Brain Products GmbH. The data are scanned by EasyCap products, sent into an amplifier and saved in the unique Brain Vision format. The Brain Vision data format consists of three separate files. The *.vhdr* file is a header for measurement, it contains metadata and links to other two files. The *.eeg* file contains the voltage values of the EEG signal. In *.vmrk* files there is information about events and their timing.

On the picture 2.5, is shown EEG-Base structure with Brain Vision files.

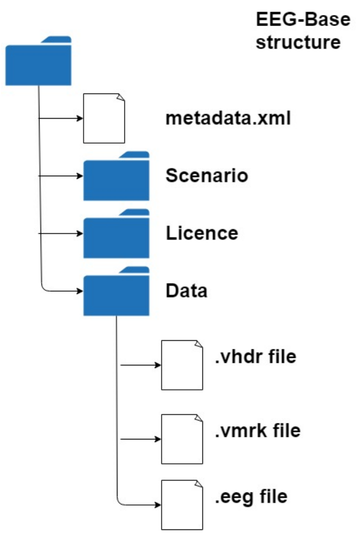


Figure 2.5: Brain Vision data in EEG-Base structure

2.7 MNE-Python

MNE-Python software is an open-source Python package for exploring, visualizing, and analyzing personal neurophysiological data such as MEG, EEG, sEEG, ECoG, and more. It includes modules for data input/output, preprocessing, visualization, source estimation, time-frequency analysis, connectivity analysis, machine learning, and statistics. [12]

In our case, it is possible to use MNE parser, because the parser can load and prase a lot of data formats include Brain Vision raw data.

2.8 Others Initiatives

2.8.1 Neurodata Without Borders

The Neurodata without borders (NWB) format started as a pilot project in the United States of America. It was first developed for AIBS (Allien Institute for Brain Science) in 2014. One year later the project resulted in NWB version 1.0. Four years later the beta version of this format (NWB 2.0) had been developed.

The NWB software has undergone many changes from 1.x to version 2.0. The first change is adding new datatypes. Then support for storing data tables and vector data. Use of tables and vectors has led to improvements in data and metadata organization. For example, lab-specific metadata, spectral analyses, storage of images, unit-based data end more. Next change is in storage timestamps or time intervals. The last change is in the readability of data by extension metadata options.

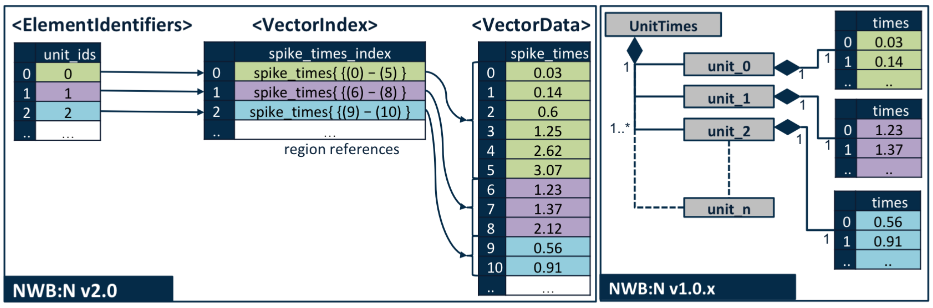


Figure 2.6: NWB: Differences between NWB v1.0 and NWB v2.0 in based data storage [14]

NWB adheres the following instructions[13]:

- Improve data presentation and distribution.
- Support cross-validation and reproducibility.
- Encourage best practices.
- Facilitate and expedite discovery.
- Share analysis tools.
- Create vital new collaborations with other fields.

2.8.2 Brain Imaging Data Sturcture

The Brain Imaging Data Structure (BIDS) is a standard of describing and storing neuroimaging data. It is based on a file structure and JSON to describe metadata. Neuroimaging experiments result in complicated data that can be arranged in many different ways. So far there is no consensus on how to organize and share data obtained in neuroimaging experiments. Even two researchers working in the same lab can opt to arrange their data in a different way. Lack of consensus (or a standard) leads to misunderstandings and time wasted on rearranging data or rewriting scripts expecting certain structure. Here we describe a simple and easy to adopt a way of organizing neuroimaging and behavioral data. [1]

On the picture, 2.7 is shown, how the BIDS works with the data. On the left side are data from the measuring instrument and on the right side are data saved in BIDS. The *.tsv* files contain tables of metadata. The *.json* contain links between metadata and data. Rest of the files are raw data files.

2.9 Summary

There are several competing formats, developed by institutions/nodes/teams active in INCF. INCF has launched an endorsement process, the candidates that would like to become standards, community and expert reviews are done. On the picture 2.8, we can see that FAIR describes the best practices of data sharing and INCF use FAIR rulers. INCF care about compliance best practices on the word.

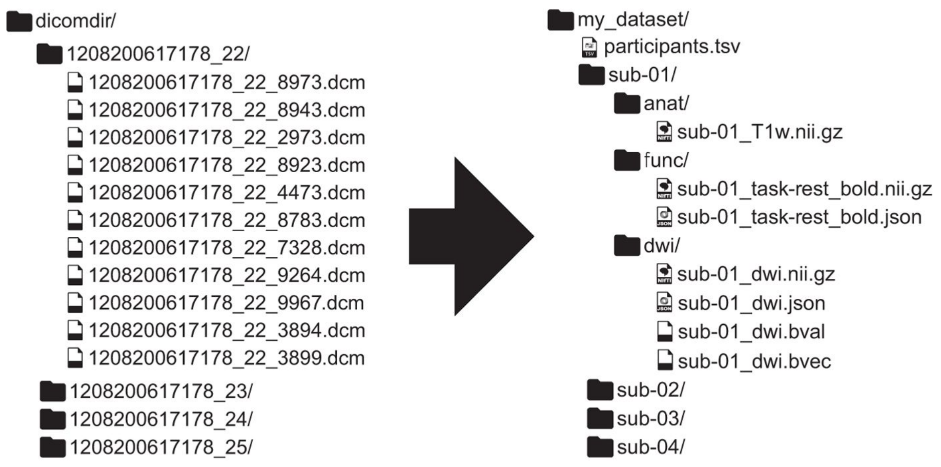

dicomdir/
1208200617178_22/
1208200617178_22_8973.dcm
1208200617178_22_8943.dcm
1208200617178_22_2973.dcm
1208200617178_22_8923.dcm
1208200617178_22_4473.dcm
1208200617178_22_8783.dcm
1208200617178_22_7328.dcm
1208200617178_22_9264.dcm
1208200617178_22_9967.dcm
1208200617178_22_3894.dcm
1208200617178_22_3899.dcm
1208200617178_23/
1208200617178_24/
1208200617178_25/
my_dataset/
participants.tsv
sub-01/
anat/
sub-01_T1w.nii.gz
func/
sub-01_task-rest_bold.nii.gz
sub-01_task-rest_bold.json
dwi/
sub-01_dwi.nii.gz
sub-01_dwi.json
sub-01_dwi.bval
sub-01_dwi.bvec
sub-02/
sub-03/
sub-04/

Figure 2.7: BIDS Structure [1]

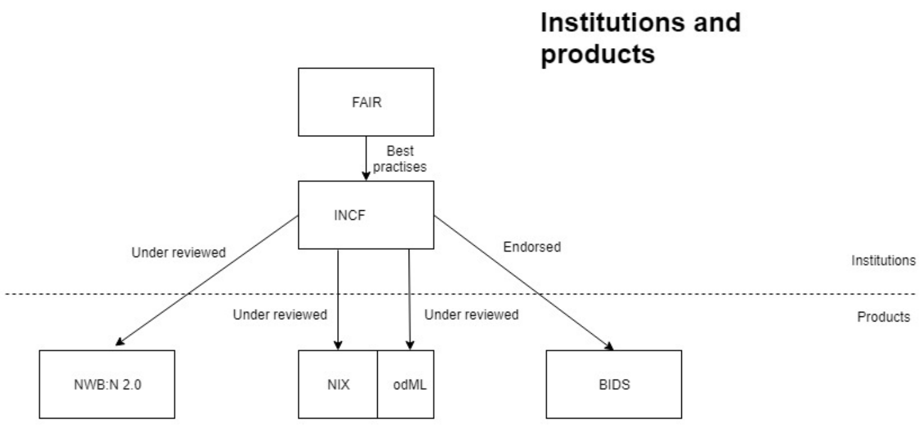


Figure 2.8: Relationships between named institutions.

3 Software Requirement Specification

The goal of this chapter to provide requirement specification of a software system to be developed.

3.1 Overall Description

3.1.1 Product Perspective

The main goal is to transform data from the EEG-Base to the NIX/odML structure. The data have to be saved with metadata in the newest version of odML (the current odML version is 1.4,2019). The newest form is necessary to share data with other researchers.

3.1.2 Product Features

The software has just one function. It takes data and metadata saved on EEG-Base portal and put them together into the NIX/odML structure.

3.1.3 User Class

Researcher
Researcher uses software to connect measured data and metadata and store them in the NIX/odML structure. The researcher can also transfer data and metadata from the older structure to current odML 1.4 and HDF5 structure.

3.1.4 Operating Environment

The program has to run on a common computer at Windows and Linux platforms. The repository for storing the resulting structures will be specified based on later analyzes.

3.1.5 Design and Implementation Constraints

The first condition for implementation software is the programming language. Every code has to be written in Python. It is necessary to use Python because of all others institutes dealing with brain data sharing have their libraries in Python. There is NIX implementation in C++, but G-Node develops NIXpy which contains Python libraries on NIX. The second constraint is that the resulting structure of data/metadata must be saved in the newest form odML (odML 1.4).

3.1.6 Documents

Two documents must be delivered with the program. The first one is code documentation written in the git-hub wiki form. The second one is a user manual also written in the git-hub form.

3.1.7 Assumptions and Dependencies

Getting access permissions to data on EEG-Base is required. The next assumption is working programs or libraries available to work with the odML structure from G-Node. The necessary libraries are **python-odML, Nixpy** and **Nix**. Both of them are on git-hub on *https://github.com/G-Node*. Currently there are well working libraries for the Linux system and the unstable version for Windows (December 19, 2018).

3.2 System Features

3.2.1 Transformation of Data into OdML 1.4

Description and priority

Take data and metadata in odML version 1.0 saved on the EEG-Base portal in odML 1.0 and transform them into odML 1.4.
High priority

Functional requirements

Load a Brain Vision data, check the existence of their metadata and update them to 1.4 version. Then links them together and save into NIX/odML file as HDF5 format. Inform about data transformation by a message on the screen.

3.2.2 Transformation of Data without OdML Structure

Description and priority

Take data and metadata in odML version 1.0 saved on the EEG-Base portal, check their structure and take data without odML and transform them into NIX with the empty odML structure.
High priority

Functional requirements

Load a Brain Vision data and create an empty metadata structure. Then links them together and save into NIX/odML file as HDF5 format. Inform about data transformation by a message on the screen.

3.2.3 Transformation of New Data into NIX/odML

Description and priority

Take data and metadata and transform them into the NIX/odML structure.
High priority

Functional Requirements

Load a Brain Vision data and create a metadata structure in odML version 1.4. Then links them together and save into NIX/odML file as HDF5 format. Inform about data transformation by a message on the screen.

4 Architecture and Design

In this chapter, we will describe the overall architecture and design of the software tool, *EEGBaseToNIX.py*, including analyses, the third party scripts, dependencies, and use libraries. Except for standard python libraries the GNode libraries and scripts developed especially for work with brain time series data saved in *.nix* format are used.

4.1 Analysis

The goal of the project is to transform data from the current Brain Vision format to the *NIX* format. Original data are saved on the EEGBase portal, and resultant nix files have to be saved on the *GIN*.

4.1.1 EEG-Base

EEG-Base is a repository for the University of West Bohemia with a graphical interface. The repository allows saving Brain Vision data measured in the Neuroinformatics lab at the University. Data stored on EEG-Base are saved as three files(.eeg, .vmrk, .vhdr). Metadata are saved in the odML format version 1.0 with additional information about GUI interface. It is possible to download data from the repository in two ways.

The first one is to download one specific measure (dataset) of the experiment. The second one is to download one experiment; then the saved file contains folders with all individual measurements.

The total number of experiments is less than twenty and any experiments are not usable, because some of them lack some files. For example, broken the metadata file or any file from Brain Vision is missing. In this case, it is faster to download them individually and check them manually.

4.1.2 Data Structure of EEG-Base

The files downloaded from EEG-Base are zip archives which contain the file *metadata.xml* and three directories *Data, Licence* and *Scenario.* The license folder contains information about the data license. The folder scenario contains the measurement description. These two folders are useless for the tool. The file *metadata.xml* contains metadata about the measured subject and information about the EEG-Base interface. The folder Data contains all three Brain Vision files or a zip file with Brain Vision files inside.

The Metadata file is writen in the odML version 1.0, and we need the version 1.4. The biggest problem is additional information about the interface. Some information is ignored, but a couple of them makes unresolved errors. The easiest solution is to remove all interfaces information.

4.1.3 Third Party Scripts

Because the NIX and odML data are linked by the block of data from NIX and section from odML it is necessary to create a new section which contains all original sections inside in metadata. Then it is possible to connect NIX and odML file by *convert.py*.

4.1.4 Dependencies

All dependencies and modules to runs the tool EEGBaseToNIX.py are described here section 4.4. Everything has to be installed on the computer for running the program. Some modules have their dependencies. This is the reason to create installation file that contains all dependencies.

The first option how to distribute a tool with all dependencies is to create a VirtualBox operating system, which contains all libraries, modules and testing environment inside. The use of this solution takes a lot of space on disk and.

The next way to export environment for the tool is to use a docker. Docker makes a *docker-image* where all dependencies are described. After docker image startup the environment is set up automatically. Docker is a better memory option, but there are not exactly define all steps to use it in *EEGBaseToNIX.py*.

The last option to set dependencies is a text file *requirements.txt*. This works only if everything that needs to be installed is available by pip install <modulename>. All bigger IDEs can work with this file also. Other dependencies could be added into a bash script. This is the easiest way to create an install script to get all important dependencies.

4.1.5 Analysis Conclusion

As the format for the university data was chosen NIX/odML format developed by the institution G-Node. The main reason is long-term cooperation G-Node and the University of West Bohemia. Other reasons are well-documented, open source codes on GitHub[4]. The MNE package was chosen for its internal Brain Vision parser and was recommended after a conversation with G-Node developers.

4.2 Third Party Scripts

A tool *EEGBaseToNIX.py* to convert data/metadata from the brain vision format to the nix format includes three following scripts: *EEGBaseToNIX.py, mnetoix.py* and *convert.py.* People from GNode wrote last two scripts. Modifying them for their use is necessary.

4.2.1 Mnetonix.py

The goal of the script is to convert files in the BrainVision and Europen Data *.edf* formats into *.nix* files without metadata conversion. Mnetonix.py was written by Achilleas Koutsou from GNode. The script was written within the hackathon, held at the University of West Bohemia in March 2019. The tool EEGBaseToNIX uses only part of script *mnetonix.py.* The used part can transform Brain Vision data to NIX files. Functions to convert *.edf* files and backward conversion from *.nix* to BrainVision or EDF files are unused. This script can be used as a separate script.

4.2.2 Convert.py

This script transforms the metadata in odML to actual format and links them with the NIX file. Michael Sonntag from GNode wrote the script convert.py. It is part of the odML converter. For simple use of this script in the *EEGBaseToNIX.py* tool it was necessary to modify interactive parts of the script. During runtime, the original script, *convert.py*, asks to connect both files and for conversion metadata to the current version. All interactive parts were removed and changed for agree answers. The next removed part was extracting odML documents from *.nix* files. The script *convert.py* can links NIX documents in odML format independently of the version. The script is not self-contained.

4.3 Programming Language

4.3.1 Python

Python is a high-level scripting programming language and supports various programming paradigms like object oriented programming, imperative, procedural or functional programming. Currently, two incompatible versions of Python 2.x and 3.x are supported. The *EEGBaseToNIX.py* tool has been developed using the Python version 3.6.

4.4 Packages and Dependencies

EEGBaseToNIX.py uses the packages and dependencies described below.

4.4.1 Nixio

Original NIX libraries are developed in C++. Nixio is a python module, which allows an interface for work with *nix* files in Python in the same way as with original NIX libraries. A special new version of nixio 1.5.0b3 was released in March 2019 for the *EEGBaseToNIX.py* tool. This module is used in *mnetonix.py* and *convert.py* scripts for creating new nix files and store data inside files.

4.4.2 Mne

MNE is a Python package for exploring, visualizing, and analyzing brain data. This package is used in *mnetonix.py.* This script uses it for Brain Vision parser inside. More information about the MNE module is available in section 2.7.

4.4.3 OdML

The odML package is mainly used in convert.py. The package provides an interface and functions to current odML structure. Mistakes and out-of-date versions of metadata are repaired and edit to actual version (actually in April 2019 is it odML 1.4). Differences between versions are described in subsection 2.4.1.

4.4.4 NumPy

NumPy is a basic package for work with multidimensional arrays. Almost all programs which work with the matrices are developed or supported by NumPy. The package is directly used in *mnetonix.py* and in the MNE package.

4.4.5 SciPy

SciPy is a python package, which provides and supports complex mathematical calculations. The package is used in *Mne* in combination with *NumPy* for work with matrices.

4.4.6 ElementTree

The *xml.etree.ElementTree* is a python module for parsing and following work with XML files. This library is a core for the *EEGBaseToNIX.py* script. Metadata files stored on EEG-Base are in odML 1.0 format with additional GUI related information. All GUI related information must be removed.

4.4.7 Subprocess

Functions from the subprocess library are used to start third party scripts and analyzes their listings. Functions of this library check the return value of third party scripts.

4.4.8 ZipFile

ZipFile is a module to work with zip files. Some brain vision data downloaded from EEGBase are in a zip folder.

4.5 Summary

To run *EEGBaseToNIX.py* is necessary to download additional modules that depend on the modules used in the scripts. A lot of used libraries and modules are included in the standard Python libraries. Some of them could be easily downloaded by pip install <modul name>.

On the picture 4.1, is a data flow diagram of tool EEGBaseToNIX. The diagram display work with data and metadata. The metadata needs to be read, rid of gui tags and convert to the current version. The Brain Vision data need to me convert to NIX and links with the metadata. The resulting file is a NIX/odML structure.

Figure Structured graph 4.2 shows the dependencies between third party products. The module xml.etree.ElementTree loads the *metadata.xml* file and parse to elements. Xml_parser remove GUI information and returns odML version 1.0. Script *nmetonix.py* converts Brain Vision data by MNE to NIX file. Script *convert.py* converts metadata to the current version odML and links it with NIX files into NIX/odML structure.

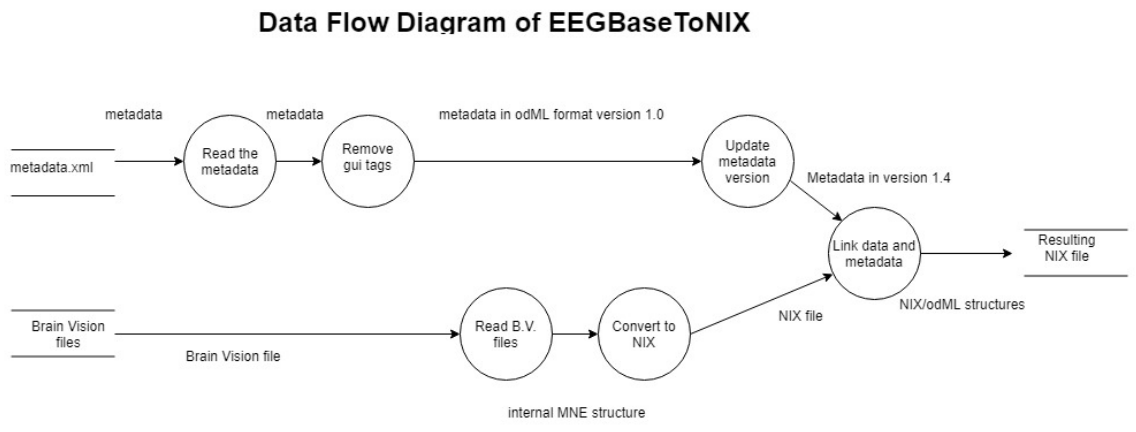


Figure 4.1: Data flow diagram of tool EEGBaseToNIX.py

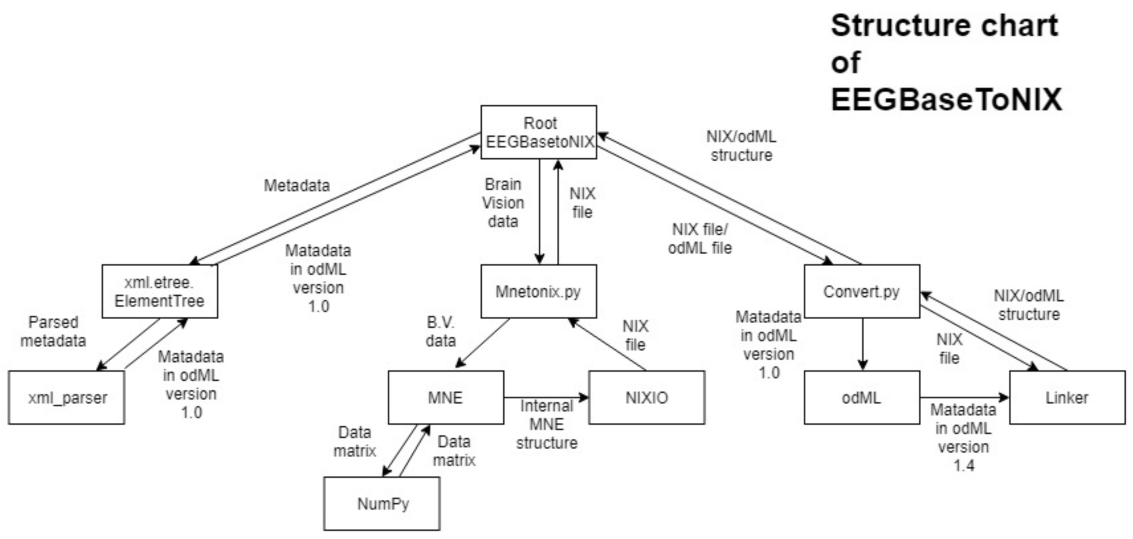


Figure 4.2: Structure chart diagram of tool EEGBaseToNIX.py

5 Implementation

In this chapter is describe implementation of the *EEGBaseToNIX.py*. Important methods are also presented and explained.

The script EEGBaseToNIX checks in which mode the script runs and content of the folder given as the first argument. If there is a *Data* folder, it tests the existence of other files and runs conversion immediately. If there is not any *Data* folder, take all folders in the directory and tested their content. For directories, which contains *Data* folder is run conversion also. During elaboration of inputs, the content of the *Data* folder is checked. If there is a zip file, it is unzipped. The next step is parsing *metadata.xml* file and creating a new section. The resulting file is saved into a new folder called *newNix*.

After preparing a new metadata file the *mnetonix.py* script is started. The script takes Brain Vision data and creates a NIX file in the folder *newNix*. If the process ends successfully, and NIX file exists, the second script *convert.py* is started. *Covert.py* checks the *metadata.xml* file version. If *metadata.xml* is not in version 1.4, it updates them to the current version. Then it links the NIX file with the *metadata* file.

At the end of the script, the result is saved in a new folder, called *newNix*. The folder is situated into the directory, which contains all data for each measurement, on the same level as *metadata.xml* and *Data* folder. In the *newNix* there are two files - metadata.xml and NIX file. The final file is in *.nix*.

5.1 Methods

Main

This is the *main* method of the program. It checks the arguments and chooses the right mode. If the target is one folder with metadata, it runs the method *convert*. If the target is a folder with more directories it runs the method *convert* for every folder inside.

Convert

The method *convert* is the most important method in the script. The input parameter is a path to the directory with metadata and data. At the beginning it checks the existance of all important files by using the method *all_vhdr_files*. Then it runs *xml_parser, run_mne_to_nix* and *nixodml-converter*. At the end it show the result of files conversion.

Nixodmlconverter

In this method, *Nixodmlconverter*, starting script to connect metadata in odML version 1.0 and NIX file.

Run_mne_to_nix

Run_mne_to_nix runs an external script to convert Brain Vision raw data files to the nix file. The result of the script is verified by testing the presence of an existing file.

All_vhdr_fies

The input parameter to the method is a path to the directory, which has to contain data and metadata. The method checks *metadata.xml* and then goes to the data folder and tries to find all files ending with *.vhdr*. For all *.vhdr* files it tries to find files with the same name, but in the *.eeg* and *.vmrk* format.

Xml_parser

Xml_parser reads the metadata file into the buffer. Then it removes all information about the interface of the EEG Base portal.

Set_splitter

The method *set_spliter* is the first method called in the main method. This method gets the current OS and sets the default splitter for work with paths in a specific OS.

5.2 Other activities

Workshop/Hackathon

A workshop/hackathon on data standards for electrophysiology was held at the University of West Bohemia from March 25th to March 30th, 2019. It was an international meeting where people from G-Node and NWB initiative were participated. Brain Vision data, their structure and data structure of EEG-Base were discussed. The people from G-Node presented NIX and odml structures. During the week it was established a new repository for EEG-Base on GIN. The second point of a hackathon program was coding the script *mnetonix.py.*

6 Validation

In this chapter, are introduced possibilities to test the *EEGBaseToNIX.py* and the results of testing. The Nix View was recommended for validation of a new NIX file. Other options are nixio or any other HDF5 viewer. There are two important parts of validation:

The first one is that the result is a real NIX file.

The second one is that the NIX file has to contain all the metadata inside.

6.1 Nixio

For basic validation the use of nixio is the best choice. The principle of the test is to open the file and load it into a buffer. If the loading does not have any error, the file is in the NIX format, according to G-Node employees. This method does not check integration of data with metadata or loss of any information from metadata.

6.2 Nix View

The Nix View is a tool developed especially for NIX files. The viewer can open NIX files and provides functions to check the content of the metadata manually. This is the best option how to test the consistency of NIX files. The current Nix Viewer works with the nix version 1.4.9, *EEGBaseToNIX.py* works with the NIX version 1.5.3, and the structure cannot be fully opened. The new version of Nix Viewer is planned to be released after official NIX LTS version.

6.3 HDF5 Viewer

Because NIX files are based on the HDF5 file format HDF5 viewer can open them. In the viewer, it is possible to see data and metadata. The metadata contains information about links to the NIX, but inserted metadata are still available for the manual check. This method and tool were chosen to test the result of *EEGBaseToNix.py.*

6.4 Positive Tests

NIX files from randomly chosen experiments are created, they are open in the HDF5 Viewer and metadata from the the NIX file and metadata from EEG-Base are checked manually.

Testing was done on three most important experiments from EEG-Base. From each experiment ten random measurements were downloaded containing Brain Vision data and metadata. The resulting NIX files were open in the HDF5 Viewer and check metadata manually in NIX files and metadata on EEG-Base.

6.5 Negative Tests

The negative test has to ensure the stability of the script and checks reactions on wrong inputs. In table 6.1 it is possible to see input and output of the script.

Input	**Output**
Missing argument	Use path to a folder as an argument.
Wrong argument	Use like an argument path to folder
File does not exist	Working with files on target path
Path to the folder without metadata	Metadata file not found
Path to the folder without .vhdr files	Some Brain Vision file is missing
Path to the folder without additional Brain Vision data	No completely .vhdr files found
Nix file created unsuccessful	Mne to nix ended with errors
File metadata, which not contains odML structure	Wrong metadata file
Connect metadata and nix failed	Convert ended with errors

Table 6.1: Table with negative tests and results.

Scripts of Third Party Testing

The part of the implementation was testing of *mnetonix.py* and *nixodml-converter* tool. In case of *mnetonix.py* there was a problem with the NIX version. The script was written for NIX version 1.5.3, but the latest available version was 1.4.9. After reporting this bug, G-Node released a new version for python libraries 1.5.3.

At *Nixodmlconvertor* there was a problem with error handling and with the version of odML. The version problem was fixed by using functions to transform metadata from the odML library.

7 Conclusion

During the implementation of the bachelor project, I got the basic knowledge about current initiatives in data sharing, data formats, and work with brain time series data. This project stems from a recent school assignment, in the course of which. I tried to collect Brain Vision data: „Measurement of motor-evoked potential“. As data collection proved to be a time-consuming task - about one hour per person – it follows that data sharing is of immense importance.

The project is aimed at analysis of options regarding data sharing, proposition and realisation the best way to share data in the University of West Bohemia saved on EEG-Base. Then was chosen a space for the data for easy sharing with others institutions/labs/research teams around the world.

The bachelor's thesis assignment is completely fulfilled. As the best-suited format were chosen the NIX and odML formats from G-Node. The reason being that the metadata file format was in odML version 1.0. As the repository for a new NIX data was chosen GIN because GIN was created for sharing neuroscience data; that makes it an ideal choice. In addition, it is being used by people from G-Node. For conversion Brain Vision files to NIX files the script *EEGBaseToNIX.py* was implemented. Based on the test results it is possible to claim that the script is stable and works correctly. The work has to make the data available for sharing, thats why the code is well-documented and placed with user documentation on GitHub [15]. This tool could be used in the laboratories around the word and wants to move their Brain Vision data to the NIX.

As the next step, it would be desirable to improve error notices in course of conversion of the collected data. At the moment, if any mistake occurs in any part of file conversion, it is difficult to detect and identify the exact problem (failed data).

List of Figures

Bibliography

[1] BIDS. *About BIDS* [online]. BIDS, 2019. [cit. 2019/03/07]. Available at: `https://bids.neuroimaging.io`.

[2] FAIR. *What is FAIR* [online]. FAIR, 2016. [cit. 2018/11/20]. Available at: `https://www.incf.org/activities/standards-and-best-practices/what-is-fai`.

[3] G-Node. *The Model* [online]. G-Node, 2018. [cit. 2018/12/01]. GitHub wiki. Available at: `https://github.com/G-Node/nix/wiki/The-Model`.

[4] G-Node. *Source code and user document* [online]. G-Node, 2018. [cit. 2019/04/30]. G-Node, Git Hub wiki. Available at: `https://github.com/G-Node`.

[5] G-Node. *Home* [online]. G-Node, 2018. [cit. 2018/11/25]. GitHub wiki. Available at: `https://github.com/G-Node/nix/wiki`.

[6] G-Node. *About* [online]. G-Node, 2018. [cit. 2018/11/25]. GitHub wiki. Available at: `https://github.com/G-Node/nix/wiki/About`.

[7] G-Node. *Represent Ephys Data* [online]. G-Node, 2019. [cit. 2019/04/07]. GitHub wiki. Available at: `https://github.com/G-Node/nix/wiki/Represent-Ephys-Data`.

[8] G-Node. *Nixpy* [online]. G-Node, 2018. [cit. 2018/12/27]. G-Node, Git Hub wiki. Available at: `https://github.com/G-Node/nixpy`.

[9] G-Node. *odML releases* [online]. G-Node, 2018. [cit. 2018/12/01]. GitHub wiki. Available at: `https://github.com/G-Node/python-odml/releases`.

[10] G-Node. *G-Node nix-odml-converter* [online]. G-Node, 2018. [cit. 2018/11/20]. GitHub wiki. Available at: `https://github.com/G-Node/nix-odML-converter`.

[11] INCF. *About* [online]. INCF, 2016. [cit. 2018/11/20]. Available at: `https://www.incf.org/about`.

[12] Martinos. *MNE* [online]. Martinos, 2018. [cit. 2018/12/18]. Available at: `http://martinos.org/mne/stable/index.html`.

[13] NWB. *NWB Goals and Values* [online]. NWB, 2017. [cit. 2019/04/23]. Official NWB page. Available at: `https://www.nwb.org/nwb-neurophysiology/`.

[14] NWB. *NWB Release notes* [online]. NWB, 2017. [cit. 2019/04/20]. Available at: `https://nwb-schema.readthedocs.io/en/latest/format_release_notes.html`.

[15] RYCHLÍK, J. *Source code and user document* [online]. Jan Rychlík, 2019. [cit. 2019/04/30]. Git Hub. Available at: `https://github.com/RychlikJan/EEGBase-odMLConvertor`.

[16] ZEHL, L. et al. Handling complex metadata in neurophysiological experiments. *Frontiers in Neuroinformatics*. 01 2014, 8. doi: 10.3389/conf.fninf.2014.18.00029.

[17] ZEHL, L. et al. Handling Metadata in a Neurophysiology Laboratory. *Frontiers in Neuroinformatics*. 07 2016, 10. doi: 10.3389/fninf.2016.00026.

A List of abbreviations

- **AIBS** - Allen Institute for Brain Science
- **BIDS** - Brain Imagin Data Structure
- **EEG** - Electroencephalogram
- **FAIR** - Findable, Accessible, Interoperable, Reusable
- **GUI** - Graphical User Interface
- **HDF5** - Hierarchical Data Format version 5
- **INCF** - International Neuroinformatics Coordinating Facility
- **JSON** - JavaScript Object Notation
- **LTS** - Long Term Support
- **NWB** - Neurodata Without Borders
- **odML** - Open Metadata Markup Language
- **OS** - Operating System
- **URL** - Uniform Resource Locator
- **US** - United states
- **XML** - eXtensible Markup Language

B User Document

B.1 Intro

The tool *EEGBaseToNIX.py* is a convertor Brain Vision data to NIX/odML file. The tool can parse and connect Brain Vision data with metadata at odML structure.

B.2 Install

EEGBaseToNIX.py works on Python3, for succesful install all dependencies is needed pip(3).

- 1) Clone or download the repository.
- 2) Go to the folder EEGBaseToNIX.
- 3) Run installLinux.sh for Linux or installWin.bat for Windows to download all dependencies.
- 4) Run *EEGBaseToNIX.py*.

B.3 Use Case

B.3.1 Data Package

Data package is a working label for a folder with metadata and data to convert. Data package structure: The package has to contains a folder called "Data" with Brain Vision data inside (.eeg, .vhdr, .vmrk), and file "metadata.xml." The file metadata have to contains data in odML format of any version.

B.3.2 Script Startup

The tool is runnable with one mandatory argument and one optional argument.

The mandatory argument is a path to the data package, where are "metadata.xml" and folder "Data" and convert them to NIX/odML. If the path enters into a folder, which is not data package, check all folders inside.

If the folders inside id a data package, try to convert the value of this folder. By this way, it is possible to convert more than one data package. The path has to be written without last separator.

Optional argument activates output of Info Logs. Log display is set by default to -off. To activate use "InfoLog=1".

C CD Contents

The thesis is accompanied by a CD with files, closely related with the tool *EEGBastToNIX.py*.

- **RychlikBachelorThesis.pdf** - the complete text of bachelor thesis
- **EEGBaseToNIX.py** - EEGBaseToNIX source code
- **mnetonix.py, convert.py** - source codes of third party scripts
- **installLinux.sh** - installation script for Linux
- **installWin.bat** - installation script for Windows
- **readme.txt** - text file with user documentation
- **requirements.txt** - list of requirements for python