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

Master Thesis

Data format for electrophysiological experiments

Pilsen 2015

Bc. Jiří Vaněk

Acknowledgment

I would like to thank my supervisor Ing. Roman Mouček Ph.D. for his advice and also I would like to thank to my whole family for their support.

Statement

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

Pilsen, May 1 2015

Bc. Jiří Vaněk

Abstract

V současné době neexistuje standard datového formátu pro ukládání elektrofyziologických dat. Tento standard je nutný pro rozšíření možností sdílení dat s dalšími výzkumníky. Tato diplomová práce se zabývá úpravou modelu pro reprezentaci ukládání dat z elektroencefalografických a experimentů využívajících metodu evokovaných potenciálů a metadat těchto experimentů. Byly prozkoumány existující datové formáty pro ukládání naměřených binárních dat a metadat a jejich ontologie, byla definována terminologie, struktura a byl vybrán vhodný formát pro ukládání těchto dat. Dále byl vytvořen program, který umožňuje ukládat do vybraného formátu naměřená data a informace o experimentu.

Abstract

Currently there is no standardized data format for storing electrophysiological data. This standard is necessary for the improvement of sharing capability. This master thesis deals with the adjustments of data and metadata model for measured data from Electroencephalography and Event-related potentials experiments. The formats for storing electroencephalography data and their ontology were examined. The adjusted file format for export of measured binary data and metadata with experiment details was chosen. Terminology and structure for the experimental metadata were defined. The program that exports the measured data and information about the experiment into the appropriate file format was also implemented.

Contents

1 Introduction

Brain research has been very popular recently. Tens of measurements are conducted every year and it is very important to store measured data and all known information about experiments and measuring conditions for later use. It was common that experiments were designed, measured and analyzed and after evaluation this recorded data was deleted. But during last years it has become more important to store experiments for later use and to share data with other researchers to provide further analyses.

At the University of West Bohemia (UWB) the research of Event-related potentials (ERP) experiments are in progress. When this thesis was written, all recorded electroencephalography (EEG) data were stored in the EEG-Base portal at the UWB. These include experiment binary data in files defined by recording software, and other information about the experiment, measured subject and measuring conditions which are stored in the database and recording system files. Because UWB cooperates with other universities and research centers, there is a need for a file format that would allow easy transfer of measured data between the EEGBase portal and other systems. Therefore a file format for exporting and transporting raw binary EEG data and metadata is needed. There is no such standardized format. Each university or firm uses its proprietary format or format defined by recording software. That complicates cooperation and sharing possibilities. Not all formats support storing data and metadata and also some of formats strictly define metadata which could be stored.

It is important to develop a standard for storing and exporting experimental data and metadata. If this standard is accepted by a larger community, it will allow easy sharing and better understanding of conducted experiments.

At first I had to get familiar with formats for storing electroencephalography and biomedical data. I explored their model, terminology and ontology. I became acquainted with the International Neuroinformatics Coordinating Facility (INCF) working group and their standard proposal. Then I checked storing options of measured experiments at the UWB. I analyzed the file formats containing data and stored metadata. Then it was necessary to search for available formats for storing EEG data, compare them and choose the best one or develop a new one. After that I had to figure out the best way to export all relevant meta-data into the chosen format. Then I developed and tested the program that exports data and metadata into the chosen format.

2 State of the Art

2.1 EEG

EEG is a non-invasive method for measuring and recording electrical brain activity along the scalp. This measurement is based on registering voltage changes from the brain neurons using electrodes, which are attached on a scalp. The ERP technique allows us to take raw EEG data, the electrical activity recorded from the brain, and use it to investigate cognitive processing. It is recorded a subject's EEG while they perform a task designed to elicit the proper cognitive response (e.g. attending to a certain type of object). To accomplish this subjects wear a mesh cap embedded with electrodes which record brain activity. [29] The position and designation of electrodes is defined in the international system 10-20 (Figure 2.1), but the scheme was extended and now defines positions for 70 electrodes. Signals from electrodes are recorded and then analyzed. The research of driver's attention, children's motor coordination disorder or mouse blindness is ongoing at University od West Bohemia. EEG measurement is not focused only on humans, but is conducted also on animals or cells. Brain activity from electrodes is recorded during measurements, and for ERP experiments, stimuli are also recorded. Stimuli are synchronized with EEG recording to determine subjects' responses.

Each year, an increasingly vast amount of neuroscience electrophysiology data is collected and reported in journal publications. However, almost none of these data are accessible to the community of theorists building integrative models of neuronal systems or to experimentalists planning new experiments. [19]

2.2 Data Sharing

A trend toward increased sharing of neuroimaging data has emerged in recent years. Nevertheless, a number of barriers continue to impede easy sharing of experiment's data. Many researchers and institutions remain uncertain about how to share data or lack the tools and expertise to participate in data sharing. The use of Electronic Data Capture (EDC) (Figure 2.2) meth-

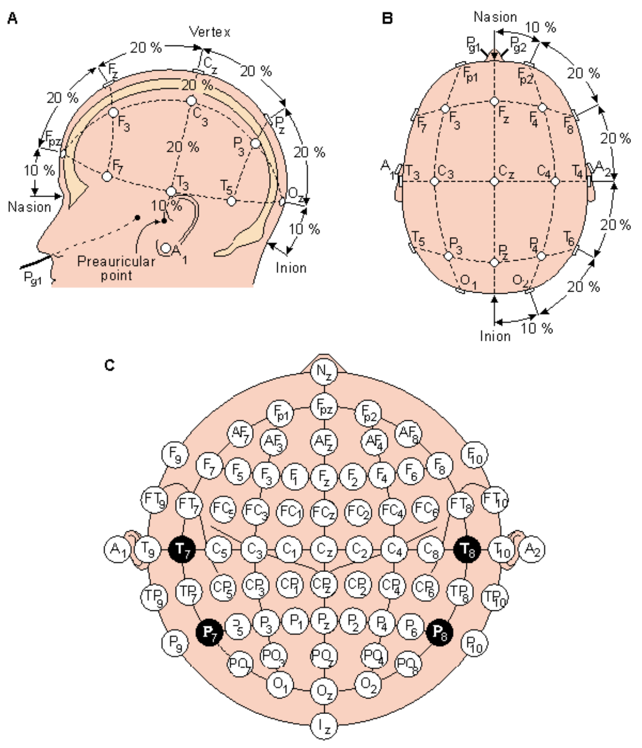


Figure 2.1: The original 10-20 international system - internationally recognized method to describe and apply the location of scalp electrodes in the context of an EEG test or experiment. [43]

ods for neuroimaging greatly simplifies the task of data collection and has the potential to help standardize many aspects of data sharing. [42] The motivation for sharing is:

- **to accelerate progress in understanding of the brain**

  Several researchers claim that more rapid scientific discoveries are possible with shared data [31] [41].

- **to improve data quality**

  The sharing data helps uncover mistakes as missing data, noise, errors, etc. and improves the quality of the data in the future experiments.

- **to reduce cost of research**

  Neuroimaging research is costly both in terms of the data acquisition costs and the time spent in data documentation. A significant amount of money could be saved from redundant data acquisition if data were shared with appropriate metadata descriptions. [42]

The situation is similar in electroencephalography. [42] [28] [19]

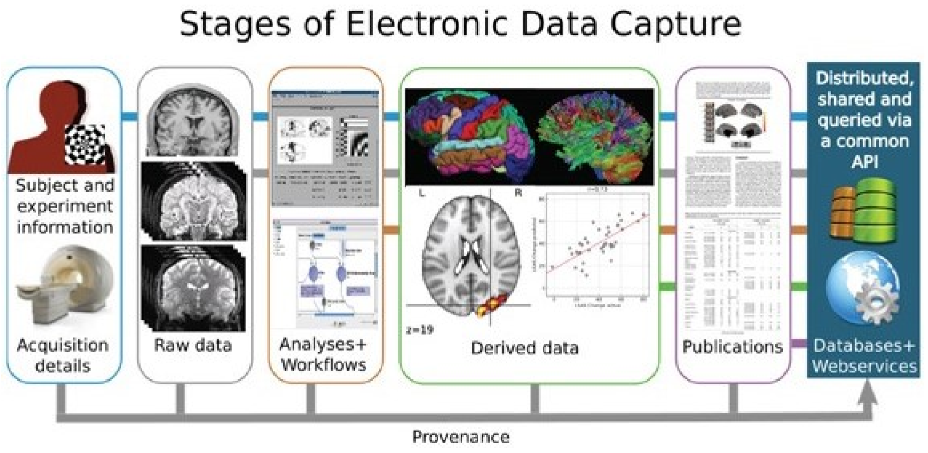

Stages of Electronic Data Capture
Subject and experiment information
Acquisition details
Raw data
Analyses+ Workflows
Derived data
Publications
Distributed, shared and queried via a common API
Databases+ Webservices
Provenance

Figure 2.2: Stages of Electronic Data Capture. [42]

2.3 Program on Standards for Data Sharing

INCF Program on Standards for Data Sharing was established for the purpose of specification the standard for storing EEG data. INCF is an international non-profit organization devoted to advancing the field of neuroinformatics and was established in 2005 in Stockholm. INCF community consists of 17 member countries and associated research groups, consortia, funding agencies and publishers in the field. The National Nodes are institutions or networks that represent each member country. The nodes are established to coordinate neuroinformatics activity within a country. [23]

Program Standards for Data Sharing aims to develop generic standard and tools to facilitate the recording, sharing, and reporting of neuroscience metadata in order to improve practices for the archiving and sharing of neuroscience data. Metadata define the methods and conditions of data acquisition and subsequent analytical processing, Metadata also describe conditions under which the actual raw-data were acquired.

The current focus of the Program on Standards for Data Sharing is in two areas: neuroimaging and electrophysiology. [19]. The most important requirement of such a standard is to accommodate common types of data used in electrophysiology or neuroimaging and also the metadata required to describe them. However we will focus on electrophysiology.

A standard way of storing metadata must be specified. The set of metadata required to describe electrophysiology data is difficult to determine a priori because the types of experiments are so varied. A flexible mechanism must be used which allows referencing and specifying values for currently existing ontologies and also accommodates information not currently systematized. Techniques to include post-experiment annotations of data, and for relating different data parts, are also required. [25]

So far, the working group entertains two approaches towards defining a standard, which may eventually be merged. One, currently named Pandora, defines a generic data model that can be used with Hierarchical Data Format 5 (HDF5) or other storage back-ends. Due to the generic nature, the data model can be used to store various kinds of neuroscience data. The other proposal, called epHDF, defines domain specific schemata for storing electrophysiology data in HDF5. [25]

The Electrophysiology Task Force of the INCF Program on Standards for Data Sharing is working on a document Requirements for storing electrophysiology data. The last version is 0.72 from 3rd November, 2014. This INCF internal document specifies all data that the standardized file format requires, defining what it must, should and may support. An overview of the stored data types is in Table 2.1. [11]

2.4 Present formats for Storing EEG Data

Most known formats for storing EEG data use the format HDF5. Also both INCF proposals use HDF5 and the Electrophysiology Task Force of the INCF Program on Standards for Data Sharing in Requirements for a standard recommends basing a standard on HDF5. [11] Some formats are proprietary and even though some of them are well documented, is due to licenses complicated to use them or edit them. So I focus on the open ones. For storing EEG neuroinformatics data many types of formats exist. The most known and used formats are Ovation [36], NeXus Format [34], NEO [32], NeuroHDF [33], EDF+ [27] and NIX (Pandora) [38].

2.4.1 Ovation

Ovation is a commercial system for managing laboratory scientific data. The Ovation natively stores data in a database, but the data can be exported in

Data type	Description
Signal source	The origin of the recorded data; for example, the identity, position, etc. of the recording. This typically refers to information about a channel name, description, and location, or the target recorded. There are at least two kinds of sources, including actors doing the recording (e.g. an electrode) and objects/actors being recorded (e.g. an subject, brain region, or neuron (real or putative))
Time series	An ordered collection of values given at defined points in time; for example, a recorded voltage signal. It may correspond to raw data, recorded from "electrodes" or "channels". Alternatively, they could be derived from a data processing step. The sampling may be done at regular time intervals (the sampling rate) or at irregular intervals (in which case a time point is required for each value).
Signal events	The Signal events identify changes in a signal; for example, spikes, synaptic potentials, artifacts. They could be raw data, for example, output of a hardware device that detects, and provides the times of spikes. They may also be generated as the result of processing data, for example, applying a spike detection algorithm on a time series signal.
Image stacks	Image stacks are an ordered collection of images; the meaning of stack dimensions must be specified. For example, a z-stack of images, a time series of images, a time-series of z-stacks, movies which are in standard formats.
Experimental events	The Experimental Events data type consists of times of events, along with values that correspond to the times. This data type can be used to describe: stimuli, trials or sweeps, time intervals, behaviors, and other events or conditions that occur during the experiment.
Other / Generic array	Other data types and a mechanism for storing data types which have not been included in the standard.

Table 2.1: Overview of data types required by Requirements for storing electrophysiology data, Version 0.72. [11]

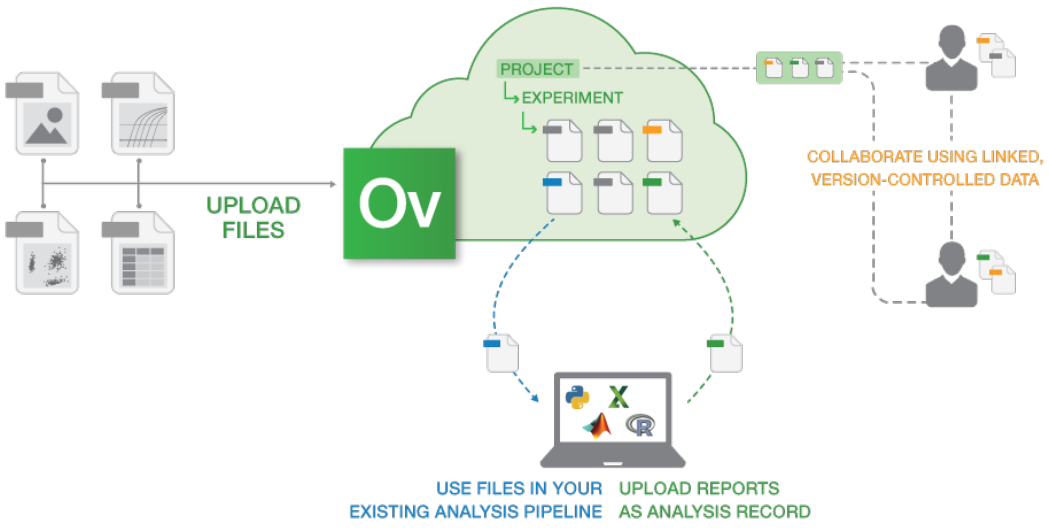

PROJECT
EXPERIMENT
UPLOAD FILES
Ov
COLLABORATE USING LINKED, VERSION-CONTROLLED DATA
USE FILES IN YOUR EXISTING ANALYSIS PIPELINE
UPLOAD REPORTS AS ANALYSIS RECORD

Figure 2.3: Ovation work model - Researchers upload their data into Ovation portal and describe them. Then they could work with data with existing analyze tools or work directly with project with linked version-controlled data. [37]

HDF5 container. Ovation is cloud based data management and collaboration portal. It organizes files by projects and experiments while maintaining relationships between files and subjects, devices, and protocols used in experiment. [37] The description of the work model of Ovation is in Figure 2.3. Although Ovation is a commercial system, the data model of Ovation is open [40].

2.4.2 European Data Format

European Data Format (EDF) is a format for storing multichannel biological and physical signals. This format has limitations for application in a field of evoked potentials, cardiology etc. A major limitation is that EDF can store only uninterrupted recordings. EDF+ was specified for that reasons. The EDF+ specification is incompatible with EDF only in allowing storage of several non-contiguous recordings. The EDF format support annotation for text, event and stimuli. An EDF and EDF+ data file consists of a header record followed by data records. The variable-length header record identifies the patient and specifies the technical characteristics of the recorded signals.

[26] Information about signals, subject and whole measurements is defined and could not be changed. EDF+ format contains this information: version of format, patient identification (sex, birth date, subjects name), record identification (start date, start time, hardware identification, a code specifying the responsible investigator, a code specifying used equipment), number of data records, duration of records and number of signals in data record.

2.4.3 NeXus Format

NeXus is a format for data from Neutron and X-ray facilities. The format is also used for muon experiments. NeXus uses HDF5 as a default file container, but for compatibility reasons it is possible to use also Hierarchical Data Format 4 (HDF4) or Extensible Markup Language (XML) for special use cases. This format supports saving data and metadata into a single file. Several schemes are available for storing metadata sections, but the terminology is not defined and informations are stored in general structures as note, log, parameter or characterization. [34]

2.4.4 NEO

Neo is a package for representing electrophysiology data in Python. This software package is able to load data from closed manufacturers' formats as Axon, Spike2, AlphaOmega, BlackRock and others. Neo implements a hierarchical data model and supports exporting to several neutral formats including HDF5, MATLAB .mat file with NEO structure (NeoHDF5, NeoMatlab,..) or other open source tools WinEdr, WinWcp, PyNN, etc. The recommended way of use of this package is converting a closed format into more standard and open formats - NeoHDF5 or NeoMatlab.

The goal of Neo is to improve interoperability between Python tools for analyzing, visualizing and generating electrophysiology data by providing a common, shared object model. In order to be as lightweight a dependency as possible, Neo is deliberately limited to presentation of data, with no functions for data analysis or visualization. Neo implements a hierarchical data model well adapted to intracellular and extracellular electrophysiology and EEG data with support for multi-electrodes. [32]

2.4.5 NeuroHDF

This format also uses HDF5 as the main container for storing data. NeuroHDF is an effort to combine the flexibility and efficiency of HDF5 for neuroscience datasets through the specification of a simple layout for different data types with minimal metadata. [33]

2.4.6 NIX

This format also uses HDF5 as a data container. This format specification closely defines an inner structure of file, especially the data part. The meta data part is defined by the The open metadata markup language (odML). The NIX project (previously called Pandora) started in the context of the Electrophysiology Task Force which is part of the INCF Datasharing Program.

NIX is one approach to this problem: it uses highly generic models for data as well as for metadata and defines standard schemata for HDF5 files representing these models. Last but not least NIX aims to provide a convenient C++ library to simplify the access to the defined format. The design principle of the data model used by NIX was to create a rather minimalistic, generic, yet expressive model that is able to represent data stored in other widely used formats or models like Neuroshare or NEO without any loss of information. Due to its generic approach, the data model is also able to represent other kinds of data used in the field e.g. image data or image stacks. [38]

This format's scheme (Figure 2.4) was taken as inspiration for my file format, because it is discussed within the INCF Datasharing Program (Chapter 2.3) and it is recommended to use the file model of NIX. The measurements are stored in blocks. Each block identifies measurement and related metadata section. Raw data (signals, stimuli) are saved in DataArrays and they are specified by Dimension, Sample, Set, Representation and Range (Figure 2.4) and could be specified by DataTag. The stimuli and artifacts are stored in SimpleTag (one stimulus) or MultiTag (more stimuli). The source of DataArrays or Tags could be specified by Source. Each section could contain link to the metadata part with measurement information. Closer description is available in Section 3.2.

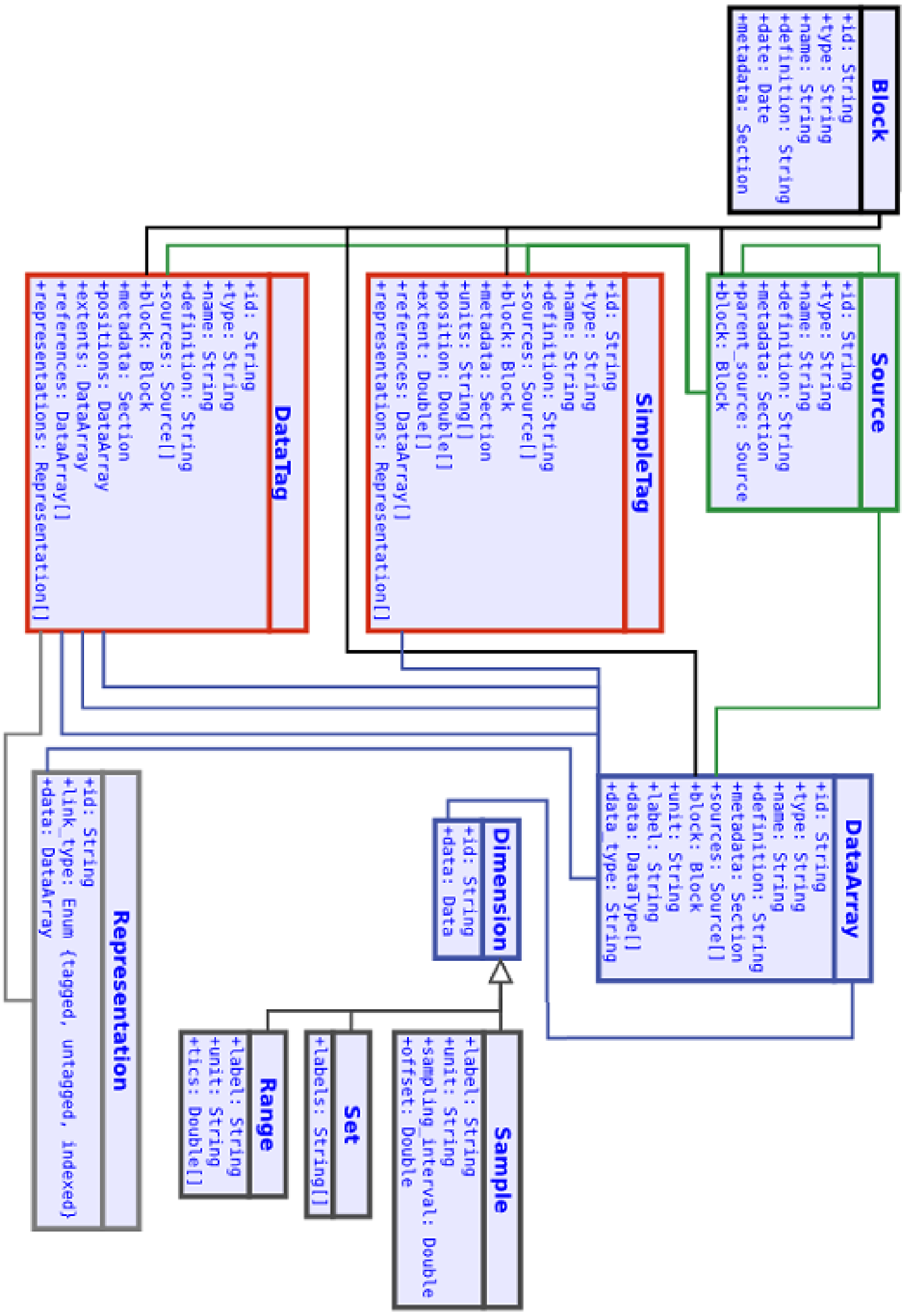


Figure 2.4: NIX data scheme. [38]

2.4.7 epHDF

This format is discussed within the INCF Program on Data Sharing standard. It uses HDF5 and deals with two issues. It is virtually impossible to anticipate all of the types of data and metadata that will need to be stored and no standard scheme exists for specifying how data in HDF5 files should be organized. [24] To address both of these difficulties, a layered approach is proposed. The first layer, which is called Hierarchical Data Format – data sharing (HDFds), provides domain-independent conventions for specifying how the data in HDF5 files are organized. Main features of HDFds are as follows:

- Enables associating external schemata to components of an HDF5 file in a manner similar to how name spaces in an XML file identify elements.
- Specifies locations and a format for storing arbitrary metadata in a HDF5 file.
- Allows linking metadata to particular data parts within a file and to external files. [24]

The second layer builds on the conventions in HDFds to specify schemata for storing basic electrophysiology data types. It is called second layer electrophysiology HDF (epHDF). The data types defined in epHDF (time series, time series segment, neural event and experimental event) are based on the entities defined in Neuroshare [7] for covering the most commonly used data types in electrophysiology. For each type, the data can be stored in whatever HDF5 numeric format is most efficient (for example 16 bit integer). For all of the data types, a metadata schema is specified to include the fields needed to make a plot of the data with correct units. [24]

The epHDF for storing metadata uses odML. An example of data annotation using odML and JavaScript Object Notation (JSON) encoding is bellow in listing 2.1. The listing shows description of amplifier hardware and its parameters. [24]

Listing 2.1: Example of data annotation using odML and JSON encoding.

```
{
  "schema": "odml:hardware/amplifier.xml",
  "model" : <string_value>,
  "manufacture" : <string_value>,
  "serial_no" : <string_value>,
  "inventory_no": <string_value>,
  "owner" : <string_value>,
  "amplifier_type": <string_value>,
  "measurement_type": <string_value>,
  "operation_mode": <string_value>,
  "switching_frequence": <numeric_value>,
  "duty_cycle" : <numeric_value>,
  "gain" : <numeric_value>,
  "high_pass_cutoff" : <numeric_value>,
  "low_pass_cutoff" : <numeric_value>,
}
```

2.4.8 Brain Vision Format

EEG data at UWB are recorded by BrainVision Recorder [14] (Figure 2.5). This program records raw data and saves it to three files, which are described in the next chapter.

The BrainVision Recorder does not allow natively recorded data in any other format. Most recordings consist of three files. The format of these files is defined in the BrainVision Recorder User Manual [13]:

- **data file**

  This is binary file which contains recorded values from recording device. The data are stored as double numbers.

- **vhdr file**

  This text file is Brain Vision Data Exchange Header File Version 1.0 and includes basic information about measurements. The format of the header file is based on the Windows INI format. It consists of various named sections containing keywords/values. The file stores basic information about measuring: coding, name of data file, name of marker file, number of channels, sampling interval in microseconds, information about binary format (IEEE_FLOAT_32) and information about channels (Channel number, channel name, resolution of unit, unit). The sample file of stored data and metadata is in listings 2.2 2.3 A.1 A.2 and table A. The sample file shows data saved in the EEGBase format.

- **vmrk file**

  This is Brain Vision Data Exchange Marker File, Version 1.0. The marker file is based upon the same principle of sections and keywords as the header file. This text file contains information about markers. The file stores marker number, type of the marker, description, position, size and channel number. The example of file is in Listing A.3.

Listing 2.2: The header file example - Information about the file format.

```
Brain Vision Data Exchange Header File Version 1.0
; Data created by the Vision Recorder
```

Listing 2.3: The header file example - Information about coding, created files, data orientation, number of recorded channels and sampling interval.

```
[Common Infos]
Codepage=UTF-8
DataFile=000007.eeg
MarkerFile=000007.vmrk
DataFormat=BINARY
; Data orientation: MULTIPLEXED=ch1,pt1, ch2,pt1 ...
DataOrientation=MULTIPLEXED
NumberOfChannels=48
; Sampling interval in microseconds
SamplingInterval=5000
```

2.5 Hierarchical Data Format

Hierarchical Data Format (HDF) is a data model, file format and library for storing extremely large and complex data collections. [22] This technology is able to store any kind of data and is used all over the world in research centers and government agencies. For example the format HDF5 is used by Cardiff

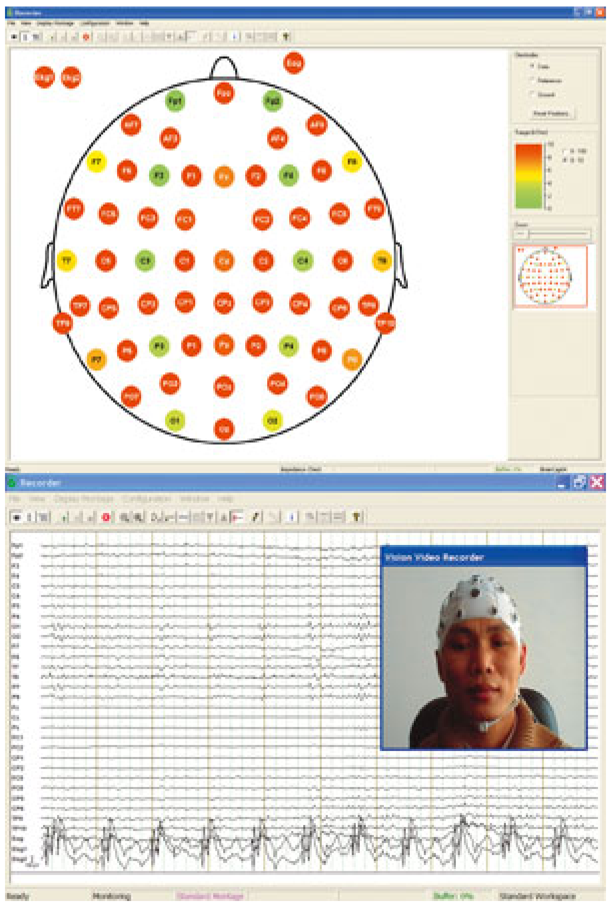


Figure 2.5: Software used for EEG recording at University of West Bohemia - BrainVision Recorder. [14]

University for resolving their problem with grid computing, Deutsche Bank for financial engineering, Diamond Light Source in synchrotron science (using NeXus format- Section 2.4.3), Laboratory for Neural Computation for bio-engineering and many others. A lot of formats for storing electrophysiology data use HDF5. The adopters are able to solve variety of problems with HDF format. (Figure 2.6). "The grouping structure in HDF5 enables applications to organize data objects in HDF5 to reflect complex relationships among objects. The rich collection of HDF5 datatypes, including datatypes that can point to data in other objects, and including the ability for users to define their own types, lets applications build sophisticated structures that match well with complex data. The HDF5 library has a correspondingly rich set of operations that enables applications to access just those components that are important." [22]

HDF is similar to XML documents, HDF files allows to specify complex data relationships and dependencies and are self-describing. Several APIs for programing languages C, C++, Fortran 90, Java and others are available for this format. HDF is open-source (Berkeley Source Distribution (BSD) license), stored data are human readable and the metadata model is easily customized.

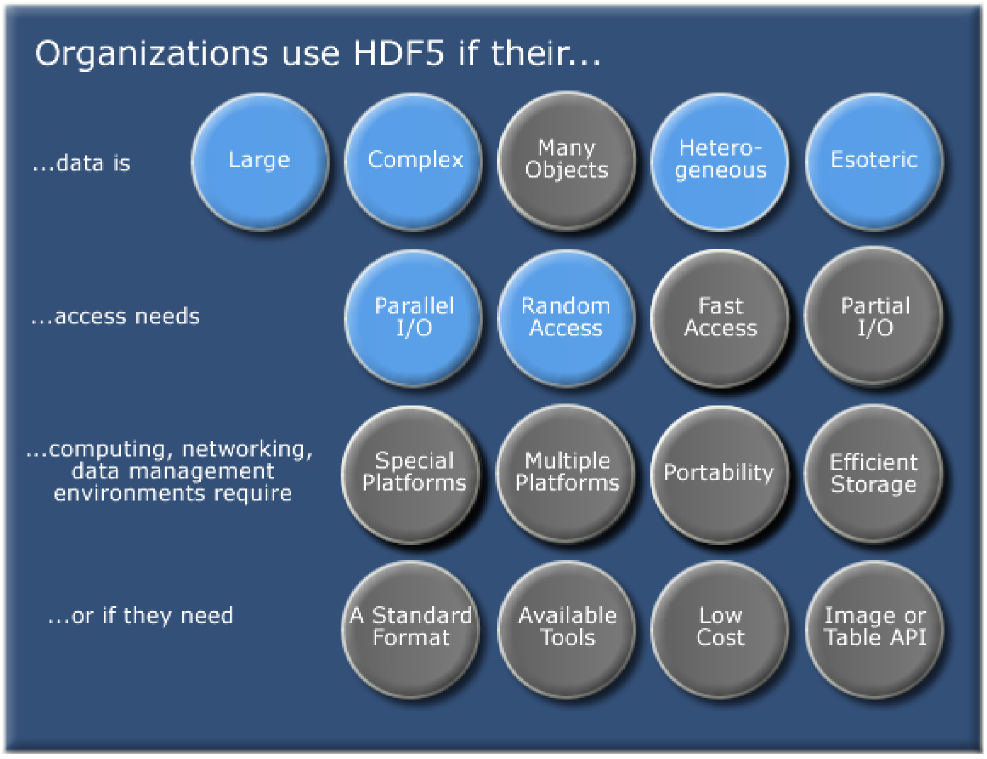


Figure 2.6: Detailed descriptions of some of the data challenges facing HDF adopters. [22]

The advantages of using HDF5: [33]

- **Compact binary data storage, extensible metadata**

  The HDF5 container could use software lossless compression. Szip and Gzip are a stand-alone libraries that are configured as optional filters in HDF5. An Application is using Szip or Gzip compression when a dataset is created and if Szip (Gzip) encoder is enabled, data is automatically compressed and decompressed by Szip (an implementation of the extended-Rice lossless compression algorithm) or Gzip compression. (Figure 2.7). HDF5 allows to use third party compression filters too.

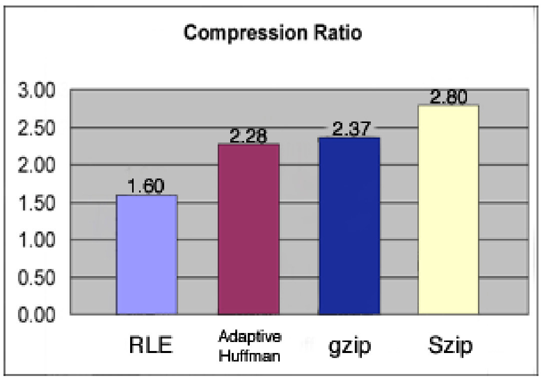


Figure 2.7: Comparison of compression performance in tests with HDF4. Size of compressed output with Szip in HDF4. The test were conducted with HDF4 format, but the compression algorithms are the same in HDF5.

- **Fast random and parallel access, efficient, scalable**

  The HDF already supports parallel I/O and there is project of the European synchrotron particle accelerator community on developing the capability for multiple reader processes to read from an HDF5 file while another process writes to the file. The HDF container supports compression.

- **Widely used in High Performance Computing**

- **Open source and cross-platform**

  The HDF5 format is open source and the HDF Group offers API in many languages (C, Java, Python, Fortran,..) for a various platforms.

- **Possible speed up the query** [20]

Possible limitations of HDF5: [33]

- **Difficulty to store variable-length string properties.**

  Storing of string properties is not simple, especially for strings with variable length.

- **Deleting a dataset does not free the space on disk.**

  The deletion of a dataset from HDF5 file does not free the space and the dataset stays in the file (but is inaccessible). The release of the space on disk requires rewriting the file.

- **Evaluating HDF5**

  There is no tool to evaluate HDF5 files. The HDF allows storing of any data and it is complicated to evaluate it.

- **Delete or update a dataset in HDF5**

  The size of the dataset cannot be reduced after it is created. The dataset can be expanded by extending one or more dimensions, with function H5Dextend. It is not possible to contract a dataspace, or to reclaim allocated space.

2.6 Ontologies

Ontologies are formalized vocabularies of terms. Yann Le Franc defined ontologies this way: "Ontologies are formal models of knowledge in a particular domain and composed of classes that represent concepts defining the field as well as the logical relations that link these concepts together." [18] The model of ontology is in Figure 2.8.

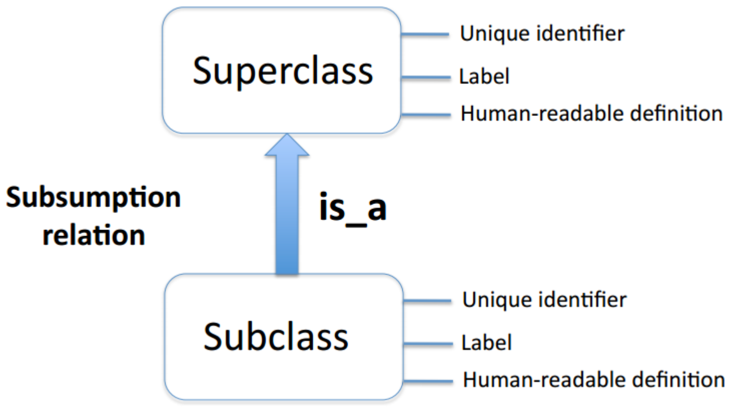


Figure 2.8: Ontology model. [18]

2.6.1 Neural ElectroMagnetic Ontology

Neural ElectroMagnetic Ontology describes classes of event-related brain potentials and their properties, including spatial, temporal, and functional (cognitive/behavioral) attributes, and data-level attributes (acquisition and analysis parameters). [6]

2.6.2 Gene Ontology

This project provides an ontology of defined terms representing gene product properties. The ontology covers three domains: cellular component, molecular function and biological process. [15]

2.6.3 Phenotype And Trait Ontology

"Phenotype And Trait Ontology is an ontology of phenotypic qualities intended for use in a number of applications, primarily defining composite phenotypes and phenotype annotation, Phenotypic qualities (properties). This ontology can be used in conjunction with other ontologies." [15]

2.6.4 The Open Biomedical Ontologies Foundry

The open biomedical ontologies (OBO) Foundry is a collaborative experiment involving developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of orthogonal interoperable reference ontologies in the biomedical domain. [10]

2.6.5 OBO Relations Ontology

Relations Ontology is a collection of relations intended primarily for standardization across ontologies in the OBO Foundry and wider OBO library. It incorporates core upper-level relations such as part of as well as biology-specific relationship types such as develops form. [44]

2.6.6 NIFSTD

Neuroscience Information Framework Standard ontology (NIFSTD) is a core component of Neuroscience Information Framework (NIF) project, a semantically enhanced portal for accessing and integrating neuroscience data, tools and information. NIFSTD includes a set of modular ontologies that provide a comprehensive collection of terminologies to describe neuroscience data and resources. [8]

2.6.7 Basic Formal Ontology

The Basic Formal Ontology (BFO) is a small, upper level ontology that is designed for use in supporting information retrieval, analysis and integration in scientific and other domains. BFO is a genuine upper ontology designed to serve data integration in scientific and other domains. Thus it does not contain physical, chemical, biological or other terms which would properly fall within the coverage domains of the special sciences. [1]

2.6.8 Computational Neuroscience Ontology

Computational Neuroscience Ontology (CNO) is a controlled vocabulary of terms used in Computational Neurosciences to describe models of the nervous system. This first release of CNO is an alpha version and should be further aligned with other ontologies accessible on Bioportal and should be made compliant with the OBO foundry recommendations. [17]

2.6.9 Ontology for Biomedical Investigations

Ontology for Biomedical Investigations is an ontology of investigations, the protocols and instrumentation used, the material used, the data generated and the types of analysis performed on it. [35]

2.7 Open Metadata Markup Language

The metadata in electrophysiology domain providing information about stimuli, data acquisition, and experimental conditions are indispensable for the analysis and the management of experimental data within a lab. However, only rarely are metadata available in a structured, comprehensive, and machine-readable form. This poses a severe problem for finding and retrieving data, both in the laboratory and on the various emerging public data

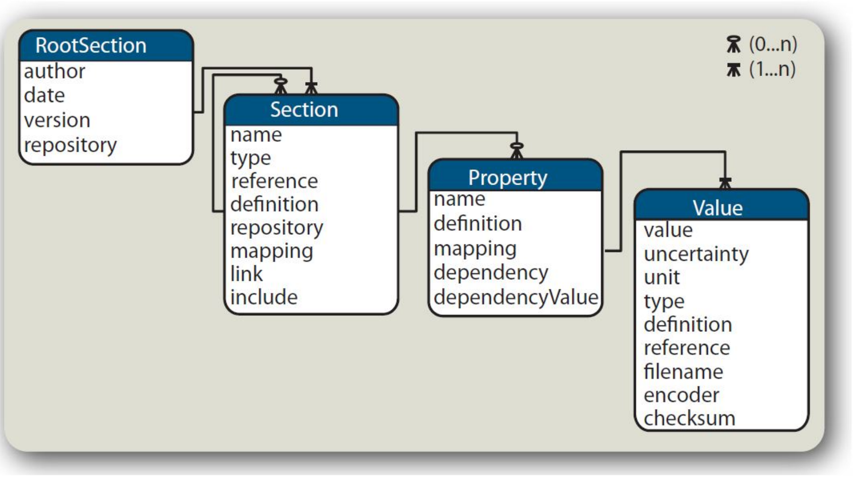


Figure 2.9: Open metadata Markup Language Entity-Relation diagram. [21]

bases. [21] The odML defines the format, not the content, so that it is inherently extensible and can be adapted to the specific requirements of any laboratory. For data sharing a correct understanding of metadata and data is only possible if the same terminology is used or if mappings between terminologies are provided. For this purpose were assembled terminologies with definitions of commonly used terms. [12]

Listing 2.4: Example of odml XML file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="odmlTerms.xsl"
   xmlns:odml="http://www.g-node.org/odml"?>
<!-- ********************************************************* -->
<!--                  Subject section                       -->
<!-- ********************************************************* -->
<odML version="1">
<repository>
http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
</repository>
<version>1.0</version>
<date>2011-01-21</date>
<section>
<type>subject</type>
<name>Subject</name>
<definition>The investigated experimental subject (animal or
   person). May contain the Cell and Preparation sections as
   subsections.</definition>

<property>
<name>Comment</name>
<value>
<type>text</type>
</value>
<definition>A general comment on a specific subject. </definition>
</property>
```

odML stores data as extended key-value in hierarchical (tree) structure (Figure 2.9). The main concept of odML is to separate the content and the structure. The big benefit of odML is that it is highly flexible and allows to save any kind of data.

”Property and Section entities are the core of the odml. A Section contains Properties and can further have subsection thus building a tree-like structure. The model further does not control the content which is a risk, on the one hand, but offers the flexibility.” [21]. So far is the odML model is implemented as a XML schema. Listing 2.4 shows the XML file structure and provides example of odML file.

3 Analysis and Design

The file format is divided into the two autonomous parts DATA and METADATA, which relate to each other, but could be read or written separately. The both parts are stored in one HDF5 container.

The data part stores all recorded binary data and the basic information about measuring for correct representation of measured data (Sampling interval, resolution and resolution unit). This section includes data from the eeg 2.4.8, vhdr 2.4.8 and vmrk 2.4.8 files. The file structure is defined by the NIX file model - (Figure 2.4). I also use the same terminology and the file structure. However our model - EEGBase model was simplified for our needs. More specific description follows in Section 3.2.

Java was selected as a programming language for program developing, because there is already parser of Brain Vision formats developed and also the EEGBase portal is written in Java. There is effort to include the export program into the EEGBase portal for easy export.

3.1 Current Formats Comparison

The comparison of the selected file formats is in Table 3.1. Formats in the comparison were selected, because they use HDF5 and their model is open. The close models could be limited by licenses and their modifications could be limited. The valuated criteria are:

- **Proprietary** - the format is open source / closed
- **Data terminology** - terminology for data (and necessary metadata) is defined
- **Data ontology** - ontology for data is defined
- **Metadata terminology** - terminology for metadata is defined
- **Metadata ontology** - terminology for metadata is defined
- **Portability** - the format has an application programming interface (API) for different languages, platforms

- **Java API** - the format has Java API (my program is in Java)
- **Extensibility of model** - the data and metadata models are extensible
- **Documentation of format** - the file format documentation and its availability

I evaluated formats with three grades compared to our needs. One is best, minus one is worst. The evaluation is subjective even though I tried to be as objective as possible.

Table 3.1: The current formats comparison.

Format	**Ovation**	**EDF+**	**NeXus**	**NEO**	**neuroHDF**	**NIX**	**epHDF**
Proprietary	-1	1	1	1	1	1	1
Data terminology	1	1	0	1	0	1	1
Data ontology	1	1	0	1	1	1	1
Metadata terminology	1	1	-1	0	-1	1	1
Metadata ontology	1	1	-1	0	-1	1	1
Portability	0	1	1	1	1	0	-1
Java API	-1	1	1	-1	-1	-1	-1
Extensibility	0	-1	-1	0	0	1	1
Documentation	1	1	1	1	0	1	0

NIX and epHDF look like the best models. Both these models are supported by INCF.The NIX project offers only C++ API and epHDF is only a standard for file format. Because my effort was integrate program for exporting measured data into the EEGBase portal [2], which is in Java, I choose Java as the programming language for my program. The HDF Group provides Java API for read and write HDF files so I was able to create my own program that writes data in the HDF5 container. Because the NIX model is better commented and provides API, I choose NIX as a initial solution for EEGBase format.

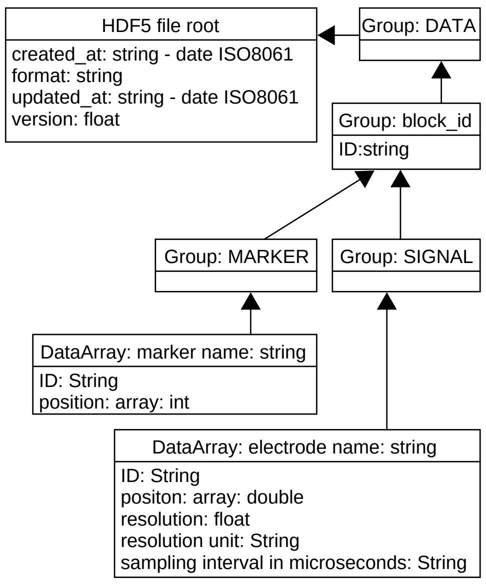


Figure 3.1: The final data model of proposed data format. The data are stored in tree structure with fixed terminology and structure. Each Group MARKER and SIGNAL could contain zero or more DataArrays with raw data.

3.2 Data Model

The data model is based on the NIX data model (Figure 2.4 and Section 2.4.6). The NIX model is able to save data from any electrophysiology experiment. But for EEG experiments the NIX model is too general. So I used only necessary parts of the model and other sections were omitted. The omitted parts are in the NIX model optional, so my format is compatible with the NIX definition. My data model is described in Figure 3.1. The data model uses the NIX scheme of Block, DataArray, MultiTag, DataTag and SimpleTag. The Block is used to divide measuring, DataArray stores raw data of signals and stimuli and MultiTag stores stimuli information and DataTag contains EEG channel information. DataArrays are divided for better distribution in my model to SIGNAL and MARKER parts. Also, the names of DataArrays correspond to names of channels. These adjustments allow better human readability and do not influence information or the model compatibility.

3.3 Model Ontology

3.3.1 Data Part

Ontology and terminology of the data part is based on the NIX model that is described above in Section 3.2 - Data model and in Figure 3.1.

3.3.2 Metadata Part - odML

Metadata are organized according to odML terminology. The German Neuroinformatics Node (G-Node) odML scheme and terminology was used for the metadata part of file, but there was some more information in the metadata scheme used at UWB, which was difficult to save with existing odML terminology. Therefore the odML model and terminology were extended. The changes and adjustments are described in Section 3.3.5 - Metadata adjustments. We also decided that it would be useful to store some more data, which are more specific and could help to describe experiments better. The existing ontologies were also searched, for example Ontology for biomedical investigations [10] [16]. However, I decided to use odML because I was able to extend odML to perfectly fit our needs and it is already used by the NIX model.

3.3.3 UWB Metadata Model

This section shows all collected metadata, which are saved with experiments at UWB and stored in the EEGBase portal [2]. The summary and comparison between the EEGBase portal information and original odML (without our changes) model is in Table 3.2.

Table 3.2: The comparison between the EEGBase portal metadata model and terminology used by the odML model (without my adjustments).

EEGBase		**odML**	
Digitization	Gain	HW/Amplifier	Gain
Digitization	Filter	HW/Filter	
Digitization	Sampling_rate	HW/DataAcq	SampleRate
Pharmaceutical[1]	Title	-	
Pharmaceutical	Description	-	
Artifact	Compensation	-	
Artifact	Reject_condition	-	
Disease	Title	-	
Disease	Description	Subject	HealthStatus
Weather	Description	-	
Weather	Title	-	
Artifact_rem_meth[2]	Title	-	
Artifact_rem_meth[3]	Description	-	
Software	Title	-	
Software	Description	-	
Hardware	Title	HW/*type*	Model
Hardware	Description	-	
Hardware	Type	-	
Project_type	Title	Experiment	Type
Project_type	Description	-	
Subject_group	Title	-	
Subject_group	Description	-	
Exper_opt_par[4]	Param_name	-	
Exper_opt_par[5]	Param_type	-	
Exper_opt_par_val[6]	Param_value	-	
Electrode_conf	Impedance	Electrode	Impedance
Electrode_conf	Desc_img_ID	-	
Electrode_system	Title	-	
Electrode_system	Description	Electrode	ElectrodeCount

Continued on next page

[1]Specification of pharmaceutics used by subject.
[2]Name of the used artifact remove method.
[3]Description of the used artifact remove method.
[4]Name of optional experiment parameter.
[5]Data type of optional experiment parameter.
[6]Value of optional experiment parameter.

Table 3.2 – *Continued from previous page*

EEGBase		**odML**	
Electrode_location	Title	-	
Electrode_location	Shortcut	-	
Electrode_location	Description	-	
Electrode_fix	Title	Electrode	
Electrode_fix	Description	Electrode	
Electrode_type	Title	Electrode	Type
Electrode_type	Description	Electrode	
Scenario	Title	-	
Scenario	Scenario_Length	-	
Scenario	Description	experiment/*type*	Protocol
Scenario	Scenario_name	-	
Scenario	Mimetype	-	
Stimulus	Description	Stimulus	Desc/Start/End
Stimulus_type	Description	-	
Person	Givenname	Person	FirstName
Person	Surname	Person	LastName
Person	Date_of_birth	Person	Birthday
Person	Gender	Person	Gender
Person	Email	Subject	ContactInfo
Person	Phone_number	Subject	ContactInfo
Person	Note	Subject	Comment
Person	Laterality	-	
Education_level[7]	Title	-	
Experiment	Start_time	Recording	Start
Experiment	End_time	Recording	End
Experiment	Temperature	-	
Experiment	Env_note	-	
Experiment	Res_group_id	Recording	ExperimenterID

3.3.4 NIX Metadata Model

The metadata model of NIX is using odML. odML is described in section 3.3.2. The whole odML terminology is too general and too extensive to list it here. So I chose the basic and most used sections which are related

[7]Highest ducation level of person.

to our model and EEG measurements and added it in Appendix B in Figures B.1 B.2 B.3 B.4. The EEGBase model contains modifications which are described below.

3.3.5 Metadata Terminology Extensions

In order to save all our metadata into the HDF5 container I extended the odML model for our metadata. These modifications were committed to G-Node respective INCF GitHub repository [9]. New sections **Environment**, **Protocol** and **Software** and several attributes to the existing sections **Person** and **Electrode** were added. All suggested changes were included into odML. All modification are listed in Table 3.3. Several attributes do not have descriptions, because their names are self-describing.

Table 3.3: Modifications of the odML model.

Name	**Property**	**Value**	**Definition**
Electrode	Usage	Ground	Usage of electrode. [8]
Electrode	Usage	Reference	Usage of electrode.[8]
Electrode	Usage	Channel	Usage of electrode.[8]
Electrode	Description	String	
Environment	Weather	String	
Environment	RoomTemperature	String	
Environment	AirHumidity	float	The air humidity in %.
Environment	Description	String	
Protocol	Description	String	Description of the experiment
Protocol	Author	person	The persons who create this protocol.
Protocol	ProtocolFile	binary	Protocol File.
Protocol	ProtocolFileURL	URL	URL of protocol file.
Protocol	Version	String	Version of the protocol.
Person	Education level	String	Highest archived education level of the person.
Person	Role	Subject	The role of this person.
Person	Email	String	Person's e-mail.
Person	PhoneNumber	String	Person's phone number.

Continued on next page

[8]Added terminology that describes usage of electrode.

Table 3.3 – *Continued from previous page*

Name	**Property**	**Value**	**Definition**
Person	Laterality	String	Handedness - The dominant hand of the subject.
Software	Name	String	The software name.
Software	Owner	String	The owner of software.
Software	Developer	String	Developer or developers firm of the software.
Software	Version	String	Version of the software.
Software	License	String	License type.
Software	LicenceStart	date	The start date of time limited license.
Software	LicenceExpiration	date	The end date of time limited license.
Software	LicenceDuration	String	Duration of the license for the software.
Software	LicenceCount	int	Number of the software license.[9]
Software	Distribution	String	Distribution type.
Software	Description	String	
Software	LicenceDuration	String	Duration of the license for the software.

I also made some recommendations, which data would be useful to store a time zone, detail information about hardware and software and details about the project and experiments. The distribution of experiments by type in odML is very useful and could be used in the EEGBase portal. The information about the project could be also helpful to identify a specific type of experiment. odML is able to save most of these metadata and has a terminology for it so the EEGBase model is able to save them into the HDF5 container. When the current metadata model at University of West Bohemia will be extended it would not be problem to save it in EEGBase format. I suggested also that terminology of UWB metadata model could use odML terminology, because odML have terminology for all currently saved information.

[9]For floating licenses

3.3.6 EEGBase Metadata Model

The model tree structure of EEGbase model metadata is in Appendix C. Because odML scheme is not mandatory but only recommended, it is possible to save, for example, experiments on mice which are also performed at UWB in cooperation with The Faculty of Medicine - it is not necessary use both person and subject model, but it is possible to use only a subject's model. (Figure 3.2). The EEGBase model will store the tree structure of metadata into the EEGBase portal for easier, complete and adjustable metadata export.

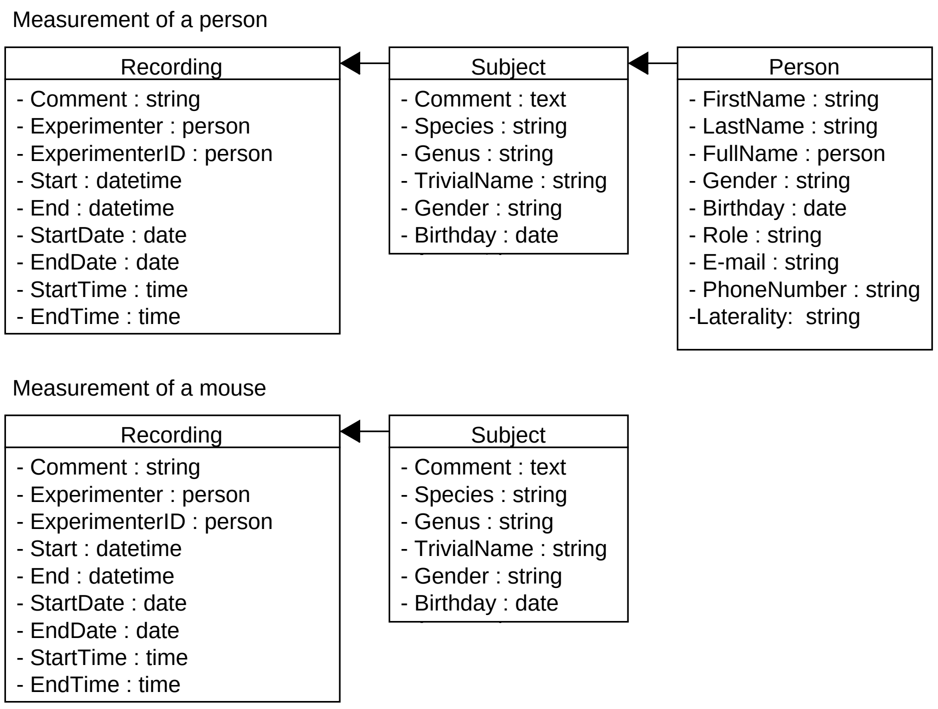


Figure 3.2: The comparison of metadata sections for saving experiments with a person or a mouse.

3.4 HDFExport Program

3.4.1 Program Specification

I chose the Java programing language because I was hoping to implement this program as a part of the EEGBase portal so it will be possible to export the data directly from the portal through an internet browser. This approach is similar to what Ovation uses (store information in database and allows export to HDF5). The problems with HDF libraries came up. Closer information about the EEGBase portal integration is in Section 4.5. The other option was to create a desktop application that uses Simple Object Access Protocol (SOAP) web services of EEGBase portal for loading data and metadata for export.

The program is designed for several use cases divided by data and metadata location (Figure 3.3):

- data and metadata export from locally stored Brain Vision files only
- data and metadata export from locally stored Brain Vision files and metadata from EEGBase portal
- data and basic metadata export from in EEGBase postal stored Brain Vision files without experiments metadata
- data and metadata export from EEGBase portal and Brain Vision files stored in EEGBase portal

Other division is by type of exported data (Figure 3.4):

- Only raw EEG data and basic metadata are exported

  Only data and metadata from the Brain Vision files are exported. Stimuli are not exported (not all measurements uses stimuli).
- Raw EEG data, basic metadata and stimuli are exported

  All data, metadata and stimuli from Brain Vision files are exported.
- Raw EEG data, basic metadata, stimuli and experiments metadata are exported

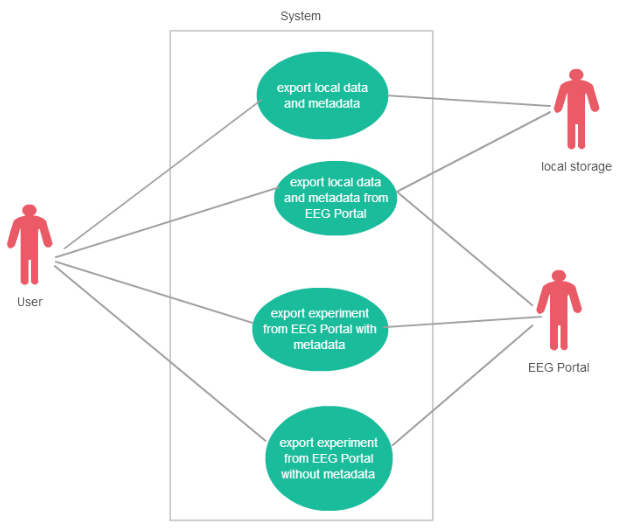

System
export local data and metadata
export local data and metadata from EEG Portal
export experiment from EEG Portal with metadata
export experiment from EEG Portal without metadata
User
local storage
EEG Portal

Figure 3.3: Use cases of HDFExport Program.

All data, metadata and stimuli from the Brain Vision files and experiments metadata from EEGBase portal are exported.

- Raw EEG data, basic metadata and experiments information are exported

  All data, metadata and experiments information are exported.

3.4.2 Architecture

The program for export is based on the three layer architecture. The Brain Vision files parser (EEGDataTransformer), web service client (EDEDClient), HDF5 writer (uses HDF Group libraries for work with HDF5 container) and Graphical user interface (GUI) are the most important parts of the program. Some supportive classes were written (IDGenerator, IsoTime). These generate unique IDs and date in ISO format.

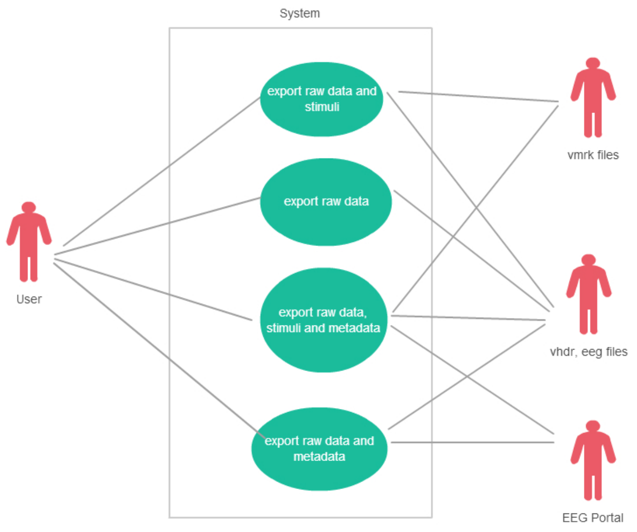

System
export raw data and stimuli
export raw data
export raw data, stimuli and metadata
export raw data and metadata
User
vmrk files
vhdr, eeg files
EEG Portal

Figure 3.4: Use cases of HDFExport Program.

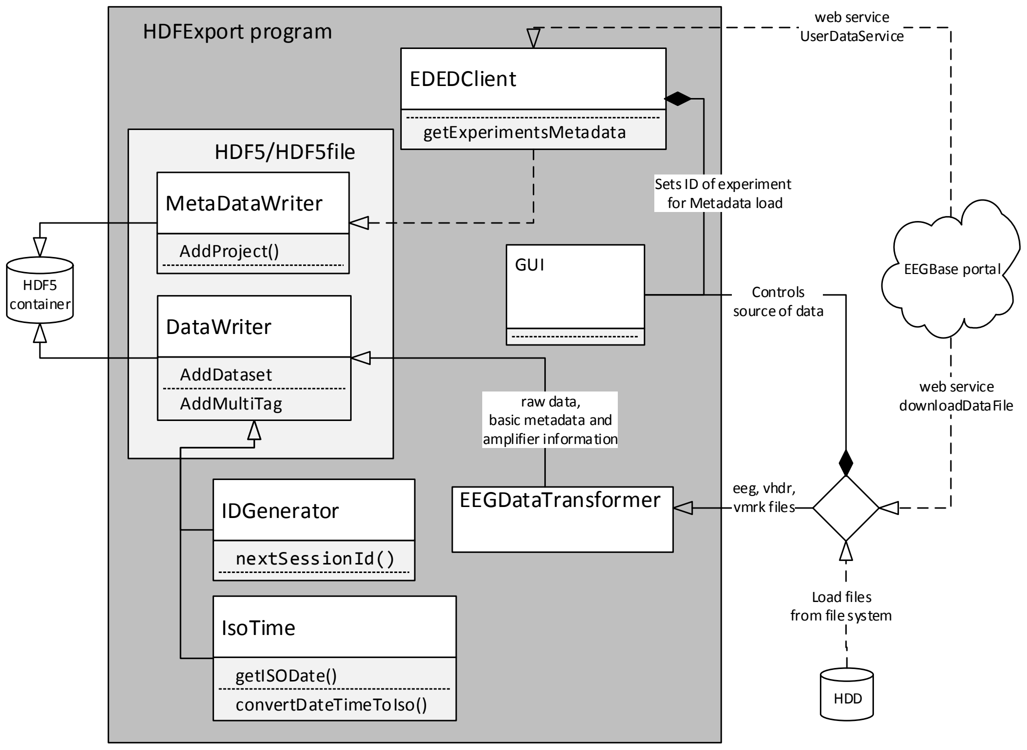


Figure 3.5: The diagram of EEGBase HDF5 export program

4 Implementation

This chapter describe the EEGBase format implementation and the EEGBase Export program implementation.

4.1 HDF Libraries

This section explains which libraries and HDF5 structures I used for HDF file operation.

The disadvantage is that the NIX project has no Java API so it was necessary to use the HDF5 Java API [4], but this API is not native. The HDF Group creates several libraries for Java. The HDF-Java wrappers consist of the Java HDF Object Package, related HDF4 and HDF5 Object Packages, and the native interface for Java HDF (JHI) and The Java HDF5 Interface (JHI5). The native interface includes the Java classes and C interface libraries.

The most intuitive way is using the HDF Object Package [3]. But I wrote my application with JHI5 which is used by HDF Object Package, because it calls directly native C functions. Therefore the program should be more quick and also more examples are available (the Java functions are similar to C and Python functions).

”The HDF Object Package does not provide a one-to-one mapping from Java methods to routines in the standard HDF4 and HDF5 libraries. The one-to-one mappings are provided via the HDF Java Native Interface products JHI and JHI5. The HDF Object Package wraps these direct mappings with a higher level object model.” [3] The diagram is in Figure 4.1.

The HDF works with five basic types of objects:

- **Group**
  A group is used as a folder in the file system, it divides objects into the logical groups. The group can contain none or more objects and can be member of another group.

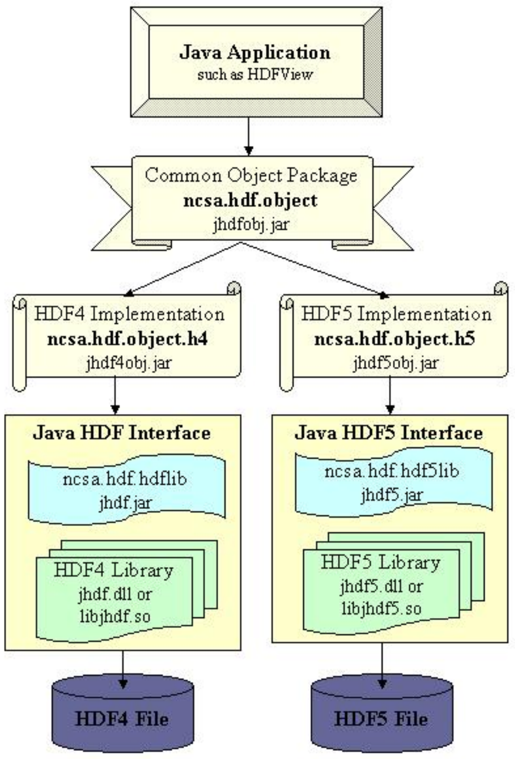


Figure 4.1: The HDF Object Package. [3]

- **Dataspace**

  This is a required part of HDF5 Dataset or Attribute. The Dataspace contains raw data and defines size and shape of data (defines the number of dimensions and size of each element).

- **Dataset**

  The Dataset is an object that describes stored data. It could contain none or more attributes and it is composed from raw data, metadata, data layout or other information necessary to write, read or interpret the stored data. The application view of a dataset is in Figure 4.3.

- **Datatype**

  The HDF5 datatype defines the storage format for a single data element. The description is in Figure 4.4. The datatype describes the storage layout of a single data element. All elements of the dataset must have the same type and the datatype of a dataset is immutable.

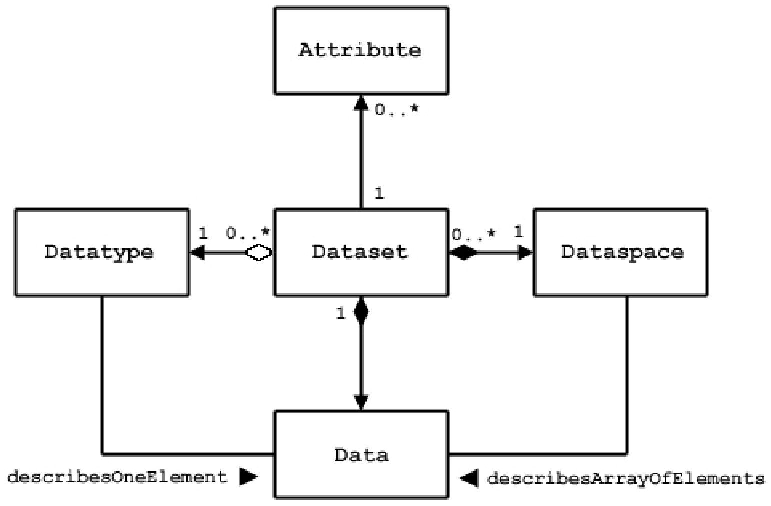


Figure 4.2: Application view of a dataset. [22]

- **Attribute**

  An HDF5 attribute is a small metadata object describing the nature and/or intended usage of a primary data object. A primary data object may be a dataset, group. Even though an attribute is not a standard HDF5 Dataset, it has several common properties: [22]

  – An attribute has a user-defined dataspace and the included metadata has a user-assigned datatype

  – Metadata can be of any valid HDF5 datatype

  – Attributes are addressed by name

  Attributes are from Datasets different in:

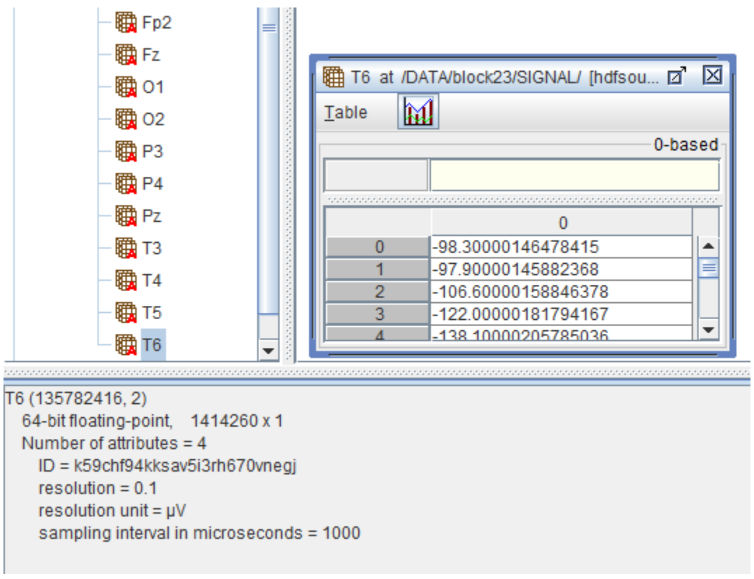
  – There is no provision for special storage such as compression or chunking

  – There is no partial I/O or sub-setting capability for attribute data

  – Attributes cannot be shared

  – Attributes cannot have attributes

  – Attributes are stored in the object header



	0
0	-98.30000146478415
1	-97.90000145882368
2	-106.60000158846378
3	-122.00000181794167
4	-138.10000205785036

```
T6 (135782416, 2)
   64-bit floating-point,    1414260 x 1
   Number of attributes = 4
      ID = k59chf94kksav5i3rh670vnegj
      resolution = 0.1
      resolution unit = μV
      sampling interval in microseconds = 1000
```

Figure 4.3: Example of HDF5 file with EEGBase format in HDFView 2.11- the Dataset with name T6 (EEG channel) with Dataspace (array of 1414260 double numbers) and four attributes.

Since all elements defined in HDF5 have names, they can be referenced by their path. [39]

4.2 Used HDF5 Data Types

The following data types were used in the HDF5 container:

- **double**

  The signal raw data are stored as the HDF5 type H5T_IEEE_F64LE.

- **float**

  Float numbers are saved as the HDF5 type H5T_IEEE_F32LE. The stimuli raw data are saved in this format.

- **string**

  All strings are stored as type H5T_C_S1. Almost all attributes, dates and IDs are saved as strings.

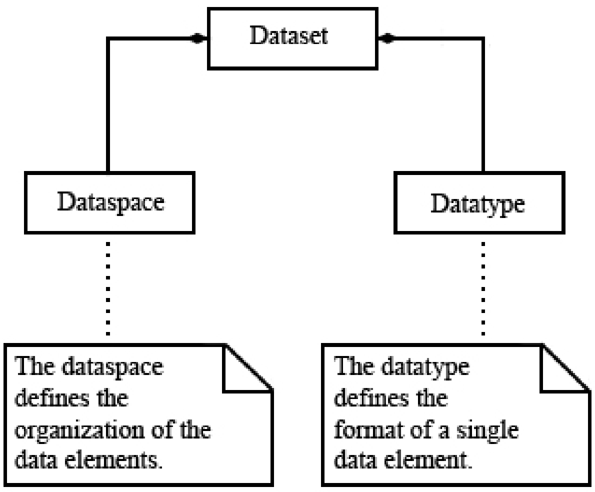

Dataset
Dataspace
Datatype
The dataspace defines the organization of the data elements.
The datatype defines the format of a single data element.

Figure 4.4: Datatypes, dataspaces, and datasets. [22]

4.3 EEGBase File Structure

The root of the file contains basic information about the file and the format. It also divides the file container into two groups Data and Metadata. (Table 4.1). The International Organization for Standardization (ISO) 8601 format was used for the date, because even though HDF5 has a data type for the date, it is not platform independent. The NIX model does not have the date format specified so I chose the norm of Czech standards (CSN) ISO 8601 [5], which is one of many possible formats and it is the Czech version of the international format ISO 8601:2004. This norm allows the user to save date and time at once and specify even the time zone. This could be important for the flawless representation of measured data. Then the final Date string is: Complete date plus hours, minutes, seconds and a decimal fraction of a second YYYY-MM-DDThh:mm:ss.sTZD (e.g. 1997-07-16T19:20:30.45+01:00) [5]. This date and time is saved as String so it is platform independent and thanks to the ISO norm is standardized.

Recordings are wrapped into the blocks (HDF groups). Each recorded channel with basic metadata is saved in a block. The recorded signal values (raw data) are stored in the Dataset with the name of the channel. Information from files eeg (Section 2.4.8) (raw data) and vhdr (Section 2.4.8) (metadata) and generated IDs are stored in the DATA section. Markers (stimuli and artifacts etc.) information are stored in a block in the section

root		
attribute	format	string
attribute	version	string
attribute	created_at	string - date ISO 8601
attribute	updated_at	string - date ISO 8601
group	DATA	
group	METADATA	

Table 4.1: The root of HDF5 file.

MARKERS and they are saved also in the Datasets with the name of the marker. (Figure 4.5).

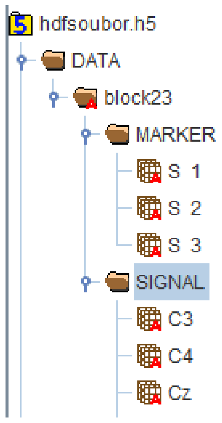


Figure 4.5: Example of HDF5 file with EEGBase format in HDFView 2.11- the DATA section.

The HDF5 container stores information about array dimensionality and a stored data type natively. (Figure 4.3).

4.4 HDFExport Program Modules

4.4.1 Brain Vision Files Parser

The EEGDataTransformer from Jan Štěbeták [47] was used as a parser for current eeg, vhdr and vrmk files. Only minor changes of this program were made. I added the class AmplifierInfo and ImpedanceInfo and the methods NumberOfChannels, SamplingInterval, getAmplifier, getImpedanceInfo into DataTransformer interface. The method getUnits() of ChannelInfo class was edited, because the returned String was in wrong format. Figure 4.6.

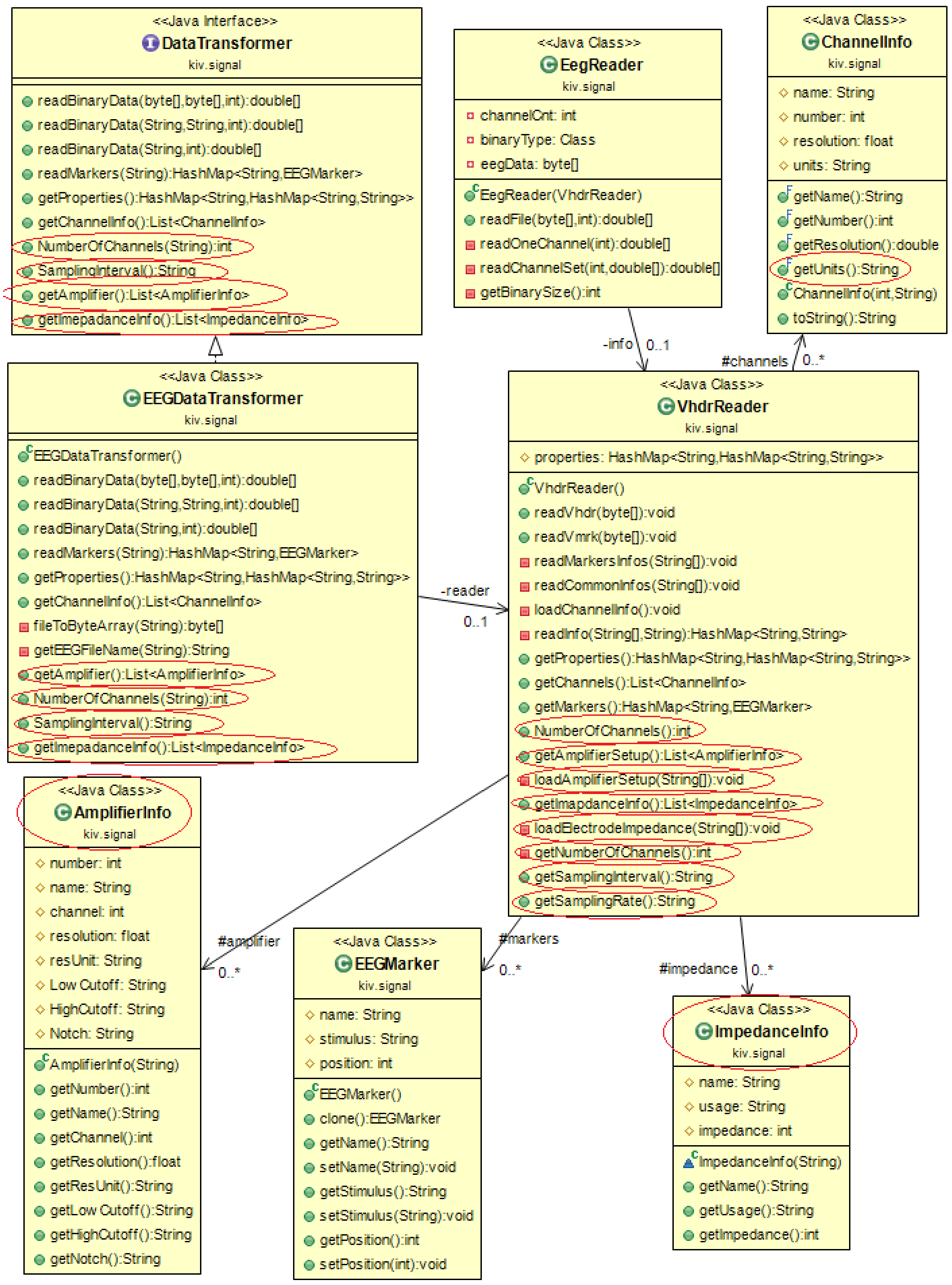

<<Java Interface>>
DataTransformer
kiv.signal
readBinaryData(byte[],byte[],int):double[]
readBinaryData(String,String,int):double[]
readBinaryData(String,int):double[]
readMarkers(String):HashMap<String,EEGMarker>
getProperties():HashMap<String,HashMap<String,String>>
getChannelInfo():List<ChannelInfo>
NumberOfChannels(String):int
SamplingInterval():String
getAmplifier():List<AmplifierInfo>
getImepadanceInfo():List<ImpedanceInfo>
<<Java Class>>
EegReader
kiv.signal
channelCnt: int
binaryType: Class
eegData: byte[]
EegReader(VhdrReader)
readFile(byte[],int):double[]
readOneChannel(int):double[]
readChannelSet(int,double[]):double[]
getBinarySize():int
<<Java Class>>
ChannelInfo
kiv.signal
name: String
number: int
resolution: float
units: String
getName():String
getNumber():int
getResolution():double
getUnits():String
ChannelInfo(int,String)
toString():String
-info
0..1
#channels
0..*
<<Java Class>>
EEGDataTransformer
kiv.signal
EEGDataTransformer()
readBinaryData(byte[],byte[],int):double[]
readBinaryData(String,String,int):double[]
readBinaryData(String,int):double[]
readMarkers(String):HashMap<String,EEGMarker>
getProperties():HashMap<String,HashMap<String,String>>
getChannelInfo():List<ChannelInfo>
fileToByteArray(String):byte[]
getEEGFileName(String):String
getAmplifier():List<AmplifierInfo>
NumberOfChannels(String):int
SamplingInterval():String
getImepadanceInfo():List<ImpedanceInfo>
-reader
0..1
<<Java Class>>
VhdrReader
kiv.signal
properties: HashMap<String,HashMap<String,String>>
VhdrReader()
readVhdr(byte[]):void
readVmrk(byte[]):void
readMarkersInfos(String[]):void
readCommonInfos(String[]):void
loadChannelInfo():void
readInfo(String[],String):HashMap<String,String>
getProperties():HashMap<String,HashMap<String,String>>
getChannels():List<ChannelInfo>
getMarkers():HashMap<String,EEGMarker>
NumberOfChannels():int
getAmplifierSetup():List<AmplifierInfo>
loadAmplifierSetup(String[]):void
getImapdanceInfo():List<ImpedanceInfo>
loadElectrodeImpedance(String[]):void
getNumberOfChannels():int
getSamplingInterval():String
getSamplingRate():String
<<Java Class>>
AmplifierInfo
kiv.signal
number: int
name: String
channel: int
resolution: float
resUnit: String
LowCutoff: String
HighCutoff: String
Notch: String
AmplifierInfo(String)
getNumber():int
getName():String
getChannel():int
getResolution():float
getResUnit():String
getLowCutoff():String
getHighCutoff():String
getNotch():String
#amplifier
0..*
<<Java Class>>
EEGMarker
kiv.signal
name: String
stimulus: String
position: int
EEGMarker()
clone():EEGMarker
getName():String
setName(String):void
getStimulus():String
setStimulus(String):void
getPosition():int
setPosition(int):void
#markers
0..*
#impedance
0..*
<<Java Class>>
ImpedanceInfo
kiv.signal
name: String
usage: String
impedance: int
ImpedanceInfo(String)
getName():String
getUsage():String
getImpedance():int

Figure 4.6: Class diagram of edited EEGDataTransformer program. New methods and classes are marked.

4.4.2 Metadata Loader

The metadata are loaded through web services, which were written by Jan Štěbeták [45], and from Brain Vision vhdr file.

The program uses the web service UserDataService and its methods for loading all experiments metadata. The client was generated from Web Service Definition Language (WSDL) file [46]. EDEDCLient [30] written by Petr Miko was used as a client for web services.

4.4.3 HDF5 Writer

The writer consists of three classes:

- HDF5file

  The the HDF5 file is created in this class and all necessary information about format (Version, Format name, Date) are written in the file. This class uses and calls methods from DataWriter and MetaDataWriter classes.

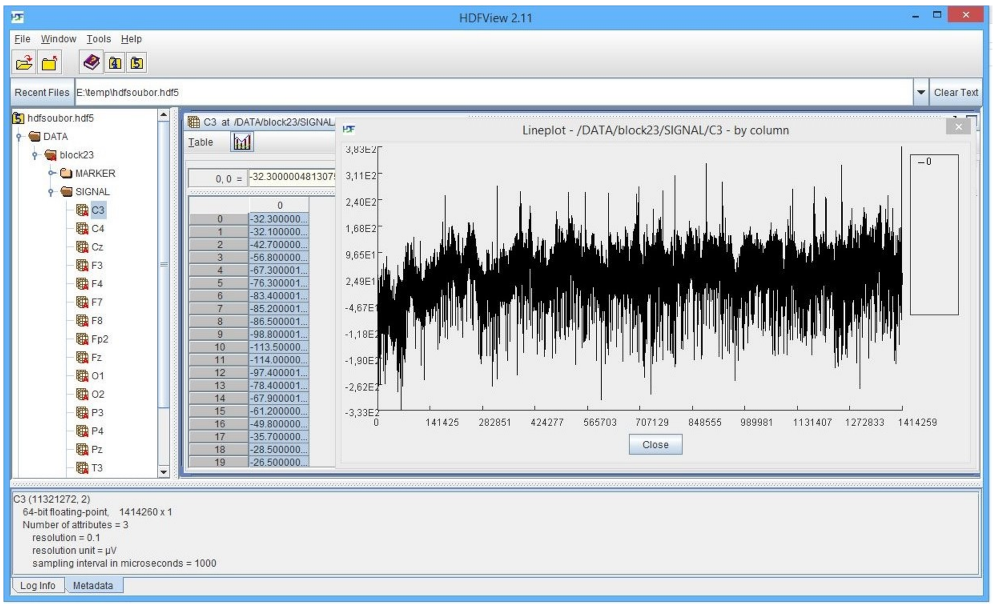
- DataWriter

  This class saves all binary data and markers to the HDF5 container. This class saves also essential metadata of raw data into attributes.

- MetaDataWriter

  This class stores all loaded metadata into METADATA part (Section 3.3.3).

The programming was not simple, because the C methods stayed the same and the wrappers only allow to be called from Java Virtual Machine. The example of the Java code and calls of C functions are in Listing 4.1. The methods for easier work with HDF5 libraries were written and the code for saving string into the HDF5 file is in Listing 4.3 and for storing binary data into the datasets is in Listing 4.2.



0, 0 = -32.300000481307

	0
0	-32.300000...
1	-32.100000...
2	-42.700000...
3	-56.800000...
4	-67.300001...
5	-76.300001...
6	-83.400001...
7	-85.200001...
8	-86.500001...
9	-98.800001...
10	-113.50000...
11	-114.00000...
12	-97.400001...
13	-78.400001...
14	-67.900001...
15	-61.200000...
16	-49.800000...
17	-35.700000...
18	-28.500000...
19	-26.500000...

```
C3 (11321272, 2)
  64-bit floating-point,   1414260 x 1
  Number of attributes = 3
    resolution = 0.1
    resolution unit = µV
    sampling interval in microseconds = 1000
```

Figure 4.7: HDF5 file with data with EEGBase format - in HDFView 2.11

Listing 4.1: Creating Group in HDF5 with Java wrappers. This code creates a new group with the name METADATA at the specified location block

```
//create Group
H5.H5Gcreate(this.block, "METADATA", HDF5Constants.H5P_DEFAULT,
    HDF5Constants.H5P_DEFAULT, HDF5Constants.H5P_DEFAULT);
```

Listing 4.2: Saving binary data into the dataset. It saves H5T_IEEE_F64LE (double) in dset_data if saving location dataset_id exists

```
if (dataset_id >= 0) // location exists
H5.H5Dwrite(dataset_id, HDF5Constants.H5T_IEEE_F64LE,
    HDF5Constants.H5S_ALL, HDF5Constants.H5S_ALL,
    HDF5Constants.H5P_DEFAULT, dset_data);
```

Listing 4.3: Saving string as attribute in HDF5.

```
//SaveString method
int attribute_space,attribute_id=-1;
int stype =-1;
//create space for one char
attribute_space = H5.H5Screate(HDF5Constants.H5S_SCALAR);
//copy size of char
```

```
stype = H5.H5Tcopy(HDF5Constants.H5T_C_S1);
//sets size of string lenght*size of char
H5.H5Tset_size(stype, savedString.length());
//create attribute in location dataset_id, with name HdfName, size
    stype
attribute_id = H5.H5Acreate(dataset_id, HdfName, stype,
    attribute_space, HDF5Constants.H5P_DEFAULT,
    HDF5Constants.H5P_DEFAULT);
//writes string into prepared attribute_id, with size of stype
H5.H5Awrite(attribute_id, stype, savedString.getBytes());
//if all was correct close space
if(attribute_space>-1){
  H5.H5Sclose(attribute_space);
}
//if all was correct close attribute
if(attribute_id>-1){
  H5.H5Aclose(attribute_id);
}
```

4.4.4 Graphical User Interface

The program has only a basic GUI, because the main purpose is to provide methods for saving all current data and metadata to the HDF5 container. The methods which are working with HDF5 container are general and it is possible to use them in any other program. All methods are in separate classes in a special package hdf5. GUI for locally stored data use case is presented in Figure 4.8.

4.5 EEGBase Portal Integration

Although my first effort was to import my program into the EEGBase portal, I was unable to do so. I tried to create a prototype which would save stored data into the HDF5 file in a web container. However I could not make Java wrappers work in the web container. I contacted HDF Group support for advice, however, the support and developers did not know how to develop such a program. I did some research and I tried to make the program work on the Tomcat server and Glassfish server but with no success. For those

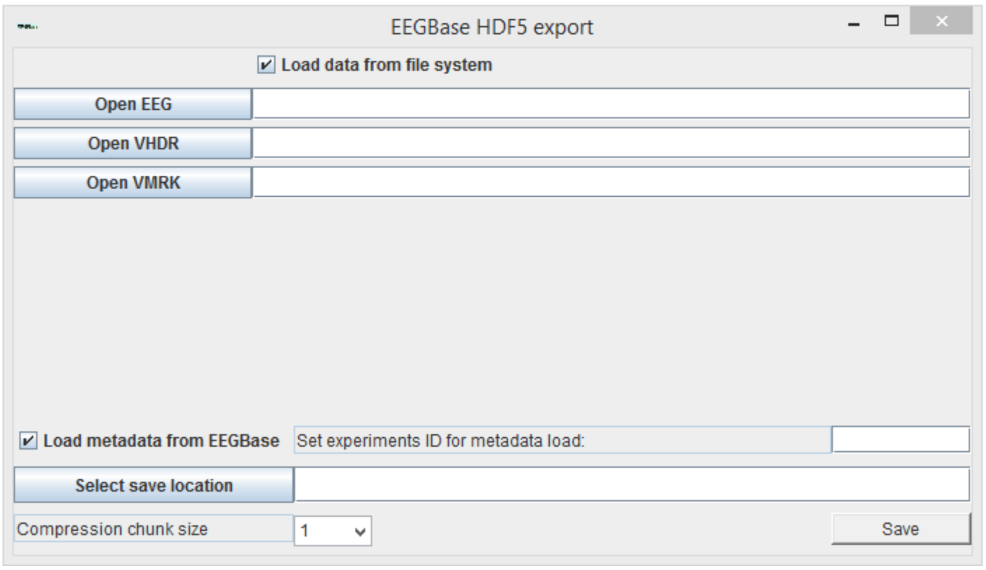


Figure 4.8: EEGBase HDF5 export program. Metadata are loaded trough web services.

reasons I chose to use the EEGBase portal SOAP web services to load data for export to the HDF5 container.

It was necessary to create a new web service, which will export all metadata of the experiment. The web service *getExperimentsMetadata* was created for that reason and it is used by my program.

5 Tests

The program was manually tested for several use cases and all created HDF5 files were verified manually (opened and the contents was checked) by official program of HDF Group HDFView [4] in version 2.11 - Figure 4.7. The following test describing writing speed into HDF5 container were conducted. (Table 5.1). The file size of created files were also tested. (Table 5.2).

5.1 Performance Tests

The write performance tests were conducted to determine time consumption of export. The tests were conducted on a standard desktop computer (processor Intel Core i7 at 3,4 GHz, 8 GB of DDR3 RAM, standard HDD with 7200 rpm). An unusually high memory consumption was detected during the test. The amount of occupied memory by the EEGExport program for big (220 MB) measurements reached up to 4 GB of memory. Further testing showed that Java Virtual Machine, in attempt to speed up export, does not free allocated memory. However, if the program was paused the amount of allocated memory was lower; the program could run with less memory. In the end the disk writing speed was a limited factor. The conversion times with GZip compression are shown in Table 5.1.

[1]Average time from four savings.

Table 5.1: Performance tests with GZip compression. Each file was saved seven times.

File size of Brain Vision files	**Time needed for conversion in ms**			
	Best	**Worst**	**Average**	**With compression**[1]
21,3 MB	3010	3501	3132	7529
51,2 MB	6160	7906	6700	26051
58 MB	7668	8262	7947	29032
87,2 MB	31364	35957	32820	44877
197,8 MB	13060	14625	13502	40168
221,3 MB	14589	16321	15197	42679
221,6 MB	14347	16108	14871	43586

5.2 File Size Test

Several test were conducted to determine the resulting file size of the HDF5 container containing all data. Data from real experiments were used for the tests. The test shows that size of the HDF5 container is influenced by exported data and the original file size is not the only decisive factor. The size of the HDF5 container is always bigger then Brain Vision files, at worst case even four times bigger in the best scenario about two times bigger. The results are shown in Table 5.2. The ratio is better for longer and bigger measurements. The GZip compression was used to reduce file size. The GZip compression is integrated in HDF libraries and it is supported natively (reading of the data does not require any special actions). The size of compression chunk was set to 256 after several tests. The file size was decreasing with bigger chunk size, but the performance was decreasing. This settings had the best results in file size compared to the impact on performance. With the GZip compression the file size ratio decreased to 1,62 in average. The influence of chunk size on file size is in Table 5.3. The comparison between GZip and SZip compression is in Table 5.4. The GZip compression is better for smaller experiments. With bigger data the SZip compression gets a smaller final file. However, most of experiments have smaller size (smaller than 100 MB) I chose the GZip compression as default compression algorithm.

Table 5.2: File size tests.

File size of Brain Vision files	**HDF5 file size**	**HDF5 file size** with compression	**Ratio** [2]
21,3 MB	80,9 MB	26,92 MB	1,26x
51,2 MB	194,2 MB	112,98 MB	2,21x
58 MB	221,1 MB	102,79 MB	1,77x
87,2 MB	329,7 MB	124,51 MB	1,43x
197,8 MB	370,9 MB	305,19 MB	1,54x
221,3 MB	414,9 MB	355,58 MB	1,61x
221,6 MB	415,5 MB	334,46 MB	1,51x

[2]HDF5 is x times bigger than original file size.

Table 5.3: File size dependency on compression chunk size.

Original file size	**HDF5 file size in MB**			
	chunk 64	chunk 256	chunk 512	chunk 1024
21,3 MB	39,76	26,92	23,30	20,46
51,2 MB	155,47	112,98	99,31	87,85
58 MB	151,31	102,79	88,60	77,96
87,2 MB	176,53	124,51	112,43	104,15
197,8 MB	365,63	305,19	292,59	285,48
221,3 MB	422,83	355,58	341,72	333,83
221,6 MB	402,81	334,46	320,22	312,10

Table 5.4: File size dependency on compression method. The chunk size is 256.

Original file size	**File size with SZip**	**File size with GZip**
21,3 MB	33,51 MB	26,92 MB
51,2 MB	138,01 MB	112,98 MB
58 MB	148,88 MB	102,79 MB
87,2 MB	145,53 MB	124,51 MB
197,8 MB	280,92 MB	305,19 MB
221,3 MB	331,01 MB	355,58 MB
221,6 MB	306,03 MB	334,46 MB

6 Evaluation and Summary

This thesis assignment was fulfilled. I examined current file formats for storing electrophysiology data and data from experiments and measurements conducted at the University of West Bohemia. I became familiar with the data and metadata model of EEG measurements and its terminology. I examined the data and metadata model of EEGBase portal.

Within my master thesis I analyzed two early file standard proposals from INCF and I tracked progress in both. I found several currently used formats, which are using HDF as a container in neuroinformatics. I chose the most suitable format for our data and usage considering the INCF recommendations.

I created my own implementation of the chosen format. I chose HDF5 container for the EEGBase format. I joined the INCF Electrophysiology Data Sharing Task Force and contributed to the odML terminology and ontology. I developed a program that transforms raw data and metadata from Brain Vision files to the EEGBase format, and I also included metadata which are stored in the EEGBase portal. I tested my format and program for several use cases and its performance.

The proposed EEGBase format is capable of storing all currently saved data and metadata and is able to save the future changes and modifications of metadata model. The developed program saves measured data into the EEGBase format. I also made a few suggestions for the UWB model. The program is currently using web services of the EEGBase portal for metadata loading. The developed libraries allow export of raw data or data with metadata. Also the GZip and the SZip compressions are included in the code, but the GUI offers only GZip compression (for tested data it had better compression results). Exporting experimental data and metadata in the EEGBase format to the HDF5 container improves sharing capabilities of the EEGBase portal and overall attractiveness of stored experiments. The next extension of the program could be loading and exporting files directly from the EEGBase portal trough web services. The GUI of EEGBase HDF5 export program could be improved to offer better selection of export options.

List of Figures

List of Tables

Listings

Glossary

API application programming interface. 29, 30, 41

BFO Basic Formal Ontology. 26

BSD Berkeley Source Distribution. 22

CNO Computational Neuroscience Ontology. 26

CSN Czech standards. 45

EDC Electronic Data Capture. 9

EDF European Data Format. 14

EEG Electroencephalography. 5, 8, 9, 11–19, 31, 35, 38, 39, 55

epHDF electrophysiology HDF. 18

ERP Event-related potentials. 8, 9

G-Node German Neuroinformatics Node. 32, 35

GUI Graphical user interface. 39, 50, 55

HDF Hierarchical Data Format. 20–23, 30, 38, 41, 50, 53, 55

HDF4 Hierarchical Data Format 4. 15, 41

HDF5 Hierarchical Data Format 5. 12, 14–16, 18, 20–24, 29, 30, 35, 36, 38, 39, 41–46, 48, 50–53, 55, 57

HDFds Hierarchical Data Format – data sharing. 18

INCF	International Neuroinformatics Coordinating Facility. 8, 11, 12, 16, 18, 30, 35, 55
ISO	International Organization for Standardization. 45
JHI	interface for Java HDF. 41
JHI5	The Java HDF5 Interface. 41
JSON	JavaScript Object Notation. 18
NIF	Neuroscience Information Framework. 25
NIFSTD	Neuroscience Information Framework Standard ontology. 25
OBO	open biomedical ontologies. 25, 26
odML	The open metadata markup language. 16, 18, 27, 28, 32, 34–37, 55, 57, 69–71
SOAP	Simple Object Access Protocol. 38, 51
UWB	University of West Bohemia. 8, 19, 32, 36, 37, 55
WSDL	Web Service Definition Language. 48
XML	Extensible Markup Language. 15, 21, 28

Bibliography

[1] *Basic Formal Ontology* [online]. 2014. [cit. 10.5.2015]. Available from: `http://ifomis.uni-saarland.de/bfo/`.

[2] *EEGbase* [online]. 2015. [cit. 10.5.2015]. RRID:nif-0000-08190. Available from: `https://eegdatabase.kiv.zcu.cz/`.

[3] *HDF Object Package* [online]. [cit. 1.5.2015]. HDF Group. Available from: `https://www.hdfgroup.org/products/java/hdf-object/index.html`.

[4] *HDF Java Products* [online]. 2014. [cit. 1.5.2015]. HDF Group. Available from: `https://www.hdfgroup.org/products/java/`.

[5] ČSN ISO 8601. *Data elements and interchange formats – Information interchange – Representation of dates and times.* Praha: ČESKÝ NORMALIZAČNÍ INSTITUT, 2005.

[6] *Neural ElectroMagnetic Ontology* [online]. 2015. 15.5.2015. Available from: `http://bioportal.bioontology.org/ontologies/NEMO/?p=summary`.

[7] *Open data specifications and software for neurophysiology* [online]. 2011. [cit. 10.5.2015]. Available from: `http://neuroshare.sourceforge.net/`.

[8] *Neuroscience Information Framework Standard ontology* [online]. 2015. [cit. 10.5.2015]. Available from: `http://bioportal.bioontology.org/ontologies/NIFSTD`.

[9] *INCF GitHub odML repository* [online]. [cit. 12.5.2015]. Available from: `https://github.com/INCF/odml-terminologies`.

[10] *The Open Biological and Biomedical Ontologies* [online]. 2015. [cit. 10.5.2015]. Available from: `http://www.obofoundry.org/`.

[11] Requirements for storing electrophysiology data, Version 0.72. Available from: `https://goo.gl/QbClkE`. Electrophysiology Task Force of the INCF Program on Standards for Data Sharing, 2014.

[12] BENDA, J. From recording to sharing of data - embedding metadata handling into the laboratory workflow using odML. *Frontiers in Neuroinformatics*. 2011, 5. `10.3389/conf.fninf.2011.08.00100`. Available from: `http://dx.doi.org/10.3389/conf.fninf.2011.08.00100`.

[13] Brain Products GmbH. *BrainVision Recorder User Manual*. Brain Products GmbH, 011 edition, December 2014.

[14] *Brain Products GmbH* [online]. 2014. [cit. 03.07.2014]. Brain Products GmbH. Available from: `http://www.brainproducts.com/index.php`.

[15] BRŮHA, P. Semantic Web and Classic Data Structures.

[16] BRINKMAN, R. R. et al. Modeling biomedical experimental processes with OBI. *Journal of Biomedical Semantics*. 2010, 1, Suppl 1, s. S7. `10.1186/2041-1480-1-s1-s7`. Available from: `http://dx.doi.org/10.1186/2041-1480-1-S1-S7`.

[17] *Computational Neuroscience Ontology* [online]. 2013. [cit. 13.6.2015]. Available from: `http://bioportal.bioontology.org/ontologies/CNO`.

[18] FRANC, Y. L. *Introduction to CNO* [online]. 2012. [cit. 15.5.2015]. Available from: `https://neuroml.org/files/NeuroML2012/YleFranc_CNO.pdf`.

[19] FRIEDRICH, S. et al. Mission and activities of the INCF Electrophysiology Data Sharing Task Force. *Frontiers in Neuroinformatics*. 2014, 8. `10.3389/conf.fninf.2014.08.00088`. Available from: `http://dx.doi.org/10.3389/conf.fninf.2014.08.00088`.

[20] GOSINK, L. et al. HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices. In *18th International Conference on Scientific and Statistical Database Management*. Institute of Electrical & Electronics Engineers (IEEE), 2006. `10.1109/ssdbm.2006.27`. Available from: `http://dx.doi.org/10.1109/SSDBM.2006.27`.

[21] Grewe, J. – Wachtler, T. – Benda, J. A Bottom-up Approach to Data Annotation in Neurophysiology. *Frontiers in Neuroinformatics*. 2011, 5. 10.3389/fninf.2011.00016. Available from: http://dx.doi.org/10.3389/fninf.2011.00016.

[22] *HDF Technologies* [online]. 2013. [cit. 03.07.2014]. The HDF Group. Available from: http://www.hdfgroup.org/.

[23] *INCF* [online]. 2015. [cit. 4.3.2015]. Available from: http://www.incf.org/.

[24] Jeffrey, T. – Friedrich, S. epHDF – a proposed standard for storing electrophysiology data in HDF5. *Frontiers in Neuroinformatics*. 2013, 7. 10.3389/conf.fninf.2013.09.00068. Available from: http://dx.doi.org/10.3389/conf.fninf.2013.09.00068.

[25] Jeffrey, T. et al. Considerations for developing a standard for storing electrophysiology data in HDF5. *Frontiers in Neuroinformatics*. 2013, 7. 10.3389/conf.fninf.2013.09.00069. Available from: http://dx.doi.org/10.3389/conf.fninf.2013.09.00069.

[26] Kemp, B. [online]. 2014. [cit. 4.5.2015]. Available from: http://www.edfplus.info/specs/edf.html.

[27] Kemp, B. – Olivan, J. European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. *Clinical Neurophysiology*. sep 2003, 114, 9, s. 1755–1761. 10.1016/s1388-2457(03)00123-8. Available from: http://dx.doi.org/10.1016/S1388-2457(03)00123-8.

[28] Liu, Z. – Ding, L. – He, B. Integration of EEG/MEG with MRI and fMRI in Functional Neuroimaging. *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine and Biology Society*. 2006, 25. ISSN 0739-5175.

[29] Luck, S. *ERPinfo.org* [online]. 2015. [cit. 22.3.2015]. Available from: http://erpinfo.org/what-is-an-erp.

[30] Miko, P. *EDEDClient* [online]. 2012. [cit. 9.1.2015]. Available from: https://github.com/stebjan/jerpa-EDEDClient.

[31] Milham, M. P. Open Neuroscience Solutions for the Connectome-wide Association Era. *Neuron*. jan 2012, 73, 2, s. 214–218. 10.1016/j.neuron.2011.11.004. Available from: http://dx.doi.org/10.1016/j.neuron.2011.11.004.

[32] *Neo* [online]. 2014. [cit. 03.07.2014]. Available from: `http://pythonhosted.org/neo/`.

[33] *NeuroHDF* [online]. 2014. [cit. 03.07.2014]. NeuroHDF Interest Group. Available from: `https://neurohdf.readthedocs.org/`.

[34] *NeXus Format* [online]. 2014. [cit. 03.07.2014]. NeXus International Advisory Committee. Available from: `http://www.nexusformat.org/`.

[35] *Ontology for Biomedical Investigations* [online]. 2015. [cit. 13.6.2015]. Available from: `http://bioportal.bioontology.org/ontologies/OBI`.

[36] *Ovation* [online]. 2014. [cit. 3.7.2014]. Physion LLC. Available from: `http://physion.us/`.

[37] *Ovation* [online]. 2014. [cit. 3.7.2014]. Physion LLC. Available from: `http://ovation.io/`.

[38] *NIX* [online]. 2014. [cit. 03.02.2015]. G-Node. Available from: `https://github.com/G-Node/nix/wiki`.

[39] *NIX* [online]. 2014. [cit. 03.02.2015]. G-Node. Available from: `https://github.com/G-Node/nix/wiki/Implementation-in-HDF5`.

[40] *Ovation Scientific Data Management System*. Physion Consulting. 2012. Release 1.4.

[41] Poldrack, R. A. The future of fMRI in cognitive neuroscience. *NeuroImage*. aug 2012, 62, 2, s. 1216–1220. `10.1016/j.neuroimage.2011.08.007`. Available from: `http://dx.doi.org/10.1016/j.neuroimage.2011.08.007`.

[42] Poline, J.-B. et al. Data sharing in neuroimaging research. *Frontiers in Neuroinformatics*. 2012, 6. `10.3389/fninf.2012.00009`. Available from: `http://dx.doi.org/10.3389/fninf.2012.00009`.

[43] Schalk, G. *BCI2000* [online]. 2009. [cit. 25.11.2014]. [rev. 2009-8-14]. Available from: `www.bci2000.org`.

[44] Smith, B. et al. *Genome Biol.* 2005, 6, 5, s. R46. `10.1186/gb-2005-6-5-r46`. Available from: `http://dx.doi.org/10.1186/gb-2005-6-5-r46`.

[45] Štěbeták, J. *EEGBase SOAP webservices* [online]. 2015. [cit. 10.5.2015]. Available from: `http://eeg2.kiv.zcu.cz:8080/webservice/`.

[46] Štěbeták, J. *Webservice WSDL file* [online]. 2015. [cit. 10.5.2015]. Available from: `http://eeg2.kiv.zcu.cz:8080/webservice/UserDataService?wsdl`.

[47] Štebeták, J. *EEGloader 2.0* [online]. 2015. [cit. 10.5.2015]. Available from: `https://github.com/stebjan/eegloader`.

A Brain Vision Files Examples

Listing A.1: The header file example - Information about format and channels and the amplifier setup.

```
[Binary Infos]
BinaryFormat=INT_16

[Channel Infos]
;Each entry: Ch<number>=<Name>,<Reference channel name>,
;<Resolution in "Unit">,<Unit>,Future extensions..
;Fields are delimited by commas, some fields might be omitted.
;Commas in channel names are coded as "\1".

Ch1=1,,0.1,µV
Ch2=2,,0.1,µV
...

Ch41=41,,0.1526,C
Ch42=42,,0.0763,mm
...
[Comment]
A m p l i f i e r S e t u p
============================
Number of channels: 48
Sampling Rate [Hz]: 200
\end{verbatim}
Sampling Interval [µS]: 5000

Channels

- - - - - - - -
```

	Name	Phys. Chn	Resol./ Unit	Low Cutoff [s]	High Cutoff [Hz]	Notch [Hz]	Series Res. [kOhm]	Gradient	Offset
1	1	1	0.1 μV	DC					
2	2	2	0.1 μV	DC					
...									
41	41	41	0.1526 C	DC					
42	42	42	0.0765 mm	DC					
...									

Listing A.2: The header file example - Information about used software filters.

```
S o f t w a r e F i l t e r s
==============================
# Low Cutoff [s] High Cutoff [Hz] Notch [Hz]
1 0.0006366 Off Off
2 0.0006366 Off Off
...
41 0.0006366 Off Off
42 0.0006366 Off Off
...
Impedance [kOhm] at 12:10:43:
1: Out of Range!
2: Out of Range!
...
41: Out of Range!
42: Out of Range!
...
Ref: Out of Range!
Gnd: Out of Range!
```

Listing A.3: The vmrk file example - information about stimuli

```
Brain Vision Data Exchange Marker File, Version 1.0

[Common Infos]
Codepage=UTF-8
DataFile=0000007.eeg

[Marker Infos]
; Each entry: Mk<Marker
   number>=<Type>,<Description>,<Position in data points>,
; <Size in data points>, <Channel number (0 = marker is
   related to all channels)>
; Fields are delimited by commas, some fields might be
   omitted (empty).
; Commas in type or description text are coded as "\1".
Mk1=New Segment,,1,1,0,20120503112002605235
Mk2=Stimulus,S 7,110437,0,0
Mk3=Stimulus,S 7,110673,0,0
..
Mk1233=Stimulus,S 3,3209901,0,0
```

B Metadata Terminology

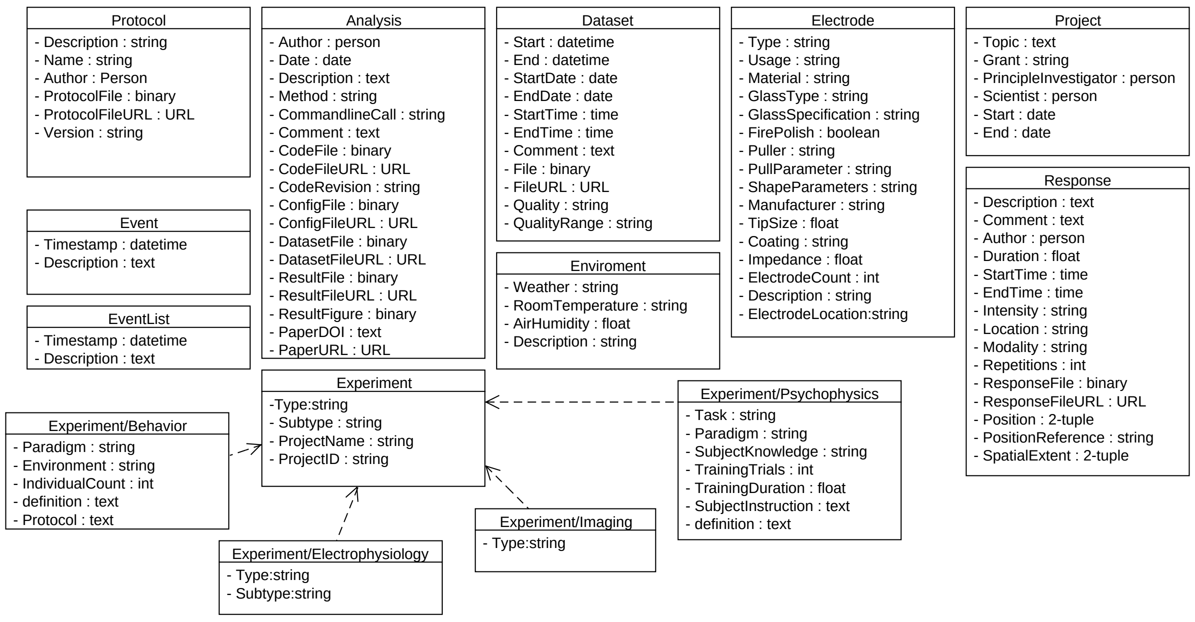


Figure B.1: The odMLterminology. Part 1.

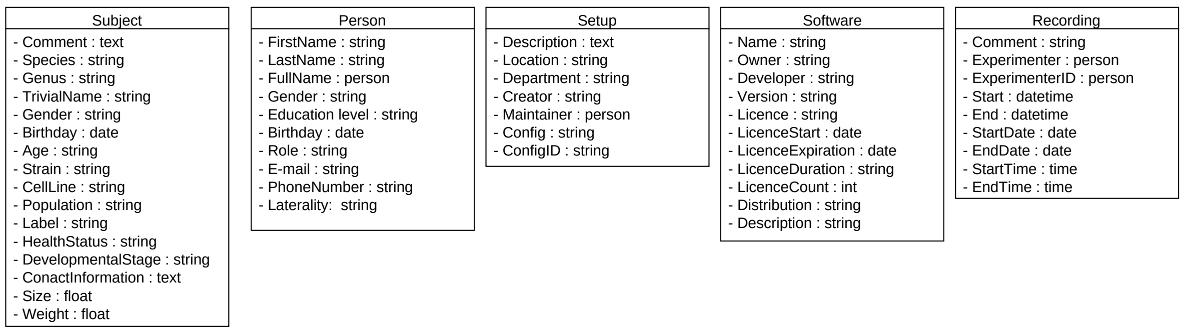


Figure B.2: The odMLterminology. Part 2.

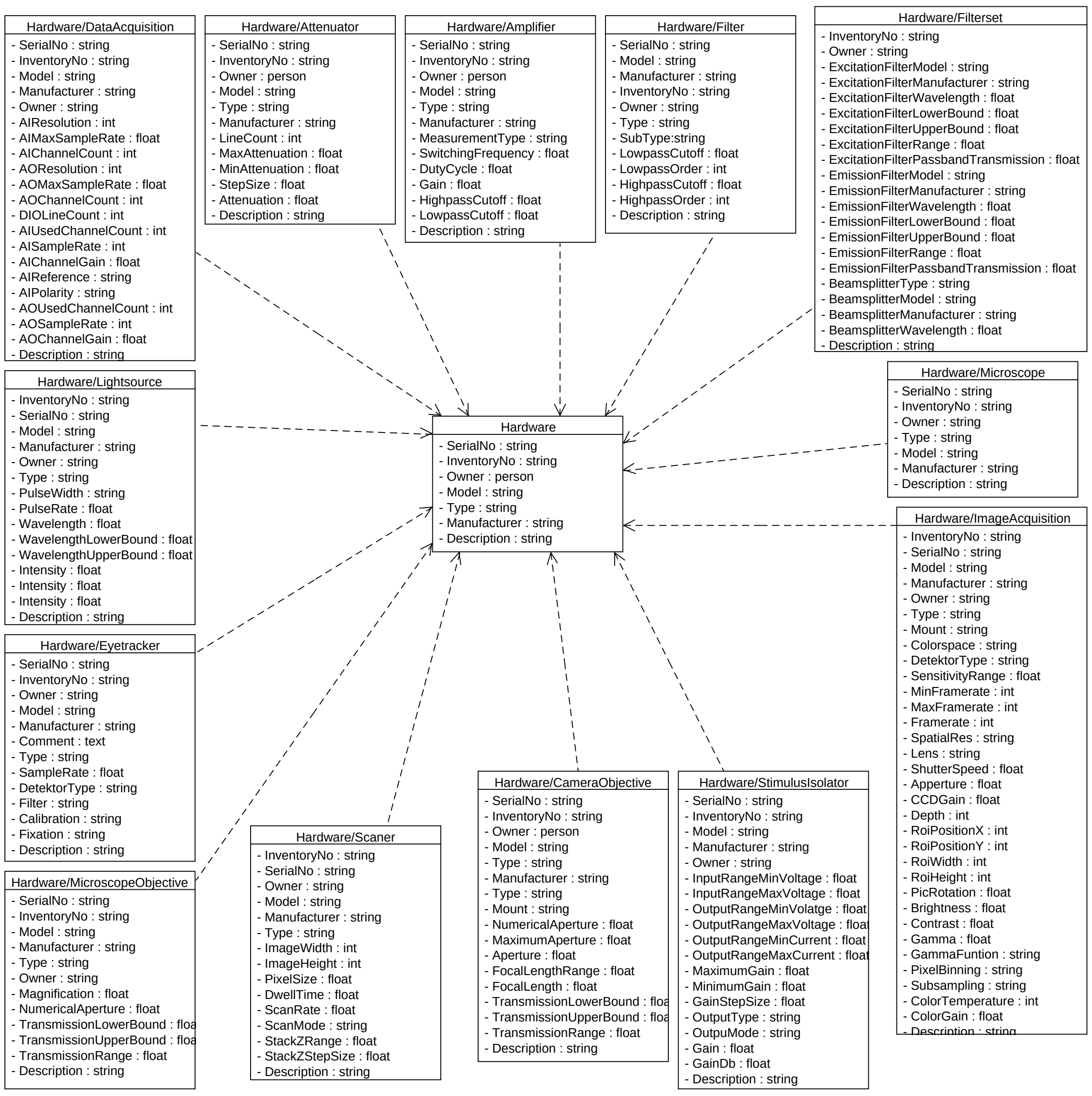


Figure B.3: The odMLterminology. Part 3.

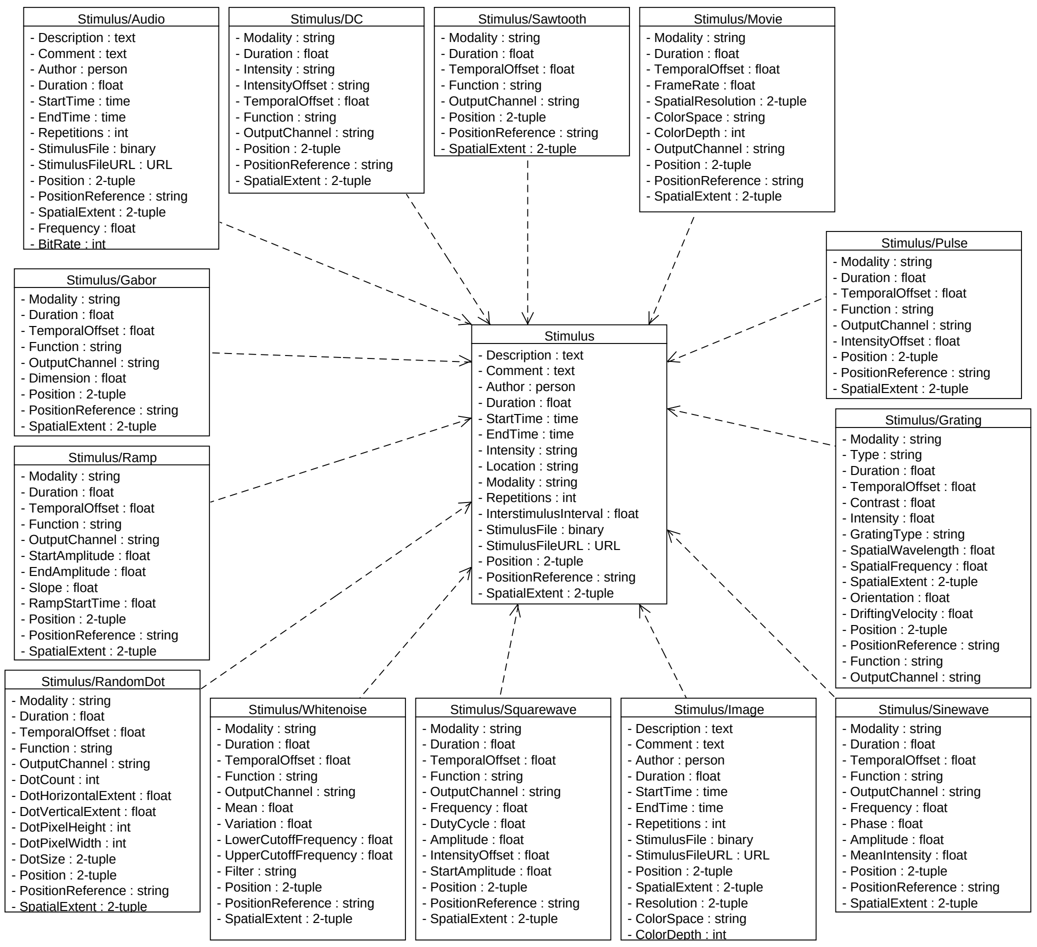


Figure B.4: The odMLterminology. Part 4.

C Metadata Scheme

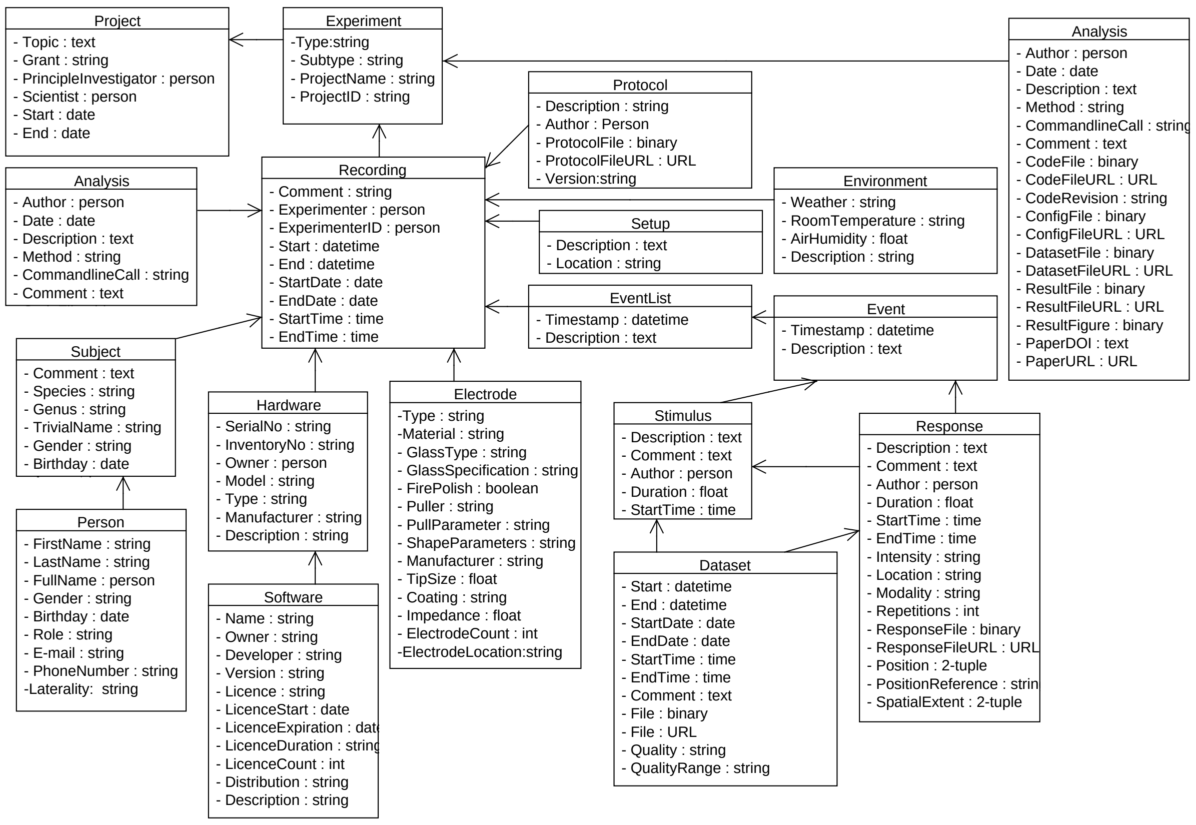


Figure C.1: Metadata scheme of EEGBase format.

D Export Program User Manual

D.1 Installation

The EEGExport program needs for its functionality installed HDF5 libraries version 2.10 or higher and Java Runtime Environment in version 1.7 or higher. The pre-build binary distribution could be obtained from official HDF Group website.

The EEGExport program do not require installation. Program is distributed as runnable jar and could be lunched with supplied "run.bat" file. **The path to installed HDF libraries must be in the bat file for the correct function filled.** (Listing D.1).

Listing D.1: Bat file

```
@ECHO OFF
rem Fill in hdf_path path to installed HDF libraries
set hdf_path=C:\HDF_Group\HDF-JAVA\2.11.0\lib\jhdf5.dll
java -Dncsa.hdf.hdf5lib.H5.hdf5lib=%hdf_path% -jar program.jar
```

D.2 Data Export

The location of hvdr, eeg and vmrk files and exported files location and file name must be set before export. The load of eeg files from EEGBase portal is not working in current version. So the option 1 (Figure D.1) must be selected.

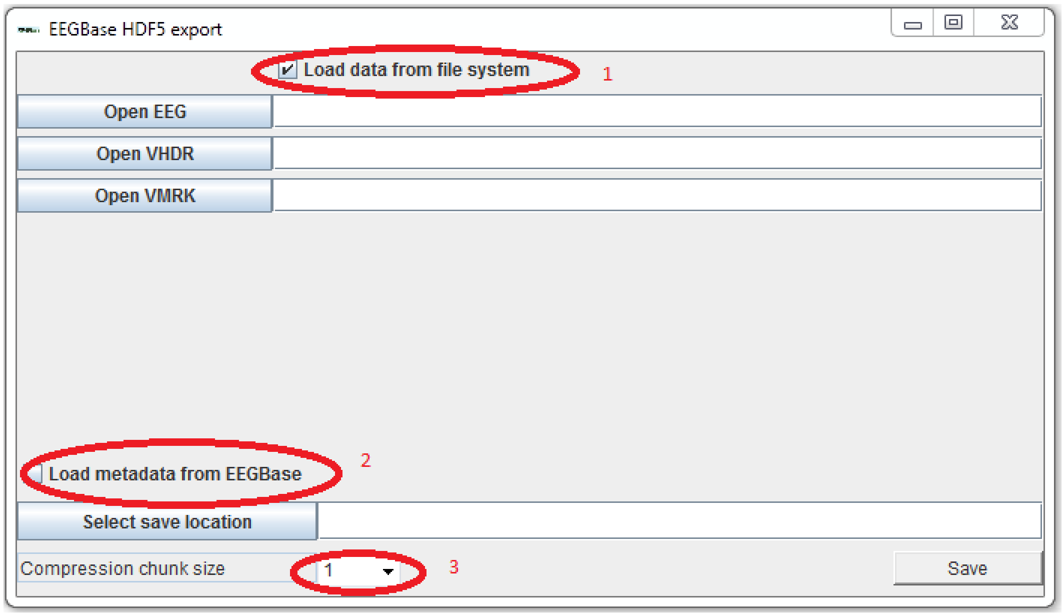


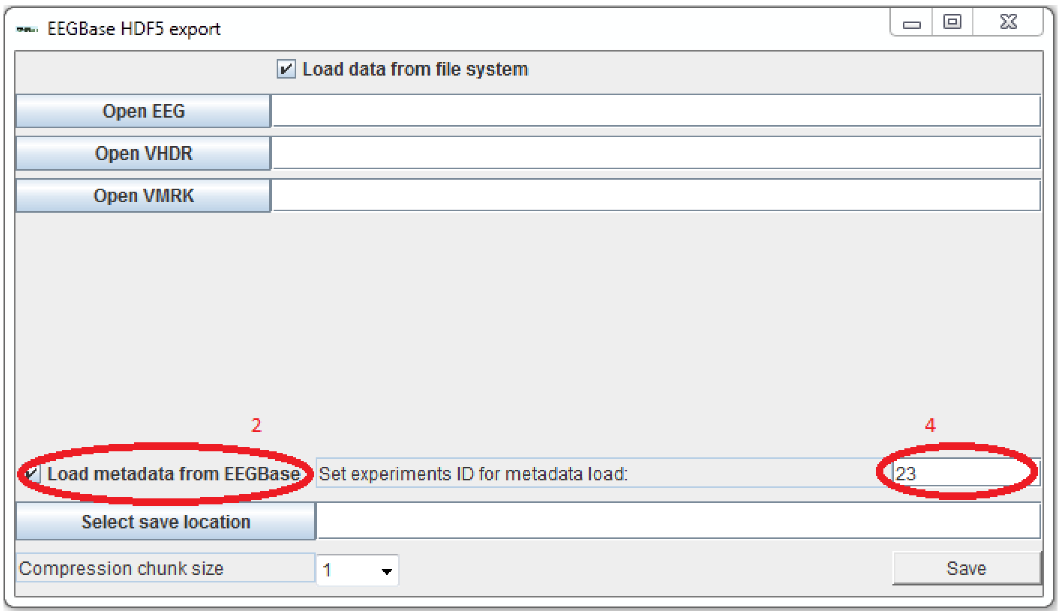
Figure D.1: The EEGExport program GUI

D.3 Metadata Export

If the metadata from EEGBase portal should be exported as well the option 2 must be checked and the number of experiment must be filled (Figure D.2).

D.4 Compression

The final file size could be influenced by compression. The amount of compression is set by chunk size. The chunk size 1 means no compression. Bigger chunk size sets bigger compression so the file size is smaller.



Load metadata from EEGBase [✓]

Set experiments ID for metadata load: 23

Compression chunk size: 1

Figure D.2: The EEGExport program GUI