FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

Archetype-based approach for modelling of electroencephalographic/event-related potentials data and metadata

Ing. Václav Papež

Doctoral thesis submitted in partial fulfillment of the requirements for a degree of

Doctor of Philosophy in specialization *Computer Science and Engineering*

Supervisor: Ing. Roman Mouček, Ph.D.

Department: Department of Computer Science and Engineering

Plzeň 2017

FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Archetypový přístup k modelování dat a metadat z oblasti elektroencefalografie a evokovaných potenciálů

Ing. Václav Papež

Disertační práce k získání akademického titulu
doktor v oboru *Výpočetní technika*

Školitel: Ing. Roman Mouček, Ph.D.
Katedra: Katedra informatiky a výpočetní techniky

Plzeň 2017

I dedicate this thesis to my grandfather who has always wished to see me finishing what I have once started.

Declaration/Prohlášení

Acknowledgements

I would like to acknowledge all members of the neuroinformatics research group at the Department of Computer Science and Technologies, University of West Bohemia (the Department), headed by Ing. Roman Mouček, Ph.D., my supervisor and friend. I would like to thank Roman and his group for giving me valuable advice throughout my Ph.D. studies, and for providing me with wonderful opportunities in exciting neuroinformatics research community.

I would also like to thank Dr Heather Leslie and Ms. Silje Ljosland Bakke, members of the openEHR community, for their consultations regarding the achetyping process, and Dr Kenan Direk and Dr Ghazaleh Fatemifar, friends and colleagues, for helping me with the language revision.

I am grateful to doc. Ing. Josef Kohout, Ph.D., head of the bio-informatics division at the Department, and doc. Ing. Přemek Brada, MSc., Ph.D., head of the Department, for their encouragement, motivation, and for allowing me to further my academic skills through various international research fellowships and events.

I would also like to express gratitude to my alma matter, University of West Bohemia, for giving me the opportunity to gain priceless knowledge, uncover my abilities and became a better man in both professional and personal life.

Finally, my deepest gratitude belongs to those who stood by me and supported me all the time; to my beloved parents and grandparents, to my darling partner Zuzana, and to my dearest friends.

Abstract

Currently, there is no common data standard in the experimental electroencephalography/event-related potential (EEG/ERP) domain. Existing standardization efforts are mainly based on the conventional approaches and use generic data formats and containers (e.g. HDF5, odML) popular in the research community. This work draws on the medical/health characteristics of EEG/ERP data and investigates the feasibility of applying openEHR (an archetype-based approach for electronic health records representation) to modelling data stored in EEGBase, a portal for experimental EEG/ERP data management. The work evaluates re-usage of existing openEHR archetypes and proposes a set of new archetypes together with the openEHR templates covering the domain. The main goals of the work are to (i) link existing EEGBase data/metadata and openEHR archetype structures; (ii) propose a new openEHR archetype set describing the EEG/ERP domain since this set of archetypes currently does not exist in public repositories.

Apart from that, the work describes common data models (e.g. relational, object-oriented) and compares their expressive power in order to (i) determine the elements, which these models have in common; (ii) build a data model hierarchy according to their expressive power. The work uses the proposed archetypes and their reference models as semantic schemata to derive a specific data model for each level of the hierarchy. Finally, the work describes a newly proposed personal electronic health records system for research purposes, which serves as a first use-case of obtained results.

Abstrakt

V současné době neexistuje obecný datový standard v oblasti experimentální elektroencefalografie a metody evokovaných potenciálů (EEG/ERP). Stávající snahy o vytvoření takového standardu jsou z většiny založeny na konvenčních přístupech a využívají generické datové formáty a kontejnery (např. HDF5, odML) oblíbené ve vědecké komunitě. Tato práce využívá medicínských/zdravotních charakteristik EEG/ERP dat a prozkoumává vhodnost použití openEHR (způsob reprezentace elektronických zdravotních záznamů založených na archetypech) k modelování dat uložených v EEGBase, portálu pro zprávu experimentálních EEG/ERP dat. Práce vyhodnocuje opětovné použití existujících openEHR archetypů a navrhuje sadu nových archetypů spolu s openEHR šablonami pokrývajícími danou oblast. Hlavními cíli práce jsou (i) propojení existujících EEGBase dat/metadat se strukturami archetypů; (ii) návrh nových archetypů popisujících EEG/ERP doménu, jelikož tyto v současné době neexistují ve veřejných repozitářích.

Krom výše uvedeného, práce popisuje stávající běžné datové modely (např. relační, objektově orientovaný) a porovnává jejich vyjadřovací sílu za účelem (i) vymezení prvků, které mají tyto modely společné; (ii) vystavění hierarchie datových modelů v závislosti na jejich vyjadřovací síle. Práce využívá navržených archetypů a jejich referenčních modelů jako sémantických schémat k odvození specifických datových modelů pro jednotlivé vrstvy hierarchie. V závěru práce popisuje nově navržený systém osobní zdravotní knížky pro výzkumné účely, který slouží jako první případ užití získaných výsledků.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ADL	Archetype Definition Language
API	Application Programming Interface
AQL	Archetype Query Language
BLOB	Binary Larger Object
BP	Blood Pressure
CARMEN	Code Analysis, Repository & Modelling for e-Neuroscience
CDA	Clinical Document Architecture
CKM	Clinical Knowledge Manager
CLOB	Character Larger Object
CNS	Central Nervous System
CNV	Cognitive Negative Variation
CSV	Comma Separated Values
CUI	Concept Unique Identifier
CWA	Close World Assumption
DTD	Document Type Definition
ECG	Electrocardiography
EDF	European Data Format

EEG	Electroencephalography
EEGBase	EEG/ERP Portal
EHR	Electronic Health Record
EMG	Electromyography
EMR	Electronic Medical Record
ER model	Entity-Relational Model
ERP	Event-Related Potentials
GP	General Practice
HDF5	Hierarchical Data Format
HL7	Health Level Seven
HTTP	Hypertext Transfer Protocol
ICD-10	International Classification of Disease 10th Revision
IDE	Integrated Development Environment
IDL	Interface Definition Language
INCF	International Neuroinformatics Coordinating Facility
IRI	Internationalized Resource Identifier
IT	Information Technology
LOV	List of Values
MEG	Magnetoencephalography
MIREOT	the Minimum Information to Reference an External Ontology Term
NEMO	Neural ElectroMagnetic Ontologies
NIF	Neuroscience Information Framework
NIFSTD	NIF Standardized Ontology
NIH	National Institutes of Health

NLP	Natural Language Processing
NWM	Neurodata Without Borders
OBD	Open Biomedical Ontologies
OBI	The Ontology for Biomedical Investigations
OCL	Object Constraint Language
odML	open metaData Markup Language
OEN	Ontology for Experimental Neurophysiology
OID	Object Identifier
OMG	Object Management Group
OMT	Object Modelling Technique
OO	Object-Oriented
OODB	Object-Oriented Databases
OOSE	Object-Oriented Software Engineering
ORDB	Object-Relational Databases
ORM	Object-Relation Mapping
OWA	Open World Assumption
OWL	Web Ontology Language
PHR	Personal Health Record
PNS	Peripheral Nervous System
R model	Relational Model
RCD	Clinical Terms Version 3 (Read Codes)
RDF	Resource Description Framework
RDFS	RDF Schema
REST	Representational State Transfer

RIF	Rule Interchange Format
RIM	Reference Information Model
RM	Reference Model
SGML	Standard Generalized Mark-up Language
SNOMED-CT	Systematized Nomenclature of Medicine - Clinical Terms
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SWRL	Semantic Web Rule Language
UK	United Kingdom of Great Britain and Northern Ireland
UML	Unified Modelling Language
UMLS	Unified Medical Language System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Mark-up Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

Part I

Opening

Chapter 1

Introduction

The domain of neuroscience is currently one of the most progressive fields in health care research (as witnessed in, e.g., the Horizon 2020 Societal Challenges list). There has been a rapid expanse in electroencephalography (EEG)/event-related potential (ERP) data resources, e.g. data from clinical EEG/ERPs, experimental EEG/ERPs, neuro-rehabilitations, assistive systems based on EEG/ERPs, household BCI (Brain-Computer Interface) devices; necessitating the need for stricter requirements on the data formats and storages used. Nevertheless, there is still paucity of matured data formats and standards in the experimental and clinical EEG spheres. Moreover, many existing software solutions have been designed only for internal purposes of the user group in which the software solution has been developed.

The EEGBase portal [33] (developed at the University of West Bohemia) is a software tool focused on annotation, storage, management and sharing of EEG/ERP experiments. EEGBase partially implements a selection of existing standardization efforts and uses semantic web technologies. Its uses are most applicable to EEG/ERP settings. This close relation imposes a limitation not only in the case of stored experiment types but also for stored experiment metadata which could be considered as stand-alone experiments (stand-alone medical reports). The main objective of this work is to utilize EEG/ERP health data characteristics to extend the interoperability potential of neuroinformatics databases like EEGBase.

The vast majority of advanced health institutions archive medical health records electronically. However, these electronic health records (EHR) are very often unstructured and therefore require a significant effort to facilitate machine readability, which consequently hinders use and exchange of such records between institutions. The standardization of communication protocols and structured data models describing health domains increases overall data interoperability, unambiguity and readiness for further analysis. Since these abilities are important for most, if not all, forms of health data, a standardized data structure is considered to be essential for its usability.

The standards supporting health data interoperability are based on (i) the explicit description of data meaning, (ii) terminology and structure separation, and (iii) controlled vocabularies integration. The openEHR [34] approach provides a multi-model, single source EHR framework. openEHR data models (archetypes) stored in the openEHR CKM (Clinical Knowledge Manager) public repository have direct application in clinical and non-clinical medical data. While the CKM contains hundreds of archetypes describing many medical domains (suitable for metadata), it lacks one for EEG, hence a set of new archetypes covering the EEG (EEG/ERP respectively) domain is proposed in this work. These archetypes are derived from the EEGBase data/metadata structure as well as from other common well-known EEG data formats. The terminology is mainly based on a controlled vocabulary taken from the *odML terminology for electrophysiology*.

openEHR archetypes by design are technologically independent constraint-based models. This work also considers generic, technologically dependent, data modelling concepts including relational model, object-oriented model, RDF, etc. Expressive powers of these concepts are evaluated, compared against each other and organized into a hierarchy. Then, a domain specific data model of each level of the hierarchy is proposed according to created archetypes.

The work is organized as follows: The context is described in part II: *Work Context, Materials and Methods*. Description and comparison of existing common data formats, models and concepts are described in part III: *Data Modelling*. Part IV: *Electronic Health Records* describes the application of EEG/ERP data health characteristics to development of a new set of openEHR archetypes. Finally, part V: *Results* evaluates these archetypes against selected criteria and EEGBase data, describes their benefits and application within recently proposed experimental EHR system, specifies their position in the semantic hierarchy, and discusses their application in clinical sphere.

Part II

Work Context, Materials and Methods

Chapter 2

Electroencephalography / Event-Related Potentials

To ease a better understanding of the characteristics of the electrophysiological data, which is examined in this work, a brief domain background is provided in this chapter, describing how the neuronal synapses emerging and what event-related potentials are and how they can be evoked. The chapter concludes by providing a high level overview of the laboratory experimental set-up.

2.1 Brain activity, neurons, synapses

The human nervous system is divided into two parts – Central Nervous System (CNS) and Peripheral Nervous System (PNS). The CNS contains brain and spinal cord; The PNS contains all other remaining parts (i.e. nerves and ganglia outside the spinal cord and the brain). The majority of nervous system is composed of neurons, these are electrically excitable cells consisting of a cell body, an axon, and network of dendrites (Figure 2.1). Enzymes and genetic material are contained in the cell body. An electrical impulse travels along the axon and activates neuron transmitters that stimulate dendrites of the neighbour neuron at synapses. Neurons have two states: (i) receiving the signal - charging and (ii) sending the signal - discharging. Overall electrical charge of the human brain can be detected and measured non-invasively via electrodes placed on the scalp in a process known as electroencephalography (EEG).

2.2 Electroencephalography

During the electroencephalography experiment, pairs of electrodes are attached to the scalp. Then, a voltage difference between paired electrodes, the potential, is measured and recorded over time and brain excitation movement can be observed. A rhythmic fluctuation of this potential difference is observable as series of peaks in the brain wave signal. An output of the measurement is represented as a set of waveforms; i.e. one waveform for each channel (pair of electrodes) [8].

The non-invasiveness of the EEG facilitates its use, however obtained data are very rough. Skull bones thickness, a distance between electrode and an active part of the brain adds a significant volume of noise into the output data. The brain signal is muffled on the skull surface and as the electrodes are not positioned directly in the excitation epicentre, they are scanning a signals from wider area of brain at once. For these reasons the signal must first be processed in order to be analysed for research.

EEG measurement time range could variate from milliseconds (e.g. brain reaction on an evoked single event) to days and more (e.g. long-term epilepsy monitoring; brain activity monitoring in patients in comma). In our neuroinformatics lab we are focusing only on short-term recordings, specifically on the Event-Related Potentials (ERP) method.

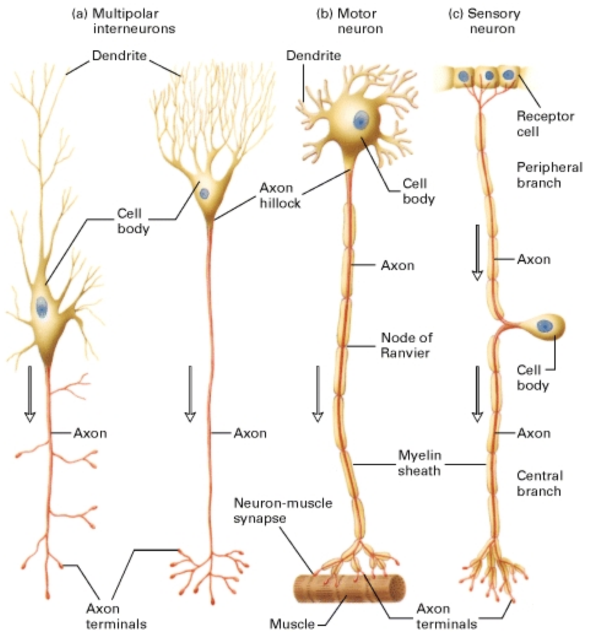


Figure 2.1 A structure of the typical mammalian neurons [42, Figure 21-1]

2.3 Event-Related Potentials

ERP[1] method is based on an evocation of cognitive or motor event and subsequent observation of the brain response for the purpose of determining a brain waveform pattern - the cognitive component. The history of the ERP method begins in the mid 1930s, but the modern ERP and cognitive component discovery came thirty years later. In 1964, neurophysiologist Grey Walter discovered the first cognitive component, the Cognitive Negative Variation (CNV), that is evoked approximately half a second before the subjects realize the movement that they are supposed to do. More recently, significant components have been discovered and a labelling system reflecting their characteristics is being used to mark them. The components are marked by letters according to the electrical polarity: P (positive), N (negative) and C (polarity is not stable). The letter is followed by a number specifying a delay between stimulus and occurred component (e.g. N1 – negative component emerges 100 milliseconds after the stimulus; P3 – positive component emerges 300 milliseconds after the stimulus). Each component has a relationship to a specific stimulus type, e.g., acoustic or visual. Stimuli components could be evoked by more then one stimulus type (e.g. P1 is evoked by an acoustic as well as a visual stimulus). Two frequently discussed components and also components that we are focused on in our neuroinformatics group are N2 and P3. N2 is a negative component emerging approximately 200 milliseconds after visual and/or auditory stimulus. A measurement protocol is focused on frequently repeated non-target stimulus (N2a) and non-frequently repeated target stimulus. A component emerging after the target stimulus means that the brain reacts on a mismatch in the pattern of frequently used stimuli. P3 is a positive component emerging 300 milliseconds after auditory or visual stimuli. The component is evoked after an unexpected event (type P3a) or an expected, but not a frequent event (type P3b). A full list of particular components and their detailed descriptions can be found in, e.g., [43].

EEG data processing for the purpose of ERP examination consists of the following:

- Measured sample (single waveform) is divided into epochs, i.e. periods delimited by a short time before and after the time-point the stimulus had been occurred
- More epochs extracted from various resource channels (paired electrodes), but capturing same event, are averaged
- Various filters for reducing signal artefacts (e.g. subject's blinking) are applied on averaged epochs
- Filters for reducing noise in the signal are reduced

[1]Sometimes also Evoked Response Potential

2.4 EEG / ERP Laboratory

The fact that EEG/ERP measurement is (i) non-invasive and (ii) realizable with a relatively low cost equipment gives researches an opportunity to perform EEG/ERP experiments in the environment of small experimental laboratories, a typical set-up is shown in Figure 2.2. Our neuroinformatics laboratory specialises in the analysis of EEG/ERP experiments and was established at the University of West Bohemia under the Department of Computer Science and Engineering (http://neuroinformatics.kiv.zcu.cz/) in 2003. Currently, our laboratory is equipped with a soundproof cabin (to screen out the measured subject from environmental disturbing effects), BrainProducts GmhB software and hardware equipment (32 channels EEG recorder BrainAmp, BrainVision recording software), own solution of hardware stimulator, PresTi Presentation software for scripting experimental (stimuli) protocols, computer for playing experimental protocols, computer for recording EEG data, USB adapter for synchronizing protocol with recording, electrode EEG caps, and real-size car simulator.

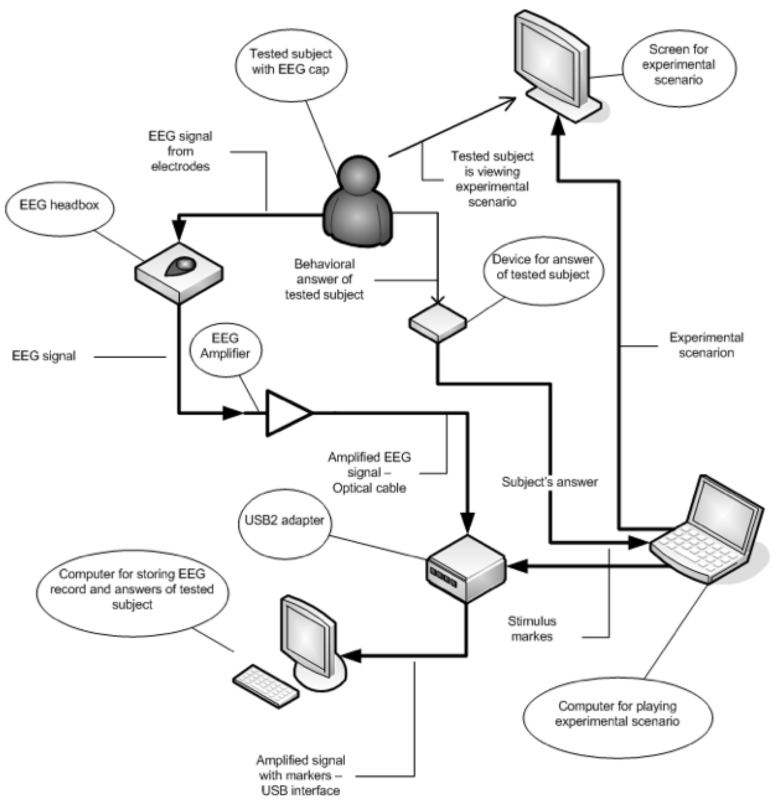

Tested subject with EEG cap
Screen for experimental scenario
EEG signal from electrodes
Tested subject is viewing experimental scenario
EEG headbox
Behavioral answer of tested subject
Device for answer of tested subject
EEG signal
EEG Amplifier
Experimental scenarion
Amplified EEG signal – Optical cable
Subject's answer
USB2 adapter
Computer for storing EEG record and answers of tested subject
Stimulus markes
Computer for playing experimental scenario
Amplified signal with markers – USB interface

Figure 2.2 Basic laboratory set-up utilized in the experimental EEG/ERP measurements [49]

Chapter 3

Neuroinformatics

Neuroinformatics is a scientific field combining neuroscience and informatics that aims to answer complex questions, which usually require sophisticated mathematical apparatus from data aggregated from diverse resources.

3.1 Aims of neuroinformatics

Answering complex neuroscience questions such as how to cure and prevent brain diseases and dysfunctions like Asperger syndrome, anorexia, epilepsy, Parkinson's disease, etc. requires deep knowledge from various neuroscience sub-fields. However, those sub-fields (e.g. moleculular neuroscience, neuroimmunology/neurovirology, neuropharmacology, neurogenetics, developmental neuroscience, cellular neuroscience etc.) are largely siloed, mutually independent and an interaction between researchers from different sub-fields is often insufficient. A common characteristic for all sub-fields is gathering a significant amount of data. This data is stored in diverse formats from domain-specific terminologies across the sub-fields. This fact negatively influences data interoperability and sharing abilities and is detrimental to the elimination of notional gaps between sub-fields. The neuroinformatics domain is supposed to provide at least a partial solution to mentioned problems. Neuroscientist Dr Marja-Leena Linne characterises neuroinformatics in her lectures as an "Integration of information across all levels and scales of nervous system – from genes to behaviour.".

A development in the neuroinformatics domain has three specializations and an intersection of these specializations represents the aim of neuroinformatics (Figure 3.1):

- Development of tools and databases for management and sharing data
- Development of tools for analysis and modelling (signal processing)

- Development of computational models of nervous system

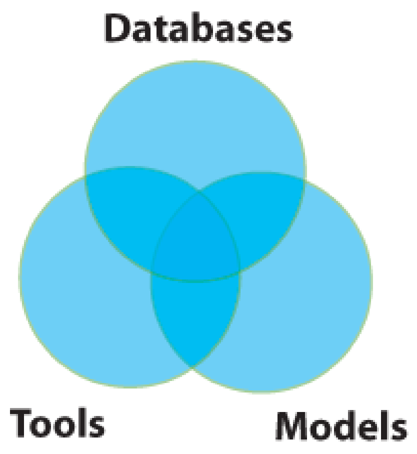


Figure 3.1 Neuroinformatics development directions (https://www.incf.org/about)

This work presented in this thesis is positioned in the first development branch, i.e. data management and sharing.

3.2 International Neuroinformatics Coordinating Facility

To coordinate the development in various neuroinformatics centres on the international level, International Neuroinformatics Coordinating Facility (INCF) has been established in 2005 in Stockholm. During the 1st INCF Workshop on Sustainability of Neuroscience Databases [30] the following recommendations at the INCF were proposed:

1. establishes a moderated web-based infrastructure with specific issues for discussion by the community
2. engages peer-reviewed journals in the process of identifying domain-specific minimal information recommendations for the sharing and sustainability of neuroscience data
3. identifies specific types of data/databases and a set of researchers who are generating and disseminating these data to form a special interest group that will develop the minimal information standards for that data/database
4. identifies specific types of models/tools and a set of researchers who are generating and disseminating these theoretical/computational models to form a special interest group that will develop the minimal information standards (in appropriate exchange formats, I/O, GUIs, etc.) for those models/tools

5. investigates existing neuroscience data/tools/models/clearinghouses and examines how they can engage in coordinating dissemination activities
6. examines how to serve as an accreditation body
7. can facilitate grass-roots recognition of need for data/database sustainability

This work contributes to points three and five.

3.3 Neuroinformatics data and metadata

There are two main data management challenges common to any domain:

- How to store the data?
- Where to store the data?

The first question is mainly related to a logical structure of the data and data annotation by metadata. The second question is related to the less abstract physical structure and more the technological solution.

Even though both challenges are partially dependent and they have to respect each other, there is an effort to separate them as possible. Format developers are working to develop a universal data container for the content of any structure, whilst data scientists are proposing suitable data structure without deep consideration of further technical implementation. This work is focused mainly on the first challenge, however, it discusses also the influences on the second one.

3.3.1 Data vs metadata

Common categories of metadata are structural and descriptive (sometimes also administrative, e.g., creation date, file type [53]). Structural metadata describes the structure in which data is stored (typically a table header) and descriptive metadata describes (or identifies) the nature and origin of data. In the case of EEG recordings, we can consider raw brain waveforms as data. The structure in which this data is stored (commonly binary data representation) is described by structural metadata. Descriptive metadata contains information about data acquisition and in most cases, provides the knowledge necessary to provide the correct technical data interpretation, e.g., the number of electrodes, electrode impedance or event timestamps. In the case of the BrainVision format (described in section 3.3.2), this metadata is included in the respective output files (.vmrk and .vhdr files). Unless stated otherwise, the metadata discussed is hereafter considered as descriptive.

Metadata related to the whole experiment (environmental conditions, subject state, etc.) is important for the context interpretation of the recording. Let us consider that the recording workflow is described by a scenario, and each scenario has its own metadata set related to the scenario raw data. With each new metadata set, like scenario metadata, the answer for the question "How to store the data/metadata?" becomes more ambiguous. These metadata sets are stored independently of each other (as they are related to different data) and cannot be easily aggregated. Moreover, if we consider an example when the EEG experiment is extended with a blood pressure (BP) measurement, then the BP measurement is a part of the metadata set related to the whole experiment, i.e. BP is included in the semantic context of the experiment. The BP measurement output is defined by two numerical values representing the systolic and diastolic blood pressures. Both values (data) have their metadata: e.g. units (mm Hg), measurement date and time or recording device. Even though the BP (meta)data is a part of the EEG experiment metadata set, it is also an autonomous fully-fledged measurement output. This example shows that the definition of what data and metadata are and where the border between data and metadata is, is a matter of perspective.

Consider the following, if data is a potential analysis input, then metadata could be considered an analysis input filter and/or an analysis parameter. The BP data and metadata from the previous example could be analysed separately or could serve as an input filter for the next EEG/ERP data analysis. Since EEG/ERP and BP recordings are logically separated for the next analysis, they have to be also separated structurally to achieve an efficient computational process. (Deeply embedded BP recording data in the EEG/ERP experiment metadata structure would be hard to process independently).

Generally, metadata can be divided into concepts according to what they are referring to. Many of those concepts are identical for differently focused systems and thus those systems should implement the same metadata set. The concept determination and metadata reference frames are taken into consideration for the proposal of a data description further in the work.

3.3.2 Formats

Although there is no common and widely accepted universal format for electrophysiology data/metadata, the current standardization initiatives/efforts and major projects and outcomes are presented.

Open metaData Markup Language

open metaData Markup Language (odML) [23] developed by the G-Node (German Neuroinformatics Node) is an explicit specification of the metadata exchange format, which is generic enough to store textual metadata from any scientific discipline. The model allows to construct a tree-like structure from the *Sections*. For each *Section*, a set of properties and values could be defined, as shown in Figure 3.2. In addition to this generic format the *odML terminology* for electrophysiology was established. The usage of the odML structure together with the terminology for electrophysiology provides a machine-readable metadata set limiting the ambiguity of the used terms.

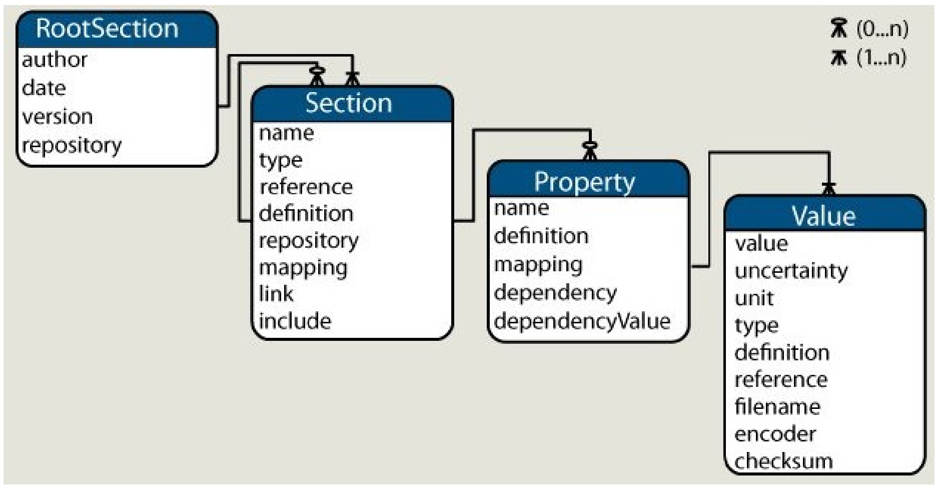


Figure 3.2 odML data model

NIX format

G-Node's follow-up project, NIX [66], extends odML by solving the related data storage issue. NIX defines a generic data model to represent data and metadata with flexible back-ends and provides an open standardized data format. The current authors' implementation combines the odML structure for metadata and the HDF5 [20] container for data, i.e. NIX defines a standard schema for HDF5 files to represent the generic model (Figure 3.3 and 3.4). The basic NIX structure defines data (typically time series) as n-dimensional *DataArrays*. Each *DataArray* is related to its data *Source* (e.g. channel specification) and has its *Dimension*. The data can be annotated by *Tags* to define its specific parts (e.g. time points or time intervals). All parts of the structure can be also annotated by additional odML metadata sections.

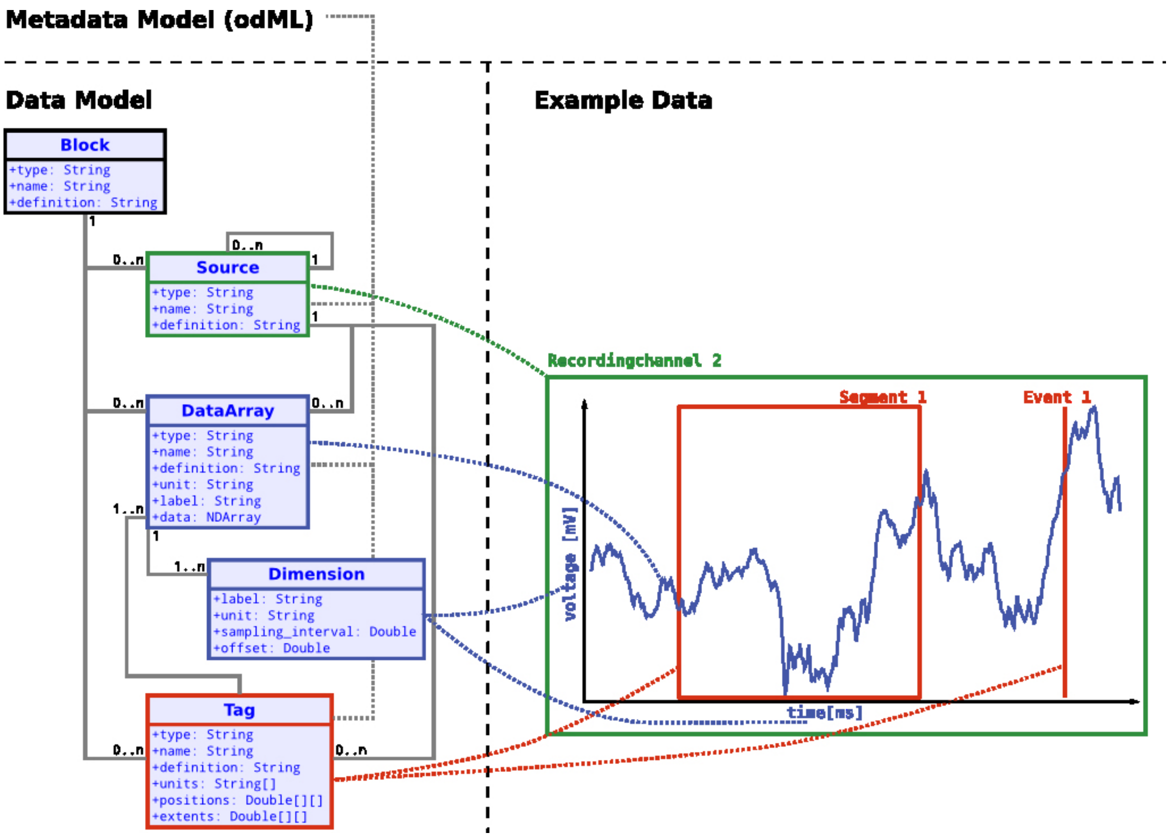


Figure 3.3 Description of an analogue signal (e.g. EEG) in NIX (https://github.com/G-Node/nix/wiki/The-Model)

Neurodata Without Borders data format

The increasing popularity of HDF5 is reflected in the Neurodata Without Borders (NWB) project, which established a new data format based just on HDF5. The NWB data format defines detailed data model which is much stricter than NIX [68]. NWB as a standalone format was proposed to avoid an additional mapping layer between NWB strict models and current solutions like NIX.

There are seven main NWB characteristics [68]:

1. Data is represented by time series classes; metadata is stored as their subclasses
2. Time series and processed data are stored with labels (HDF5 attributes) that identify its structure and content
3. Files are organized/separated by different kinds of data (recorded data, analysis results, stimuli, etc.)

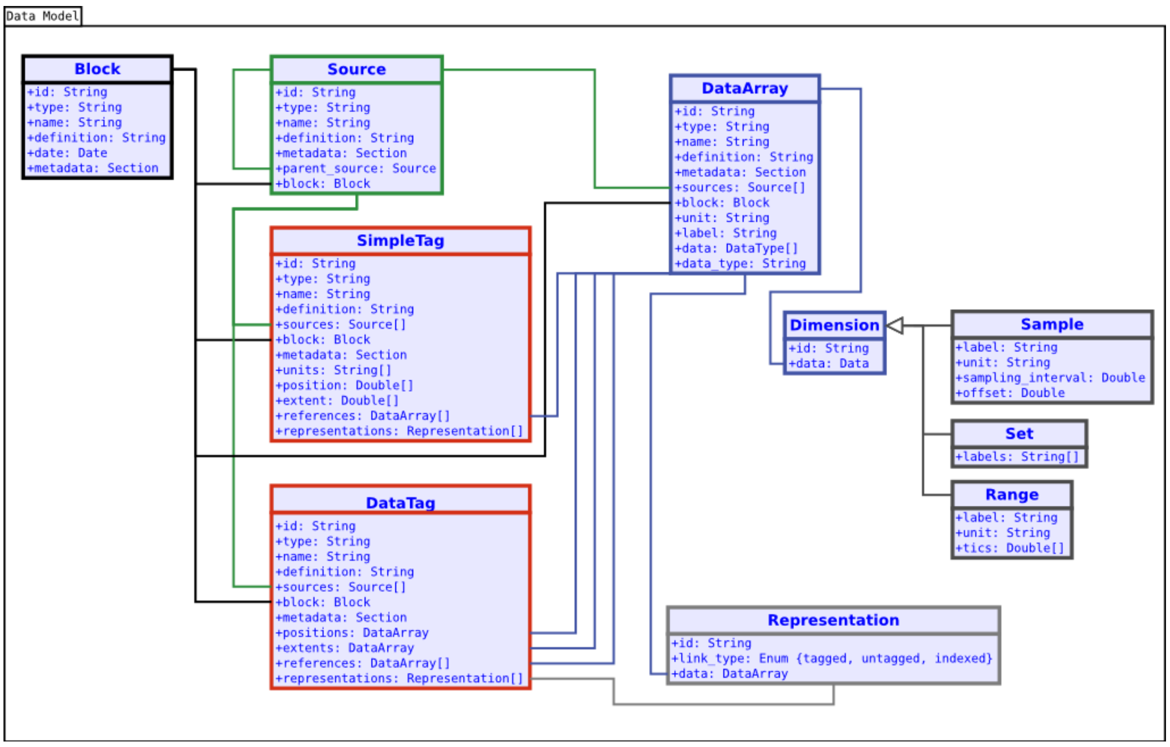


Figure 3.4 Definition of the NIX data format (https://github.com/G-Node/nix/wiki/Model-Definition)

4. Information about intervals are linked directly to the time series
5. Compatibility with HDFView is ensured
6. Format features expressed in the specification language are human and machine readable
7. Additional extensibility is supported via specification language

Since NWB is still in its early phase of development, it will not be further considered.

EEG BrainVision data format

The EEG BrainVision data format [7] developed by Brain Products GmbH[1] organizes raw data and metadata from recordings into three files: (i) a textual *INI*[2]*-like* header file containing descriptive metadata, (ii) a textual *INI-like* marker file containing event timestamps, and (iii)

[1]http://www.brainproducts.com/

[2]Configuration/Initialization files with *.INI* extension known mainly from Microsoft platforms (MS DOS, Windows)

a binary representation of raw brain waveforms. The header file describes technical metadata, e.g., number of channels, channel resolution, data format, information about binary/ASCII data representation, or sampling interval. Metadata related to a subject or experiment itself is not included. A mind-map of this structure is shown in Figure 3.5.

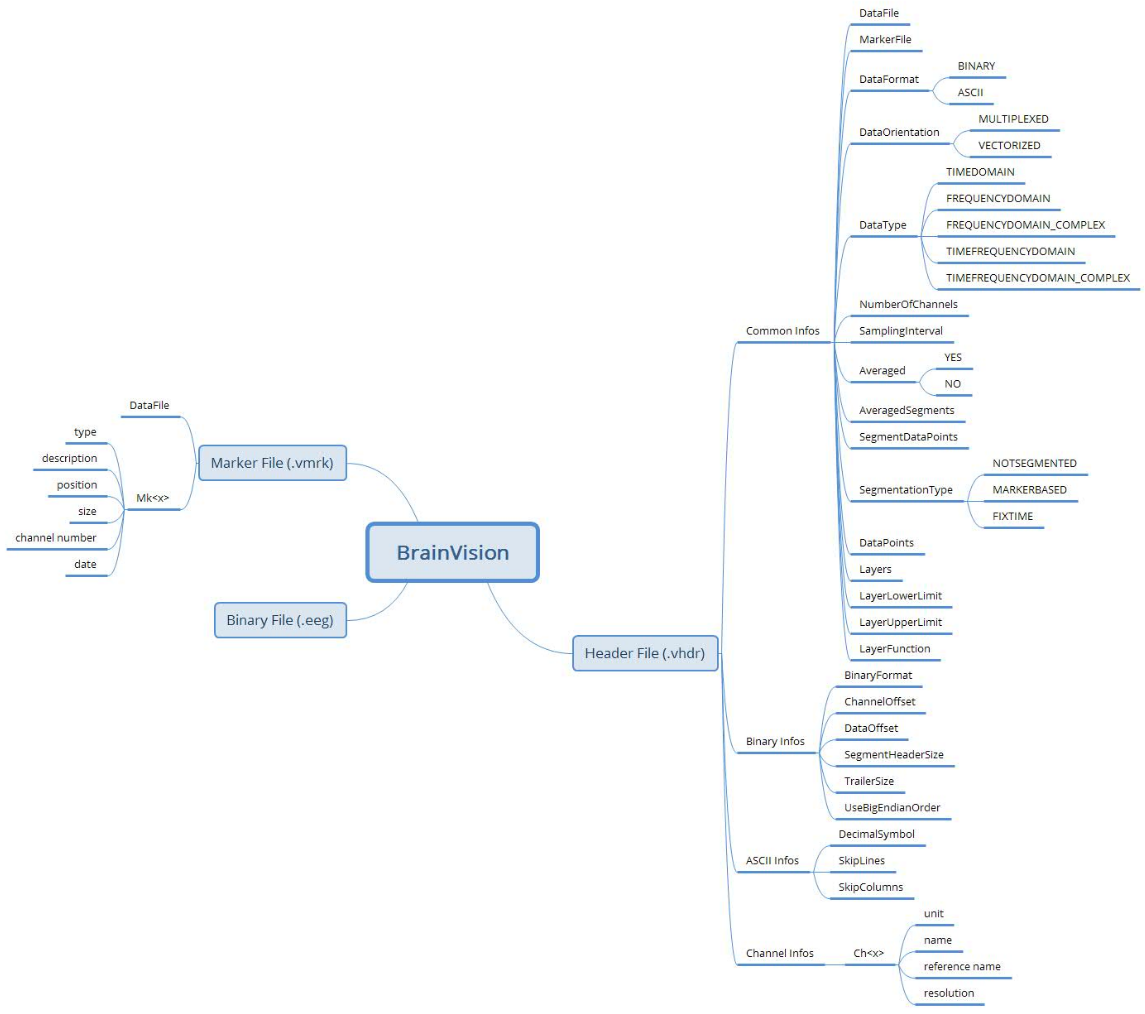


Figure 3.5 Structure of the BrainVision EEG data format

European Data Format

European Data Format (EDF) and its extension EDF+ is another standardization effort and data format for EEG, sleep recording, ECG, ElectroMyoGraphy (EMG) and evoked potentials. EDF+ can save annotations and analysis results [35], [36]. Just as the BrainVision format, EDF+ is strictly focused on the recording itself. EDF+ organizes recording content into

Header records and *Data records* sections. The *Header records* section includes both file and recording metadata, e.g. the data format version, patient's ID, start time of the recording or number of channels in the recording. The *Data records* section contains consecutive data for all channels. A mind-map of this structure is shown in Figure 3.6.

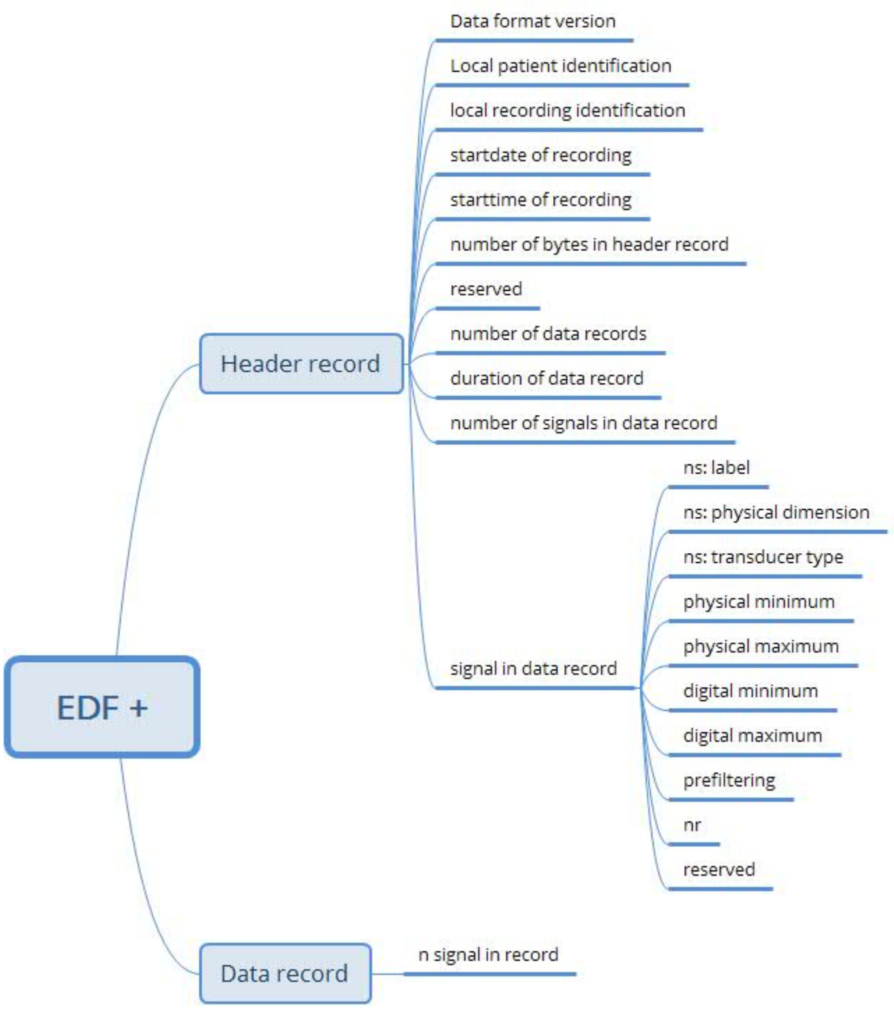


Figure 3.6 Structure of EDF+ data format

3.3.3 Ontologies and terminologies

odML terminology for electrophysiology

As the odML format is proposed to be generic and domain independent, it can serve also for a terminology specification storage. The authors provide a terminology for describing experimental electrophysiology domain[3]. As the domain of experimental EEG measurements is composed of both measurement and analysis, the terminology covers both parts. EEGBase (section 3.3.4) implements this terminology, and moreover it contributed to the terminology extension [69]. As a standalone terminology, the odML terminology for electrophysiology suffers with the absence of term referencing/dereferencing ability.

Neural ElectroMagnetic Ontologies

Neural ElectroMagnetic Ontologies[4] (NEMO) project, founded by National Institutes of Health in Oregon, is fully focused on EEG and MagnetoEncephaloGraphy (MEG) domains. The NEMO project deals with raw EEG data and data analysis provision, data storage tools, and ontology describing the data. Moreover, NEMO provides an analysis toolkit for ERP. However, the NEMO ontology resource, like many other existing ontological resources, suffers with lack of terms for accurately annotating the electrophysiological data [40]. Moreover, NEMO is no longer a sustained project.

NIF Standardized Ontology and NeuroLex

NeuroLex project[5], supported by NIF (section 3.3.4), is a lexicon of neuroscience terms, which aims to provide a unified terminology. NeuroLex draws on the NIF Standardized Ontology (NIFSTD), which includes a set of modular ontologies providing a comprehensive collection of terminologies (approx. 60000 concepts) to describe neuroscience data and resources.

The Ontology for Biomedical Investigations

The Ontology for Biomedical Investigations (OBI)[6] is an integrated ontology for the description of life-science and clinical investigations.

Systematized Nomenclature of Medicine - Clinical Terms

Systematized Nomenclature of Medicine - Clinical Terms (SNOMED-CT)[7] [65] is one of the most comprehensive existing multilingual clinical healthcare terminology providing scientifically validated clinical content. SNOMED-CT is organized in systematic collections of terms,

[3]https://github.com/G-Node/odml-terminologies/tree/master/v1.0
[4]http://nemo.nic.uoregon.edu/wiki/
[5]http://neurolex.org/
[6]http://obi-ontology.org/
[7]http://www.snomed.org/snomed-ct

where the terms are labelled by unique codes, definitions, synonyms, etc. SNOMED-CT is implemented in the national health care service systems in significant number of countries including the UK, USA, Australia, Canada, and Sweden. SNOMED-CT describes clinical content of electronic health records (EHR) and covers clinical findings, symptoms, diagnoses, procedures, body structures, organisms and other etiologies, substances, pharmaceuticals, devices and specimens. Therefore, its ability to describe, e.g., experimental laboratory device set-ups or protocols, is limited due to lack of terms.

Ontology for Experimental Neurophysiology

Ontology for Experimental Neurophysiology [40] (OEN) provides a formal explicit representation of experimental neurophysiological data/metadata. The OEN development was initialized under the INCF Program on Standards for Data Sharing (section 3.2) as the answer to insufficient ontological resources in the domain. The OEN is purposed with a view to provide a controlled vocabulary to standardize descriptions of domain resources. OEN development is divided into two branches: (i) neurological concepts and (ii) devices and methods. The development of a terminology describing neurophysiological concepts (e.g. action potential, synaptic plasticity) is difficult as these concepts transverse multiple ontology branches. Within the project it was informed by a strategy using web-based surveys and detailed literature analyses. The development of the *devices and methods* branch firstly aims for describing specific lab devices and methods enriching odML terminology descriptions and drawing on the EEGbase data (Section 3.3.4). Determined terms from various resources (EEGBase, odML terminology for electrophysiology, Neurolex, etc.) are mapped with existing ontology resources (e.g. NEMO, OBI) and predefined terms are incorporated in the OEN using the Minimum Information to Reference an External Ontology Term (MIREOT) principle [15].

OEN has a potential to become a referential terminology for other data format projects including this one, however, the project has been suspended in its early development stage when some of the involved researches left the group, and its future is uncertain. For that reason, a referential terminology for this work is odML terminology for electrophysiology as a substitution for OEN.

3.3.4 Databases

The most common way to store neuroinformatics data and metadata is via dedicated databases. Their functionality is usually beyond a simple provision of persistence storage. The neuroinformatics databases, which aim is data sharing, should be able to share data or provide an interface for processing tools. Data format plays a lead role in case of data sharing since the proprietary and closed third-party solutions could force valuable datasets to be excluded on a inter-system level. Regardless of type of data, requirement of specific software for reading the data builds a barrier between the shared data and scientists whom the data are intended for. Efficiency of data sharing is closely related also to data structures and descriptions used. Separately readable experiment metadata is beneficial for data search portals like Neuroscience Information Framework (NIF).

EEGBase

EEGBase [33], the portal for experimental data management and sharing, has been proposed and developed within the software and hardware infrastructure [50] of our laboratory. EEG-Base strongly emphasises an effective separation of data and metadata and their storing in commonly-known well-described formats.

The EEGBase data is stored in the BrainVision EEG format. The EEGBase metadata related to the experiment is stored in the odML structure and can be easily serialized into the XML format. In addition to the stored data and metadata, EEGBase also contains experimental scenarios describing the recording work-flow and used stimulation. The EEGBase scenarios are most often designed in the Presentation® software tool (developed by Neurobehavioral Systems, Inc.[8]. However, the scenarios we diverse, ranging from a flashing light to a pre-scripted computer game level and it is hard to predict a specific format the scenario is stored in. Therefore, EEGBase handles scenarios as multimedia attachments of the particular experiment. An example of the EEGBase experiment (from the project investigating developmental coordination disorders in children) structure is shown in Figure 3.7. From the perspective of storage technologies, EEGBase uses combination of relational database (for data) and noSQL database (for metadata).

A list of other portal solutions follows, but for their insufficient features and/or support, EEGBase as a live project with contributing community serves as the main resource for this work.

CARMEN Portal

CARMEN (Code Analysis, Repository & Modelling for e-Neuroscience) project of the UK Neuroinformatics Node represents a concept of a virtual laboratory intended for experiments in the domain of neuroscience. Its main idea is to accompany the experimenters for whole research process, i.e. from recording to analysis. Apart from the storage for experimental

[8]http://www.neurobs.com/

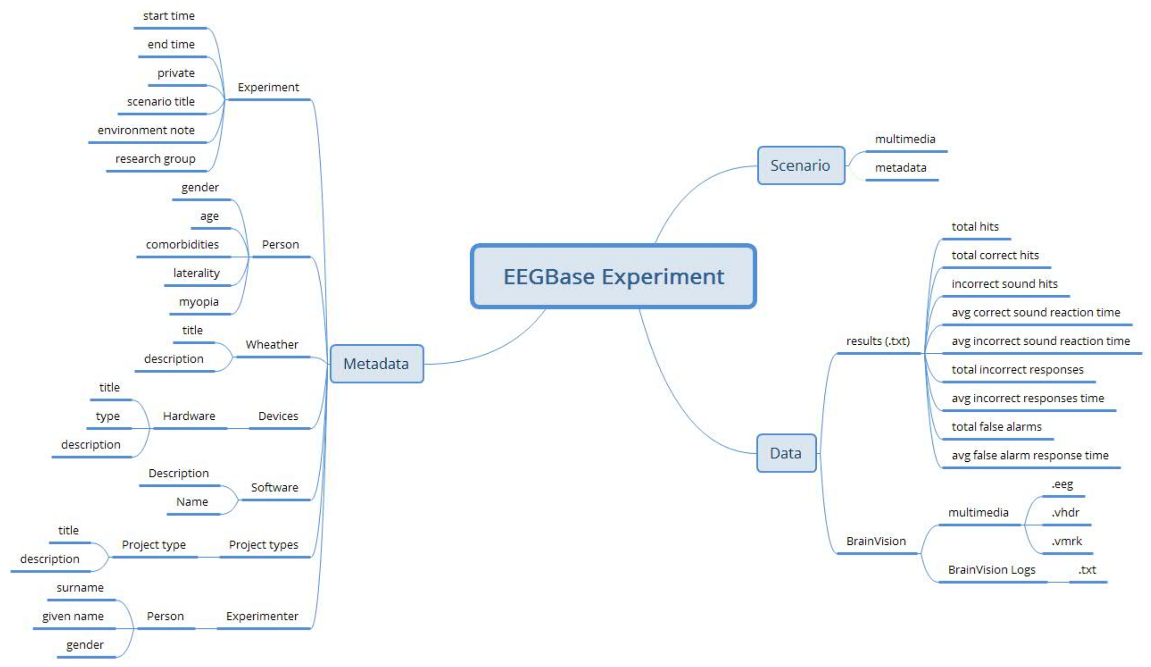


Figure 3.7 An example of the EEGBase experiment structure consisting of the data, metadata and scenario sections. This example also illustrates that even in such a relatively simple structure there is no clearly defined border between data and all metadata.

data, CARMEN provides a persistent storage of services. Thus, collaborators can upload their routines as web services, share them and execute them on the server side. CARMEN structure is shown in Figure 3.8.

However, as promising CARMEN project appears, the latest update in version 2 is dated to October 2014, and even though there is no official statement, the latest update announced the termination of the project.

G-Node data management and data sharing platform

G-Node data management and data sharing platform is a portal developed by the German Neuroinformatics Node [27]. The platform provides basic storage and sharing abilities and access to an international neuroinformatics discussion forum. Similar to CARMEN project, the lack of support and updates as well as a very small community participating to data sharing might announce the upcoming end of project effectiveness.

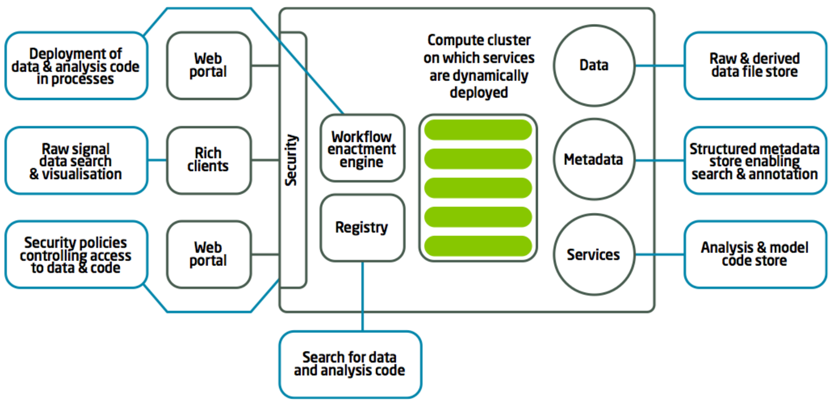

Deployment of data & analysis code in processes
Web portal
Raw signal data search & visualisation
Rich clients
Security policies controlling access to data & code
Web portal
Security
Workflow enactment engine
Registry
Compute cluster on which services are dynamically deployed
Data
Metadata
Services
Raw & derived data file store
Structured metadata store enabling search & annotation
Analysis & model code store
Search for data and analysis code

Figure 3.8 A structure of the CARMEN portal [70]

J-Node

A software infrastructure of Japanese INCF Node (J-Node) provides a set of specialized platforms.

- Visiome Platform
- Brain Machine Interface Platform
- Invertebrate Brain Platform
- Neuro-Imaging Platform
- Dynamic Brain Platform
- Cerebellar Platform
- Brain Transcriptome Database
- Comprehensive Brain Science Network Platform
- Mouse Phenotype Database
- Brain Science Dictionary
- ViBrism Database
- Kanphos Database

- Simulation Platform
- Riken BSI Research Database Portal

Each platform provides a dedicated solution for particular research domain including a storage for experiments, methods, articles, etc. However, as the J-Node platforms are heterogeneous with absence of any common data standard, these will not be considered further.

Neuroscience Information Framework

The neuroscience Information Framework (NIF) [45] established under the National Institutes of Health (NIH) Blueprint for Neuroscience Research, is a dynamic inventory of the resources in neuroscience. The NIF provides a search engine working over the registered web-based resources: data, materials, and tools. The searching abilities of the engine depends on the level of resource registration into NIF Resource registry. There are three registration levels [11]:

1. Only resource URL is registered by NIF; search engine is able to find the resource but cannot access any dynamic content.
2. The resource provides a web-service allowing NIF to query the resource metadata automatically [32].
3. Final step allows NIF to access resource data directly; data are mapped on the NIFSTD and resource provides Content-Based Query Interface; NIF creates a virtual federated database from registered resources.

NIFSTD is proposed to describe neuroscience resources not data. Therefore, NIF is not considered further.

3.3.5 Data format features

Currently, there is no set of data format desired features, i.e. desiderata, specified. The only feature that is common between current formats is machine readability. Some of the formats also provide a mechanism to annotate data by controlled terminologies/ontologies (e.g. NIX). To specify a comprehensive desiderata for EEG/ERP data format, the work of Mo and colleagues [47] was used. The authors reviewed a set of features in the domain of computable *electronic health record*-driven phenotypes and proposed a subset of those that are most important [47]:

1. *Recommendations for clinical data representation to support phenotype*

   (a) *Structure clinical data into queryable forms*
   (b) *Recommend a common data model, but also support customization for the variability and availability of electronic health records data among sites*

2. *Recommendations for phenotype representation models*
   (a) *Support both human-readable and computable representations*
   (b) *Implement set operations and relational algebra*
   (c) *Represent phenotype criteria with structured rules*
   (d) *Support defining temporal relations between events*
   (e) *Use standardized terminologies, ontologies, and facilitate reuse of value sets*
   (f) *Define representations for text searching and natural language processing*
   (g) *Provide interfaces for external software algorithms*
   (h) *Maintain backward compatibility*

As the EEG/ERP data have health data qualities, the specified desiderata have a potential to be useful also in this domain; each particular desideratum was evaluated as relevant/irrelevant for the EEG/ERP data format (Table 3.1).

Both desiderata from the first category, originally proposed for clinical data and electronic health records data, can be applied on experimental data and experiment metadata instead. These desiderata are related with the content of chapters 4 and 5.

Relevant desiderata from the second category (considered as *Recommendations for EEG experiment representation models*), i.e. (d), (e), (g), and (h), are related with the content of chapters 6 and 7.

How the proposed solution meets the desiderata is discussed in chapter 8.

Table 3.1 Relevance of computable phenotypes desiderata for a purpose of EEG/ERP data format

Desideratum	**Relevance**	**Comment**
Structured data in queryable form	Relevant	Relevant also for data with experimental provenance (e.g. EEG/ERP)
Common and flexible data model	Relevant	Relevant also for data with experimental provenance (e.g. EEG/ERP)
Human-readable and computable representation	Relevant	Relevant for EEG experiment metadata
Set operations and relational algebra	Irrelevant	Computable representations of data processing algorithms, where set operations and relational algebra are useful and which would be a part of data description, are not in the scope of this work.
Structured rules	Irrelevant	Computable representations of data processing algorithms, where structured rules are useful and which would be a part of data description, are not in the scope of this work.
Temporal relations	Partially relevant	Date/time of data acquisition has to be specified in data format. However, temporal relations could be handled by superior systems/processing algorithms, e.g., in order to construct time-series from multiple experiments.
Standardized nomenclature	Relevant	Relevant for interoperability of shared data.
Text searching & NLP	Irrelevant	NLP is not in the scope of this work, thus it will not be discussed further.
External interfacing	Relevant	Data format has to be accessible computationally regardless the domain.
Backward compatibility	Relevant	Historical data should be processable regardless the domain.

Part III

Data Modelling

Chapter 4

Data models and storages

An aim of the thesis is to present an innovative approach to data/metadata description in the EEG/ERP domain. Existing data modelling concepts and logics (relational algebra, object-oriented concept, description logic) are described in this chapter.

4.1 (Entity-)Relational model

A relational model consists of:

- a collection of time-varying tabular relations,
- relationships ensuring the entity and reference integrity,
- relational algebra.

4.1.1 Elements

Relational model (R model) is composed of the following elements [14]:

Domain *A domain is a set of values of similar type. A domain is "simple" if all of its values are atomic.*

Relation *Let D_1, D_2,..., D_n, be n (n > 0) domains (not necessarily distinct). The Cartesian product x{D_i: i = 1,2, ...,n} is the set of all n-tuples <t_1, t_2,...,t_n> such that $t_i \in D_i$ for all i. A relation R is defined on these n domains if it is a subset of this Cartesian product. Such a relation is said to be of degree n.*

Attribute *For each tuple component we associate not only its domain, but also its distinct index. This we call an attribute. The n distinct attributes of a relation of degree n distinguish the n different uses of the domains upon which that relation is defined. A tuple then becomes a set of pairs (A:v), where a is an attribute and v is a value drawn from the domain of A, instead of a sequence (v_1, v_2, ...,v_n).*

A relation then consists of a set of tuples, each tuple having the same set of attributes.

4.1.2 Relationships between relations

If a collection of time-varying tabular relations meets following criteria:

- all domains are simple
- no two tuples are duplicated
- row order is irrelevant
- column order is irrelevant
- attributes are atomic values

than a collection of data represented by these relations is called **relational database**.

The relationships between the relations are handled by an apparatus of **Primary** and **Candidate keys**. *K is a candidate key of relation R if it is a collection of attributes of R with the following time- independent properties:*

1. No two rows of R have the same K-component.
2. If any attribute is dropped from K, the uniqueness property (1) is lost.

One candidate key is selected as the primary key for each relation. A set of all primary keys is called a primary domain. No primary key is allowed to be null, i.e. entity integrity.

Suppose an attribute a of a compound (multi-attribute) primary key of a relation R is defined on a primary domain D. Then, at all times, for each value v of a in R there must exist a base relation (say S) with a simple primary key (say B) such that v occurs as a value of B in S, i.e. reference integrity.

Finally, a list of unique attribute names is called a **schema** for relation R with degree n.

The ER model describes a concept in a tabular manner, where the table is a relation, the column is an attribute and the row is a tuple representing data. A relation is defined as a Cartesian product of domains. Entity-Relational Model (ER model) [12] describes the real-world concepts by entities and a relationship between them as their Cartesian product.

When the entities are expressed by R model, we can say, that ER model extends semantic abilities of R model. ER model also allows us to define cardinalities over the relationships.

4.1.3 Relational algebra

Relational algebra [14] extends semantics of ER models and defines queries on them. Basic set of relational algebra's operators includes:

Set operators Relational algebra supports set operators of union, intersection and difference. As the operators are binary operators applicable over two relations, both relations have to be of the same domain. The outcome of the operation is a new relation of the same domain.

Selection; THETA-SELECT *Theta-select* operator is a binary operator applicable above relations and attributes: $<$, $\leq$, $=$,$\geq$, $>$, $\neq$. If binary operator is "=", then the *theta-select* operator is called SELECT.

Projection *Projection* is a relation created as a subset of the original projection where all redundant rows are dropped.

Join; THETA-JOIN *Theta-join* is a union of attributes of two relations into single one. Theta-join is conditioned by a binary operator above two attributes (from distinct relations). If the binary operator is "=", *theta-join* is called EQUI-JOIN. If the redundant rows are dropped from final EQUI-JOIN, it is known as a NATURAL-JOIN (e.g. JOIN based on foreign keys in relational databases).

Division *Given relations R(A, B1) and S(B2) with B1 and B2 defined on the same domain(s), then, R[B1 + B2]S is the maximal subset of R[A] such that its Cartesian product with S[B2] is included in R. This operator is the algebraic counter-part of the universal quantifier.*

4.1.4 ER model construction and graphical representation

The ER model has strictly specified element properties and abilities but there is no formal specification of the model representation. However, a set of conventions exists.

Running example: One laboratory with name is involved in n projects. Each project with name and identifier (ID) engages in at least one experiment. Each experiment with name and ID consists of n recordings. One experiment can be realized only in one project. One recording with a date and ID could serve to more than one experiment. Experiments and projects have supervisors with names and surnames.

We can consider subjects as entities, predicates as relationships and objects as attributes. Pronouns and numerals specify cardinalities. Such classification creates a basic skeleton of the model from the natural language.

A list of the entities and their attributes follows:

- Laboratory (name, supervisor, projects)
- Project (project_id, name, supervisor, experiments)
- Experiment (experiment_id, name, supervisor, recordings)
- Recording (recording_id, date)
- Supervisor (name, surname)

The attributes *project*, *experiment*, and *recording*, are sets of entities related to different relation and together with the attribute *supervisor* breach the condition of attribute atomicity. To prevent this, the attribute *supervisor* is substituted by atomic supervisor's ID; relational algebra operator NATURAL-JOIN allows us to keep the relationship persistent and to fix the atomicity in case of un-atomic attributes.

- Laboratory (**name, supervisor_id**, projects)
- Project (**project_id**, name, supervisor_id, experiments)
- Experiment (**experiment_id**, name, supervisor_id, recordings)
- Recording (**recording_id**, date)
- Supervisor (**supervisor_id**, name, surname)

Attributes in bold represent candidate keys, which are identical to selected primary keys. The primary key for the *Laboratory* relation is composed of two attributes. For simplicity of next examples, explicit identifier *laboratory_id* will be added to the *Laboratory* relation and set as the primary key. The real-world concept in the running example is limited to one laboratory. This limitation influences the uniqueness of the identifiers. These are unique for the concept only and as the concept reflects close-world assumption, it cannot be guaranteed there is no other experiment with the same identifier within different laboratory.

Relationships and cardinalities are specified in [subject-predicate-object] triples as:

- ONE Laboratory is involved in MANY Projects
- ONE Project involves MANY Laboratories
- ONE Project engages MANY Experiments
- ONE Experiment is engaged by ONE Project
- ONE Experiment consists of MANY Recordings
- ONE Recording belongs to MANY Experiments
- ONE Projects contains MANY Recordings
- ONE Supervisor has MANY Projects
- ONE Project has ONE Supervisor
- ONE Experiment has ONE Supervisor
- ONE Supervisor has MANY Experiments

When redundant and transitive rules are merged/dropped, following schematic list of relationships are left (an exclamation mark notes that the attribute is mandatory):

- Laboratory m! . . . . n Project
- Project 1! . . . . n Experiment
- Experiment m! . . . . n Recording
- Experiment n. . . . .1! Supervisor
- Project n. . . . .1! Supervisor

A graphical representation of a common ER model (derived from the set of relationships and relations) with stereotype notation; notation for entities/relations, attributes, and relationships is shown in Figure 4.1.

An overview of the running example is shown in Figure 4.2. To meet the criteria of attribute atomicity, m:n relationships need to be decomposed into decomposition tables that provide mapping of keys in 1:n cardinality.

The created model is specific in description of its entities. However, some attributes are common for multiple entities, e.g., *supervisor*, *director*. Both could be generalised as a *person* since they represent a human being with name and surname. Additional flag can

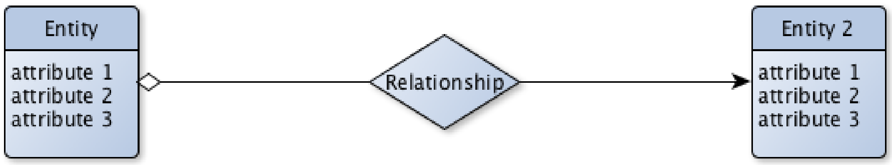


Figure 4.1 Entity, attribute and relationship graphical stereotype; the empty diamond represents optional occurrence of the related foreign key, filled arrow represents mandatory occurrence of exactly one related foreign key.

explicitly annotate particular person type, e.g., value=1 means *director*; value=2 means *supervisor* etc. For an illustrative purpose, *Laboratory* could be generalized as a *Research Centre* which includes laboratories, faculties, universities, private sector research centres etc.

- Research centre (**research_centre_id**, name, director_id, flag)
- Person (**person_id**, name, surname, flag)

Adequate relationship to this generalization is

- Research centre m!. . . . . n! Person

A loss of expressiveness is obvious. We cannot restrict that research centre has just one director; faculty has n! students, laboratories has n stuff etc. A more suitable solution would adapt an object orient data model.

4.1.5 Structured Query Language

Structured Query Language (SQL) is a standard language for managing, i.e. storing, manipulating, and retrieving data in relational databases. SQL draws on the expressiveness of the ER model, relational algebra operators included. Thus, SQL could also support set, aggregative and comparative operators over queried data. SQL statements can *SELECT*, *INSERT*, *DELETE* and *UPDATE* data according to restrictions specified in *WHERE* clause. Particular RDB implementations support various SQL extensions including procedural extensions, but those are not related to the expressiveness of ER models.

4.2 Object-oriented model

Semantics of the Object-Oriented (OO) concept is not formally specified, however, it is designed to handle some of limitations of the ER concept. OO constructs missing in relational

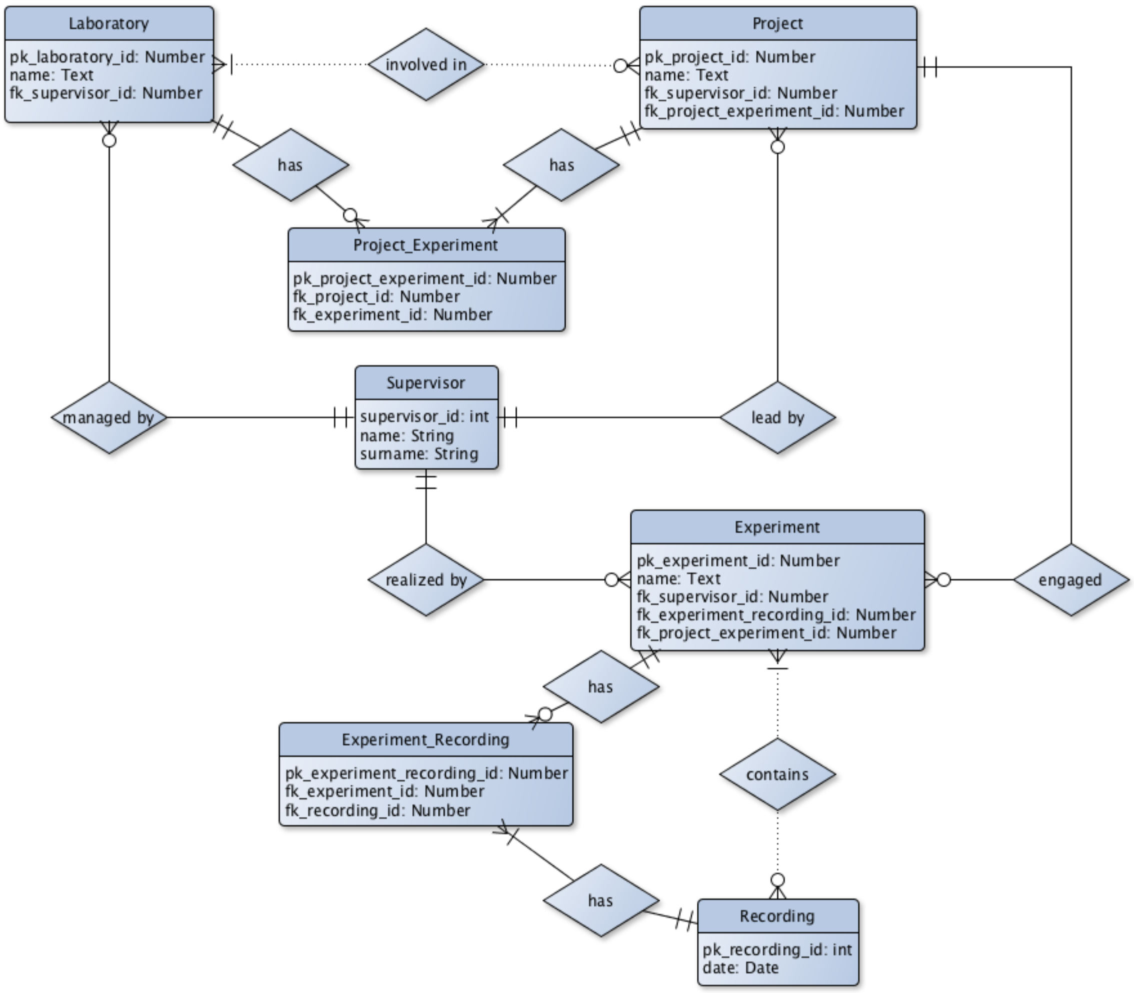


Figure 4.2 An example of the ER diagram; m:n relationships were decomposed into decomposition tables; dashed relationships representing original m:n relationships are visualized for the illustrative purposes only.

algebra (e.g. inheritance) significantly increase the expressive power of the model. There is no formal definition of the OO concept nor consensus of its constituent, however, following core features [38] are included in most of the OO model implementations (programming languages, databases, etc.).

4.2.1 The core

Object The object is an abstract representation of a real-world entity. Any real world bordered domain can be expressed as a finite set of mutually connected objects. No two completely equal objects can exist within the same domain as no two rows in same relation can be identical. Even though the object characteristics are identical, the object identifier (OID), a mandatory identifier for each object, is unique and thus each object in a specific domain is unique as well. OID construction approaches can be divided into two groups:

- Logical OIDs
  - Instance identifier and class identifier: OID contains information of the class; valid until object reclassification
  - Instance identifier only: class membership is not implicitly specified
- Physical OIDs
  - OID contains information depending on physical storage (e.g. memory address): object is persistent, but it is not possible to migrate it to the other physical storage

A nested collection of objects representing a real-world entity is a complex object, e.g., an experiment containing a collection of objects representing recordings. The collection implements one of the following orthogonal constructors (each constructor can be used for any object) [62]:

- Set; unordered complex object
- List; ordered complex object
- Tuple; complex object with predefined structure

Attributes Attributes define characteristics/state of the object. A state of the object in time t is defined by values of the attributes. The attribute can be an object or a value/literal.

Methods The behaviour of objects is defined by methods. Methods provide read/write access to attributes and thus those change the state of the object. Methods are encapsulated in the object and they are evoked from outside.

Class A class defines attributes, methods and an initial state of the object.

Inheritance Inheritance is an ability of the class (superclass) to derive another class (subclass) that has the same attribute and method sets and can be extended further by additional attributes and methods. The inheritance is:

- single: each class is a descendant of exactly one superclass; from structural perspective, single inheritance enables building of tree-like structures

- multiple: each class is a descendant of one or more superclasses; from structural perspective, multiple inheritance enables building of rooted graph-like structures (a class lattice [38])

An inheritance process from the root to the leaves is called specialization, and from the leaves to the root, generalization.

4.2.2 OO model example

Inheritance and complex objects ensure structural abilities similar to those of relationships between relations in ER models. For the running example, the following class candidates were determined:

- Class Laboratory (name, director, list of projects)
- Class Project (id, name, supervisor, list of experiments)
- Class Experiment (id, name, supervisor, list of recordings)
- Class Recording (id, date, name)
- Class Supervisor (id, name, surname)

Implementation of generalization/inheritance concept into the class candidates leads to creation of new generalized classes:

- Class Research centre (name, director)
- Class Laboratory extends Research centre (list of projects)
- Class Project (id, name, supervisor, list of experiments)
- Class Experiment (id, name, supervisor, list of recordings)
- Class Recording (id, date, name)
- Class Person (name, surname)
- Class Supervisor extends Person (id)
- Class Director extends Person (id)

4.2.3 Unified Modelling Language

Unified Modelling Language (UML) was developed as a response to the increasing popularity of OO programming languages and complex development methodologies. The first UML version was developed in 1995 as a combination of OMT[1] [64], and OOSE[2] [31]. In 1997, the Object Management Group (OMG)[3] accepted UML 1.1 as a standard. The current version, UML 2.0, was released in 2009. Figure 4.3 shows the evolution of UML. Only UML 2.0 version is considered hereafter.

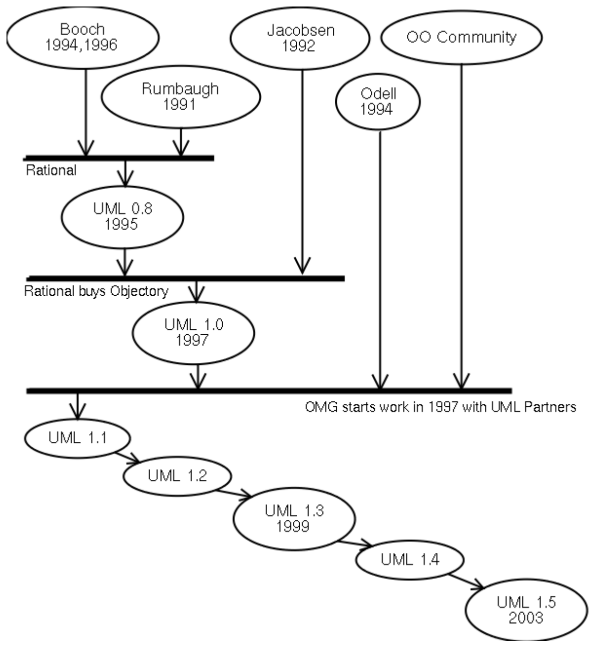


Figure 4.3 Evolution of the UML (http://vinci.org/rlv/d/uml/history.html)

UML is a modelling language for graphical representation of all aspects of the software development process including designing of software structure and behaviour. UML contains 14 basic diagrams which, together, create a complex software model independent on its further implementation. UML is primarily focused on the OO concept, however, also ER models can be represented via UML. Generation of the specific source code from the UML description can be partially automated by tools like Eclipse IDE *UML to Java Generator* plug-in.

The diagrams can be divided into the following groups [63]:

- Structure diagrams

[1]Object Modelling Technique
[2]Object-Oriented Software Engineering
[3]http://www.omg.org/

  - Class diagram
    - defines static structure of the system
    - defines data model, specific classes with all attributes, methods, relationships, inheritance and encapsulation
  - Component diagram
    - defines division of the system into smaller parts with the delimited functions
    - describes how the classes are divided into components
  - Composite structure diagram
    - defines internal structure of classes and how the methods and functions cooperate together
  - Deployment diagram
    - defines how the components will be physically distributed in an information technology (IT) environment
  - Object diagram
    - defines state of the objects in a specified system state/time
  - Package diagram
    - defines how the system is divided into packages
  - Profile diagram
    - defines stereotypes, tagged values and constraints for platforms/domains

- Behaviour diagrams
  - Activity diagram
    - defines process activities
  - State machine diagram
    - defines a progress of process activities
  - Use case diagram
    - defines how the system is supposed to be used, who the actors are, and what role the actors have
  - Communication diagram
    - defines how objects communicate together during specified activity

- Sequence diagram
    - defines how the system works in various activities (what is happening inside the system between classes or components)
- Timing diagram
    - defines system behaviour on the time line

The class diagram will be further discussed as the only representative diagram that describes a data model.

Apart from the diagrams, UML also specifies exchange formats (e.g. CORBA IDL[4], XMI[5]) and a language for expression restrictions, which cannot be represented graphically (OCL[6]).

UML describes itself as a UML model as well (Figure 4.4).

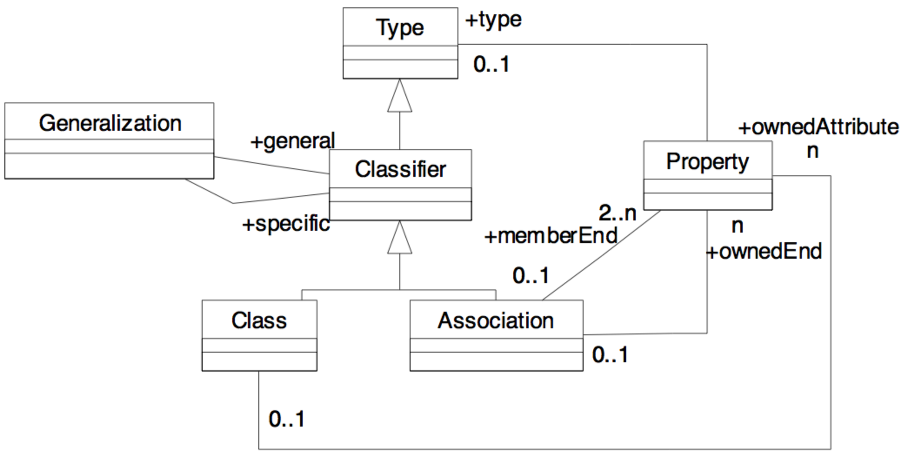


Figure 4.4 UML Kernel (http://www.omg.org/spec/ODM/1.0/PDF/)

4.2.4 UML Class diagram construction

The following section evaluates the class diagrams for the purpose of data modelling, therefore, some constructs are not considered (e.g. encapsulation).

UML supports the following graphical syntax for core features:

Class Class is represented by rectangles labelled by a class name at the top.

[4]Interface Definition Language
[5]XML Metadata Interchange
[6]Object Constraint Language

Attribute Attributes are listed in the middle part of a class rectangle. Initial symbol +/-/# defines encapsulation of the attribute (public, private and protected access restrictions) followed by an attribute name. A data type (i.e. primitive or complex object) of the attribute is specified after colon.

Method Methods described by the same graphical notation that of the attributes are listed in the last part of the class rectangle (Figure 4.5).

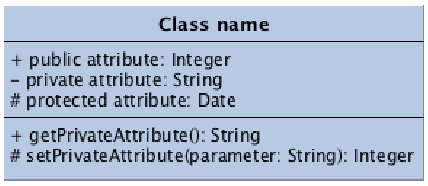

Class name
+ public attribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer

Figure 4.5 An example of a class represented by the UML Class Diagram

Inheritance / generalization *Generalization* is illustrated as an arrow *descendant* -> *parent* (Figure 4.6).

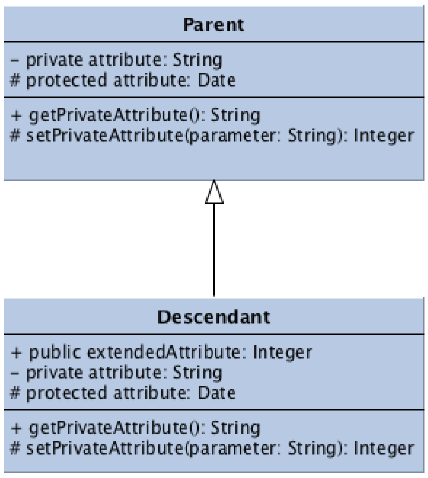

Parent
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer
Descendant
+ public extendedAttribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer

Figure 4.6 An example of class generalization in the UML Class Diagram

Association / aggregation / composition *Association* - a binary or ternary operator, representing an alternative to the relationships in the ER model; generally associated with two or three (ternary operator) classes (Figure 4.7).

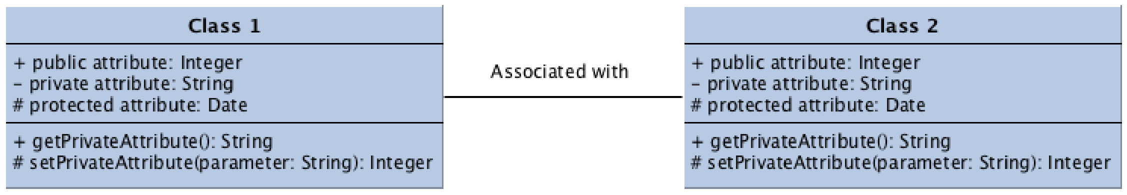

Class 1
+ public attribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer
Associated with
Class 2
+ public attribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer

Figure 4.7 An example of class association in the UML Class Diagram

Aggregation - a binary operator providing a semantically stronger relationship, where class A is a component of a class B, with class A is in the relationship. Multiplicity (cardinality) can be specified over the relationships (Figure 4.8). While ER model supports cardinality definition as 1:1, 1:n, and n:m relationships; UML supports multiplicity specifying concrete number of class instances allowed to be define (infinity is marked as a symbol *), e.g. 1 – 0..*; 1..5 – 1..*.

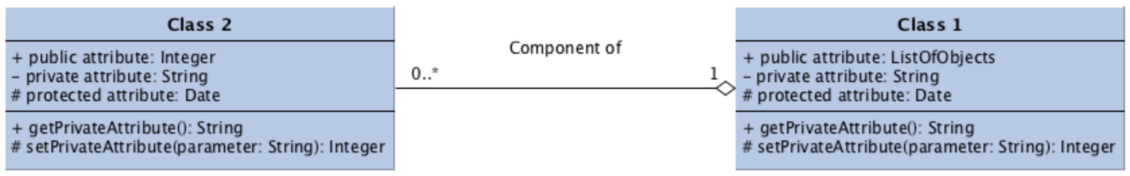

Class 2
+ public attribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer
Component of
0..*
1
Class 1
+ public attribute: ListOfObjects
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer

Figure 4.8 An example of class aggregation in the UML Class Diagram (empty diamond in the relationship)

Composition - a binary operator defining that an instance of a first class cannot exist without the existence of instance of a second class. *Composition* is stricter form of *aggregation*. Multiplicity can be specified over the relationships (Figure 4.9).

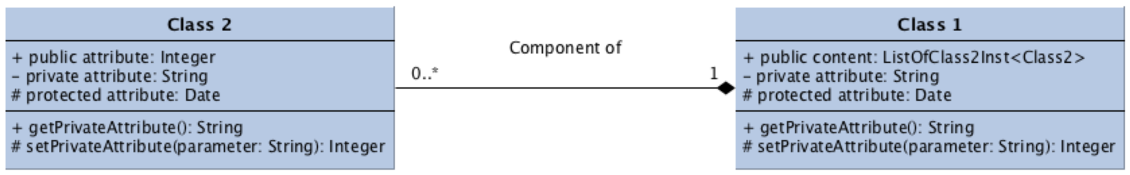

Class 2
+ public attribute: Integer
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer
Component of
0..*
1
Class 1
+ public content: ListOfClass2Inst<Class2>
- private attribute: String
# protected attribute: Date
+ getPrivateAttribute(): String
# setPrivateAttribute(parameter: String): Integer

Figure 4.9 An example of class composition in the UML Class Diagram (filled diamond in the relationship)

Other relationships supported by the UML Class diagram (e.g. dependency and realization) and advanced constructs (e.g. multiple inheritance/classification, stereotypes, parametrized classes) are out of the scope of this work. Our running example can be represented through the aforementioned elements and operators (Figure 4.10).

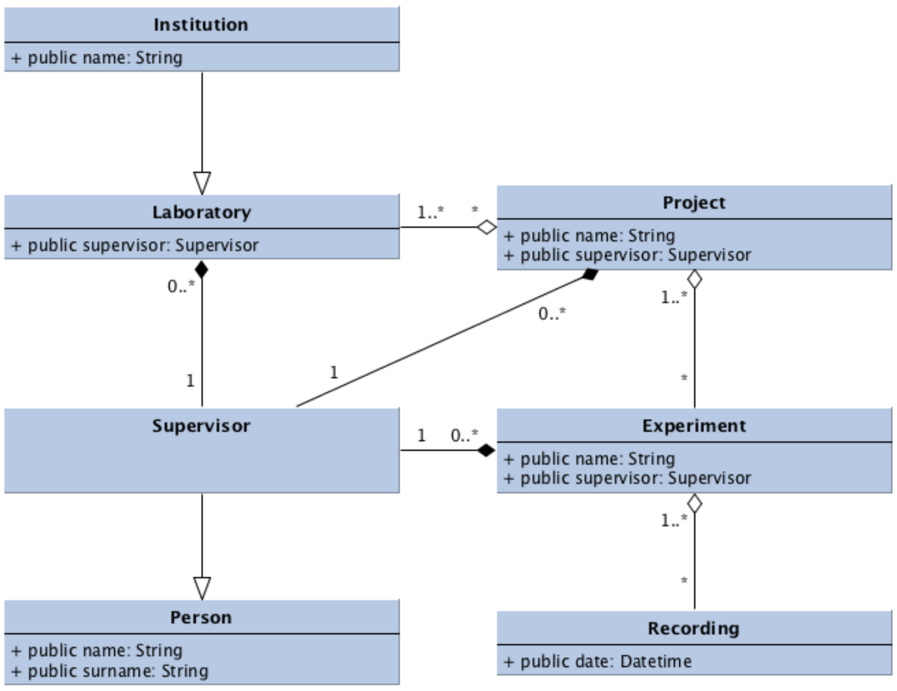


Figure 4.10 An example of the UML Class diagram

4.2.5 Object-relational databases

Object-Relational Databases (ORDB) are RDBs extended by additional OO features. Table attributes in the ORDB can be of complex object type, e.g., arrays. Not all OO model features, e.g., generalization, are supported in ORDBs.

4.2.6 Object-oriented database

Object-Oriented Databases (OODB) are based on OO core features. They provide inheritances, associations, aggregations etc. OODBs are not widely used and remain a more experimental concept.

4.3 eXtensible Markup Language

Extensible Mark-up Language (XML) is mainly proposed for a structured data exchange purpose. XML does not represent a modelling concept but a technological implementation. It is described here for its diverse expressive power.

XML is derived from the W3C's SGML (Standard Generalized Mark-up Language). Dependencies of the mark-up language family based on SGML are shown in Figure 4.11.

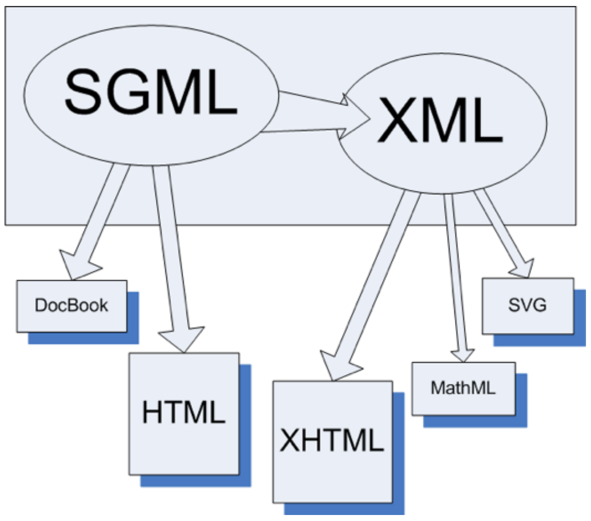


Figure 4.11 Family of SGML-based languages [26]

A template for a pre-defined XML structure can be specified by XML Schema Definition (XSD) or Document Type Definition (DTD). XML document can be validated accordingly to XSD or DTD afterwards. DTD is an older standard, which is no longer commonly used today and it is not discussed further. XSD supports pre-defined data types, multiplicity, inheritance, encapsulation, and substitution.

XML describes structured data, thus it can describe tabular structures and ER models. The abilities of XSD allow to describe also OO model. While XML describes instances, i.e. data, XSD describes classes.

Only XML in version 1.1 is considered further (a difference between versions 1.0 and 1.1 is in the character sets used [16].

There are four basic XSD elements:

Entity Entity is defined as a pair [declaration-specification]. Declaration is an *element* in XSD document referring to a specification, i.e. a *sequence* of attributes and/or elements.

Attributes XSD supports defining attributes in two different constructs: *attribute* and *element*.

Relationships Relationships are represented by nested elements.

Cardinality Cardinality is specified in an attribute section of each element as a minimal occurrence number (*minOccur*) and maximal occurrence number (*maxOccur*).

4.4 XML within relational databases

RDB, ORDB, and OODB store data and metadata in their inner representation. However, there can be various situations when keeping specific subset of data in its original format, e.g., XML-based experiment protocol in EEGBase, is convenient.

XMLType is an extended data type in Oracle 9i database and all later versions. It operates over a Character Larger Object (CLOB) type but it is adapted to XML files, i.e. it supports XML validation and direct querying XML data. XMLType exists in three modifications according to knowledge of an XML structure: structured, unstructured, and binary storage type. Analogously to the large objects types (CLOB, Binary Larger Object (BLOB)), there can be only one XMLType column per table.

4.4.1 Structured XMLType

Structured XMLType, sometimes called *Object-Relational*, is designated for frequently repeated XML documents of the same schema. XSD must be specified in advance prior to table creation [39]. Within this type, data in XML can be managed through the database system. Only one schema per table can be registered.

4.4.2 Unstructured XMLType

Unstructured XMLType does not require XSD in advance (if XSD is available, it could be used for XML validation). It supports querying an XML document by XPath queries in the database. Multiple XML documents of various schemata can be stored in one database table.

4.4.3 Binary XMLType

Binary XMLType uses XSD and pre-parses XML as the structured XMLType does but it does not require a schema registration in advance. XML document is parsed and stored in a special binary format. Various documents of different schemata can be stored in one column.

Table 4.1 compares all three types. Implementation of XMLType into EEGBase is evaluated in [55] .

Table 4.1 Comparison of the various XMLType datatypes in Oracle 11g

Test	**Structured**	**Unstructured**	**Binary**
More than one XMLType per table permitted	No	No	No
Schema required before table creation	Yes	No	No
Pre-parsed schema	Yes	No	No
XMLType data is post-parsed	No	No	Yes
XPath query	Yes	Yes	Yes
More than one schema per table permitted	No	No	Yes
Universality	No	Limited	Yes
Suitability for one XML schema	Yes	No	Yes
Suitability for multiple XML schemata	No	No	Yes
Suitability for non-schema XML	No	Yes	Yes

4.5 Ontology-based model

Ontologies and ontological modelling provide advanced semantics for Linked Data, a core of a semantic web concept introduced by Wide Web Consortium (W3C) in 2001. The main idea of the semantic web is to extend current HTTP abilities and to provide unified modelling and querying resources over the existing internet infrastructure.

4.5.1 Linked Data

Current World Wide Web (WWW) creates a web from documents identified by URL and interconnected by hyper-links. Linked Data interconnects data; i.e. every resource (data) is identified by the URI/IRI[7] and thus resources with the same URI have same definition/meaning. Linked Data specifies four basic rules[8]:

- *Use URIs as names for things.*
- *Use HTTP URIs so that everyone can look up those names.*
- *When user looks up a URI, provide useful information, using the standards (RDF (Resource Description Framework), SPARQL (SPARQL Protocol and RDF Query Language)).*
- *Include links to other URIs so that they can discover more things.*

[7]RFC 3986: https://www.ietf.org/rfc/rfc3986.txt
[8] http://www.w3.org/DesignIssues/LinkedData.html

Searching results across Linked Data web are not only presented as textual/numeric documents but also as a set of identified data. The inventor of Linked Data, Sir Tim Berners-Lee, is likening data to relationships[9]. Linked data can forms a massive collection of a logically connected information[10].

Resource Description Framework

The simplest representation of a relationship between two data resources is a key-value pair. A set of key-value pairs creates a list or dictionary-like dataset with unified relationships. Diverse relationship types can be specified by triples [source - relationship - target]. A set of triples can create a graph structure where the sources are represented by graph nodes, relationships stand for named edges and targets stand for graph nodes or leaves. Such triples enable a description of any relationship between two resources and link data together, i.e. create Linked Data structures.

Resource Description Framework (RDF), developed by W3C, provides a framework for modelling graphs based on triple statements. RDF uses a natural language terminology and names the triple elements as *subject*, *predicate* and *object*. Based on the Linked Data recommendations, subjects and predicates are the resources identified by URI. The objects are resources identified by URI or literals, i.e. specific values.

The predicates alternate the relationships in ER models and the associations in OO models including extended abilities. Apart from the user-defined predicates, RDF supports the following predefined ones:

- rdf:ID
- rdf:type
- rdf:resource
- rdf:Property
- rdf:Bag
- rdf:Seq
- rdf:Alt
- rdf:Description

[9]http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html
[10]Information is a collection of data interpreted by its logical context

The *rdf* prefix labels a namespace the predicates are taken from. In the case of W3C RDF the namespace is "http://www.w3.org/1999/02/22-rdf-syntax-ns#". Descriptions of the individual predicates can be found in the official RDF and RDF Schema specification[11]. Various RDF serializations exist, e.g., Turtle, RDF/XML.

RDF Schema

RDF Schema (RDFS) extends RDF of a set of pre-defined predicates and objects including the object *Class* and the predicate *subClassOf* known as the OO model elements. It is important to note, that there is a difference between the meaning of the class in the OO model and in RDF. A class in the RDF is a set of individuals with a set of properties those individuals have in common. A subclass of such class is a subset of those individuals. An OO model class specifies the properties of an instance/individual in advance and serves as a template for the instance creation. OO model subclass inherits all previously defined properties and, alternatively, add additional ones. The term of inheritance is lost in RDF classes.

RDFS supports following primitives (their definition can be found in the official specification[12]):

- rdfs:Class
- rdfs:label
- rdfs:comment
- rdfs:domain
- rdfs:range
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:Container
- rdfs:Literal
- rdfs:Property

[11]https://www.w3.org/TR/rdf-schema/
[12]https://www.w3.org/TR/rdf-schema/

SPARQL Protocol and RDF Query Language SPARQL Protocol and RDF Query Language is an RDF graph alternative to SQL. In contrary to SQL, SPARQL only supports data retrieval, and not data modification. Query results can be retrieved in a tabular structure (SELECT), graph structure (CONSTRUCT), single result RDF (DESCRIBE), and boolean type (ASK) answering a question whether the query has a solution or not. SPARQL supports its own SPARQL algebra based on the relational algebra and thus supports set, aggregative and comparative operators [25].

4.5.2 Ontologies

The term *ontology* has a philosophical background and various meanings, however, the following definitions are relevant for the ontology meaning in the field of data modelling.

- *An ontology is an explicit specification of conceptualization.* – Thomas Robert Gruber (1994)
- *Ontologies are defined as a formal specification of shared conceptualization.* – Willem Nico Borst (1997).

The ontologies can be divided into three non-disjunctive groups [67]:

- Terminological and lexical ontologies; extended thesauruses
- Information ontologies; extended database schemata
- Knowledge ontologies; logical theories representing knowledge

Terminological and lexical ontologies will be named as *terminologies*, hereafter. By the term *ontology* the information ontology is meant.

Open World Assumption

A key factor influencing ontology development is the Open World Assumption (OWA). Close World Assumption (CWA)-driven approaches (e.g. ER model, OO model) consider explicitly described elements only. Basically, what is not described, does not exist. This approach enables a "negative" constraint: e.g., *patients with no diabetes are those who **have no** diabetes diagnosis in their health records.* If there is no such a health record in the database, patient can be classified as non-diabetic. The same example does not work in OWA-driven approaches, which consider that everything what is not explicitly eliminated, could exist. For instance, there can potentially be a health record declaring patient's status unless a class closure is explicitly specified.

OWA is a logical consequence of the basic ontology philosophy, where existing described domain should not be described again. Therefore, it is impossible to propose all-describing ontology and only minimalistic stand-alone ontologies, generic enough to be useful from any domain, should be developed. Such ontologies can be eventually extended for particular needs and/or connected to other ontologies, but they should not contain elements for which their reinventing would be initiated.

Given by the fact, that it is quite easy to prove that some element exists but difficult, or impossible, to prove it does not, OWA approach force the developers to greatly change their way of thinking over data models.

Web Ontology Language

Web Ontology Language (OWL) supports constructs based on the predicate logic to creation of computable ontologies. The current version, OWL 2[13], provides sets of classes, properties, individuals and data values. OWL ontologies are composed from the following entities [17], [28]:

- Classes
- Object properties
- Data properties
- Datatypes
- Annotation properties
- Individuals

Classes There are two types of classes:

1. Primitive
2. Defined

Primitive classes are defined by necessary conditions only, i.e. by their super-classes. Defined classes can be defined by necessary conditions and by sufficient conditions, i.e. by equivalent classes. Defined classes can be specified implicitly, i.e. by their definition only. Such classes

[13]https://www.w3.org/TR/owl2-overview/

cannot be referenced and serve mostly for computational purposes only. Named classes (explicitly specified Defined classes), have asserted referenceable names/URIs. As it was aforementioned, classes serve as the containers for individuals. Class axioms support set operators: *union*, *intersection*, and *complement*; and definition of disjoint classes. A number of individuals asserted to the class can be restricted from a zero occurrence to infinity, precise number included (similar to the multiplicity property in OO models).

Object properties Object properties create relationships between two objects, i.e. relationships which can be established between two individuals of two classes. Object property has domain (subject) and range (subject) specified in its statement. Object properties support definition of equivalent property, sub-property, disjoint with other property and inverse property. Following characteristics can be added to object properties:

- Functional property: For each instance x, property P can acquire only unique value y; P(x,y1) and P(x,y2) cannot exist, e.g., *hasHusband(Woman, Man).*

- Inverse functional property: The object value x can be the value of property P for a single instance x, where distinct instances x1 and x2 can exist; P(y,x1), P(y,x2), e.g., *hasBiologicalMother(Child, Woman).*

- Transitive property: If pair (x,y) and (y,z) are both instances of transitive property P, then also (x,z) is an instance of property P, e.g., Pizza *hasIngredient(Pizza, Herb Seasoning)* and *hasIngredient(Herb Seasoning, Basil)*, then *hasIngredient(Pizza, Basil).*

- Symmetric property: If the pair (x,y) is an instance of a symmetric property P, then also (y,x) is an instance of P, e.g., *hasBrother(Vaclav, Bob)*, then *hasBrother(Bob, Vaclav).*

- Asymmetric property: If the pair (x,y) is an instance of an asymmetric property P, then (y,x) cannot be an instance of P, e.g., if *hasChild(Theresa, Hermione)*, then *hasChild(Hermione, Theresa)* cannot exist.

- Reflexive property: Property P relates all elements to themselves P(x,x), e.g., *isEqual(number,number)* for all real numbers.

- Irreflexive property: No elements can be related to itself with irreflexive property P, e.g., *notEqual(x,x)*; an example of the asymmetric property is also irreflexive

Data properties Data properties create relationships between individuals and their values and data types. Object properties support definitions of equivalent property, sub-property, disjoint with other property and inverse property. *Functional* characteristics can be asserted to the data property.

Datatypes OWL supports a basic set of predefined datatypes including all XSD datatypes.

Annotation properties Annotation properties allow us to assign a textual annotation to class or individual, e.g., human readable label, comment, version info, compatibility, etc.

Individuals Individuals (also Instances or Members) are specific data elements asserted to one or more classes. Each individual has a URI, thus its existence is unambiguous. An individual can specify a set of other individuals to which it is or it is not equal. Literals are asserted to individuals via Data properties.

Dialects The number of OWL features and options provides us a powerful resource to build complex ontologies facilitating automatic non-trivial inference computation. However, high complexity of ontologies increases probability of inconsistency and may cause difficulties in the inference computations. To separate the risk-potential constructs from the safe ones and to categorize ontologies according to their complexity, three OWL dialects are specified.

1. OWL Lite: The simplest dialect designed mainly for building classification hierarchies. Constraints are limited to *IntersectionOf* operator only, cardinalities are restricted to values 0/1. OWL Lite is a subset of OWL DL.

2. OWL DL: The full description logic-based dialect designed to provide maximal expressiveness retailing the computational abilities and completeness. The dialect supports all aforementioned elements and full cardinalities. Both OWL DL and OWL Lite require separation of classes, instances, properties and values. OWL DL is a subset of OWL Full.

3. OWL Full: The dialect supports all OWL constructs without any constraints. OWL Full allows us to augment or redefined the pre-defined constructs. The owl:class and owl:Data/ObjectTypeProperty are not defined as subclasses of rdf:class and rdf:Property as they are in OWL Lite/Full, but as equivalent classes. Thus, the classes, instances, properties, and values are not strictly separated. For instance, one resource with a URI can stand for a class and for a property in the same time. OWL Full supports "unlimited" expressiveness, but its computational abilities are not guaranteed.

As the OWL DL supports rich expressiveness with preserved computability, only this dialect will be considered hereafter.

Profiles Aside from the dialects, OWL 2 is also divided into the profiles (Figure 4.12) [48]. The profiles, likewise the dialects, reduce OWL 2 expressiveness in order to meet better computational abilities (*reasoning*) for different application scenarios.

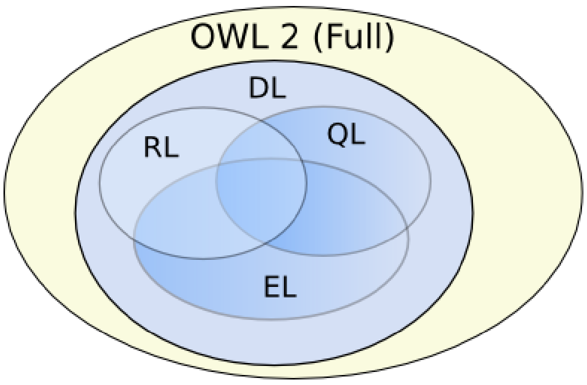


Figure 4.12 Venn diagram of the OWL 2.0 profiles (http://www.w3.org/TR/2009/WD-owl2-overview-20090327/)

- OWL EL: Suitable for a large number of classes/properties
- OWL QL: Suitable for simple ontologies with large number of individuals
- OWL RL: Suitable in the case when both scalable reasoning and rich expressive power are needed

Running example As the OWL supports a wide collection of constructs, its graphical representation becomes complicated for simple ontologies. For that reason, only fragments of the example ontology are shown. The ontology is designed for the demonstration purpose only in terms of its re-usability. Figure 4.13 shows the classes, object properties, domains and ranges. Figure 4.14 shows the class Laboratory and asserted class axioms. The ontology example was designed in the ontology editor Protégé[14].

Serialization The OWL 2 can be serialized into five standardized machine-readable formats. Table 4.2 shows a list of formats/syntaxes together with their main benefits. OWL 2 structure and its connection to the serialization formats are shown in Figure 4.15.

Semantic reasoners Semantic reasoners are generalized OWA-driven inference engines which infer logic non-trivial consequences within the ontology. Computed inferences can be

[14]http://protege.stanford.edu/

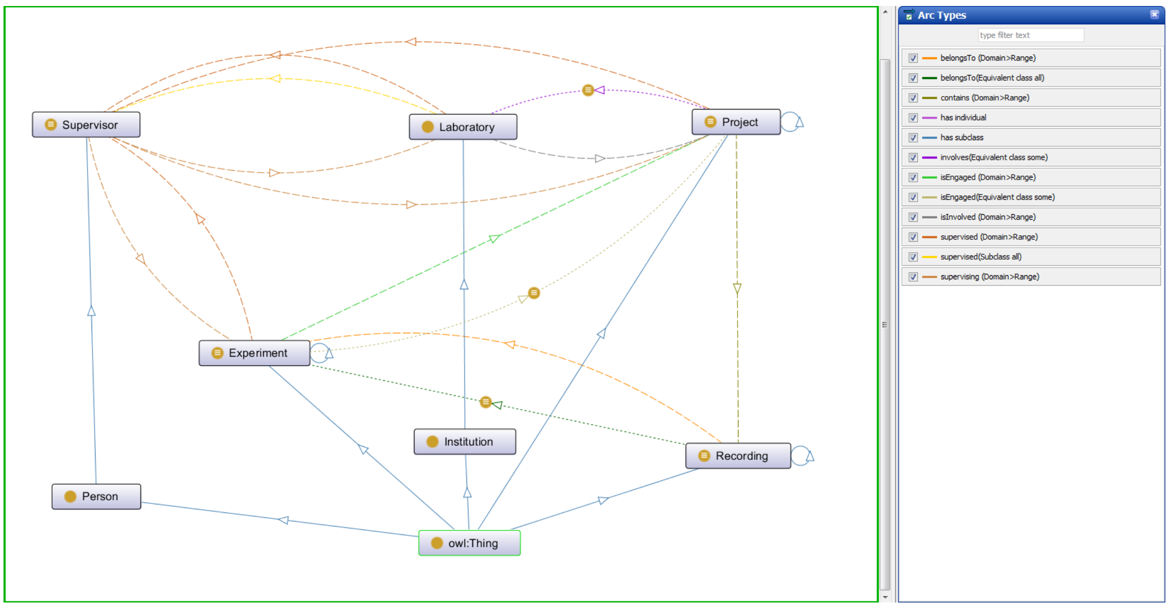


Figure 4.13 A laboratory ontology example visualized by OntoGrap (https://protegewiki.stanford.edu/wiki/OntoGraf); The graph shows classes, object properties, domains and ranges.

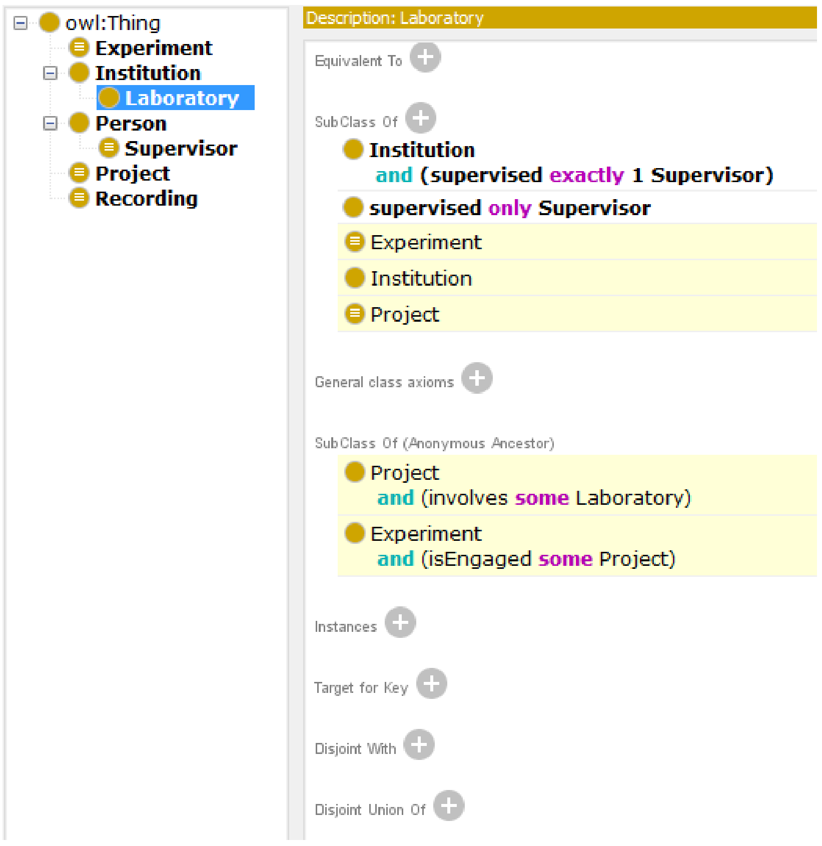
Table 4.2 Overview of the OWL syntaxes

Syntax	**Usage**
RDF/XML	commonly known universal serialization; suitable for exchange
OWL/XML	suitable for processing by XML tools
Functional Syntax	follows the ontology structure, not abstract syntax
Manchester Syntax	human-friendly format
Turtle	human-friendly format

further stored as the new ontologies. To obtain CWA results, inferences can be queried by SPARQL including SPARQL algebra.

Reasoners provide mainly the following services:

1. Realisation
2. Classification
3. Satisfiability
4. Entailment
5. Consistency



Institution
and (supervised exactly 1 Supervisor)
supervised only Supervisor
Experiment
Institution
Project

Project
and (involves some Laboratory)
Experiment
and (isEngaged some Project)

Figure 4.14 An example of Laboratory class from the example ontology; Class hierarchy is shown on the left side; the right side shows explicitly defined class axioms on white background and axioms automatically inferred by semantic reasoner on the yellowish background.

A list of the available reasoners, including Fact++[15], Hermit[16], Pellet[17], and ELK[18], and their descriptions is provided by the OWL group at the University of Manchester[19].

Semantic Web Rule Language Semantic Web Rule Language (SWRL) [29] is designed to specify rules in the expressive power of OWL DL constructs. SWLR rules are readable and intuitive to write. However, not all semantic reasoners support SWRL. SWRL rules are substitutable by common OWL DL constructs.

[15]http://owl.man.ac.uk/factplusplus/
[16]http://www.hermit-reasoner.com/
[17]https://github.com/stardog-union/pellet
[18]https://www.cs.ox.ac.uk/isg/tools/ELK/
[19]http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

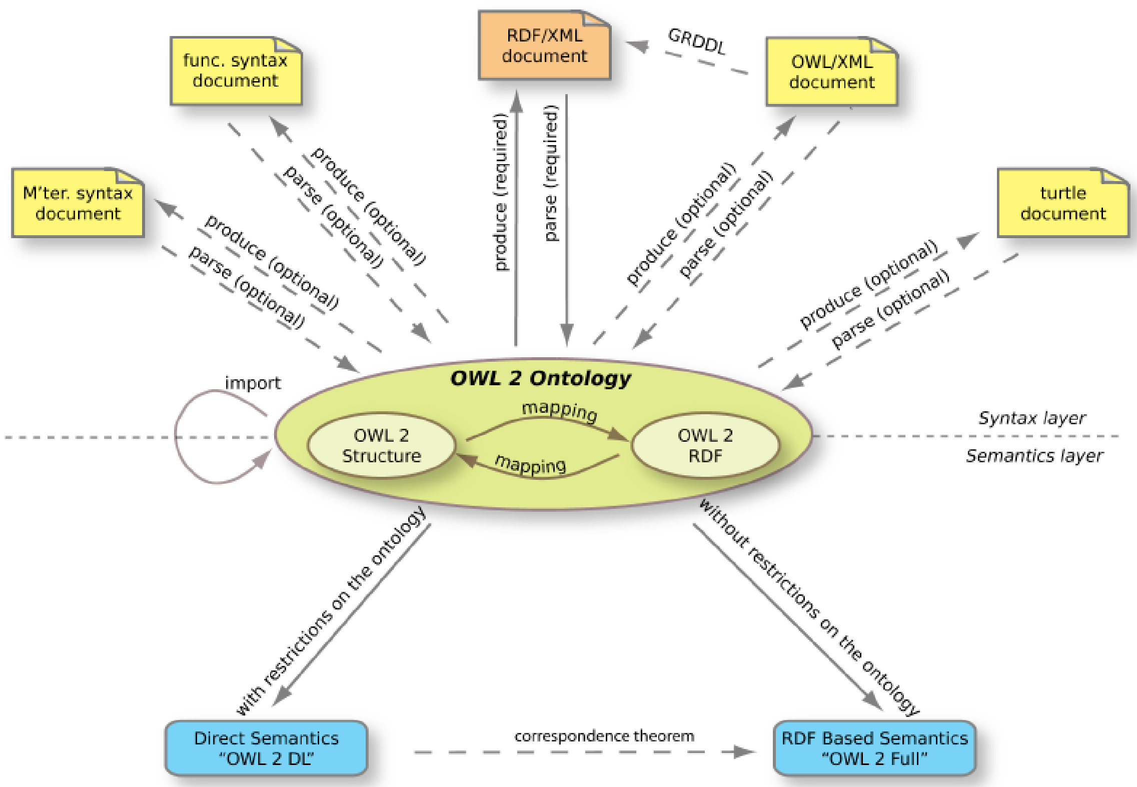


Figure 4.15 The structure of OWL 2.0 (http://www.w3.org/TR/2009/WD-owl2-overview-20090327/)

Open Biomedical Ontologies

Open Biomedical Ontologies (OBO)[20] provide a set of biomedical controlled vocabularies. OWL is primarily designed to provide a computable ontology, describing any domain in a generic (top-down) approach. OBO is designed to provide an ontology describing biological domain in a specific (bottom-up) approach. OBO does not support automatic reasoning. Transformation mechanisms between OBO and OWL (or vice versa) exist but losses of expressiveness are unavoidable. While OWL is popular with computer scientists, OBO is more popular with computational biologists. Since the OBO usage is very narrow, it is not considered further.

[20]http://www.obofoundry.org/

4.5.3 Semantic web

The current WWW can be perceived as a decentralized knowledge-base composed from hyper-linked documents. The semantic web can be considered as an extension of WWW using aforementioned concepts. The semantic web architecture is shown in Figure 4.16.

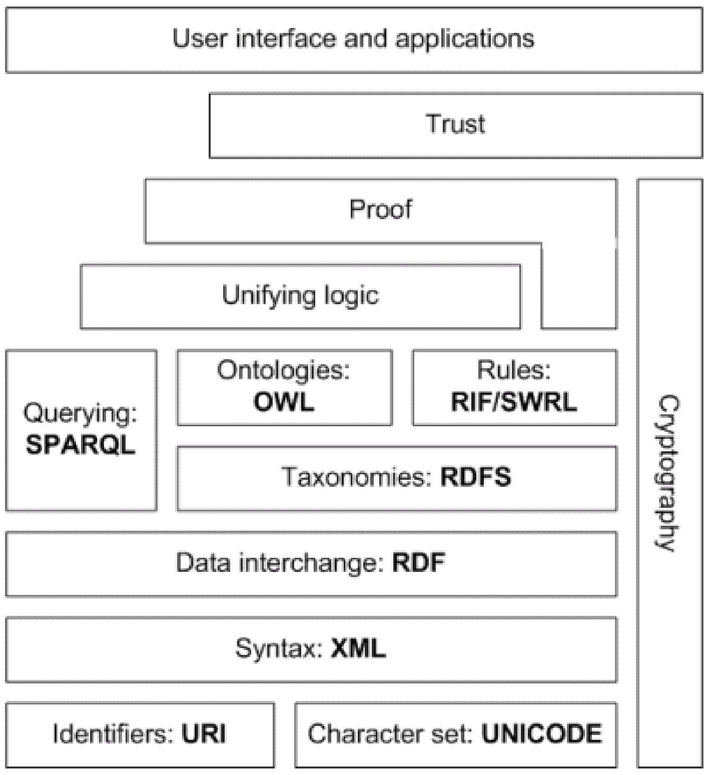


Figure 4.16 An architecture of the Semantic Web [54]

A bottom layer specifies the identifiers in a URI syntax and codeset. The second layer specifies standardized serialization format as the XML for RDF (third layer) graphs, namespaces and XML schemata. Above the RDF graphs, taxonomy extensions (RDFS), ontologies (OWL), rule languages (SWRL or Rule Interchange Format (RIF)) and SPARQL are specified. The unifying logic layer deals with automatic deduction of information from ontologies and includes semantic reasoners. The proof layer is designed for verification of the deduced expressions. The thrust layer ensures trustworthiness of returned data.

It is important to mention, that not all layers are fully implemented (e.g. *Trusts*) so far and thus the overall architecture is theoretical.

4.6 NoSQL databases

The noSQL (not-only SQL) databases represent a specific storage category, which is not based, unlike RDBs, on the table structure. Main purpose of the noSQL DBs is to provide a data storage system suitable for a huge amount of unstructured data, data which cannot be structured or data which could be structured but tabular representation is not convenient (OO model, RDF). Three major representatives within a wide variety of NoSQL database concepts are:

- Key-value repositories based on associative arrays, where a unique key is related to an arbitrary value, i.e. dictionary concept
- Document databases providing a repository for encapsulated data in the standard formats like XML, JSON etc.
- Graph databases adapted to graph-structured data, e.g. RDFs

NoSQL databases will be discussed and used further in this work.

Chapter 5

Mapping between the models of various levels of expressiveness

This chapter describes relationships between features and elements of the data models described in chapter 4, it discusses differences in their mutual expressive power and highlights incomparable elements/features. Consequently, it proposes a set of mapping rules for preserving maximal compatibility and interoperability across the models within the same domain.

5.1 Spoken word to ER model

From a perspective of ER model, a spoken word has nearly unlimited expressive power. Therefore, initial assumption is that only simple grammar sentences are considered and those are from strictly specified domain, where word unambiguity is ensured. Correct identification of nouns, adjectives, verbs and numerals (implicit and explicit) is helpful in case of ER model design. Also position of nouns in the sentence is crucial, i.e. subjects and objects. Precise determination of [subject-predicate-object] triples is key for almost any data model design. Table 5.1 shows the basic relationships.

Table 5.1 Relationships between spoken word and ER model elements

Spoken word	**ER model**
Noun (Subject)	Entity/Attribute
Noun (Object)	Attribute
Verb (Predicate)	Relationship
Numeral	Cardinality
Other	Dependency, constraint, etc.

5.2 ER model to spoken word

Reverse relationships from Table 5.1 can be used to construct simple sentences. Those sentences should be meaningful (semantically, not grammatically) without any additional semantics, otherwise the model is probably poorly proposed, i.e. entities and relationships could be identified in unsuitable way.

5.3 ER model to OO model

Mapping between ER model and OO model is straight as the OO concept fully supports all elements/features supported by ER concept. Table 5.2 shows OO model equivalents for ER model elements/features.

Table 5.2 OO model equivalents for ER model elements/features

ER model	**OO model**
Entity	Class
Attribute	Attribute
Relationship	Association
Cardinality	Multiplicity

5.4 OO model to ER model

As OO model has stronger expressive power than ER model, some elements/features must be simplified (and thus, some expressiveness is lost) for backward mapping (Table 5.3). The set of OO model elements/features was selected according to those supported by UML class diagram.

5.5 OO model to XSD

XSD supports various ways to express same or similar concepts (e.g. attributes) and that makes the XSD expressive power slightly stronger than that of OO model. There is no semantic difference if an attribute is defined as an *XML attribute* or an *XML element*. However, an additional user-defined rule precisely specifying how those two ways should be used, could draw on this ability and extend the expressiveness; e.g. *XML elements* could carry attributes directly related to described data model and *XML attributes* could describe the external context. Ambiguous mapping proposals are seen in Table 5.4 and 5.5.

Table 5.3 Mapping of OO model into ER model elements/features

OO model	**ER diagram**
Class	Entity
Attribute	Attribute
Encapsulation	N/A
Methods	N/A
Multiplicity	Cardinality; multiplicity has to be simplified as follows (*?* substitutes any natural number but 1 or 0) • 1 ->1 • ? –>n (restriction NOT NULL) • * ->n • 0..? ->n • ?..? –>n
Association	Relationship
Aggregation	Relationship with loss of the “is part of” information
Composition	Relationship; the entity of a composite class has to carry a foreign key referring to a component class entity
Dependency	N/A
Specialization/Generalization	N/A

Table 5.4 Mapping of OO model into XSD elements

OO model	**XSD**
Class	Complex type
Complex attribute	Element / Sequence type / Complex content / Complex type
Primitive attribute	Element / Attribute / Simple content / Simple type
Multiplicity	Minimal occurrence / Maximal occurrence
Generalization/Specification	Extension
Association	Group
Aggregation/Composition	Extension / Group / Group attribute

5.6 XSD to OO model

Adequately to previous section, inverse mapping from XSD to OO model elements is proposed in Table 5.5.

Not all available XSD 1.1 elements are described in Table 5.5. More elements can be found in W3C recommendation[1]. Their usage is very specific and therefore, they are not to be described here.

[1]https://www.w3.org/TR/xmlschema11-1/

Table 5.5 Mapping of XSD elements into OO model elements

XSD	**OO model**
Complex type	Class/Attribute
Sequence	N/A
All	Multiplicity 1
Choice	Multiplicity 0..*
Occurrence	Multiplicity ?..?
Element	Attribute/Class
Attribute	Attribute
Simple content	Attribute
Complex content	Attribute
Extension	Generalization/Specialization
Group	Aggregation/Composition/Association/Generalization
Attribute group	Aggregation/Composition
Element any	N/A
Element anyAttribute	N/A
Substitution	N/A (Dependency/Association/Generalization)

5.7 ER model to RDF/OWL

The relationship between two relations in ER model can be considered as a *[subject-predicate-object]* triple. In the same manner, the columns of one table/relation can be assigned to the relation as a triple *column-"assigned_to_relation"-table* (Table 5.6).

Table 5.6 RDF/OWL equivalents for ER model elements/features

ER model	**RDF/OWL**
Entity	rdf:class
Attribute	rdf:resource
Relationship	rdf:property
Cardinality	owl:cardinality

Detail information about the transformation is provided in, e.g. [44]. It is important to realize that the outcome graph reflects only the structure of ER model and no implicit semantics. Additional semantics can be achieved with pre-defined ontology, which can be included into mapping process. There are some tools, which transform relational database schema into RDF graph or provide SPARQL endpoint over the data in relational database with the ontology support including D2RQ[2], whose architecture is shown in Figure 5.1.

[2]http://d2rq.org/

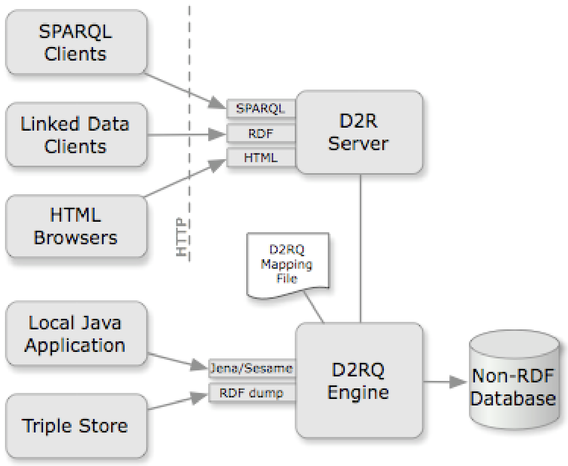


Figure 5.1 D2RQ Architecture [5]

The main purpose of D2RQ is to provide an interface allowing users to query the RDB by SPARQL. During the creation process of the virtual RDF graph the additional ontology can be included and linked by proprietary D2RQ mapping. This ontology must fully correspond with RDB model and has to be created manually. D2RQ platform is separated into three parts:

- D2RQ mapping language
- D2RQ Engine
- D2R Server

The mapping language is designed for a specification of the relationships between ER model and the ontology. The engine processes the mapping and queries connected to repositories. And finally, server provides a user interface (SPARQL, HTML and RDF based).

EEGBase implements D2RQ [57] in combination with Apache Jena[3] and OWL API[4]. D2RQ is used for a construction of Jena-based RDF graph, which is serialized by OWL API.

5.8 OO model to RDF/OWL

Even though both, OO model and RDF/OWL, support the concept of classes, it is important to realize that the meaning of them is different. While OO model classes can be considered as templates for instances carrying data modelled in advance, RDF/OWL classes can be

[3]https://jena.apache.org/
[4]http://owlapi.sourceforge.net/

considered as sets of individuals. This is reflected also in the absence of inheritance/generalization feature. However, as the arbitrary predicate can be constructed, the possibility of user-defined predicates representing OO model relationships is ensured. Table 5.7 shows the potential equivalents between OO model and RDF/OWL elements.

Table 5.7 Mapping of OO model into RDF/OWL

OO model	**RDF/OWL**
Class	rdf:class
Attribute	rdf:class
Encapsulation	N/A
Methods	N/A
Multiplicity	owl:restriction (minQualifiedCardinality, maxQualifiedCardinality, someValuesFrom, allValuesFrom,cardinality)
Association	owl:objectProperty, owl:equivalentClass, rdfs:subClassOf
Aggregation	owl:objectProperty, owl:equivalentClass, rdfs:subClassOf
Composition	owl:objectProperty, owl:equivalentClass, rdfs:subClassOf, owl:Union
Dependency	owl:objectProperty, owl:equivalentClass
Specialization/Generalization	owl:objectProperty, owl:equivalentClass, rdfs:subClassOf

EEGBase implements a transformation of the OO model into RDF graph using JenaBean[5]. The solution is based on the Object-Relation Mapping (ORM) framework Hibernate. The transformation rules are in-scripted into Java Beans as Java annotations. The main disadvantage of the solution is its double transformation: from RDB to OO model by Hibernate and from OO model to RDF graph by JenaBeans.

5.9 XML and XSD to RDF/XSD

RDF provides the rdf:XMLLiteral datatype within which whole XML can be stored as a literal. However, this approach does not append the content of the XML into the queryable RDF graph.

A potential transformation of XML supported by XSD into OWL/RDF graph is realized via incremental extension of the outcome graph by progressive passing of XSD (i.e. a main graph structure) and XML (i.e. creation of the individuals).

XSD element equivalents in RDF/OWL elements are adequate to *OO model to RDF/OWL* mapping. UIRs can be created from the XSD namespaces, element names and element paths

[5]http://code.google.com/p/jenabean/

within the XSD. As XSD describes a structure of XML, the mapping also refers to classes and property definitions. Table 5.8 shows the potential mapping relationships.

Table 5.8 Mapping of XSD elements into RDF/OWL elements

XSD	**RDF/OWL**
Complex type	rdf:class, owl:equivalentClass, rdfs:subClassOf
Sequence	N/A
All	owl:allValuesFrom
Choice	owl:cardinality, owl:someValuesFrom
Occurrence	owl:minQualifiedCardinality, owl:maxQualifiedCardinality, owl:someValuesFrom
Element	rdf:class
Attribute	rdf:class
Simple content	owl:dataProperty, owl:literal
Complex content	rdf:class
Extension	owl:objectProperty
Group	owl:objectProperty, owl:union, owl:equivalentClass
Attribute group	owl:objectProperty, owl:union, owl:Intersection
Element any	N/A
Element anyAttribute	N/A
Substitution	N/A (alternative – Dependency/Association/Generalization)

XML elements carrying data could be asserted to pre-created classes as individuals of those classes. URIs of these individuals could be constructed from the logical path from the XML root to the individual element. RDF and RDFS uses the same datatype set as XSD does, therefore, the literals can be used as they are.

A practical use-case of such transformation is the experiment protocols in EEGBase. As the exported RDF graph reflects the EEGBase RDB structure, protocols are kept in XML files and thus exported as string literals. Decomposition of these XML files supported by valid XSD would extend the RDF graph of the protocol content.

5.10 RDF/OWL to models with less expressiveness

As RDF/OWL could acquire a very high complexity and expressiveness in its OWL2 DL dialect, backward mapping is a matter of manual work and thus no mapping proposal is presented here.

5.11 Semantic hierarchy

Described data modelling concepts provide various features ensuring various levels of expressiveness that are not always mutually transformable. Thus, a hierarchy of the concepts can be constructed, where the expressiveness grows with higher levels. In such a structure each level carries as much information from the lower level as possible. While bottom-up steps lever the expressive power of its predecessor, top-down steps may be associated with a loss of information [58]. Figure 5.2 shows the hierarchy composed on basis of the mentioned concepts apart from XSD since it cannot be considered as fully-fledged standalone modelling concept even though its expressiveness slightly differs from the other concepts. The figure shows that ER model uses only a terminology from the dictionary level, OO model uses terminology, attributes, relationships and cardinalities from the ER model level etc. Despite the fact, that the ontology is the most distant from the simple key-value dictionary/terminology, the ontology rid of the properties/predicates is essentially a terminology. An important ability of all layers is their potential to be expressed by triples [subject, predicate, object]; a dictionary which is composed of pairs implicitly uses one predicate in common. Therefore, all layers can be serialized into an RDF graph and queried in a unified way using the same terminology.

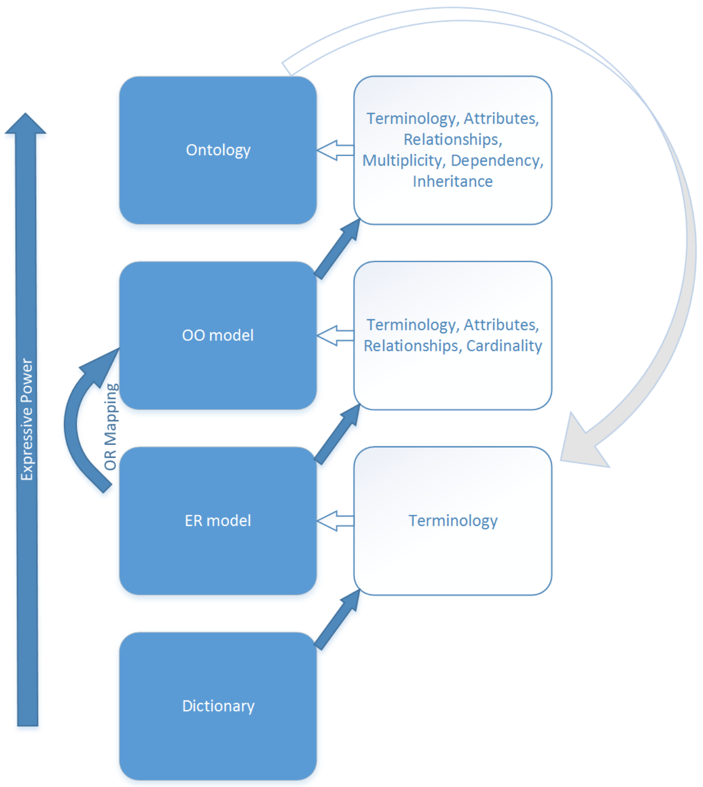

Ontology
OO model
ER model
Dictionary
Terminology, Attributes, Relationships, Multiplicity, Dependency, Inheritance
Terminology, Attributes, Relationships, Cardinality
Terminology
Expressive Power
OR Mapping

Figure 5.2 Semantic hierarchy of data models: Blue boxes represent discussed models; White boxes show the features carried from the lower model to a model with stronger expressive power

Part IV

Electronic Health Records

Chapter 6

Electroencephalography recording as an Electronic Health Record

This chapter evaluates the data/metadata as records with medical/health characteristics. Since the other electrophysiological examinations, e.g., ECG, are considered as a routine medical records, and thus, they are described in details within various medical/clinical terminologies (e.g. SNOMED-CT), EEG/ERP can be deliberate in the same way. However, there is a lack of EEG/ERP descriptions in these terminologies/standards.

6.1 Computable health and medical records

There are various types of records which differ in their intended usage but not necessarily in their content [22], [52], [21].

Electronic Medical Records (EMR) mostly stand for an electronic alternative of paper medical records. In general, EMR system is an electronic equivalent to a medical record file cabinet. EMRs are stored at one place, i.e. General Practice (GP), managed by the physician of that GP and they are not supposed to be shared or integrated into any complex system. The institution/GP is responsible for the privacy and security. A collection of patient's EMRs can be consider as a patient's state history in the field of that the institution/GP is operating in.

Electronic Health Records (EHR) are intended to cope a more complex role over various institutions a thus provide information about patient state across multiple domains. Essentially, EHRs describe symptoms, diagnosis, laboratory tests, clinical examination findings, prescriptions, treatments and procedures, referrals, etc., gathered from all levels of health care systems (e.g. primary care, secondary care, mortality register, etc.) [61]. EHRs are designed to be shared inter-institutionally. EHRs are structured, semi-structured and unstructured. The sharing ability and potential utilization for research purposes (e.g. EHR-driven phenotyping) are also eased by implementing controlled clinical terminologies and ontologies, standardized data formats, communication protocols etc.

Personal Health Records (PHR) represent health records fully maintained by patient. Patient can gather health data from his/her physicians (textual EMRs, x-ray images, etc.), household medical devices (glucometer, sphygmomanometer, etc.), smart phone applications/trackers (heart rate, sleep data, activity statistics, etc.) and others. PHR carries information about patient's health history for patient's personal usage.

According to these concepts typical user groups of overlaying information systems can be determined (Figure 6.1). Patient is a user of the personal (typically user-centric) system and owns his/her PHRs. A physician is a user of the EMR/EHR (typically data-centric) system and maintains patient's data. There can also be a situation when patient has read access to his data in EMR/EHR systems (e.g. IZIP[1]), or the physician has access to data in PHRs granted by the patient (e.g. Microsoft HealthVault[2]). Apart from patients and physicians, a third user group consists of third-parties involved into health care processes. Typical representatives of this group are insurance companies providing expense information based on the interventions recorded in EHRs.

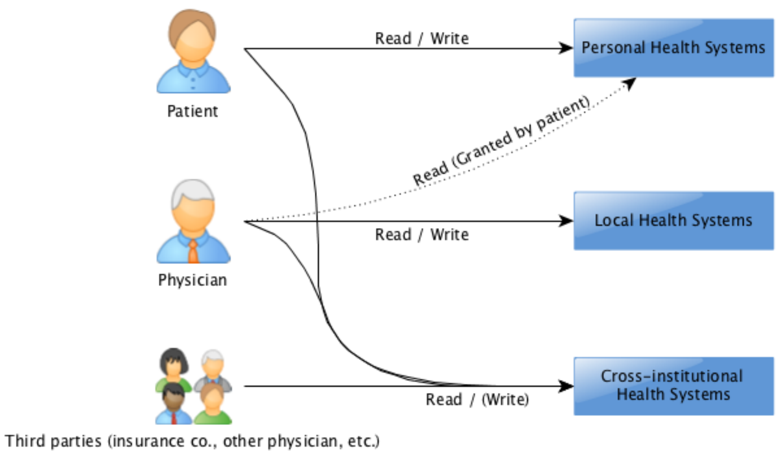

Patient
Physician
Third parties (insurance co., other physician, etc.)
Read / Write
Personal Health Systems
Read (Granted by patient)
Read / Write
Local Health Systems
Read / (Write)
Cross-institutional Health Systems

Figure 6.1 Abstract graphical representation of health systems user groups [56]

[1]http://www.izip.cz/
[2]https://www.healthvault.com/

6.2 Overview of clinical/medical standards

Standards implemented over structured EHRs and EHR systems can leverage data unambiguity, interoperability and ability for scientific purposes. There are two main representatives: HL7 and openEHR. For the deeper context, also ISO/CEN 13606 and Czech national Dasta standard are described.

6.2.1 CEN/ISO 13606

The European standard CEN/ISO 13606 [51] is *designed to achieve semantic interoperability in the electronic health record communication*[3]. The standard is based on the so-called Dual Model architecture which consists of the OO generic reference models (RMs) and specific archetypes defined as constraint-based models of domain entities. RMs contain entities for representing any information of EHR. Archetypes define a clinical concept, i.e. knowledge. While RMs are immutable, archetypes are mutually independent and they are being developed in particular needs. According to the official CEN/ISO 13606 website[4], RMs contain:

1. a set of primitive types
2. a set of classes defining the EHR building blocks; any annotation in an EHR must be an instance (so-called entity) of one of these classes; CEN/ISO 13606 defines six types of entities: folder, composition, section, entry, cluster, and element
3. a set of auxiliary classes describing the context information; these are supposed to be attached to the EHR annotations (e.g. versioning information)
4. a set of classes describing demographical data

Archetypes are composed from header, definition and ontology sections. The header section contains metadata about the archetype. The definition section contains a description of the clinical concept according to RM used. The description section constraining the entities on[5]:

- range of attributes of primitive types
- existence of attributes
- cardinality of attributes

[3]http://www.en13606.org/the-ceniso-en13606-standard

[4]http://www.en13606.org/the-ceniso-en13606-standard/reference-model

[5]http://www.en13606.org/the-ceniso-en13606-standard/archetype-model

- occurrences of objects
- attributes of complex objects

The ontology section binds the entities with terminologies and definitions.

Archetype Definition Language (ADL) for archetype manipulation is a formal computable language for expressing and serializing archetypes.

6.2.2 openEHR

openEHR[6] is an open domain-driven platform for developing flexible e-health systems. This platform, with respect CEN/ISO 13606, presents a set of generic RMs as an abstract specification of elements/processes in the health sphere (EVALUATION, ACTION, OBSERVATION, INSTRUCTION, ADMINISTRATION ENTRY). The RM also defines the structure of (i) archetype sets (COMPOSITION, SECTION) and (ii) datapoints (CLUSTER, ELEMENT, TREE). Data types are also defined by openEHR RM (Figure 6.2).

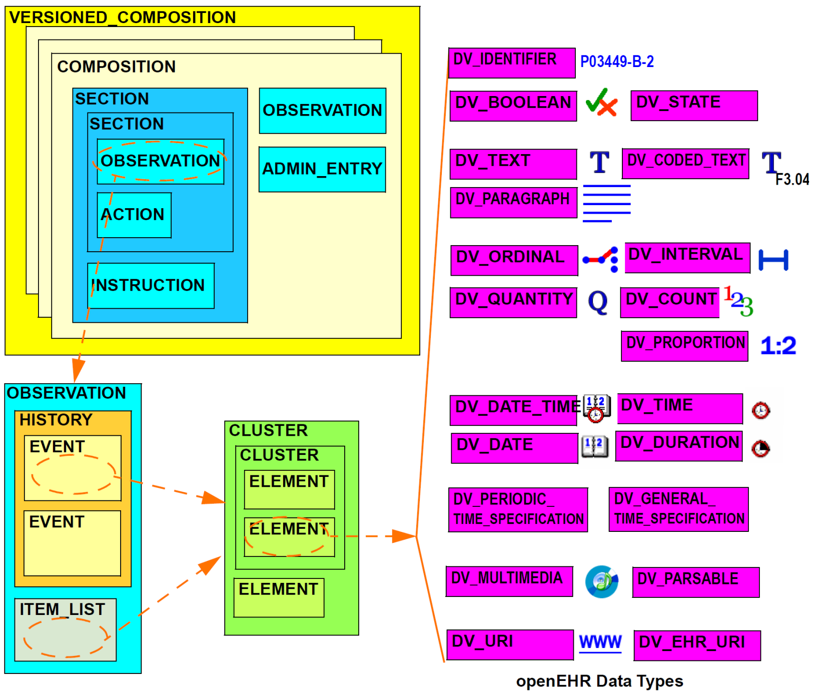


Figure 6.2 openEHR RM elements [4]

Above the RM, the archetypes, abstract representations describing particular domains or their parts, are modelled (e.g. ECG or blood pressure). While RM is only an OO-based

[6]http://openehr.org/

abstract model, which can be expressed, e.g., by UML (Unified Modelling Language), openEHR archetypes can be expressed in the machine-readable ADL (Figure 6.3). The archetype structure consists of five mandatory sections: archetype ID, concept, language, definition, and ontology and four optional sections: specialization, description, invariants, and revision history. While some sections represent an archetype metadata set, the core of the archetype is specified in definition and ontology sections. The definition section describes the domain the archetype is focused on. Each attribute (so-called datapoint in the openEHR terminology) is denoted by its internal code/ID. These IDs are paired with their real names, binding codes, and definitions in the ontology part. The binding sub-part refers to the terms used from an external resource (ontology, terminology, etc.), typically SNOMED-CT or odML terminology in the case of this work. The revision history part of the archetype carries archetype metadata dealing with the changes in structure of a once-approved archetype. Each archetype has its life-cycle state specified in its metadata set according to its phase in the publishing process. [59]

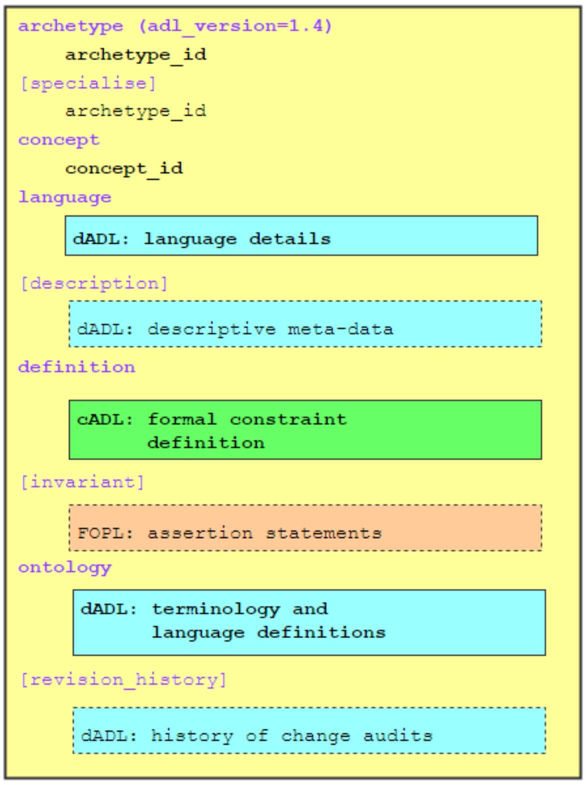

archetype (adl_version=1.4)
archetype_id
[specialise]
archetype_id
concept
concept_id
language
dADL: language details
[description]
dADL: descriptive meta-data
definition
cADL: formal constraint definition
[invariant]
FOPL: assertion statements
ontology
dADL: terminology and language definitions
[revision_history]
dADL: history of change audits

Figure 6.3 ADL Archetype Structure [2]

Archetypes should be designed in a general way to enhance their re-usability. Afterwards, the archetypes are usually implemented via templates, the third layer of the openEHR multilayer approach, which allows users to define more restrictions, connect two or more archetypes together, and reduce the datapoint set of the archetype. To maximize re-usability of the existing archetypes, public repositories called Clinical Knowledge Managers (CKMs)[7] were built. Since no EEG archetype exists in the CKM so far, there was an opportunity to create it. [59]

openEHR also includes grammar specifications of a querying language, the Archetype Query Language (AQL). AQL is developed for searching clinical data found in archetype-based EHRs, however no current formal implementation of a query engine exists.

Even though openEHR is not matured in some ways (e.g. in the quality of development tools), it has been successfully deployed at various places in the UK, Australia, and Russia. Moreover, with respect to its active community and openness, the framework has a chance of becoming more widespread in the future. [59]

6.2.3 Health Level Seven

Health Level Seven (HL7)[8] [18] set of standards operates on the seventh level of the communication ISO/OSI model[9], i.e. the application level. A primary purpose of HL7 is to standardize communication process between various health care systems. HL7 handles a data format as well as a content of clinical, administrative, financial and logistical data. Currently, two supported versions exist: HL7v2.x and HL7v3. The differences between version two and three are significant and thus neither backward compatibility is supported.

HL7 version 2.x

HL7v2.x messages use a proprietary textual format. Its syntax is similar to a Comma Separated Values (CVS)) principle. Higher subversions of the HLv2.x are backward compatible. An example of the HL7v2.3 message shown in Listing 6.1 is taken from the Priority Health website[10].

Listing 6.1 An example of the HL7v2.3 message

```
MSH|^~\&|CERNER||PriorityHealth||||ORU^R01|
    Q479004375T431430612|P|2.3|
PID|||001677980||SMITH^CURTIS||19680219|M
    ||||||||||929645156318|123456789|
PD1||||1234567890^LAST^FIRST^M^^^^^NPI|
```

[7]http://www.openehr.org/ckm/

[8]http://www.hl7.org/

[9]https://www.iso.org/standard/20269.html

[10]http://www.priorityhealth.com/provider/manual/office-mgmt/data-exchange/hl7/hl7-samples

```
OBR|1|341856649^HNAM_ORDERID|000002006326002362|648088^Basic
    Metabolic Panel|||20061122151600|||||||||1620^Hooker^
   Robert^L||||||20061122154733|||F
   |||||||||||20061122140000|
OBX|1|NM|GLU^Glucose Lvl|59|mg/dL|65-99^65^99|L|||F
   |||20061122154733|
```

HL7 version 3

HL7v3 goes further and it aims to cover all workflows in the health care domain. It draws on an OO principle and presents the Reference Information Model (RIM). EHR messages are designed according to RIM and stored in XML files. The main part of HL7v3 is the Clinical Document Architecture (CDA) [18]. CDA defines a specification of messages construction respecting RIM. RIM and CDA messages can be considered as the equivalents to RM and archetypes in ISO/CEN 13606 or openEHR. An openEHR community defines various weaknesses in CDA [9] and proposes their archetypes in a way to eliminate those limitations. A transformation between HL7 CDAs and ISO/CEN 13606 (openEHR perspectively) is a subject worth of investigation in, e.g. [46]. The mapping without loss of precision is not guaranteed, however some mappings are defined.

HL7 is mainly designed for a data transfer between institutions, while CEN/ISO 13606 and openEHR are focused on the explicit specification of the clinical content and workflow. Moreover, their design is proposed for user-centric systems, unlike that of HL7. The openEHR concept and features can help in the context of problems related to data/metadata separation.

HL7, mainly in the version 2, is internationally implemented in various countries over whole world including Australia, UK, China, USA etc.

Currently, HL7 does not support publicly available repository of the domain-based constraint models, and thus, there is no explicit definition of the EEG/ERP domain available.

6.2.4 DASTA

DASTA[11], in contrary to aforementioned standards, is not a big player on the field of EHR standards. However, it is the only representative of the Czech national data standard respected by the Ministry of Health and national health care system. DASTA, which was established in 1997, is an open data standard for the communication between information systems. It also includes controlled vocabulary for annotating EHRs by clinical codes. The

[11] http://www.dastacr.cz/

vocabulary contains a set of Lists of Values (LOV) covering clinical, laboratory, statistical and administrative domains. Besides the LOVs, DASTA also includes tools and documents. DASTA does not support predefined generic models like RMs or RIM.

Currently, there is no detailed EEG/ERP domain description in DASTA LOVs.

Chapter 7

EEG/ERP domain in openEHR concept

This chapter describes the development process of the new openEHR archetypes describing the experiments in EEG/ERP domain. The content is mostly taken from [59].

7.1 Experimental EEG/ERP domain in EHR standards

Since most data and metadata collected during experimental work in the EEG/ERP domain could be classified as health data, it is beneficial to apply some recommendations and frameworks, designed primarily for clinical health data and Electronic Health Records (EHR), to them:

- a unified and explicitly defined concept controlled by higher authorities,
- a unified data description regardless of the institution in which the data has been acquired (e.g. in a hospital or experimental laboratory),
- knowledge of the concept that makes data sharing simplified and data interpretation unambiguous,
- predefined data structures validated by domain experts in a ready-to-use form that could be used within the data structures describing the EEG/ERP experiment,
- an implicit logical separation of data and metadata,
- analytical tools and storage solutions driven by a unified concept,
- a middle-ware for potential mapping between existing formats.

HL7 is mainly designed for data transfer between institutions, whilst CEN/ISO 13606 and openEHR are focused on the explicit specification of clinical content and work-flow. Moreover, their design is proposed also for user-centric systems (unlike HL7). The openEHR concept and features could be helpful in the context of problems related to data/metadata separation.

7.2 Design of EEG archetypes and templates

Firstly, it is convenient to mention, that the purpose of the work is not to compete with the existing proposals and solutions, but to be in line with them and their current potentials.

The design procedure of a basic EEG/ERP archetype set can be described in the following steps:

1. the concepts and their sub-parts are determined within the EEGBase experiment structure, NIX structure and EDF+ attributes,
2. the same and/or similar concepts are aggregated,
3. the redundant and unimportant attributes are eliminated,
4. the determined concepts and their semantically most suitable openEHR RMs are matched (when this is not possible, the concept is separated into smaller pieces and matched with the RM Cluster),
5. the archetypes corresponding to the determined concepts are implemented in ADL,
6. the archetypes are bound with external terminology,
7. the templates covering the EEGBase experiment structure are implemented.

7.2.1 Concepts determination

Figure 7.1 shows an example of the concepts determined from the EEGBase experiment structure. Since the experiment context is superior to the experiment itself from the hierarchical perspective, it should be a responsibility of the system which implements the archetypes to link it properly with external archetypes/data (e.g. subject's/experimenter's demographical information or information related to the project are not described inside the new archetypes). The subject's overall characteristics and state were composed from other health records which were not directly connected to the EEG experiment. Therefore, the red boxes in the figure are not further considered in the archetypes design. The NIX and EDF+ attributes were processed in the same way.

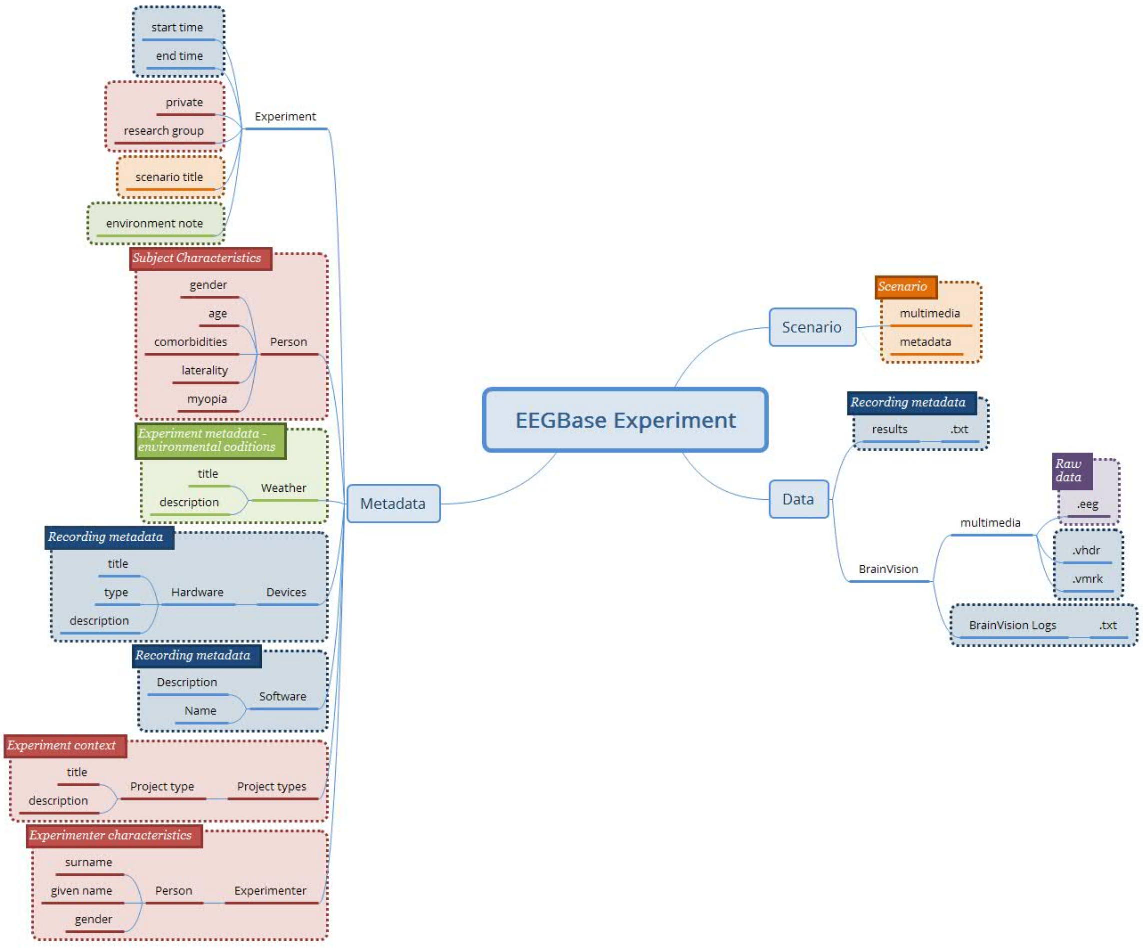


Figure 7.1 The concepts determined from the EEGBase experiment structure (blue boxes - metadata related to the recording itself; purple box - the binary file representing raw brain waveforms; orange boxes - data and metadata related to the experiment/trial scenario; green boxes - the experiment metadata; red boxes - the context of the experiment and the description of the tested subject).

7.2.2 Concepts aggregation and attributes elimination

During this phase the attributes of the same concepts, but of various data representations (EEGBase, NIX, EDF+), were aggregated. The following rules were applied:

- if two or more attributes are identical, then only one is kept,

- single attributes are kept,
- if two or more attributes are semantically identical (i.e. they have the same meaning but different representation), then only one is kept following the resource priority: 1. NIX; 2. EEGBase; 3. EDF+
- unimportant attributes or attributes with no explicit meaning (e.g. the attributes related to specific resource structures like IDs) are removed

The final attribute set is shown in Figure 7.2.

Matching concepts with openEHR reference models

When the processes of the concepts determination and aggregation were completed, the matching of concepts characteristics with the characteristics of openEHR RMs was necessary. openEHR presents five RMs representing the medical entry types (*Observation*, *Evaluation*, *Action*, *Instruction* and *Administrative Entry*), two RMs expressing the entry structure (*Composition* and *Section*) and three RMs characterizing the data structure (*Element*, *Cluster* and *Structure*). Because of the evident EEGBase metadata diversity (recording metadata, experiment metadata, scenario metadata, etc.), the final model had to be composed of multiple archetypes. As an alternative to a data container, openEHR provides so-called RM *Composition*; a *Report Composition* archetype exists in the current CKM. This archetype is perfectly suitable also for EEG/ERP experiment data and metadata. The following archetypes are then connected to the *Report* archetype via an inner reference solution - a so-called slot (Figure 7.3).

1. Problem/Diagnosis (Evaluation RM)
2. Medication order (Instruction RM)
3. Experiment scenario (Cluster RM)
4. EEG/ERP Result (Observation RM)

The *Problem/Diagnosis* archetype (Figure A.8), *Evaluation RM*, that already exists in the CKM was included for the specification of various diagnoses, which are closely related to the EEG/ERP experiment itself and not to the subject's overall health status (e.g. sleep deprivation induced for purposes of the experiment).

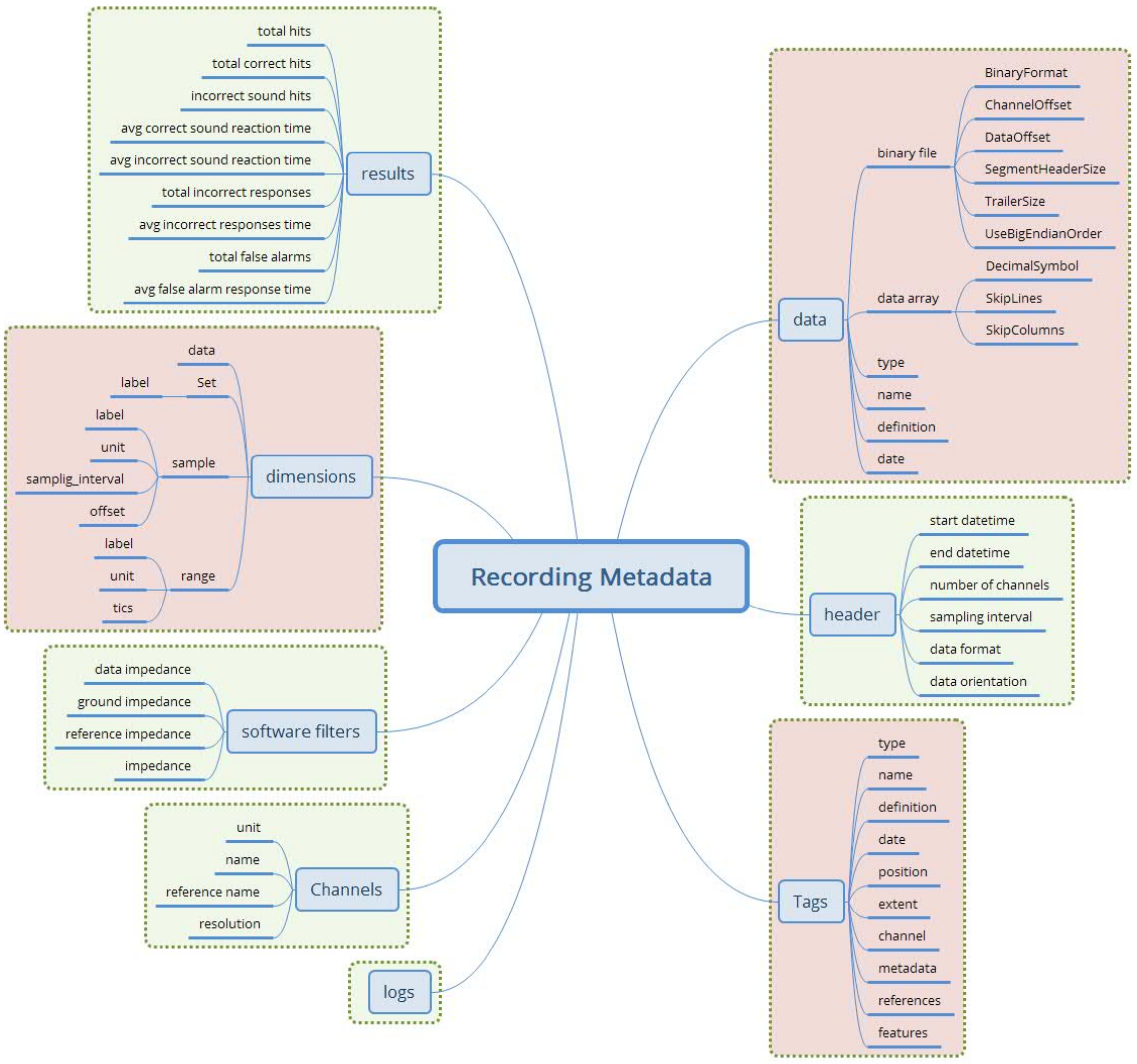


Figure 7.2 The merged recording metadata concept (green - EEGBase/BrainVision attributes; red - NIX attributes). This metadata set contains the attributes taken from the EEGBase and NIX structures. The attributes are aggregated according to the predefined set of rules. No attribute from EDF+ is used.

The *Medication order* archetype (Figure A.7), *Instruction* RM, that is also stored in CKM serves for the description of the medication or other substances that are prescribed/administered to the subject for the purpose of the experiment only, i.e. it excludes the medication prescribed to the subject for any other reasons, the long-term medication included.

The *Cluster* RM archetype is the simplest archetype structure that serves for the tree-like data structure description without any reference to a specific clinical work-flow. Since the

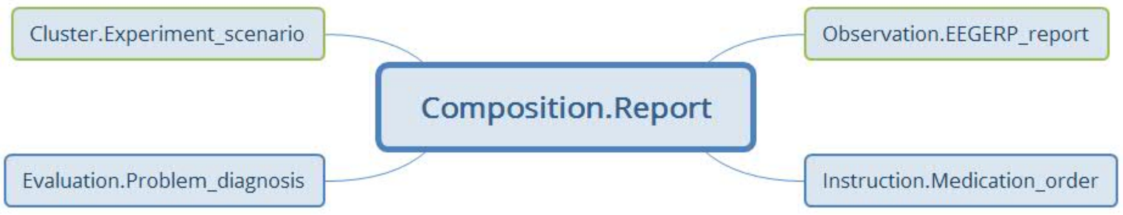

Cluster.Experiment_scenario
Observation.EEGERP_report
Composition.Report
Evaluation.Problem_diagnosis
Instruction.Medication_order

Figure 7.3 The newly proposed (green) and existing (blue) archetypes connected into *Composition* RM.

Experiment scenario has no semantic nor structural relations to other specific RMs (*Medical Entry* RMs), it is designed as a *cluster*.

Finally, data acquisition, i.e. data and recording metadata, corresponds to the *data* and *protocol* part of the *Observation* RM (its characteristics and structure can be seen in Figure 7.4). The *Observation* RM also covers information about time events and the subject's state (it differs from the *Problem/Diagnosis* archetype). Therefore, the *EEG/ERP Result* archetype could include the whole recording concept and become the most complex of all the proposed archetypes.

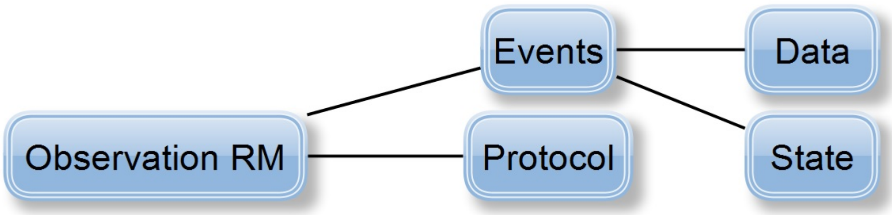


Figure 7.4 Definition of *Observation* RM [3]: *Protocol* attribute describes the way the observation was realized (e.g. used hardware/software). *Events* attribute represents the series of time events during the observation, i.e. a set of events represents the observation history. Each event has its *Data* attribute, i.e. the observation results, and its *State* attribute describing the subject's state related to the observation (e.g. the subject's position - standing/sitting).

Besides the selected concepts and archetypes corresponding to them, two more archetypes describing *Software* (Figure A.9) and *Stimulus* (Figure A.17 and A.18), also missing in the CKM, were determined as necessary. The base for these archetypes was taken from the odML electrophysiological terminology.

Since the odML structure can be expressed in a tree-like form, the *Cluster* RM could be used for any odML structure without any loss of expressive power. Figure 7.5 displays the transformation rules (dotted lines) and transformations of odML sections into openEHR clusters. The odML structure manipulates four basic elements: the *root section*, *section*, *property*, and *value*. Each part also has its attribute set. The root section represents one archetype and its attribute set was directly mapped to the archetype metadata set. Each odML section/subsection can be represented as a datapoint of the type cluster/nested cluster in the archetype body. odML *Section* attributes were transformed into cluster metadata. Datapoints were then constructed from odML properties and their values specify, e.g., data types, definitions, units, or enumerations. Enumerations were transformed into the archetype body as predefined internal codes of the datapoints. While odML has no predefined datatype set, the datatypes used were manually compared and paired (Table 7.1) with the openEHR datatype set (according to the *odML terminology* data meaning and nature).

Table 7.1 odML and openEHR datatypes pairs

odML terminology	**openEHR datatype**	**notes**
Person	N/A	slot for the demographic archetype instances
Date	date time	
Text	text	
Int	count	
String	text	
Float	quantity	
Binary	multimedia	
URL	URI	
Datetime	date time	
Time	date time	in case of start and end time presence at once, those times are merged in an *interval* of datetimes
Boolean	boolean	
2-tuple	N/A	substituted by cluster containing two text/quantity/count types

The mapping could be used for any odML "sections to cluster" archetype transformation. Therefore, the possibility of the further extension of the EEG/ERP archetype composition according to odML sections is ensured. Since the odML format is used for NIX metadata as well as for EEGBase experiment metadata, there is a possibility that new archetypes will be needed in the future. As the odML terminology is machine-readable, the proposed mapping could be used for the development of a tool for automatic odML to ADL transformation. All newly proposed archetypes are presented in detail in section 7.3.

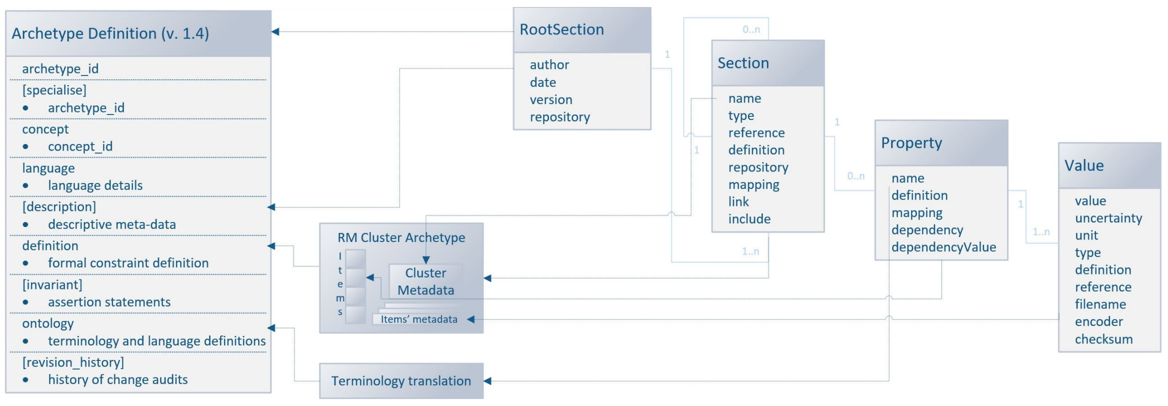


Figure 7.5 Mapping of the odML structure to the openEHR archetype based on *Cluster* RM. The left side of the figure shows the archetype sections: the sections in square brackets are optional; the sections without brackets are mandatory. The right side of the figure shows the odML data model [23] with relations (the light blue lines) between odML elements. The dashed lines show the relations/transformation rules between models. The middle part of the figure refers to the additional steps necessary for the transformation.

7.2.3 Binding archetypes with external terminologies

Referencing terms in odML terminology for electrophysiology

It is also critical to bind archetype datapoints with external terminologies. The *odML terminology for electrophysiology* represents a comprehensive controlled vocabulary. Furthermore, since odML is used within NIX and EEGBase and parts of the terminology serve as a base for recently proposed archetypes (described in Section 7.3), it is suitable to bind archetype terms to the odML terminology. However, the odML terms have no explicitly defined referenceable (and dereferenceable) IDs. Since the terminology is stored in a public repository (https://github.com/G-Node/odml-terminologies) as a set of XML files, a particular term (the XML element the term is being kept in) of the *odML terminology* can be referenced. This reference can be labelled with a unique ID. These IDs/pointers were created in four steps:

1. the file containing the root section corresponding to the particular concept was localized,
2. the XPath (the language for addressing parts of an XML document [13]) query to the specific XML element containing the searched term was constructed,
3. the paths obtained in step 1 and 2 were aggregated into one string,
4. a short unique alias (that serves as an identifier to the string created in step 3) was assigned.

As a running example, the term *Reference*, representing a reference electrode, was selected. The *reference electrode* term is located (from the structural point of view) in the *Electrode* section with the property *Usage* and value *Reference*.

All odML terminology XML files are available in a public repository and accessible via the Uniform Resource Locator (URL). In our example, the URL pointing to the file containing the *Electrode* section is shown in Listing 7.1.

Listing 7.1 The URL of the Electrode section

```
http://portal.g-node.org/odml/terminologies/v1.0/electrode/
    electrode.xml
```

The pointer to the reference electrode can be expressed as it is shown in Listing 7.2; the hash separates the section name *Electrode* from the file URL. The colon separates the property name *Usage* from the section name and the slash separates the property name *Usage* from the enumerated item.

Listing 7.2 Step 1: Pointer to the Reference term in the Electrode odML section

```
http://portal.g-node.org/odml/terminologies/v1.0/electrode/
    electrode.xml#Electrode:Usage/Reference
```

The diagram in Figure 7.6 shows the pointer string construction.

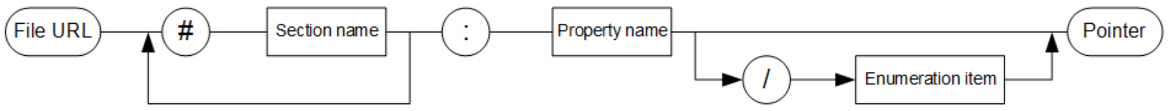


Figure 7.6 The syntax diagram describing the pointer string construction.

In the second step, an XPath string querying the element with the searched term from the selected XML file is constructed. Listing 7.3 shows the XPath query for the running example.

Listing 7.3 Step 2: the XPath query for the term Reference within the electrode odML section

```
odML[@version="1"]/section/property/name[text()="Usage"]/../value[
    text()="Reference"]
```

Within the third step, the XPath query (step 2) and the pointer string (step 1) are merged together. The final string (Listing 7.4) contains the path to the particular file and the XPath query to the particular element. Thus the pointer string can be automatically dereferenced by a machine.

Listing 7.4 Step 3: The merged pointer string and XPath query

```
http://portal.g—node.org/odml/terminologies/v1.0/electrode/
    electrode.xml#/odML[@version="1"]/section/property/name[text()
    ="Usage"]/../value[text()="Reference"]
```

The string from step 3 is not suitable for direct usage inside the archetype. When this string is used as an external terminology code inside the binding section of the archetype, existing tools and libraries like the *Ocean Informatics Archetype Editor* or openEHR Java libraries evaluate the archetype as invalid. For that reason, aliases were created. Each alias consists of the prefix ODMLID, a 3-digit section number and a 3-digit term number. The list of aliases, section numbers, XPath queries, pointer strings, etc. are kept in a separate table, which is a supplemental resource for the archetypes and templates. In our running example, the alias bound with the particular internal archetype code is ODMLID007013.

Mapping of the terms from odML terminology for electrophysiology to Unified Medical Language System Meta-thesaurus

Even though there is no natively supported referencing/dereferencing system for the terms from the odML terminology for electrophysiology, previous section 7.2.3 introduced the system which enables that. However, since the odML terminology is not widely accepted in the medical community, also mapping to existing respected controlled terminologies is useful.

Unified Medical Language System (UMLS) [6] is a unified interface over many controlled medical terminologies including SNOMED-CT, International Classification of Disease 10th Revision (ICD-10), and Clinical Terms Version 3/Read Codes (RCD). UMLS maps semantically equal terms and their codes from various terminologies internally and defines its own codding system overlaying these mapping structures. Thus, UMLS Concept Unique Identifier (CUI) refers to a single term/definition within the meta-thesaurus including many contextual references. Thanks to the mapping structures, UMLS also provides information about concept relations. UMLS meta-thesaurus provides a Representational State Transfer (REST) Application Programming Interface (API). Apart from that, there are UMLS searching tools using Natural Language Processing (NLP) including MetaMap[1] [1].

The complexity of UMLS makes the meta-thesaurus non-trivial to use. Provided REST API enables to search for a term with a knowledge of that term, its CUI or its code in a native coding system. UMLS release version can be also specified as well as a subset of resources included in the search. Searching according to a specific term, which is the only way to

[1]https://metamap.nlm.nih.gov/

search odML terms, also allows users to specify how strict the results can be, i.e. approximate match, and exact match. The approximate match option might cause significant number of results, e.g., there is 898 results for the odML term *electrode*. A search with the exact match option returns four results only.

MetaMap is an NLP tool originally proposed for recognizing UMLS concepts in a free text. Nevertheless, since it supports many of setting parameters it can be utilized for reducing the number of ambiguous search results mentioned above. In case of the *electrode*, MetaMap with appropriate parameters specified returns two results: general definition of electrode and electrode as a device component. That way, all the odML terms were annotated by CUIs.

Current archetype binding system supports 1:1 mapping only. Therefore, it is not possible to link multiple external terms with one internal code, i.e. to specify synonyms. Therefore, current version of the archetypes includes the mapping to the odML terms only.

7.3 Proposed archetypes

7.3.1 EEG/ERP Result archetype

The *EEG/ERP Result* archetype is based on the *Observation* RM, covers the EEG recording and uses the attributes from NIX, EEGBase (the BrainVision EEG format) and EDF+. These attributes are divided into *data* and *protocol* parts and turned into the archetype's datapoints. Figure 7.7 shows the final archetype structure.

The protocol part (Figure A.3) describes the metadata important for data interpretation and recording reproduction. The protocol describes recording conditions and the way the recording was conducted. The *Hardware* and *Electrode* slots are connected with the *Device* archetype (designed by openEHR developers together with the *Device details* archetype; Figures A.11 and A.12) and describe the used hardware and electrodes, respectively. The *Software* slot is connected to the newly created archetype *Software* of the type *Cluster*. The structure of the *Software* archetype is taken from the odML terminology for electrophysiology section of the identical name (Figure 7.8). Finally, the *Environment* slot is connected to the *Environment* (Figure A.14) archetype existing in the CKM. Crucial cardinalities (e.g. at least one electrode per recording is required) are explicitly specified in the archetype body.

The *Person's state* part (Figure A.4) is natively supported by the *Observation* RM. The state is restricted only to the subject's position and textual description. The complex subject's state characteristics can be expressed by the *Problem/Diagnosis* archetype of the *Evaluation* RM within the same *Report Composition*.

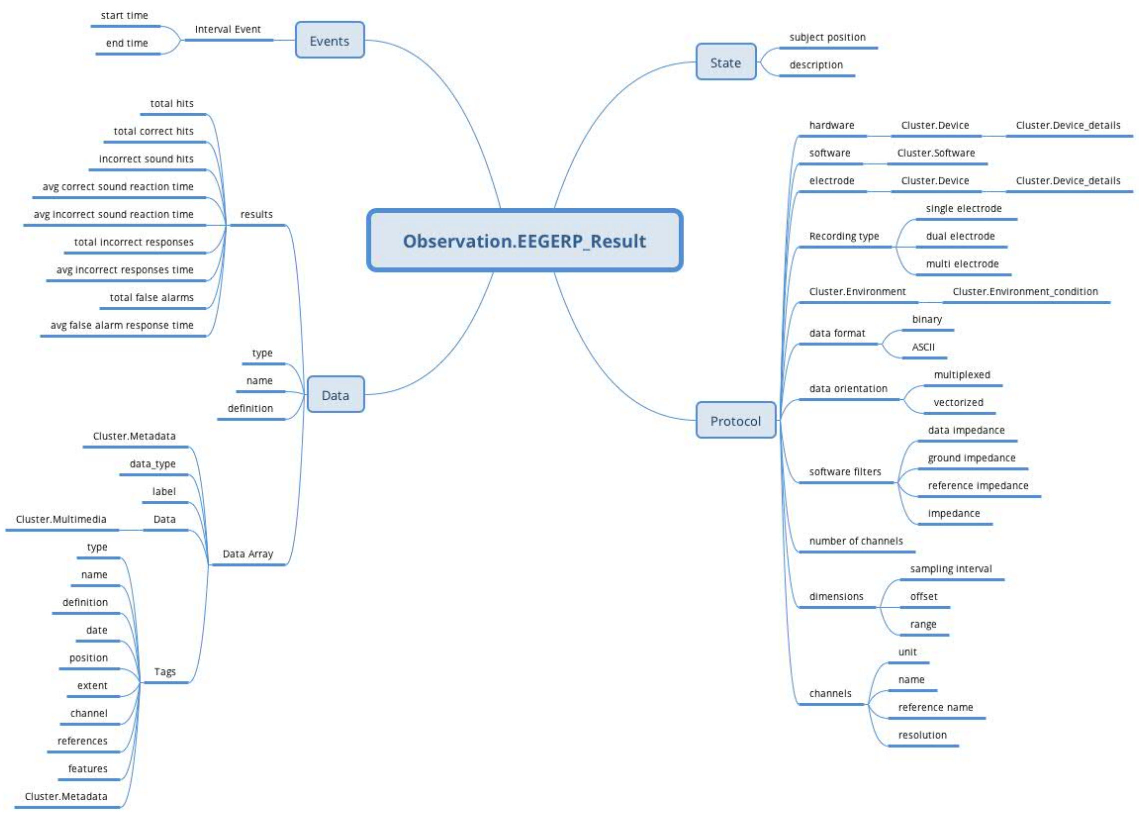


Figure 7.7 *EEG/ERP Result* archetype (*Observation* RM) structure: The archetype contains all attributes (*Protocol*, *Data*, *Events*, *State*) supported by *Observation* RM to cover recording conditions, results, time specifications and subject's position.

The *Events* part, also natively supported by the Observation RM, represents events as time intervals when the recording was done.

Finally, the *Data* part (Figure A.2) is composed of two major branches. The *Data Array* branch allows the storage of binary data (e.g. in a BrainVision .eeg file) as complex structures compatible with NIX. The datapoint structure is designed primarily according to NIX and extended with some EEGBase (BrainVision) attributes. The *Results* branch provides a summary of recording results. As this part is based on the EEGBase experiment attributes, it contains the attributes closely related to the EEG/ERP experiments only (e.g. statistics of stimuli events).

7.3.2 Experiment scenario archetype

The *Experiment scenario* archetype (Figure 7.9 and A.5) based on the RM *Cluster* is derived from the EEGBase experiment attributes dealing with a scenario description. The *stimulus*

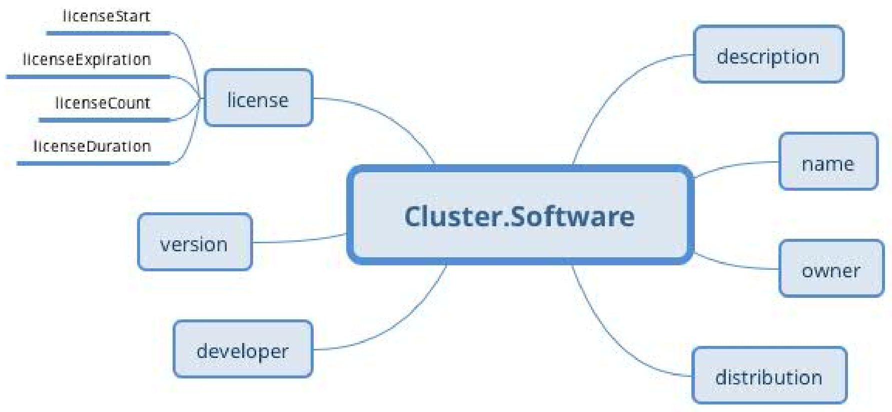


Figure 7.8 *Software* archetype (*Cluster* RM) structure: The archetype contains basic software information identical to that of the odML terminology for electrophysiology.

slot is connected to the *Stimulus* archetype (not presented for the huge amount of datapoints), which is derived from the odML section of the identical name. While odML distinguishes between various stimuli types, the archetype aggregates all datapoints into one structure and a particular stimulus can be determined using templates (see section 7.3.5).

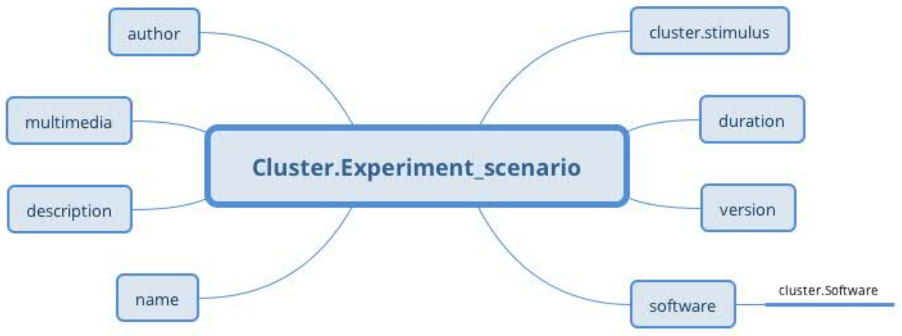


Figure 7.9 *Experiment scenario* archetype (*Cluster* RM) structure: The archetype contains general information about the scenario, the original scenario files as a multimedia attachment, information about software in which the scenario was built (via *Software* archetype), and information about used stimulus, if needed (via *Stimulus* archetype).

7.3.3 Problem/Diagnosis archetype

The *Problem/Diagnosis* archetype based on the *Evaluation* RM is used as published in the CKM. The archetype is designed to describe a single health issue that impacts the physical, mental and/or social well-being of the subject. In the EEG/ERP experiment, it allows us to specify the experimental condition of the subject only. While health issues are described in an unstructured format, additional information (e.g. date/time of resolution, severity) is structured.

7.3.4 Medication order archetype

The *Medication order* archetype based on the *Instruction* RM is used as published in the CKM. The archetype is designed to describe a single item administered to the subject. In the EEG/ERP experiment, it allows us to specify the medications given to the subject for the purpose of the experiment only. Besides the item identification, the archetype allows us to specify the medication order in more than 20 datapoints.

7.3.5 openEHR templates

Since archetypes are mainly used via templates, some testing templates covering the selected types of EEG/ERP experiments were created. As these templates are too large to visualize, only template for the *developmental coordination disorder in children* experiment is shown in Figures 8.1, 8.2, 8.3, and 8.4.

Also the templates focusing directly on stimuli were proposed according to the odML terminology. These templates reduce the original set of the *Stimulus* archetype datapoints to a subset according to the particular stimulus type characteristics (e.g. Movie, Pulse or Ramp). As in case of experiment templates a graphical representation is too large to be presented here.

Part V

Results

Chapter 8

Results

This chapter presents the results of the thesis as follows: (i) a summary of the archetypes created in chapter 7; (ii) an example of the real experiment taken from EEGBase expressed in archetype-based structure; (iii) a position of the proposed archetypes in the semantic hierarchy; (iv) an evaluation of the results against the selected desiderata.

8.1 Summary of created archetypes

Chapter 7 described a creation process of openEHR archetype composition describing experiments in EEG/ERP domain. Within that process five new archetypes were created and six were reused. Table 8.1 presents a list of all the used and proposed archetypes. Figures of all proposed archetypes visualized in Ocean Informatics Archetype Editor[1] and corresponding odML sections where necessary are shown in Appendix 1, Figures A.1 - A.30. New archetypes were passed to the incubator within the CKM.

Aside from the archetypes, templates for various EEG/ERP experiments from EEGBase (e.g. developmental coordination disorder in children or driver's attention) and templates for particular stimuli were also designed. All templates were created and visualized in Ocean Informatics Template Designer[2]. However, only one template is presented here in Figures 8.1, 8.2, 8.3, and 8.4 due to size of templates graphical representations.

[1]http://openehr.org/downloads/archetypeeditor
[2]http://openehr.org/downloads/modellingtools

Table 8.1 List of the archetypes covering the EEG/ERP domain

Archetype	**Reference Model**	**Status**
Report	Composition	CKM
Medication Order	Instruction	CKM
Problem/Diagnosis	Evaluation	CKM
EEG/ERP Result	Observation	New
Experiment Scenario	Cluster	New
Software	Cluster	New
Stimulus	Cluster	New
Device	Cluster	CKM
Device details	Cluster	CKM
Environment	Cluster	New
Environmental Conditions	Cluster	CKM

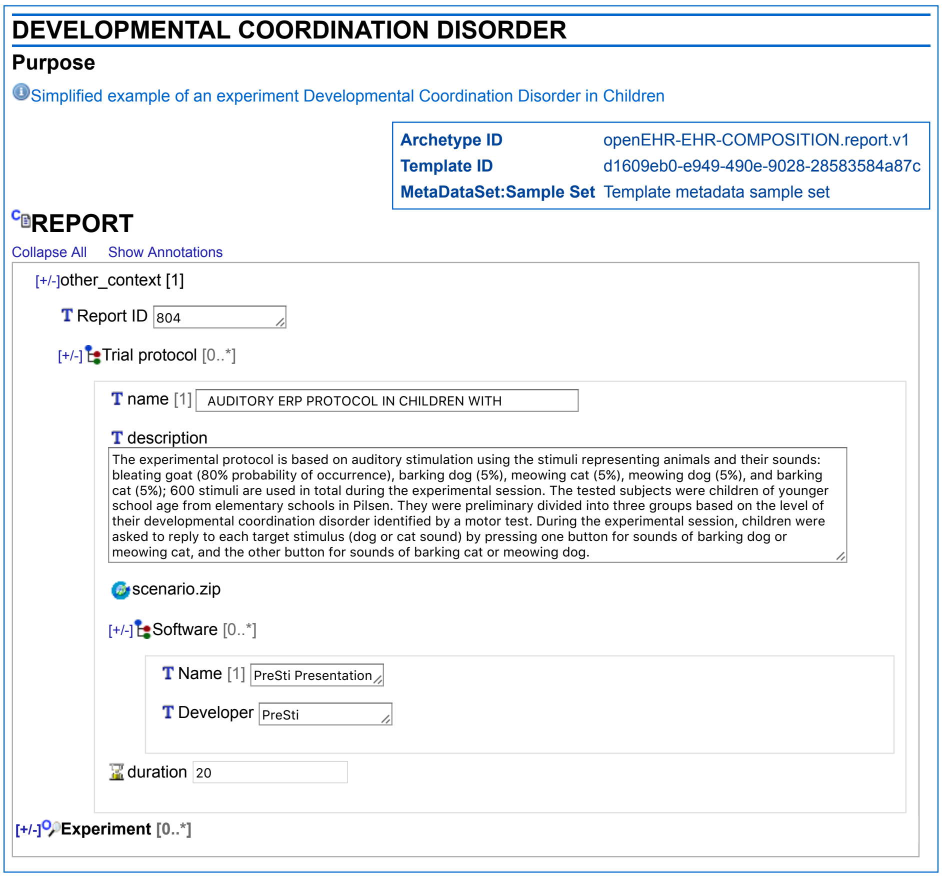
8.2 Exemplary EEG/ERP experiment described in an archetype-driven way

To present how the archetypes cover the domain, an exemplary experiment from EEGBase focused on a developmental coordination disorder in children, was manually mapped on the archetype datapoints. Figures 8.1, 8.2, 8.3, and 8.4 illustrate the simplified experiment, using one software, one part of hardware set-up, one electrode definition, and one data tag. Figures show the experiment content inserted into the form generated from the template specified over the new archetypes.

The experiment focused on the developmental coordination disorder in children was selected as a one of the most complex experiments in EEGBase containing a binary protocol/scenario, separated experiment results, and wide spectrum of odML sections in its metadata set. As it is seen, the used archetype set and corresponding template cover the experiment data and metadata.

8.3 openEHR archetypes in the semantic hierarchy

The components of the openEHR solution, i.e. reference models, archetypes, templates, and external terminologies, overlay multiple layers of the semantic hierarchy.



DEVELOPMENTAL COORDINATION DISORDER

Purpose

Simplified example of an experiment Developmental Coordination Disorder in Children

Archetype ID	openEHR-EHR-COMPOSITION.report.v1
Template ID	d1609eb0-e949-490e-9028-28583584a87c
MetaDataSet:Sample Set	Template metadata sample set

REPORT

Collapse All Show Annotations

[+/-]other_context [1]

T Report ID 804

[+/-] Trial protocol [0..*]

T name [1] AUDITORY ERP PROTOCOL IN CHILDREN WITH

T description

The experimental protocol is based on auditory stimulation using the stimuli representing animals and their sounds: bleating goat (80% probability of occurrence), barking dog (5%), meowing cat (5%), meowing dog (5%), and barking cat (5%); 600 stimuli are used in total during the experimental session. The tested subjects were children of younger school age from elementary schools in Pilsen. They were preliminary divided into three groups based on the level of their developmental coordination disorder identified by a motor test. During the experimental session, children were asked to reply to each target stimulus (dog or cat sound) by pressing one button for sounds of barking dog or meowing cat, and the other button for sounds of barking cat or meowing dog.

scenario.zip

[+/-] Software [0..*]

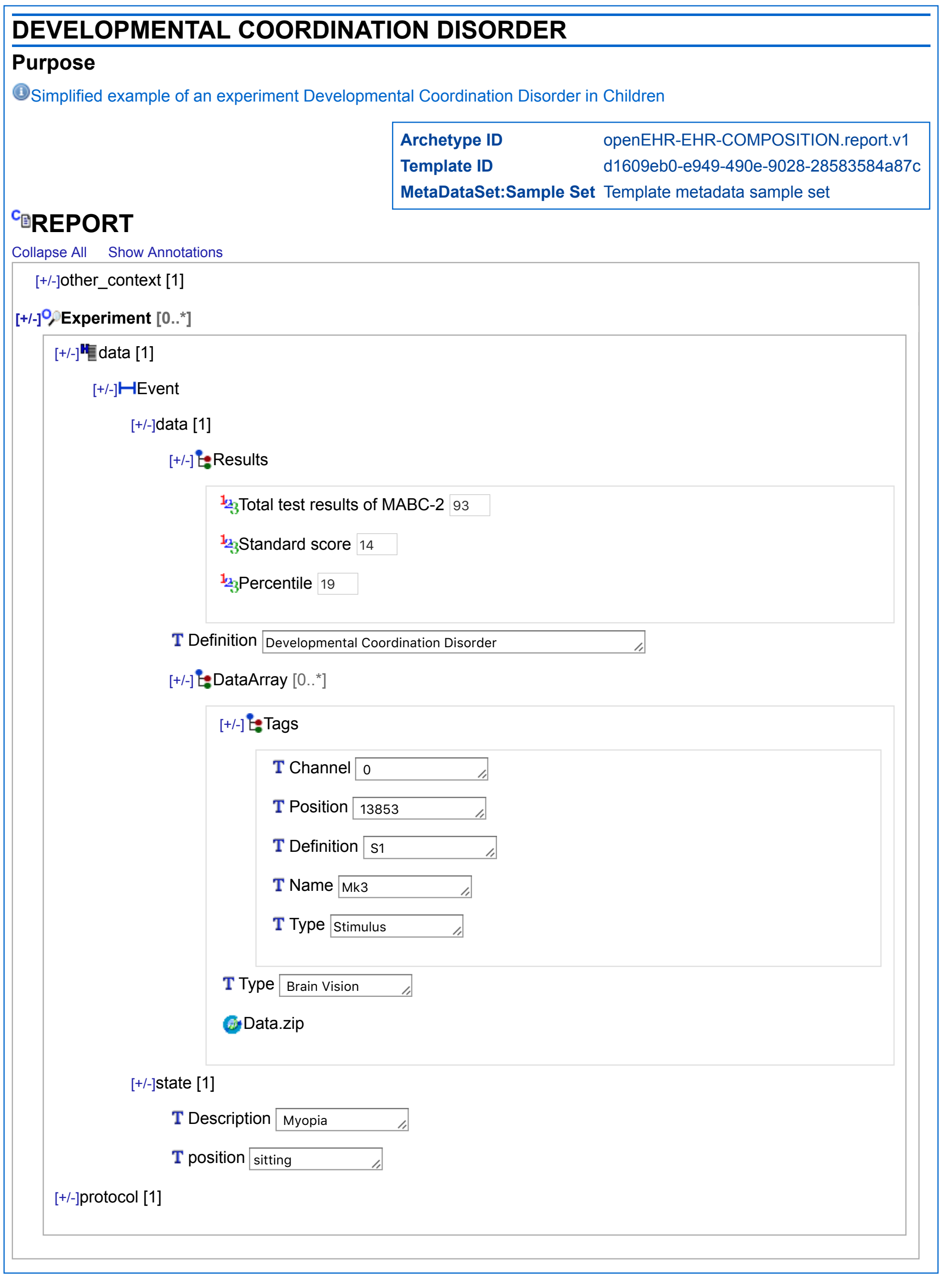
T Name [1] PreSti Presentation

T Developer PreSti

duration 20

[+/-] Experiment [0..*]

Figure 8.1 Web form generated from the template dedicated to experiments focused on the developmental coordination disorder in children. The form contains part of real trial protocol/scenario data from EEGBase.



DEVELOPMENTAL COORDINATION DISORDER

Purpose

Simplified example of an experiment Developmental Coordination Disorder in Children

Archetype ID	openEHR-EHR-COMPOSITION.report.v1
Template ID	d1609eb0-e949-490e-9028-28583584a87c
MetaDataSet:Sample Set	Template metadata sample set

REPORT

Collapse All   Show Annotations

[+/-]other_context [1]

[+/-]Experiment [0..*]

[+/-]data [1]

[+/-]Event

[+/-]data [1]

[+/-]Results

Total test results of MABC-2 93

Standard score 14

Percentile 19

Definition Developmental Coordination Disorder

[+/-]DataArray [0..*]

[+/-]Tags

Channel 0

Position 13853

Definition S1

Name Mk3

Type Stimulus

Type Brain Vision

Data.zip

[+/-]state [1]

Description Myopia

position sitting

[+/-]protocol [1]

Figure 8.2 Web form generated from the template dedicated to experiments focused on the developmental coordination disorder in children. The form contains part of real experiment data from EEGBase. For a demonstration purpose, one data tag is shown.

8.3.1 Dictionary

odML terminology and/or the selected UMLS terms reside at the very bottom of the hierarchy as the dictionary. This serves for any model in the hierarchy derived from the archetypes.

8.3.2 Entity-relational model

From the perspective of openEHR solution, ER model lies at the lower level of the semantic hierarchy. However, as the archetypes do not contain any encapsulation, methods, nor dependencies, the loss of expressiveness can potentially influence only multiplicity and generalization/specialization features. OO model multiplicity can be simplified according to rules specified in Table 5.3. As the generalization/specification is not supported, openEHR multi-layer approach has to be simplified, thus, three basic approaches can be determined:

1. Reference model - centric: The reference model is completely represented by the ER diagram. No archetype is modelled directly as a table nor any datapoint as a column. Thus, the archetypes are mapped on a "network of foreign keys" carried by reference model tables. Then, EHR data values are stored in one or few "data tables" and the EHR is reconstructed from the relationships between reference model tables. Such an approach is flexible and any type of archetype and its data can be stored in that database. Another advantage is its constant number of tables and immutable model structure. However, the potential number of relationships and unstructured data values in data tables would decrease the performance of the database. Additional implementation of templates would bring even more relationships and queries composing (i.e. select) and decomposing (i.e. insert, update, delete) data according to particular template would be extremely complex.

2. Archetype-centric: For each archetype a set of tables corresponding to particular reference model (e.g. for Observation RM it is data, protocol, etc.) is created. The tables represent archetypes and their values are fragments of stored EHRs. openEHR templates are represented as the views, virtual tables, where the relationships between various archetype tables are handled by additional auxiliary table. Stored data is well structured and in case of a small amount of archetypes used, the performance of the database is not negatively affected. However, with each newly added archetype the database structure must be changed and new tables added. Also the datapoins with a cardinality higher than one must be stored in a separate table. Thus, in case of a complex EHR system, the amount of tables would be unmanageable and that would have an negative impact on sustainability.

3. Template-centric: Relational tables are designed according to a specific template. This approach is thrifty with relationships. However, information about archetype structures is completely lost, i.e. data related to a specific archetype cannot be recognized and selected from the template data, because all data is part of one template. Moreover, for each template a new set of tables must be created which could cause data redundancy (e.g. Device archetype would be part of more than one template and thus information about device is scattered across many tables).

Overall complexity, efficiency and flexibility of ER model strongly depends on the chosen approach and granularity. Since this section discusses the models for experimental EEG/ERP data, the second approach was selected. Due to the size of the diagram, only a section is shown in Figure 8.5. The first approach would be useful in case of complex openEHR-driven system, where the significant amount of archetypes used is expected (e.g. the system described in section 9.1). The third approach was not fitting for its information loss.

The outcome of the transformation of openEHR model into ER model is a model with a unified terminology and specified table structures and relationships between them. There are two main drawback of the created model:

1. Data model suffers from loss of information when compared to original openEHR model.

2. Data model is not flexible in the matter of structural changes. The lack of flexibility disrupts one of the desiderata described in section 8.4.

8.3.3 Object-oriented model

Reference models are natively designed as the OO models. Archetypes can be considered as the classes inherited from the classes of RM and thus they represent a part of RM OO model. Templates, natively described in XML, semantically stand slightly above the OO model (described in section 4.3). However, they can also be considered as the classes derived from the archetypes. There is an important difference in the *RM classes - archetype classes* relationships and *archetype classes - template classes* relationships. While one archetype class is inherited from one RM, some template classes might require multiple inheritance if the templates cover more than one archetype. Since multiple inheritance is not being supported by many OO-driven technologies, templates can be expressed as classes containing collections of instances of various archetype classes.

8.3.4 Ontology

Similarly to the components of openEHR solution, ontology structure can be decomposed to generic conceptual structure (RM), specific classes definition (archetypes) and additional logic specification (templates). A comparable approach is used and discussed in [60].

Regarding to a conceptual structure, a reference model can be translated to OWL as follows:

- For each class in the RM a new OWL class is created
- For each relationship in the RM a new OWL Object property is created
- For each cardinality related to the relationship a new OWL restriction is created
- For each attribute in the RM classes a new OWL Datatype property is created

A transformation of RM and ADL archetype structures into OWL is discussed in [19], [37], [24], or [41].

A transformation of an archetype ontology section into OWL requires a proper mapping to external resources (where possible) in order to sustain the main ontology philosophy, i.e. re-usability. A derivation of a proper ontological description from the current terminological resources is in the scope of the OEN project [10], [40]. As it was described in section 3.3.3, selected resources, including odML terminology, are processed by Ontofox[3] tool which utilizes the MIREOT principle. Ontofox searches for the provided terms in the selected ontology, extracts the minimum information needed to reference these terms and serializes the output into a new RDF graph.

Finally, the additional logic based on the openEHR templates, best practices, consultations with domain experts, etc., must be added manually. OEN development is still in the scope of future work.

8.4 Evaluation of openEHR archetypes with the selected desiderata

The abilities of openEHR to meet the desiderata for computable representations of EHR-driven phenotypes were evaluated in [61]. The desiderata for EEG data format create a subset of original phenotyping desiderata, therefore, the evaluation is similar.

[3]http://ontofox.hegroup.org/

8.4.1 Structured data in queryable form

The main goal of openEHR is to provide a framework in order to keep stored EHRs structured. Therefore, the desideratum on structured data is supported from the nature of openEHR philosophy. openEHR also specifies proprietary archetype query language (AQL) developed for searching clinical data found in archetype-based EHRs. The AQL grammar is specified, however, there is a lack of AQL-based search engines. Additionally, since there is no technological restriction on openEHR-based systems, the technologies with implemented query engines (relation databases supporting SQL queries, noSQL databases, etc.) can be used, and thus the data is in a queryable from.

8.4.2 Common and flexible data model

As described above, openEHR implementation does not rely on a specific technology. Therefore, the desideratum on a common and flexible data model depends on the abilities of openEHR as well as the abilities of implemented underlying technologies. Generally, openEHR supports a common and flexible data model.

8.4.3 Human readable and computable representations

openEHR archetypes support both human readable and computable representations. Individual archetypes are defined in machine-readable ADL and individual templates are stored as XML. Provided XSD allows automated template validation and transformation into human-readable forms via Extensible Stylesheet Language Transformations (XSLT)[4].

8.4.4 Standardized and re-usable terminology

openEHR archetypes support a binding mechanism to pair internal codes with external standardized controlled clinical terminology/ontologies (e.g. SNOMED-CT, odML terminology for electrophysiology, OEN). Additionally, the CKM repository supports archetype sharing and re-usability.

8.4.5 External data and interfacing

openEHR provides a Java-based API for archetype interfacing and handling. Via this API, also external data and libraries can be potentially implemented.

[4]https://www.w3.org/TR/xslt

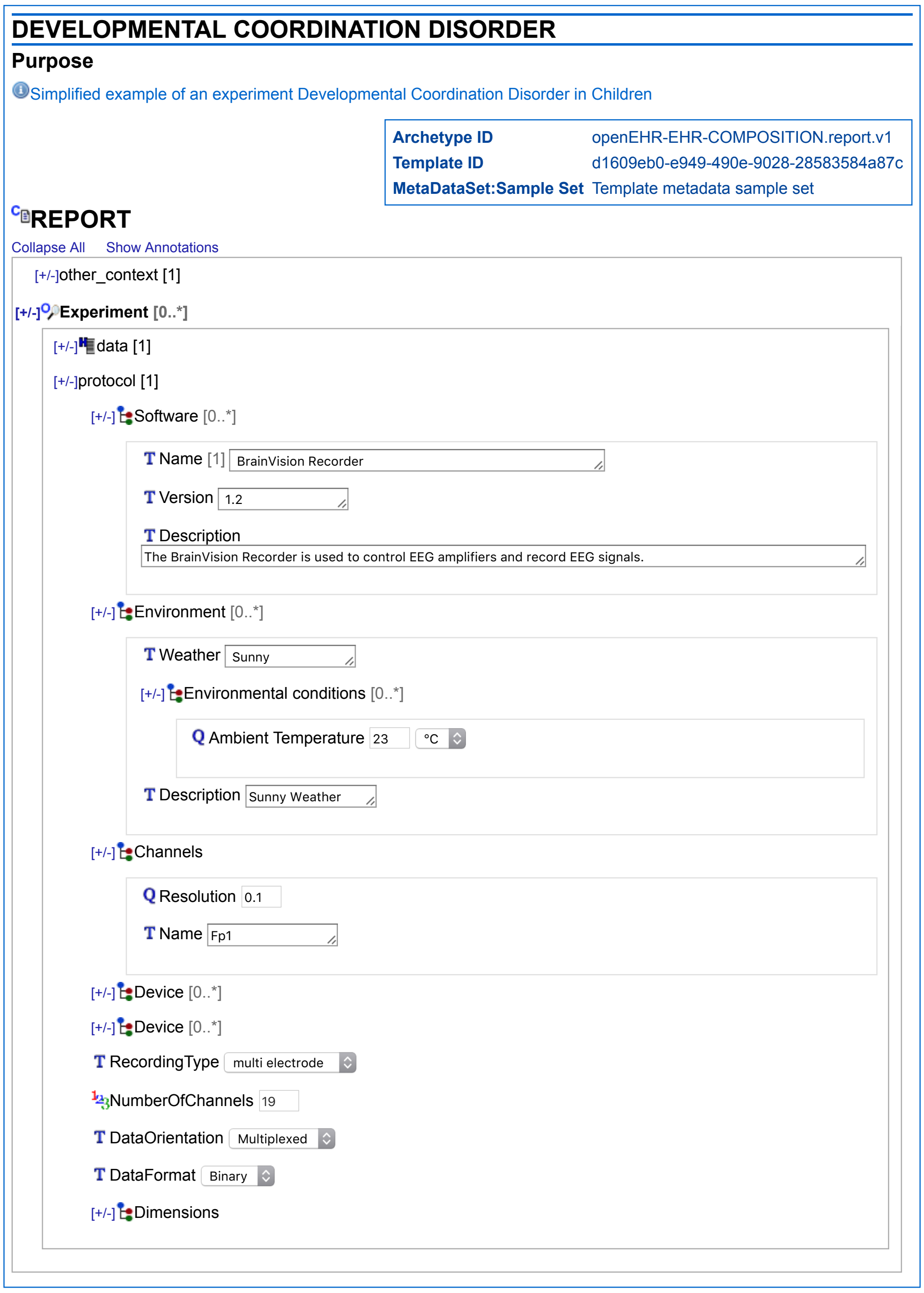
8.4.6 Backwards compatibility

openEHR archetypes in ADL are backward compatible (ADL v2 to v1.4). Backward compatibility is also increased by binding with external controlled terminologies. Finally, ADL as a plain-text mark-up language enables archetypes to be stored in a revision control systems (e.g. git, svn), which can facilitate the tracking of changes during archetype development.

Table 8.2 shows a summary how openEHR archetypes met the desiderata.

Table 8.2 The ability of openEHR archetypes to meet the selected desiderata for EEG/ERP data format

Desideratum	**Evaluation**
Structured data in queryable form	Supported
Common and flexible data model	Not ensured
Human-readable and computable representation	Supported
Standardized nomenclature	Supported
External interfacing	Supported
Backward compatibility	Supported



DEVELOPMENTAL COORDINATION DISORDER

Purpose

Simplified example of an experiment Developmental Coordination Disorder in Children

Archetype ID	openEHR-EHR-COMPOSITION.report.v1
Template ID	d1609eb0-e949-490e-9028-28583584a87c
MetaDataSet:Sample Set	Template metadata sample set

REPORT

Collapse All Show Annotations

[+/-]other_context [1]

[+/-]Experiment [0..*]

[+/-]data [1]

[+/-]protocol [1]

[+/-]Software [0..*]

T Name [1] BrainVision Recorder

T Version 1.2

T Description
The BrainVision Recorder is used to control EEG amplifiers and record EEG signals.

[+/-]Environment [0..*]

T Weather Sunny

[+/-]Environmental conditions [0..*]

Q Ambient Temperature 23 °C

T Description Sunny Weather

[+/-]Channels

Q Resolution 0.1

T Name Fp1

[+/-]Device [0..*]

[+/-]Device [0..*]

T RecordingType multi electrode

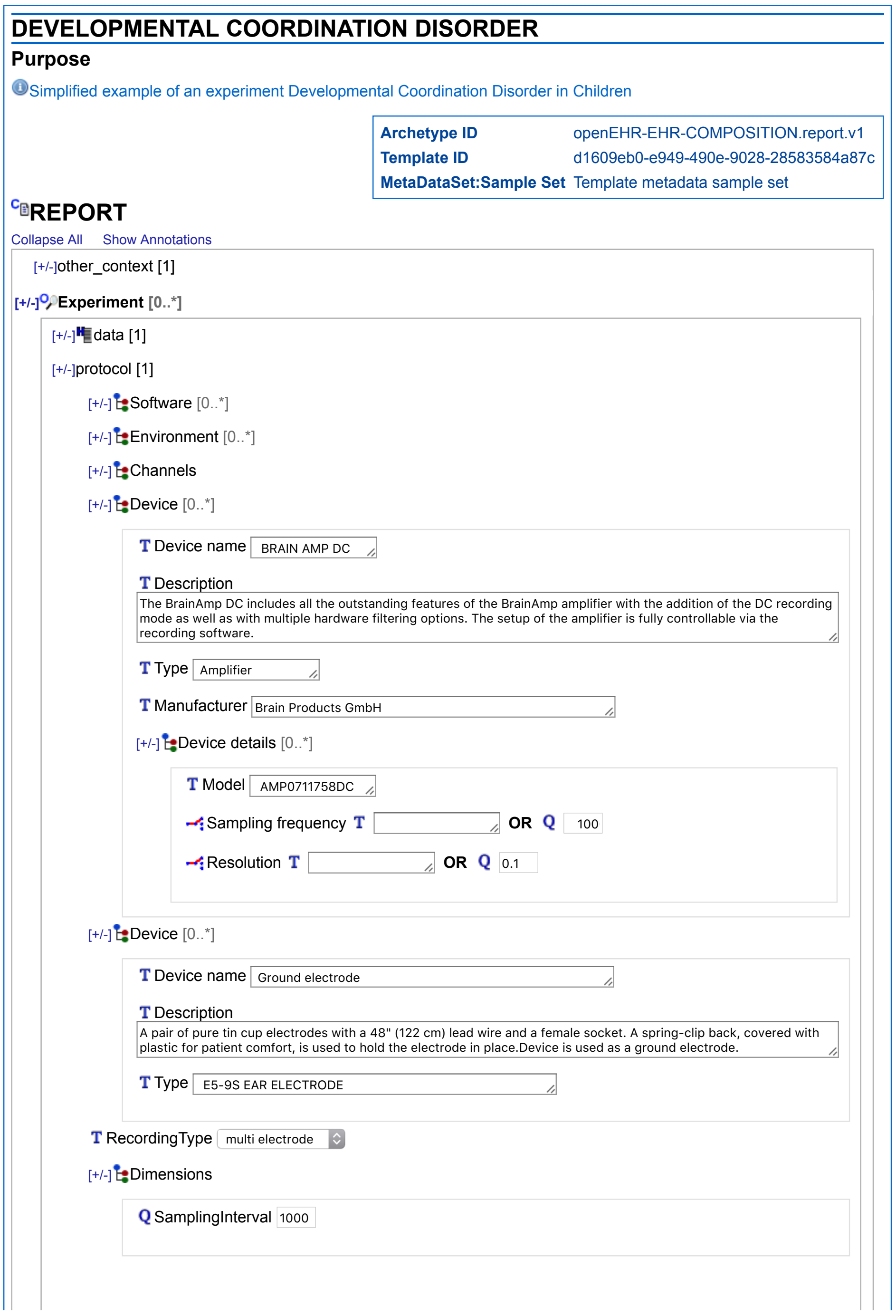
NumberOfChannels 19

T DataOrientation Multiplexed

T DataFormat Binary

[+/-]Dimensions

Figure 8.3 Web form generated from template dedicated to experiments focused on the developmental coordination disorder in children. The form contains part of real experiment protocol data from EEGBase (part 1). For a demonstration purpose, one software and one channel are shown.



DEVELOPMENTAL COORDINATION DISORDER

Purpose

Simplified example of an experiment Developmental Coordination Disorder in Children

Archetype ID	openEHR-EHR-COMPOSITION.report.v1
Template ID	d1609eb0-e949-490e-9028-28583584a87c
MetaDataSet:Sample Set	Template metadata sample set

REPORT

Collapse All Show Annotations

[+/-]other_context [1]

[+/-]**Experiment** [0..*]

[+/-]data [1]

[+/-]protocol [1]

[+/-]Software [0..*]

[+/-]Environment [0..*]

[+/-]Channels

[+/-]Device [0..*]

T Device name: BRAIN AMP DC

T Description: The BrainAmp DC includes all the outstanding features of the BrainAmp amplifier with the addition of the DC recording mode as well as with multiple hardware filtering options. The setup of the amplifier is fully controllable via the recording software.

T Type: Amplifier

T Manufacturer: Brain Products GmbH

[+/-]Device details [0..*]

T Model: AMP0711758DC

Sampling frequency T OR Q 100

Resolution T OR Q 0.1

[+/-]Device [0..*]

T Device name: Ground electrode

T Description: A pair of pure tin cup electrodes with a 48" (122 cm) lead wire and a female socket. A spring-clip back, covered with plastic for patient comfort, is used to hold the electrode in place.Device is used as a ground electrode.

T Type: E5-9S EAR ELECTRODE

T RecordingType: multi electrode

[+/-]Dimensions

Q SamplingInterval: 1000

Figure 8.4 Web form generated from the template dedicated to experiments focused on the developmental coordination disorder in children. The form contains part of real experiment protocol data from EEGBase (part 2). For a demonstration purpose, one hardware device and one electrode are shown.

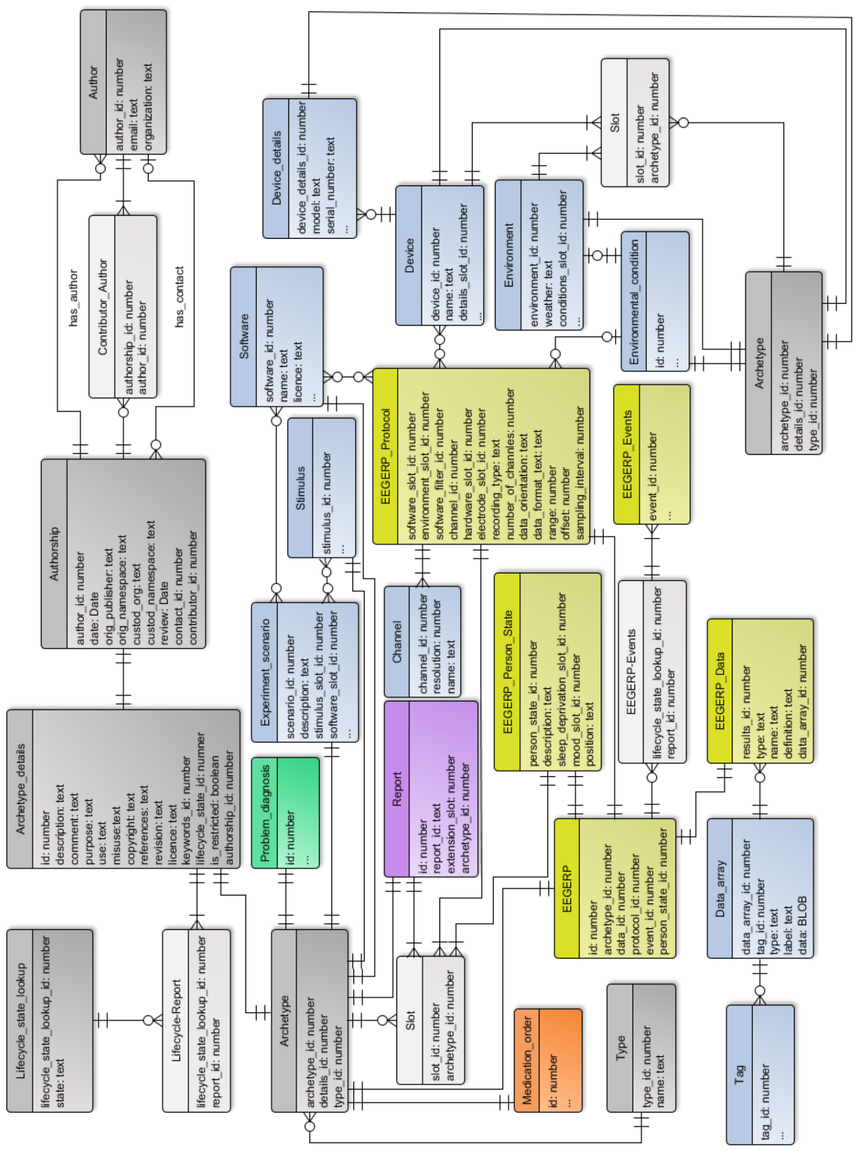


Figure 8.5 Entity-relational model derived from the openEHR archetype set for EEG/ERP experiments with archetype-centric approach; light grey - auxiliary tables, dark grey - generic archetype data, yellow - *EEG/ERP Results* archetype (RM Observation) tables, violet - *Report* archetype (RM Composition) table, green - *Problem diagnosis* archetype (RM Evaluation) table, orange - *Medication order* archetype (RM Instruction) table, blue - various cluster tables representing stand-alone cluster-based archetypes as well as clusters which stand as datapoints in other archetypes. The diagram shows only a selection of the original size. *Archetype* and *Slot* tables are shown twice in the diagram for the clarity of the figure and to avoid intersecting the relationship lines.

Chapter 9

Discussion

This chapter discusses the application of the results as follows: (i) openEHR-driven system for research use as a first use-case of the proposed archetypes; (ii) potential of the archetypes in the clinical sphere.

9.1 openEHR-driven Electronic Health Record system

As a follow-up project to the EEGBase an EHR system for research use was proposed and started being developed. The system is designed to provide universal approach to store health and medical data, which cannot be stored in EEGBase. Aside from this, the system is a proof-of-concept for an openEHR-driven storage for EEG/ERP data.

Recently, our research group expanded to other electrophysiology subdomains (e.g. ECG or EMG). However, EEGBase which effectively stores experimental data and metadata from the EEG experiments, is not universal to store any types of electrophysiology data/metadata (section 3.3.1).

9.1.1 EEGBase data/metadata limitations

Storing additional electrophysiology data in EEGBase, including ECG, can be handled in following ways:

Stand-alone ECG experiment could by stored in EEGBase in a same way as stand-alone EEG experiments do. Only minor changes in the EEGBase database model are necessary. Since EEG experiment metadata is stored in a schema-independent noSQL database, ECG experiment metadata can be stored identically. EEG data is stored in schema-dependent relational database in the native format for the measurement device. Thus, ECG data in a native format for the measurement device can be also stored in the EEGBase data model. The limitation comes when the ECG experiment is supposed to be part of EEG experiment.

ECG as a part of EEG metadata set can be stored in the noSQL document database with other EEG metadata in its native, typically binary, format. ECG metadata must be stored as a part of the same noSQL *document* as the binary ECG data. Computable abilities of ECG recording are limited since the recording is a part of the metadata set related to superior EEG experiment.

9.1.2 Main philosophy

The aforementioned facts established the concept of an EHR-driven system for research use fulfilling the following requirements:

1. The system is user-centric.
2. The system is designed for a personal and research, not clinical, use.
3. The system is universal to store almost any type of bio-data.
4. Stored data is annotated by controlled ontologies/terminologies.
5. The system supports storage of both medical and "fitness" data.
6. The system supports an analysis interface over the cross-domain data, i.e. a border between data and metadata is flexible from the perspective of the analysis.

Since openEHR concept addresses many of these challenges, especially point four and six, the system is proposed as openEHR-driven. The archetype definitions represent a knowledge core of the system. A user interface and data storage adapt dynamically to the structure of currently used archetype/template. Abstract architecture of the system is shown in Figure 9.1.

9.1.3 Roles and use-cases

The basic user roles were specified:

1. Subject
2. Researcher
3. Administrator

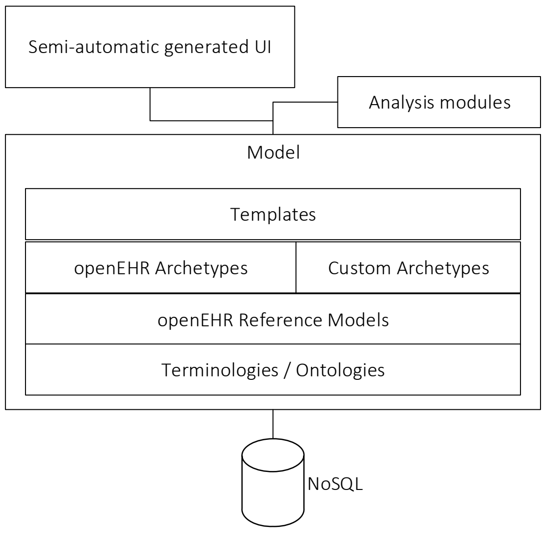


Figure 9.1 An architecture of the openEHR-driven EHR information system

Current version of API prototype implements two roles: administrator and subject. The main use-cases are shown in Figure 9.2.

9.1.4 Architecture and technologies

REST Server

The system is designed as a Java REST server application built on Jersey[1] framework running in Apache Tomcat container[2] using openEHR Java libraries [3] handling the archetypes, i.e. ADL parsing, OET parsing, archetype serialization, etc. An overall architecture is shown in Figure 9.3, an architecture of the REST server application is shown in Figure 9.4

Persistent storage

The first prototype utilized Elasticsearch[4] noSQL database and search engine, which supports a so-called *mapping*. The *mapping* specifies a schema over one or more document types and thus provides partial validation of inserted documents, e.g., EHRs. An *index* was created for each openEHR RM within the database *cluster* and corresponding *node*. A *mapping* specifying particular document *type* was created for each archetype in the system and

[1]https://jersey.java.net/
[2]http://tomcat.apache.org/
[3]https://github.com/openEHR/java-libs
[4]https://www.elastic.co/products/elasticsearch

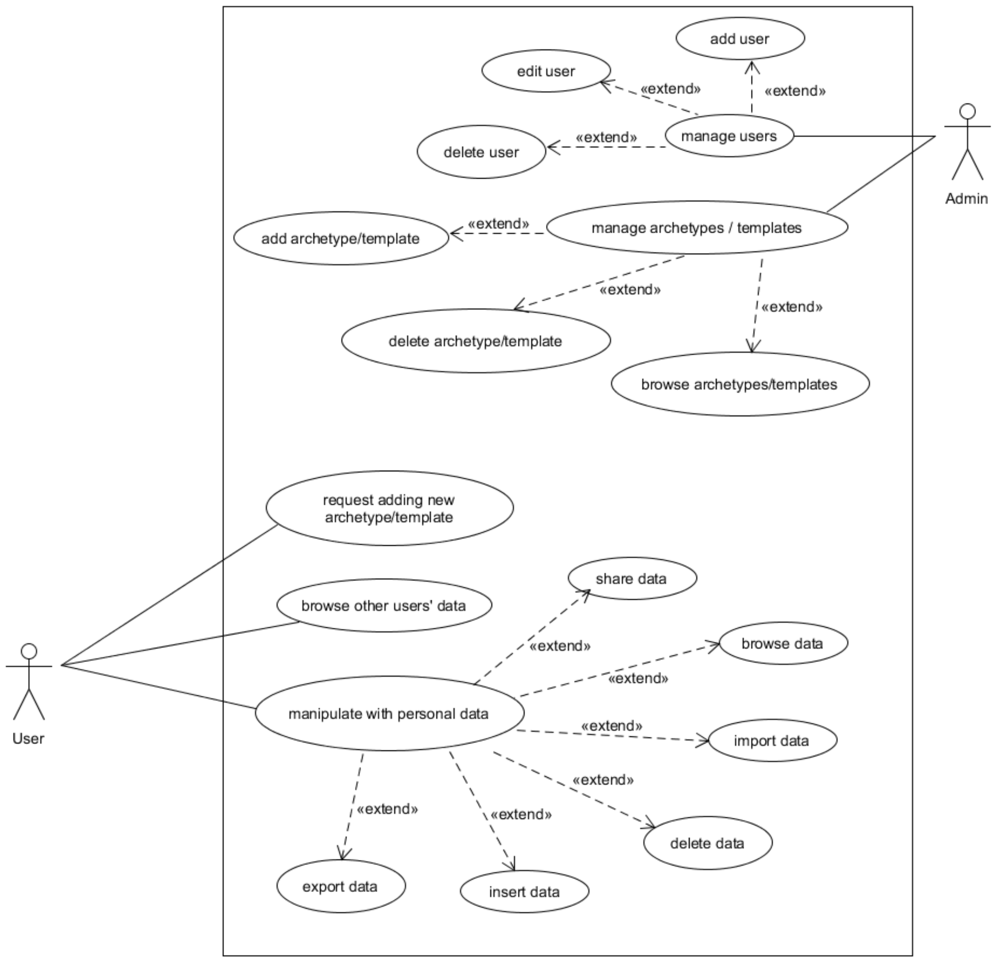


Figure 9.2 Typical use-cases of the openEHR-driven EHR information system (https://github.com/ghessova/SemanticEHR/wiki)

associated with corresponding *index*. Testing EHRs were inserted as JSON documents and the data fields were annotated by the archetype datapoint ID.

Elasticsearch provides a great scalability and supports document schema-based validation via *mappings*, which is an uncommon feature in the noSQL world. Additionally, new *types*, *mappings* and *indices* can be added on-the-fly without required outage of the system. However, the Elasticsearch is designed primarily as a search engine and not as a persistent storage. Due to that fact, there are some crucial limitations including absence of a transaction mechanism. Since the data integrity is essential when speaking of sensitive user data, the current development version utilizes a Mongo DB[5] instead of Elasticsearch.

[5]https://www.mongodb.com/

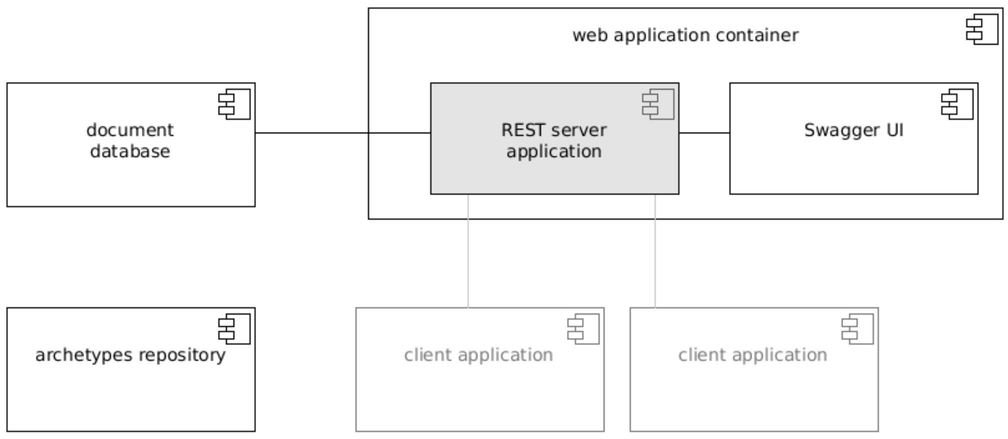


Figure 9.3 An overall architecture of the proposed EHR system (https://github.com/ghessova/SemanticEHR/wiki/Architecture)

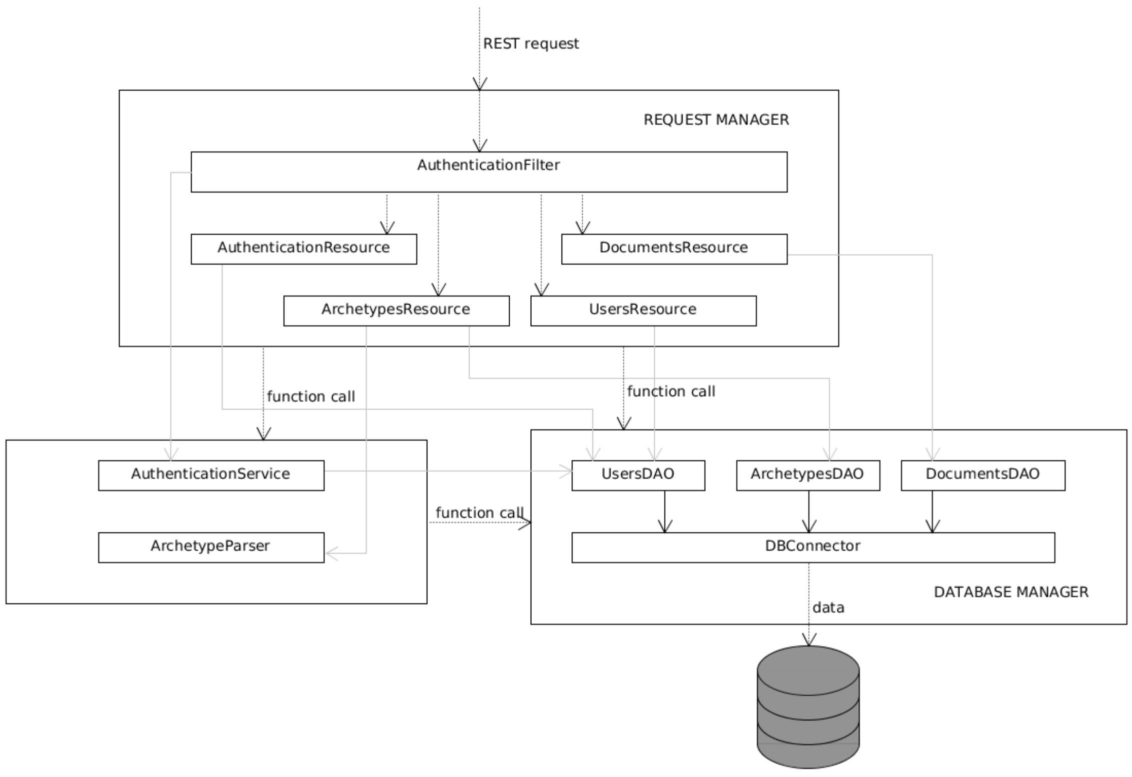


Figure 9.4 An architecture of the REST server application at the back-end of the proposed EHR system (https://github.com/ghessova/SemanticEHR/wiki/Architecture)

Front-end

Currently, two front-end prototypes are implemented:

1. a server-side prototype where the code is processed on the server side; the solution is implemented in a free version of Java-based framework Vaadin[6].

2. a client-side prototype where the code is processed on the client side, i.e. web browser; solution is implemented in a JavaScript-based framework Angular JS[7].

The client-side front-end is preferred to prevent the hosting server from overburdening.

Mobile client

Mobile client development is still in the phase of architectural proposal. It is designed as a hybrid mobile application to prevent separate development for various platforms (e.g. Android, iOS, Windows Phone) utilizing the Ionic[8] framework. Since the application is designed to import data from the third-party resources/applications, there will be an interface to communicate with Google Fit[9], Apple Health[10] or Microsoft Health[11] platforms.

9.1.5 Development team

There is no active development team at the moment and the project has been suspended. However, I would like to acknowledge bachelor and master degree students who were involved in the development and provided a considerable effort and enthusiasm.

- Daniel Horák, Ondřej Byrtus, Jan Schröpfer: Automatically generated client-side front-end (Angular JS framework)

- Ondřej Pražák, Gabriela Hessová, Patrik Kořínek: REST Server application implementation (Jersey framework), database implementation (MongoDB)

- Lukáš Rostás, David Herman, Michal Medek: Database implementation (MongoDB), mobile client architecture proposal

- Ondřej Havlíček: Automatically generated server-side front-end (Vaadin framework)

[6]https://vaadin.com/home
[7]https://angularjs.org/
[8]https://ionicframework.com/
[9]https://www.google.com/fit/
[10]https://www.apple.com/ios/health/
[11]https://www.microsoft.com/microsoft-health

- Jiří Matyáš: Proposal of a new archetype describing sleep data based on the Sleep as Android[12] application and Sleep Domain Ontology[13]; prototype of a mobile client for Android
- Martin Graubner, Markéta Wolfová, Marek L'uptáčik, Adam Barák: Prototype of an Elasticsearch storage
- Karol Kalaga, Mateusz Łysień: Storing of archetypes in Elasticsearch

9.2 EEG/ERP archetype for clinical purposes

Developed archetypes describing EEG/ERP domain reflect recent data formats used in electrophysiology and address the absence of such archetypes in the CKMs. The archetypes serve to conduct a research in EEG/ERP domain and narrow the gap between data retrieved from experimental EEG/ERP measurements and clinical data. Exemplified usage of these archetypes is in the experimental EHR system described in Section 9.1.

Potential application of the archetypes in a clinical sphere is limited. The publication and approval process, where the archetypes are reviewed by clinicians and domain experts, is a long-term one. However, an incubator for the archetypes within the CKM was created and further improvement is expected.

[12]http://sleep.urbandroid.org/cs/
[13]http://purl.bioontology.org/ontology/SDO

Chapter 10

Conclusion

10.1 Results summary

The main aim of this work was to investigate the potential benefits of EHR description frameworks utilized to EEG/ERP data/metadata modelling. openEHR, the archetype-based EHR description framework, was selected and applied to EEGBase portal data and metadata. The main EEGBase constraint is the strict separation of experiment data and experiment metadata. This feature may cause serious difficulties with further extension of the metadata set, especially by complex structures taken from other measurements (e.g. blood pressure measurement).

The investigation has shown that the experiment data from EEGBase could be mapped as composition of newly proposed and already existing archetypes. Moreover, the archetypes have been developed in-line with the already existing EEG formats (NIX, EDF+, and the BrainVision EEG format), respecting the odML terminology for electrophysiology. Therefore, the data stored in these structures could be easily mapped into our archetype structures. The proposed mapping allows us to present the existing data in the form that is typical for the health domain/health records. Apart from the created archetypes and templates, an algorithm for referencing/dereferencing odML terms was proposed. Also an alternative for the odML terminology, a mapping of used terms to UMLS meta-thesaurus, was made.

The archetype publication process, i.e. the process during which the proposed archetype goes through various lifecycle phases, is currently in progress and an incubator for the archetypes (the virtual space where the archetypes are accessible to domain experts for additional comments and modifications) was created within the official openEHR CKM.

For the evaluation of computable abilities of the proposed archetypes a set of desired features was selected. These desiderata were originally proposed for computable phenotype representations, however, their subset is relevant also for the domain of EEG/ERP data modelling. Proposed solution met all selected criteria.

For testing with real data, the archetype structures were mapped on the most complex experiment in EEGBase focused on the developmental coordination disorder in children.

Aside from that, the work determined the elements in common data modelling concepts including relational model, object-oriented model, ontology, etc, compared the expressive power of the concepts and organized the concepts into a semantic hierarchy. For each level, a specific model was derived from the reference archetypes. Currently, an ontology development, at the top level of the hierarchy, stays incomplete. However, this development is in the scope of the OEN project.

Currently, there is no specific deployment of the proposed archetypes. The implementation of the whole archetype concept into the EEGBase portal is a challenge from the architectural point of view. However, the concept of a personal EHR system based on openEHR (currently under development) implements the EEG/ERP archetypes. The key benefit of this personal EHR system and/or improved EEGBase is in the granularity of experiment data/metadata and in the existence of the flexible border between data and metadata. The system design allows researchers to choose which dataset can be used as data (the analysis input) and which dataset can be used as metadata (analysis attributes and filters). This approach also enables researchers to do a reverse analysis; e.g., the blood pressure could be analysed according to the body mass index (BMI) criterion or the BMI could be analysed according to the blood pressure criterion. Furthermore, all datasets could be processed using the same software tools (e.g. the tools based on openEHR Java libraries).

The potential use of the developed EEG/ERP archetypes in the clinical domain is beyond the scope of this work.

10.2 Future work

Proposed models on various levels of semantic hierarchy could unify EEG/ERP data resources independently on the technology used. However, to achieve unified data access, a single access point over the resource must be implemented. Such solution provides, e.g., OpenLinks Virtuoso[1]. Virtuoso server has potential to provide a federative database composed of various data resources similar to NIF. Nevertheless, unlike NIF, no ontology would be necessary for data access and retrieved data would have openEHR archetype structures/annotation.

Ontology development at the top level of a semantic hierarchy is still incomplete. Even though the OEN development was initialized, currently the project is not in progress and

[1]https://virtuoso.openlinksw.com/

future work is desired. OEN has potential to serve as a reference resource suitable for backward improvement of underlying models in the hierarchy.

Also the development of personal EHR system for research use should continue in the future to fully address its aims.

Given the fact that openEHR respects CEN/ISO 13606 and that translation rules between HL7v3 and openEHR exist, the created archetypes can be transformed to HL7v3 models / CEN/ISO 13606 archetypes and serve in the EHR systems based on HL7v3 / CEN/ISO 13606. Together with the completion of the archetype publication process, these domain transformations should be performed in the future. Then, the current gap between experimental and clinical data descriptions could be reduced.

References

[1] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[2] Thomas Beale. Archetype Definition Language 2 (ADL2) Specification, june 2016. URL http://www.openehr.org/releases/AM/latest/docs/ADL2/ADL2.html.

[3] Thomas Beale. EHR Information Model, may 2016. URL http://www.openehr.org/releases/RM/latest/docs/ehr/ehr.html.

[4] Thomas Beale, Sam Heard, Dipak Kalra, and David Lloyd. Openehr architecture overview. *The OpenEHR Foundation*, 2006. URL http://www.openehr.org/releases/BASE/latest/docs/architecture_overview/architecture_overview.html.

[5] Chris Bizer, R Cyganiak, J Garbers, O Maresch, and C Becker. The d2rq platform. *On-line (Last access May. 16): http://www4.wiwiss.fuberlin.de/bizer/D2RQ/spec*, 2009.

[6] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[7] BrainProducts. *BrainVision Recorder: User Manual*. Brain Products GmbH, http://www.brainproducts.com/, 2013.

[8] Encyclopædia Britannica. Electroencephalography, 01 2016. URL http://www.britannica.com/science/electroencephalography.

[9] E Browne. Openehr archetypes for hl7 cda documents. *Ocean Informatics*, page 47, 2008.

[10] P Bruha, V Papez, Anita Bandrowski, Jan Grewe, R Moucek, Shreejoy Tripathy, et al. The ontology for experimental neurophysiology: a first step toward semantic annotations of neurophysiology data and metadata. In *Front. Neuroinform.(Conference Abstract: Neuroinformatics 2013)*, volume 26, 2013. doi: 10.3389/conf.fninf.2013.09.00026. URL http://www.frontiersin.org/neuroinformatics/10.3389/conf.fninf.2013.09.00026/full.

[11] Petr Bruha. Eeg/erp portal a prostredky semantickeho webu. Master's thesis, University of West Bohemia, 2011.

[12] Peter Chen. Entity-relationship modeling: historical events, future trends, and lessons learned. In *Software pioneers*, pages 296–310. Springer, 2002.

[13] James Clark, Steve DeRose, et al. Xml path language (xpath) version 1.0, 1999.

[14] Edgar F Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems (TODS)*, 4(4):397–434, 1979.

[15] Mélanie Courtot, Frank Gibson, Allyson L Lister, James Malone, Daniel Schober, Ryan R Brinkman, and Alan Ruttenberg. Mireot: The minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, 2011.

[16] John Cowan, J Paoli, Francois Yergeau, Eve Maler, CM Sperberg-McQueen, and T Bray. Extensible markup language (xml) 1.1. *W3C CR CR-xml11-20021015*, 2002.

[17] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. *W3C Recommendation February*, 10, 2004.

[18] Robert H Dolin, Liora Alschuler, Sandy Boyer, Calvin Beebe, Fred M Behlen, Paul V Biron, and Amnon Shabo. Hl7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, 13(1):30–39, 2006.

[19] P Elkin, Martin Kernberg, Thomas Beale, Sam Heard, Mark Shafarman, Robert Dolin, and Dale Nelson. Hl7 template and archetype architecture. *HL7 Template Special Interest Group*, page 203, 2004. URL http://www.hl7.org/library/committees/template/HL7_Atlanta_10_20_04.doc.

[20] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, AD '11, pages 36–47, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0614-0. doi: 10.1145/1966895.1966900. URL http://doi.acm.org/10.1145/1966895.1966900.

[21] Dave Garets and Mike Davis. Electronic medical records vs. electronic health records: yes, there is a difference. *Policy white paper. Chicago, HIMSS Analytics*, pages 1–14, 2006.

[22] P Garrett and J Seidman. Emr vs ehr—what is the difference. *HealthITBuzz, January*, 2011.

[23] Jan Grewe, Thomas Wachtler, and Jan Benda. A bottom-up approach to data annotation in neurophysiology. *Frontiers in Neuroinformatics*, 5(16), 2011. ISSN 1662-5196. doi: 10.3389/fninf.2011.00016. URL http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2011.00016/abstract.

[24] Birger Haarbrandt, Thomas Jack, and Michael Marschollek. Automated transformation of openehr data instances to owl. In *Health Informatics Meets EHealth: Predictive Modeling in Healthcare–From Prediction to Prevention. Proceedings of the 10th EHealth2016 Conference*, volume 223, page 63. IOS Press, 2016.

[25] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. Sparql 1.1 query language. *W3C recommendation*, 21(10), 2013.

[26] David P. Heitmeyer. Sgml, xml, html, and xhtml, 2007. URL http://cscie12.dce.harvard.edu/lecture_notes/2007-08/20080130/slide26.html.

[27] Andreas V.M. Herz, Ralph Meier, Martin P. Nawrot, Willi Schiegel, and Tiziano Zito. G-node: An integrated tool-sharing platform to support cellular and systems neurophysiology in the age of global neuroinformatics. *Neural Networks*, 21(8):1070 – 1075, 2008. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2008.05.011. URL http://www.sciencedirect.com/science/article/pii/S0893608008001160. NeuroinformaticsNeuroinformatics.

[28] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer. *W3C recommendation*, 27(1):123, 2009.

[29] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79, 2004.

[30] Jaap Van Pelt Jack Van Horn. 1st incf workshop on sustainability of neuroscience databases. In *Nature Precedings*, doi:10.1038/npre.2008.1983.1, 2008. http://dx.doi.org/10.1038/npre.2008.1983.1.

[31] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. Pearson Education India, 1993.

[32] Petr Jezek and Roman Moucek. Database of eeg/erp experiments. In *HEALTHINF*, pages 222–227, 2010.

[33] Petr Jezek and Roman Moucek. System for eeg/erp data and metadata storage and management. *Neural Network World*, 22(3):277, 2012.

[34] Dipak Kalra, Thomas Beale, and Sam Heard. The openehr foundation. *Studies in health technology and informatics*, 115:153–173, 2005.

[35] Bob Kemp and Jesus Olivan. European data format 'plus'(edf+), an edf alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9): 1755–1761, 2003.

[36] Bob Kemp, Alpo Värri, Agostinho C Rosa, Kim D Nielsen, and John Gade. A simple format for exchange of digitized polygraphic recordings. *Electroencephalography and clinical neurophysiology*, 82(5):391–393, 1992.

[37] Ozgur Kilic, Veli Bicer, and Asuman Dogac. Mapping archetypes to owl. *Middle East Tech. Univ., Ankara, Turkey, Tech. Rep. TR-2005-3 [Online]. Available: http://www. srdc. metu. edu. tr/webpage/publications/2005/MappingArchetypestoOWLTechnical. pdf*, 2005.

[38] Won Kim. Object-oriented databases: Definition and research directions. *IEEE Transactions on Knowledge & Data Engineering*, 2(3):327–341, 1990.

[39] Miroslav Kral. Experimentalni lekarsky informacni system: zpracovani heterogennich nestrukturovanych dat. Master's thesis, University of West Bohemia, 2010.

[40] Yann Le Franc, Anita Bandrowski, Petr Bruha, Vaclav Papez, Jan Grewe, Roman Moucek, Shreejoy J Tripathy, and Thomas Wachtler. Describing neurophysiology data and metadata with oen, the ontology for experimental neurophysiology. *Frontiers in Neuroinformatics*, 8(44), 2014. ISSN 1662-5196. doi: 10.3389/conf.fninf.2014.18.00044. URL http://www.frontiersin.org/neuroinformatics/10.3389/conf.fninf.2014.18.00044/full.

[41] Leonardo Lezcano, Miguel-Angel Sicilia, and Carlos Rodríguez-Solano. Integrating reasoning and clinical archetypes using owl ontologies and swrl rules. *Journal of biomedical informatics*, 44(2):343–353, 2011.

[42] Harvey Lodish. *Molecular cell biology*. W.H. Freeman, New York, 2000. ISBN 0-7167-3136-3.

[43] S.J. Luck. *An Introduction to the Event-Related Potential Technique*. MIT Press, 2014. ISBN 9780262324069. URL https://books.google.com/books?id=y4-uAwAAQBAJ.

[44] Yanhui Lv and ZM Ma. Transformation of relational model to rdf model. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pages 506–511. IEEE, 2008.

[45] Luis Marenco, Yuli Li, Maryann E Martone, Paul W Sternberg, Gordon M Shepherd, and Perry L Miller. Issues in the design of a pilot concept-based query interface for the neuroinformatics information framework. *Neuroinformatics*, 6(3):229–239, 2008.

[46] C McCay, D Kalra, and R Worden. Results of investigating the transformability between hl7 v3, openehr and en/iso 13606. *NHS Connecting for Health, Leeds*, 11, 2008.

[47] Huan Mo, William K Thompson, Luke V Rasmussen, Jennifer A Pacheco, Guoqian Jiang, Richard Kiefer, Qian Zhu, Jie Xu, Enid Montague, David S Carrell, et al. Desiderata for computable representations of electronic health records-driven phenotype algorithms. *Journal of the American Medical Informatics Association*, 22(6):1220–1230, 2015.

[48] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, et al. Owl 2 web ontology language profiles. *W3C recommendation*, 27:61, 2009.

[49] Roman Moucek and Pavel Mautner. Attention of drivers during dual task - eeg/erp experiment. In *Kognice a umely zivot*, pages 219–223. Slezska univerzita, 2009. ISBN 978-80-7248-516-1.

[50] Roman Moucek, Petr Bruha, Petr Jezek, Pavel Mautner, Jiri Novotny, Vaclav Papez, Tomas Prokop, Tomas Rondik, Jan Stebetak, and Lukas Vareka. Software and hardware infrastructure for research in electrophysiology. *Frontiers in Neuroinformatics*, 8(20), 2014. ISSN 1662-5196. doi: 10.3389/fninf.2014.00020. URL http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00020/abstract.

[51] Pilar Munoz, Jesus D Trigo, Ignacio Martinez, Adolfo Munoz, Javier Escayola, and Jose Garcia. The iso/en 13606 standard for the interoperable exchange of electronic health records. *Journal of Healthcare Engineering*, 2(1):1–24, 2011.

[52] Houston Neal. Ehr vs emr: what's the difference?, 2011. URL "http://profitable-practice.softwareadvice.com/ehr-vs-emr-whats-the-difference/".

[53] NISOPress. Understanding Metadata. *National Information Standards*, 20, 2004.

[54] Marek Obitko. Ontologies and semantic web. *URL: http://obitko. com/tutorials/ontologies-semantic-web*, 2007.

[55] V. Papez and R. Moucek. Model of storage of erp protocols in eeg/erp portal. In *2012 5th International Conference on BioMedical Engineering and Informatics*, pages 1275–1279, Oct 2012. doi: 10.1109/BMEI.2012.6512944.

[56] V. Papez and R. Moucek. Archetypes development in electrophysiology domain: Electroencephalography as a personal ehr system module. In *HEALTHINF 2015 - 8th International Conference on Health Informatics, Proceedings; Part of 8th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2015*, pages 611–616, 2015. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84938821933&partnerID=40&md5=ee59d85bcd80a78393f4a6e93fe53c68.

[57] Vaclav Papez. Neuroinformatic database and semantic web. Master thesis, University of West Bohemia, Pilsen, Czech Republic, 06 2010.

[58] Vaclav Papez and Roman Moucek. Data and metadata models in electrophysiology domain: Separation of data models into semantic hierarchy and its integration into eegbase. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 539–543. IEEE, 2013.

[59] Vaclav Papez and Roman Moucek. Applying an archetype-based approach to electroencephalography/event-related potential experiments in the eegbase resource. *Frontiers in Neuroinformatics*, 11, 2017.

[60] Vaclav Papez, Spiros Denaxas, and Harry Hemingway. Evaluation of semantic web technologies for storing computable definitions of electronic health records phenotyping algorithms. American Medical Informatics Association (AMIA) 2017 Annual Symposium, 2017.

[61] Vaclav Papez, Spiros Denaxas, and Harry Hemingway. Evaluating openehr for storing computable representations of electronic health record phenotyping algorithms. *IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, 2017.

[62] Mario Piattini and Oscar Diaz. *Advanced database technology and design*. Artech House, Inc., 2000.

[63] Karel Richta. Unifikovany modelovaci jazyk uml. *Systems Integration*, page 386, 2003.

[64] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. *Object-oriented modeling and design*. Prentice-hall Englewood Cliffs, NJ, 1991.

[65] Michael Q Stearns, Colin Price, Kent A Spackman, and Amy Y Wang. Snomed clinical terms: overview of the development process and project status. In *Proceedings of the AMIA Symposium*, page 662. American Medical Informatics Association, 2001.

[66] Adrian Stoewer, Christian Johannes Kellner, Jan Benda, Thomas Wachtler, and Jan Grewe. File format and library for neuroscience data and metadata. *Frontiers in Neuroinformatics*, 8(27), 2014. ISSN 1662-5196. doi: 10.3389/conf.fninf.2014.18.00027. URL http://www.frontiersin.org/neuroinformatics/10.3389/conf.fninf.2014.18.00027/full.

[67] Vojtech Svatek. Ontologie a www. *Praha: Vysoka skola ekonomicka*, 2002.

[68] Jeffery L. Teeters, Keith Godfrey, Rob Young, Chinh Dang, Claudia Friedsam, Barry Wark, Hiroki Asari, Simon Peron, Nuo Li, Adrien Peyrache, Gennady Denisov, Joshua H. Siegle, Shawn R. Olsen, Christopher Martin, Miyoung Chun, Shreejoy Tripathy, Timothy J. Blanche, Kenneth Harris, György Buzsaki, Christof Koch, Markus Meister, Karel Svoboda, and Friedrich T. Sommer. Neurodata without borders: Creating a common data format for neurophysiology. *Neuron*, 88(4):629 – 634, 2015. ISSN 0896-6273. doi: http://dx.doi.org/10.1016/j.neuron.2015.10.025. URL http://www.sciencedirect.com/science/article/pii/S0896627315009198.

[69] Jiri Vanek. Data format for electrophysiological experiments. Master's thesis, University of West Bohemia, 2015.

[70] Paul Watson, Phillip Lord, Frank Gibson, Panayiotis Periorellis, and Georgios Pitsilis. Cloud computing for e-science with carmen. In *In 2nd Iberian Grid Infrastructure Conference Proceedings*, pages 3–14, 2008.

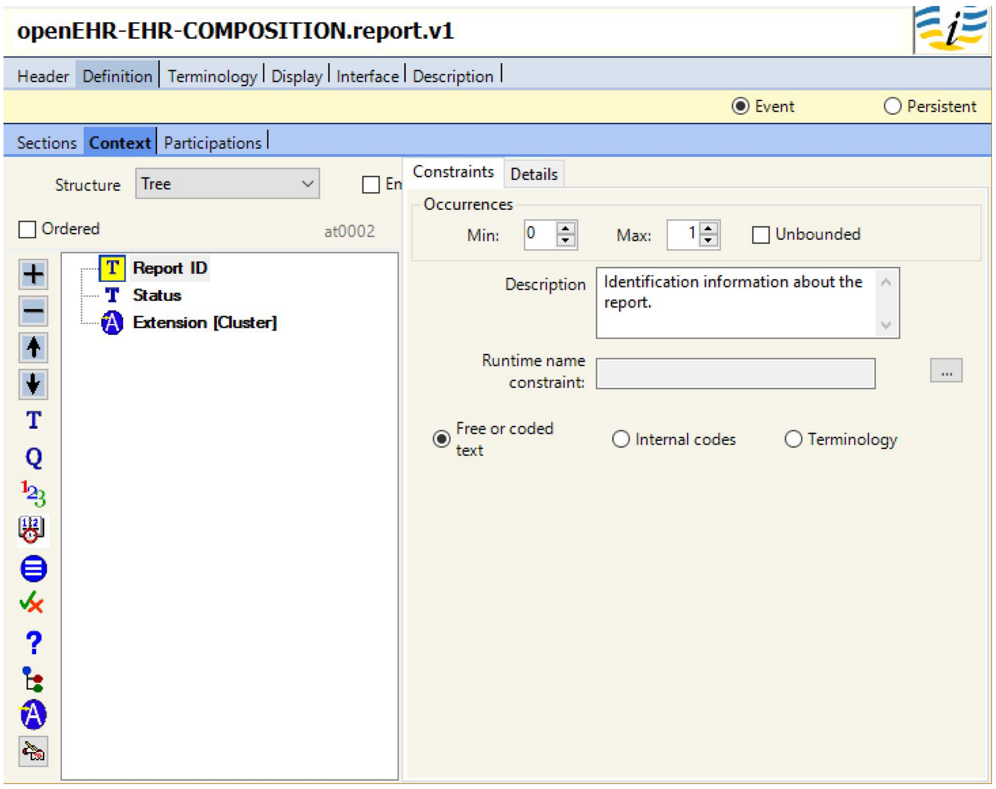
Part VI

Appendices

Appendix A

List of proposed archetypes, templates and related odML sections

Report archetype



openEHR-EHR-COMPOSITION.report.v1

Header | Definition | Terminology | Display | Interface | Description

[✓ Event] Persistent

Sections | Context | Participations

Structure: Tree

Ordered: ☐ at0002

- Report ID
- Status
- Extension [Cluster]

Constraints | Details

Occurrences — Min: 0 Max: 1 ☐ Unbounded

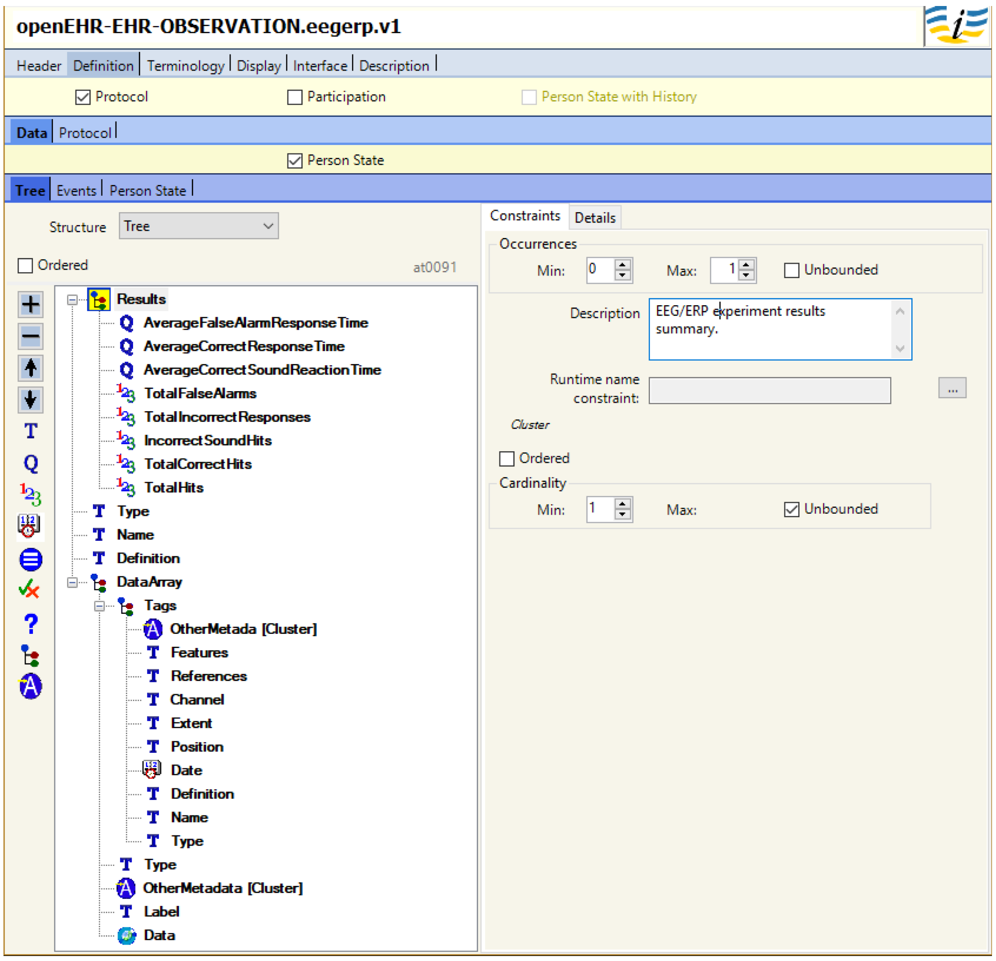
Description: Identification information about the report.

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

Figure A.1 Definition of the *Report* archetype (*Composition* RM)

EEG/ERP experiment results archetype


openEHR-EHR-OBSERVATION.eegerp.v1

Header | Definition | Terminology | Display | Interface | Description

☑ Protocol ☐ Participation ☐ Person State with History

Data | Protocol

☑ Person State

Tree | Events | Person State

Structure: Tree

☐ Ordered at0091

- Results
  - AverageFalseAlarmResponseTime
  - AverageCorrectResponseTime
  - AverageCorrectSoundReactionTime
  - TotalFalseAlarms
  - TotalIncorrectResponses
  - IncorrectSoundHits
  - TotalCorrectHits
  - TotalHits
- Type
- Name
- Definition
- DataArray
  - Tags
    - OtherMetada [Cluster]
    - Features
    - References
    - Channel
    - Extent
    - Position
    - Date
    - Definition
    - Name
    - Type
  - Type
  - OtherMetadata [Cluster]
  - Label
  - Data

Constraints | Details

Occurrences — Min: 0 Max: 1 ☐ Unbounded

Description: EEG/ERP experiment results summary.

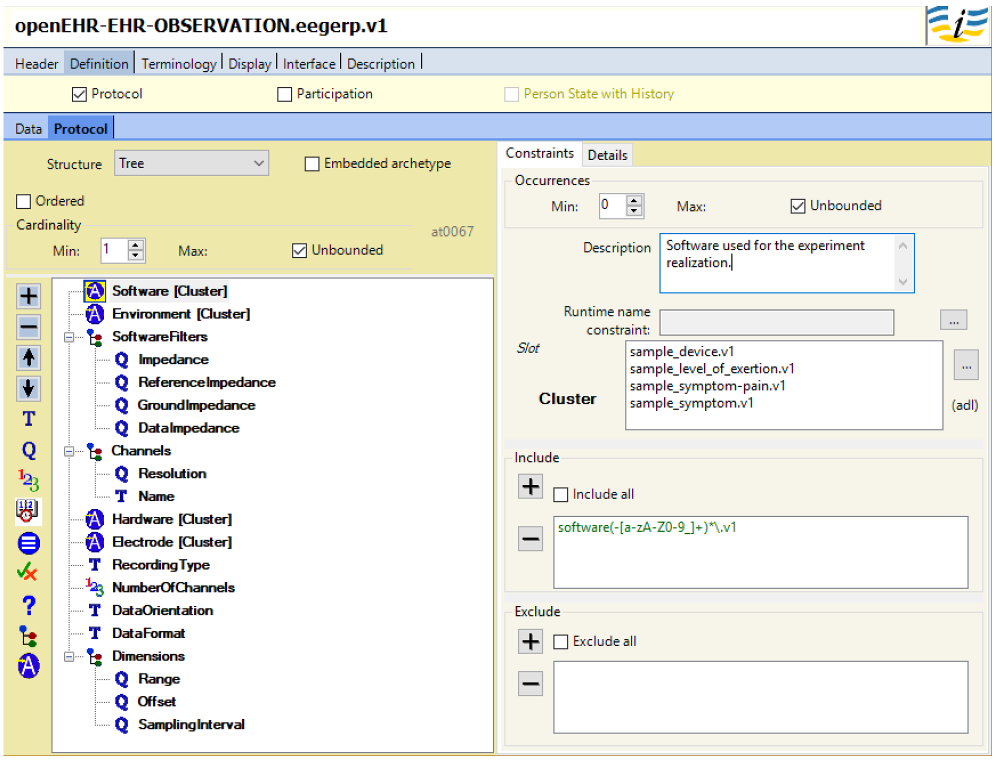
Runtime name constraint:

Cluster

☐ Ordered

Cardinality — Min: 1 Max: ☑ Unbounded

Figure A.2 Definition of the *EEG/ERP experiment results* archetype (*Observation* RM): Data section



openEHR-EHR-OBSERVATION.eegerp.v1

Header | Definition | Terminology | Display | Interface | Description

☑ Protocol ☐ Participation ☐ Person State with History

Data | Protocol

Structure: Tree ☐ Embedded archetype

☐ Ordered

Cardinality — at0067

Min: 1 Max: ☑ Unbounded

- Software [Cluster]
- Environment [Cluster]
- SoftwareFilters
  - Impedance
  - ReferenceImpedance
  - GroundImpedance
  - DataImpedance
- Channels
  - Resolution
  - Name
- Hardware [Cluster]
- Electrode [Cluster]
- RecordingType
- NumberOfChannels
- DataOrientation
- DataFormat
- Dimensions
  - Range
  - Offset
  - SamplingInterval

Constraints | Details

Occurrences — Min: 0 Max: ☑ Unbounded

Description: Software used for the experiment realization.

Runtime name constraint:

Slot — Cluster:

sample_device.v1
sample_level_of_exertion.v1
sample_symptom-pain.v1
sample_symptom.v1

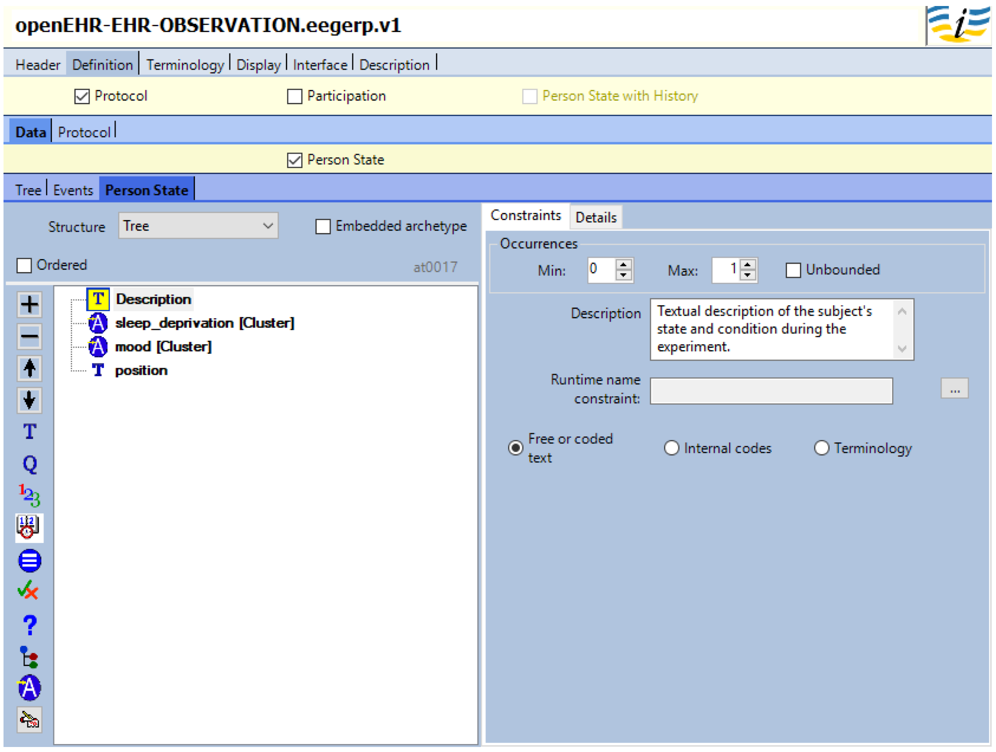
(adl)

Include — ☐ Include all

software(-[a-zA-Z0-9_]+)*\.v1

Exclude — ☐ Exclude all

Figure A.3 Definition of the *EEG/ERP experiment results* archetype (*Observation* RM): Protocol section



openEHR-EHR-OBSERVATION.eegerp.v1

Header | Definition | Terminology | Display | Interface | Description

[✓ Protocol] Participation Person State with History

Data | Protocol

[✓ Person State]

Tree | Events | Person State

Structure: Tree

Embedded archetype

Ordered

at0017

- Description
- sleep_deprivation [Cluster]
- mood [Cluster]
- position

Constraints | Details

Occurrences

Min: 0 Max: 1 Unbounded

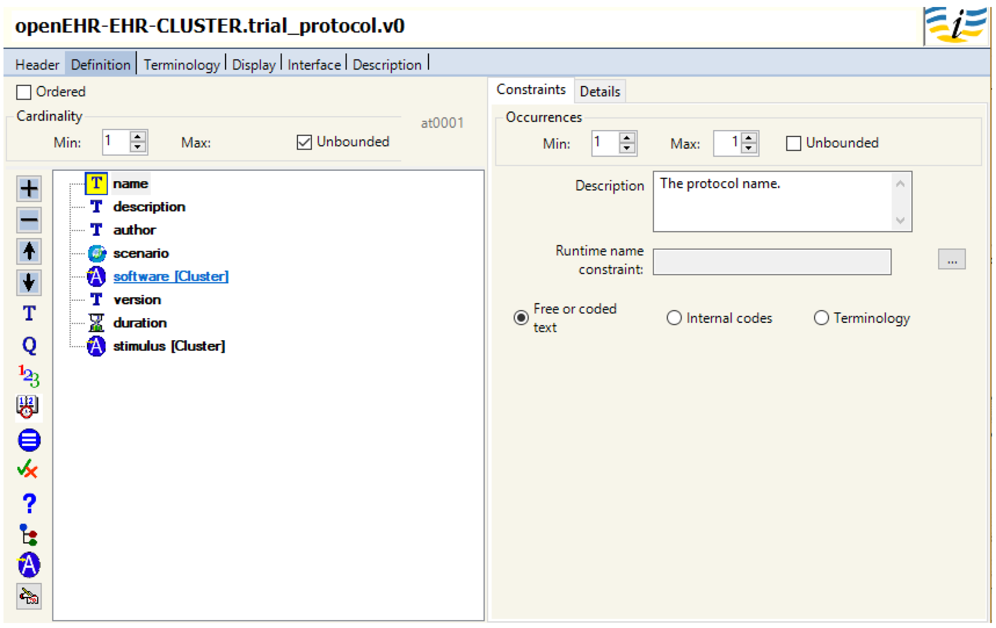
Description: Textual description of the subject's state and condition during the experiment.

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

Figure A.4 Definition of the *EEG/ERP experiment results* archetype (*Observation* RM): State section

Trial/Experiment protocol archetype


openEHR-EHR-CLUSTER.trial_protocol.v0

Header | Definition | Terminology | Display | Interface | Description

☐ Ordered

Cardinality

Min: 1 Max: ☑ Unbounded

at0001

- name
- description
- author
- scenario
- software [Cluster]
- version
- duration
- stimulus [Cluster]

Constraints | Details

Occurrences

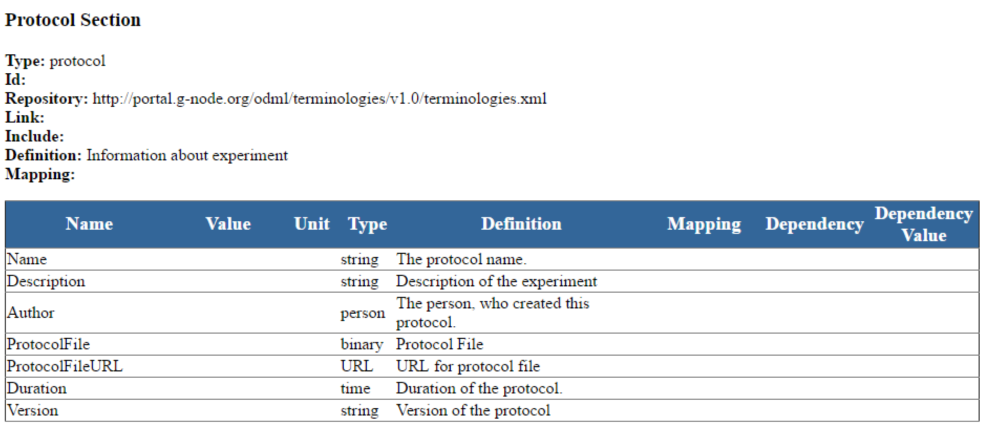
Min: 1 Max: 1 ☐ Unbounded

Description: The protocol name.

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

Figure A.5 Definition of the *Trial/Experiment protocol* archetype (*Cluster* RM), which corresponds to the odML Protocol section (Figure A.6) specification

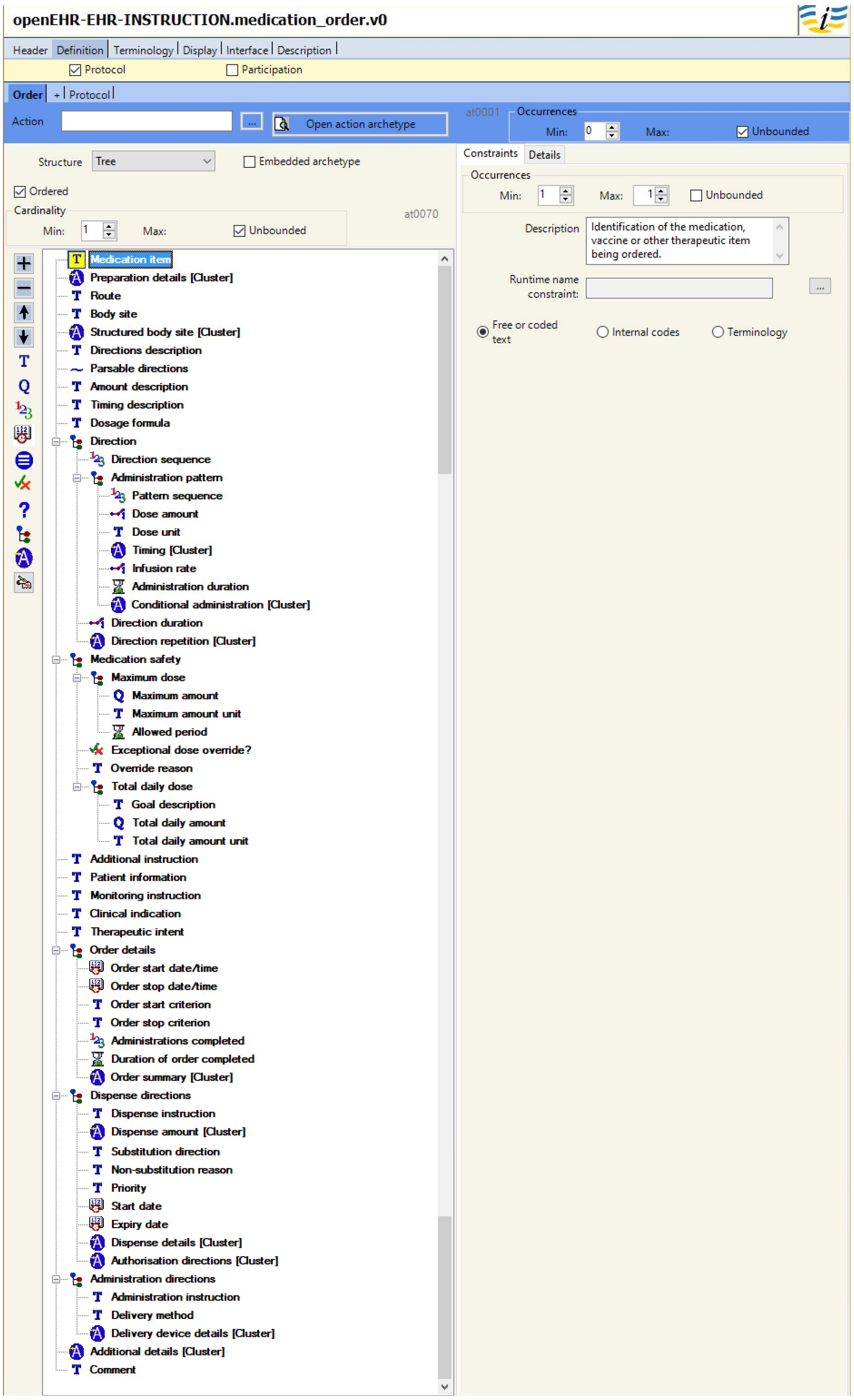

Protocol Section

Type: protocol
Id:
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Information about experiment
Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Name			string	The protocol name.			
Description			string	Description of the experiment			
Author			person	The person, who created this protocol.			
ProtocolFile			binary	Protocol File			
ProtocolFileURL			URL	URL for protocol file			
Duration			time	Duration of the protocol.			
Version			string	Version of the protocol			

Figure A.6 Specification of the odML *Protocol* section

Medication order archetype



openEHR-EHR-INSTRUCTION.medication_order.v0

Header | Definition | Terminology | Display | Interface | Description

[✓ Protocol] Participation

Order | + | Protocol

Action: [empty] ... Open action archetype

at0001 Occurrences — Min: 0 Max: [✓ Unbounded]

Structure: Tree — Embedded archetype

[✓ Ordered]

Cardinality — Min: 1 Max: [✓ Unbounded]

at0070

Constraints | Details

Occurrences — Min: 1 Max: 1 — Unbounded

Description: Identification of the medication, vaccine or other therapeutic item being ordered.

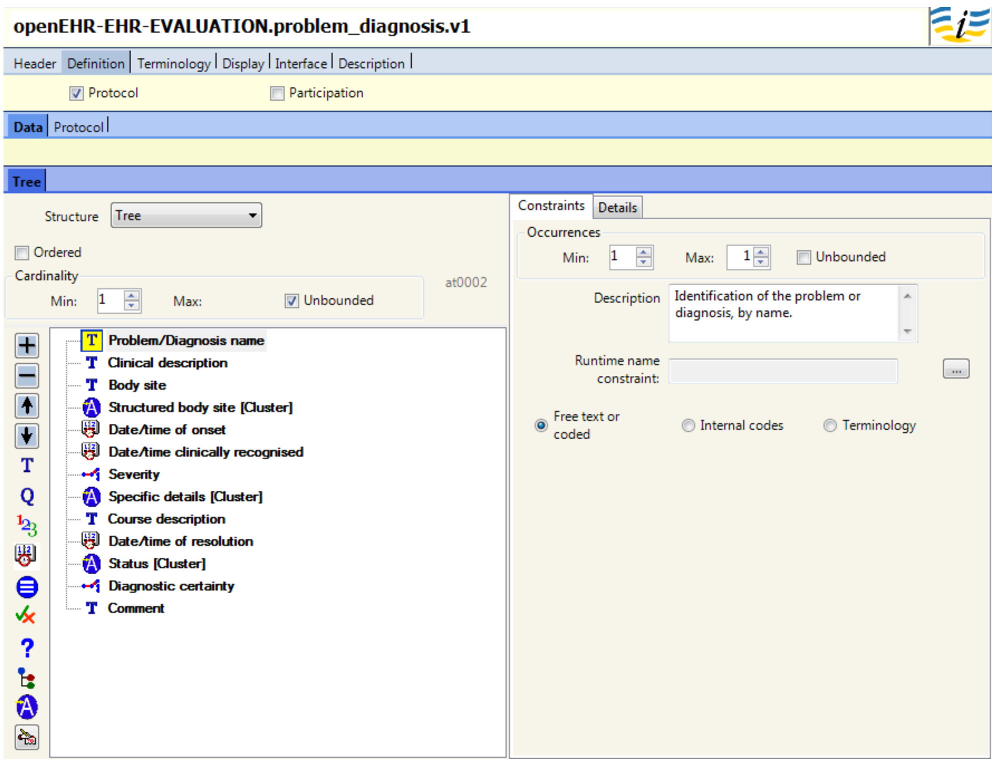
Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

- Medication item
- Preparation details [Cluster]
- Route
- Body site
- Structured body site [Cluster]
- Directions description
- Parsable directions
- Amount description
- Timing description
- Dosage formula
- Direction
  - Direction sequence
  - Administration pattern
    - Pattern sequence
    - Dose amount
    - Dose unit
    - Timing [Cluster]
    - Infusion rate
    - Administration duration
    - Conditional administration [Cluster]
  - Direction duration
  - Direction repetition [Cluster]
- Medication safety
  - Maximum dose
    - Maximum amount
    - Maximum amount unit
    - Allowed period
  - Exceptional dose override?
  - Override reason
  - Total daily dose
    - Goal description
    - Total daily amount
    - Total daily amount unit
- Additional instruction
- Patient information
- Monitoring instruction
- Clinical indication
- Therapeutic intent
- Order details
  - Order start date/time
  - Order stop date/time
  - Order start criterion
  - Order stop criterion
  - Administrations completed
  - Duration of order completed
  - Order summary [Cluster]
- Dispense directions
  - Dispense instruction
  - Dispense amount [Cluster]
  - Substitution direction
  - Non-substitution reason
  - Priority
  - Start date
  - Expiry date
  - Dispense details [Cluster]
  - Authorisation directions [Cluster]
- Administration directions
  - Administration instruction
  - Delivery method
  - Delivery device details [Cluster]
- Additional details [Cluster]
- Comment

Figure A.7 Definition of the *Medication order* archetype (*Instruction* RM); the archetype is designed by openEHR and taken from CKM

Problem diagnosis archetype



openEHR-EHR-EVALUATION.problem_diagnosis.v1

[✓] Protocol
[ ] Participation

Structure: Tree
[ ] Ordered
Cardinality — Min: 1 Max: [✓] Unbounded
at0002

- Problem/Diagnosis name
- Clinical description
- Body site
- Structured body site [Cluster]
- Date/time of onset
- Date/time clinically recognised
- Severity
- Specific details [Cluster]
- Course description
- Date/time of resolution
- Status [Cluster]
- Diagnostic certainty
- Comment

Occurrences — Min: 1 Max: 1 [ ] Unbounded

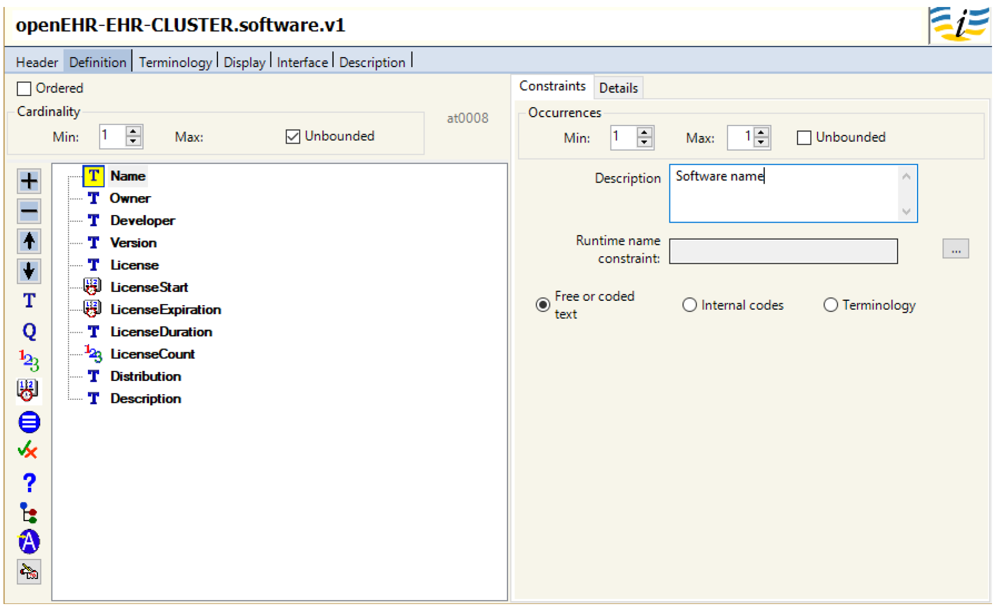
Description: Identification of the problem or diagnosis, by name.

Runtime name constraint:

[✓ Free text or coded] Internal codes Terminology

Figure A.8 Definition of the *Problem diagnosis* archetype (*Evaluation* RM); the archetype is designed by openEHR and taken from CKM

Software archetype



openEHR-EHR-CLUSTER.software.v1

☐ Ordered

Cardinality Min: 1 Max: ☑ Unbounded

at0008

- Name
- Owner
- Developer
- Version
- License
- LicenseStart
- LicenseExpiration
- LicenseDuration
- LicenseCount
- Distribution
- Description

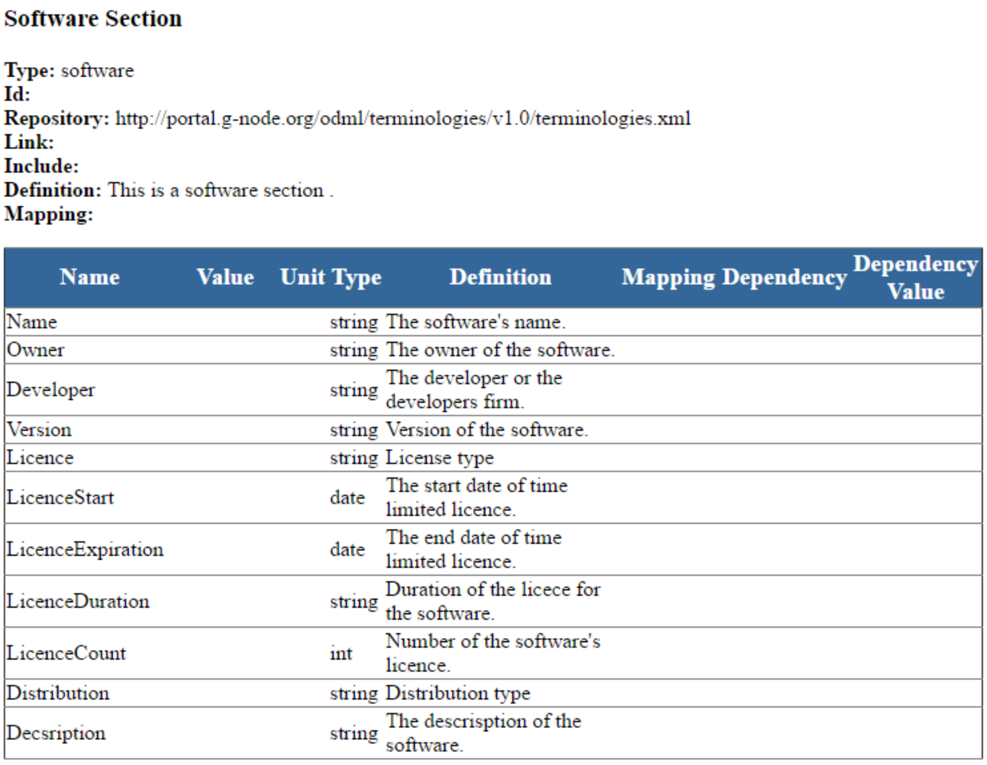
Occurrences Min: 1 Max: 1 ☐ Unbounded

Description: Software name

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

Figure A.9 Definition of the *Software* archetype (RM Cluster), which corresponds to the *Software* section (Figure A.10) in the odML terminology



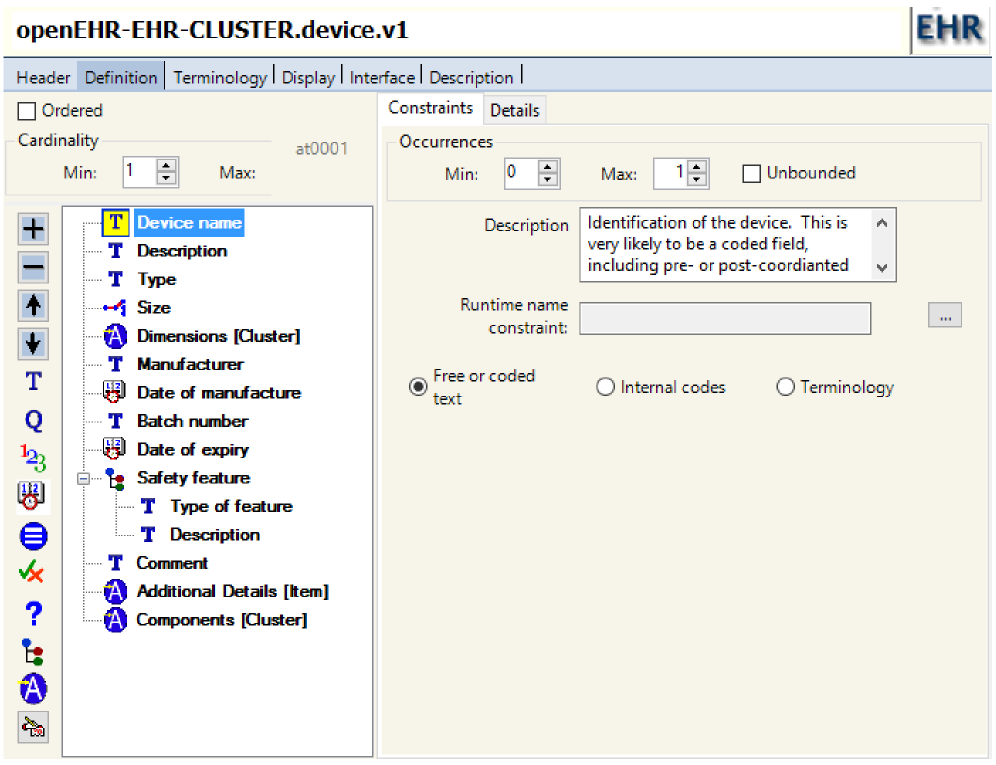
Software Section

Type: software
Id:
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: This is a software section .
Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Name			string	The software's name.			
Owner			string	The owner of the software.			
Developer			string	The developer or the developers firm.			
Version			string	Version of the software.			
Licence			string	License type			
LicenceStart			date	The start date of time limited licence.			
LicenceExpiration			date	The end date of time limited licence.			
LicenceDuration			string	Duration of the licece for the software.			
LicenceCount			int	Number of the software's licence.			
Distribution			string	Distribution type			
Decsription			string	The descrisption of the software.			

Figure A.10 Specification of the odML *Software* section

Device and Device Details archetypes



openEHR-EHR-CLUSTER.device.v1

Header | Definition | Terminology | Display | Interface | Description

☐ Ordered

Cardinality Min: 1 Max:

at0001

- Device name
- Description
- Type
- Size
- Dimensions [Cluster]
- Manufacturer
- Date of manufacture
- Batch number
- Date of expiry
- Safety feature
  - Type of feature
  - Description
- Comment
- Additional Details [Item]
- Components [Cluster]

Constraints | Details

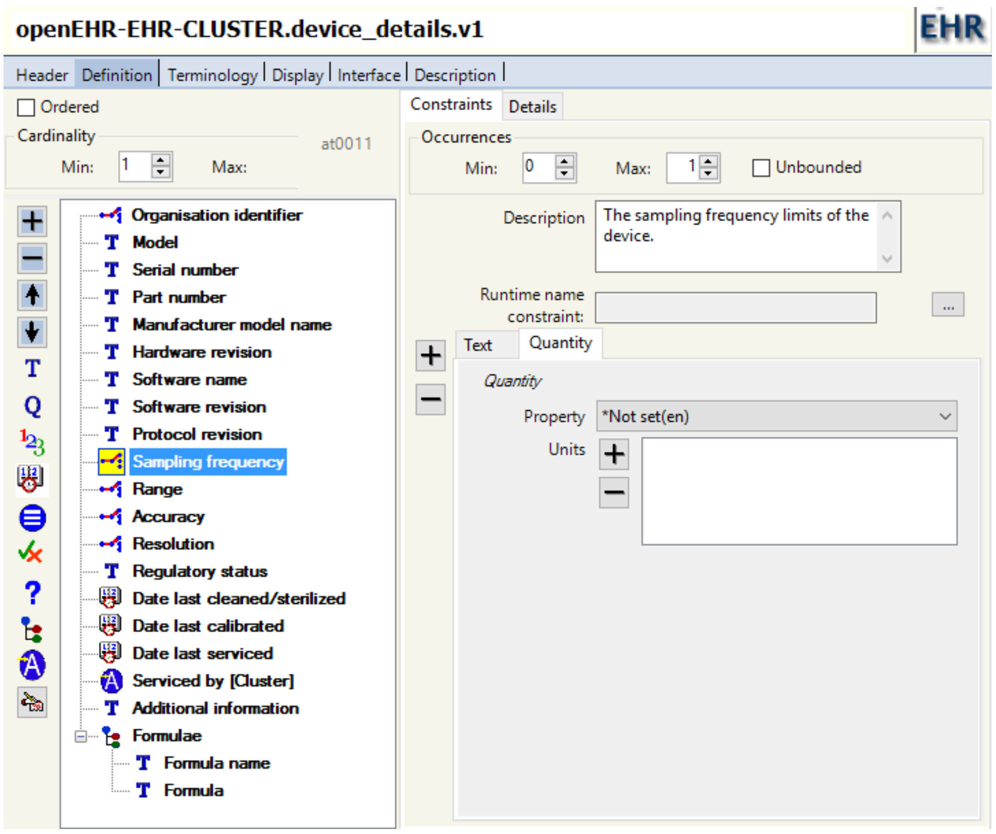
Occurrences Min: 0 Max: 1 ☐ Unbounded

Description: Identification of the device. This is very likely to be a coded field, including pre- or post-coordianted

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

Figure A.11 Definition of the *Device* archetype (*Cluster* RM); the archetype is designed by openEHR and taken from CKM


openEHR-EHR-CLUSTER.device_details.v1

☐ Ordered

Cardinality — Min: 1 Max:

at0011

Occurrences — Min: 0 Max: 1 ☐ Unbounded

- Organisation identifier
- Model
- Serial number
- Part number
- Manufacturer model name
- Hardware revision
- Software name
- Software revision
- Protocol revision
- Sampling frequency
- Range
- Accuracy
- Resolution
- Regulatory status
- Date last cleaned/sterilized
- Date last calibrated
- Date last serviced
- Serviced by [Cluster]
- Additional information
- Formulae
  - Formula name
  - Formula

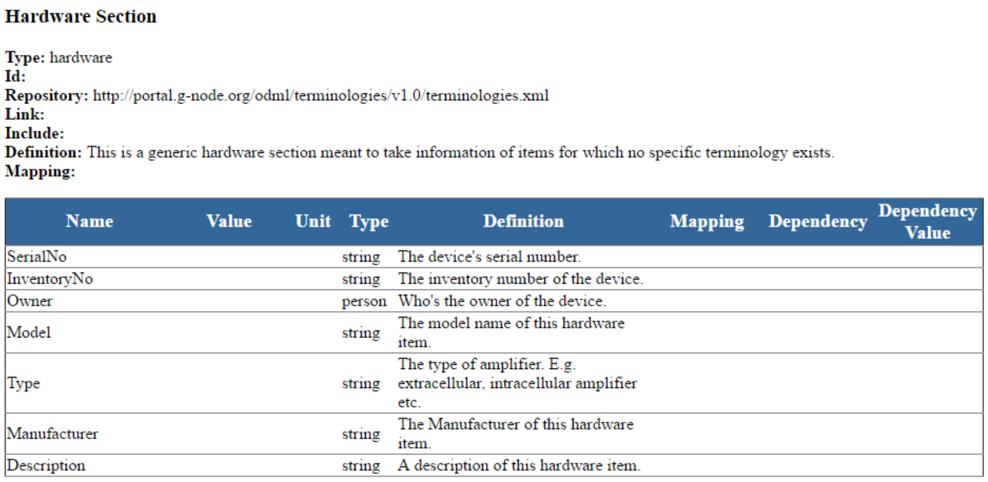
Description: The sampling frequency limits of the device.

Runtime name constraint:

Property: *Not set(en)

Units:

Figure A.12 Definition of the *Device details* archetype (*Cluster* RM); the archetype is designed by openEHR and taken from CKM

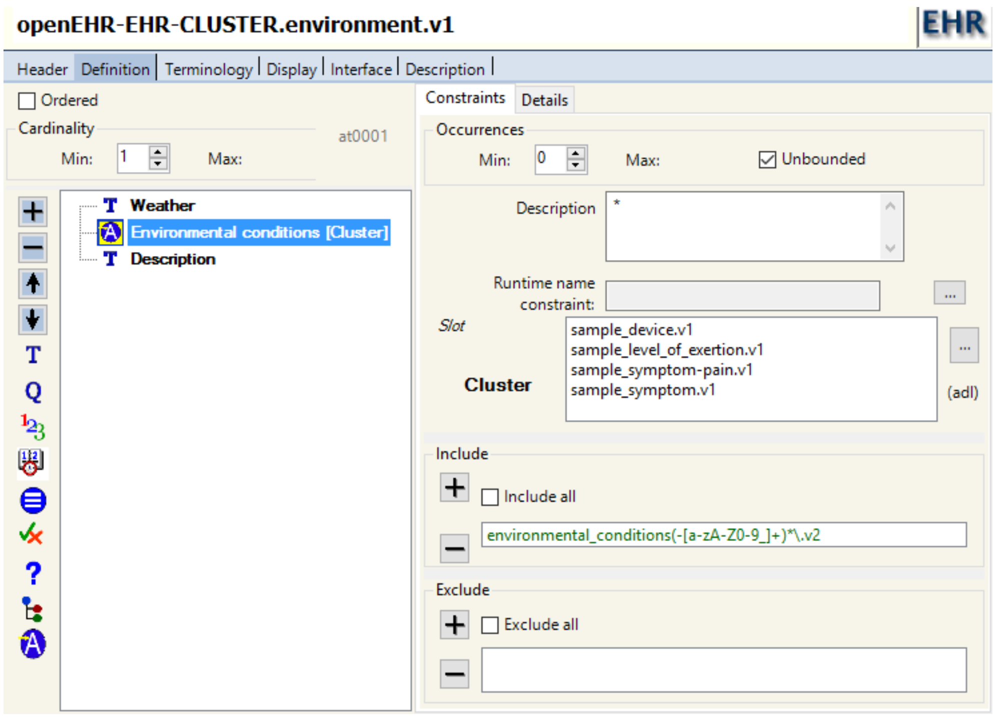

Hardware Section

Type: hardware
Id:
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: This is a generic hardware section meant to take information of items for which no specific terminology exists.
Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
SerialNo			string	The device's serial number.			
InventoryNo			string	The inventory number of the device.			
Owner			person	Who's the owner of the device.			
Model			string	The model name of this hardware item.			
Type			string	The type of amplifier. E.g. extracellular, intracellular amplifier etc.			
Manufacturer			string	The Manufacturer of this hardware item.			
Description			string	A description of this hardware item.			

Figure A.13 Specification of the odML *Hardware* section, which is expressible by *Device* and *Device details* archetypes (Figure A.11 and A.12)

Environment and environmental conditions archetypes


openEHR-EHR-CLUSTER.environment.v1

☐ Ordered

Cardinality — Min: 1 Max:

at0001

- Weather
- Environmental conditions [Cluster]
- Description

Occurrences — Min: 0 Max: ☑ Unbounded

Description: *

Runtime name constraint:

Slot

Cluster

sample_device.v1
sample_level_of_exertion.v1
sample_symptom-pain.v1
sample_symptom.v1

(adl)

Include

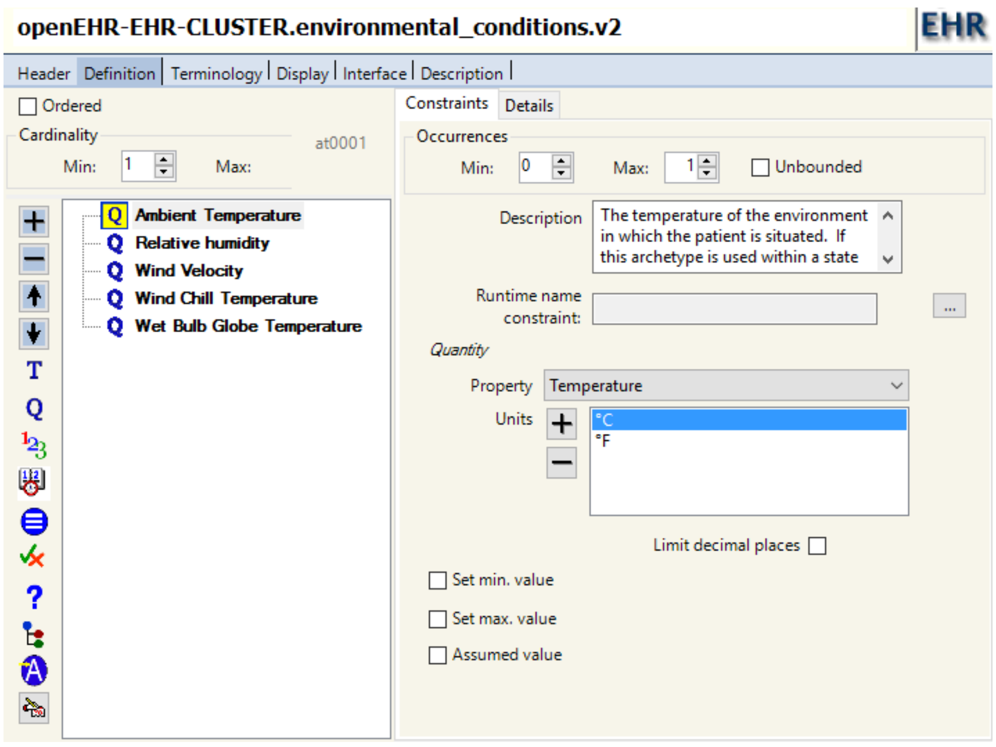
☐ Include all

environmental_conditions(-[a-zA-Z0-9_]+)*\.v2

Exclude

☐ Exclude all

Figure A.14 Definition of the *Environment* archetype (*Cluster* RM), which corresponds to the odML *Environment* section (Figure A.16)


openEHR-EHR-CLUSTER.environmental_conditions.v2

Header | Definition | Terminology | Display | Interface | Description

☐ Ordered

Cardinality Min: 1 Max:

at0001

- Ambient Temperature
- Relative humidity
- Wind Velocity
- Wind Chill Temperature
- Wet Bulb Globe Temperature

Constraints | Details

Occurrences Min: 0 Max: 1 ☐ Unbounded

Description: The temperature of the environment in which the patient is situated. If this archetype is used within a state

Runtime name constraint:

Quantity

Property: Temperature

Units: °C, °F

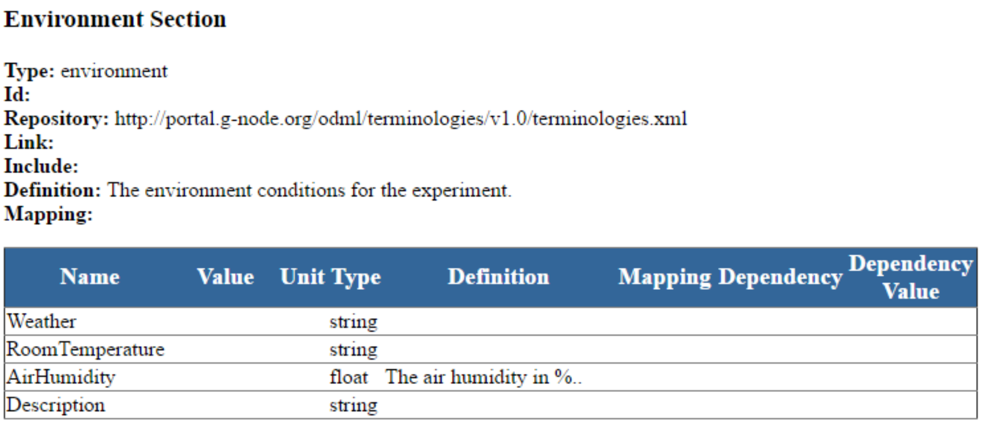
Limit decimal places ☐

☐ Set min. value

☐ Set max. value

☐ Assumed value

Figure A.15 Definition of the *Environmental conditions* archetype (*Cluster* RM); the archetype is designed by openEHR and taken from CKM

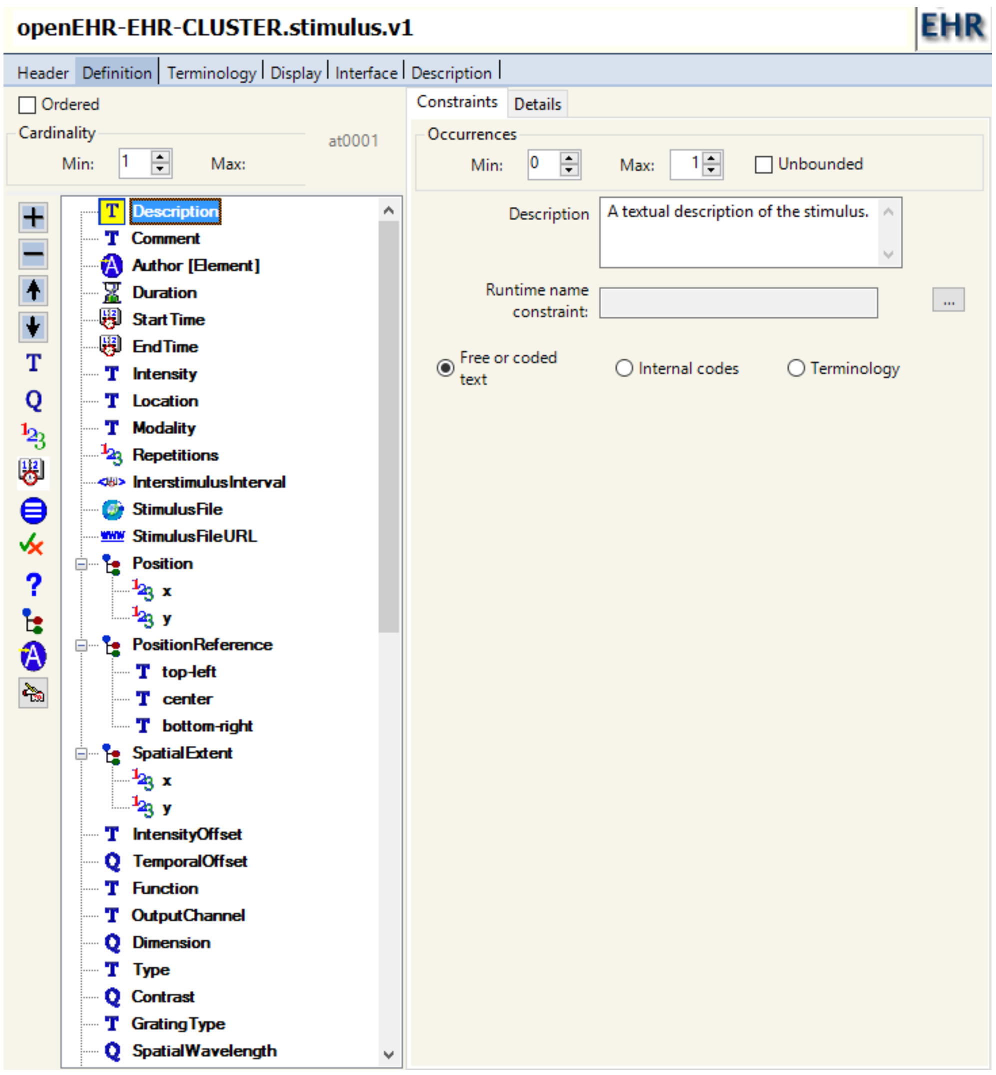

Environment Section

Type: environment
Id:
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: The environment conditions for the experiment.
Mapping:

Name	Value	Unit Type	Definition	Mapping Dependency	Dependency Value
Weather		string			
RoomTemperature		string			
AirHumidity		float	The air humidity in %..		
Description		string			

Figure A.16 Specification of the odML *Environment* section

Stimuli archetypes



openEHR-EHR-CLUSTER.stimulus.v1

Header | Definition | Terminology | Display | Interface | Description

Ordered: ☐

Cardinality — Min: 1 Max:

at0001

Constraints | Details

Occurrences — Min: 0 Max: 1 Unbounded: ☐

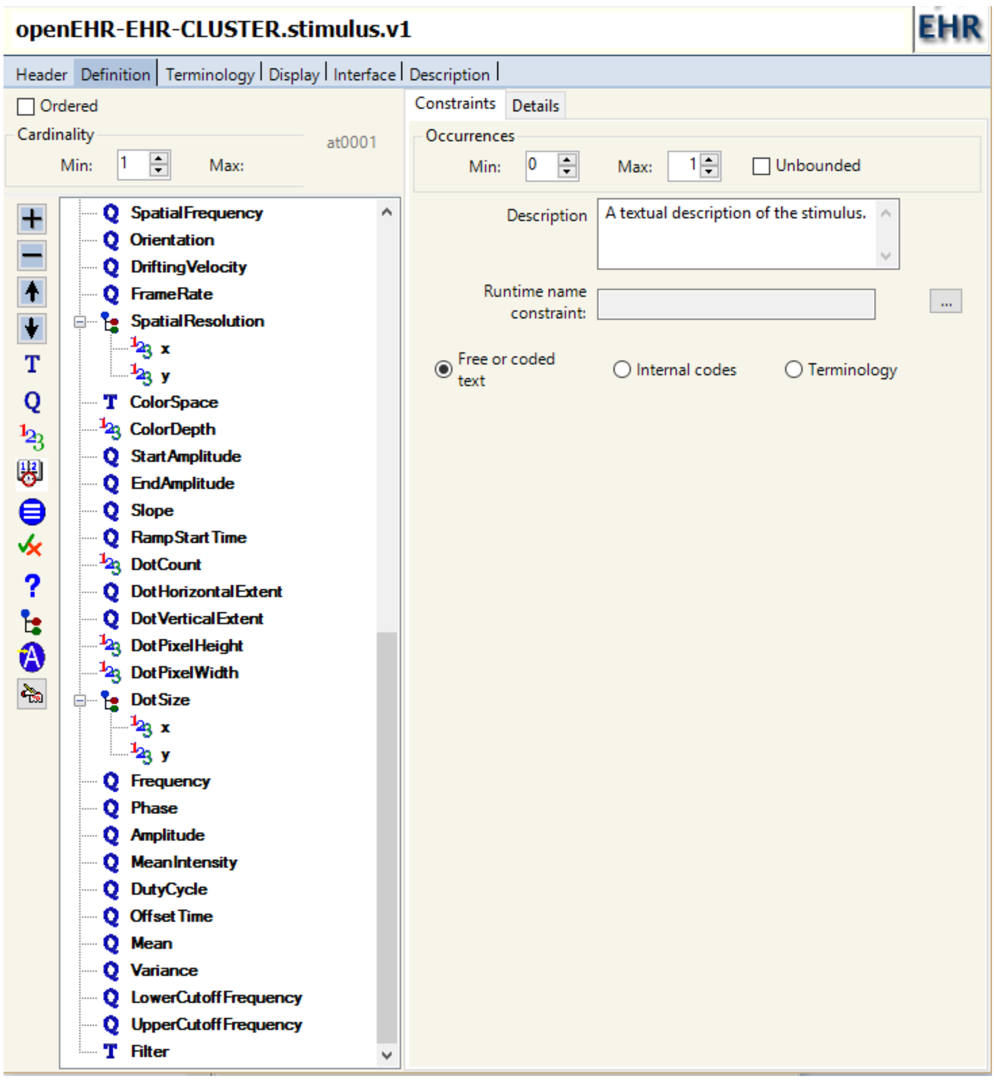
Description: A textual description of the stimulus.

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

- Description
- Comment
- Author [Element]
- Duration
- Start Time
- End Time
- Intensity
- Location
- Modality
- Repetitions
- InterstimulusInterval
- StimulusFile
- StimulusFileURL
- Position
  - x
  - y
- PositionReference
  - top-left
  - center
  - bottom-right
- SpatialExtent
  - x
  - y
- IntensityOffset
- TemporalOffset
- Function
- OutputChannel
- Dimension
- Type
- Contrast
- GratingType
- SpatialWavelength

Figure A.17 Definition of the *Stimulus* archetype part 1, (*Cluster* RM), which corresponds to the union of the various stimuli sections (Figure A.19 - A.30) in the odML terminology


openEHR-EHR-CLUSTER.stimulus.v1

Header | Definition | Terminology | Display | Interface | Description

☐ Ordered

Cardinality

Min: 1 Max:

at0001

Constraints | Details

Occurrences

Min: 0 Max: 1 ☐ Unbounded

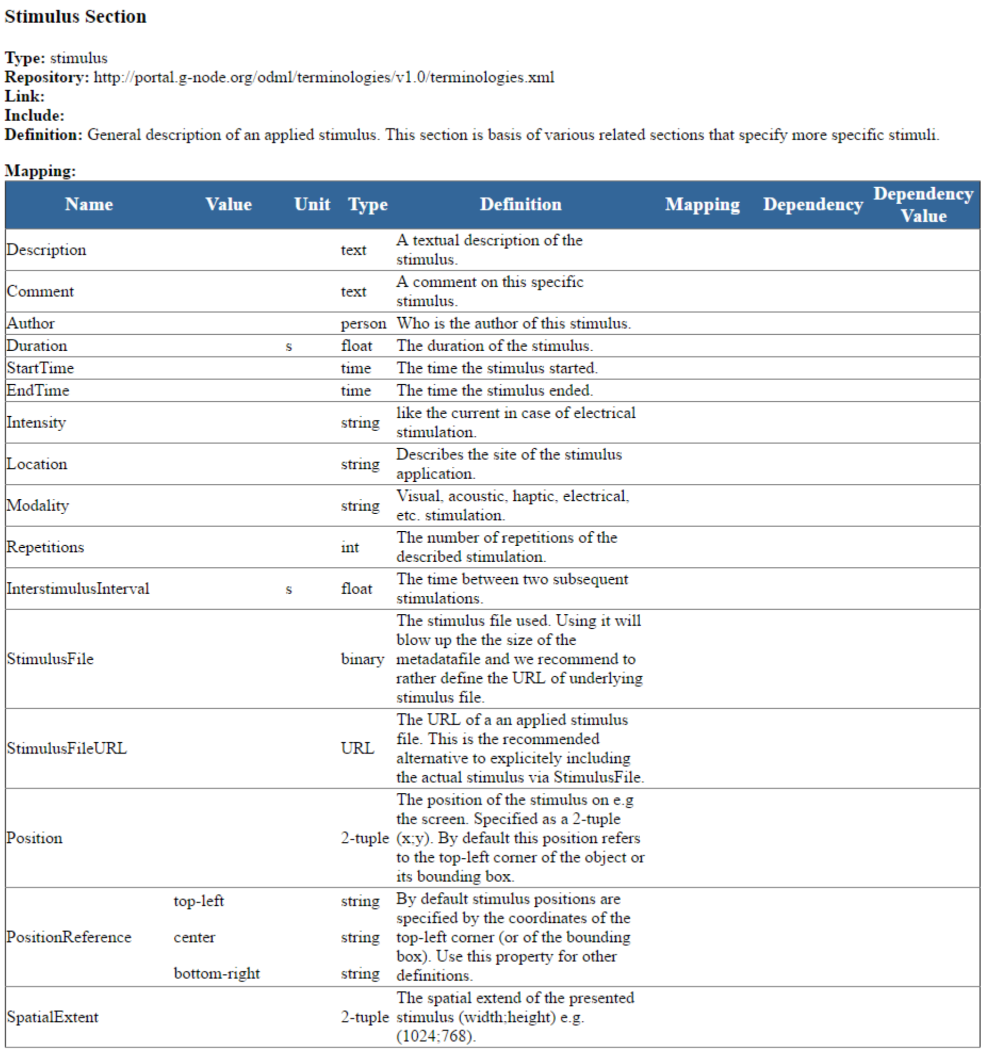
Description: A textual description of the stimulus.

Runtime name constraint:

[✓ Free or coded text] Internal codes Terminology

- SpatialFrequency
- Orientation
- DriftingVelocity
- FrameRate
- SpatialResolution
  - x
  - y
- ColorSpace
- ColorDepth
- StartAmplitude
- EndAmplitude
- Slope
- RampStartTime
- DotCount
- DotHorizontalExtent
- DotVerticalExtent
- DotPixelHeight
- DotPixelWidth
- DotSize
  - x
  - y
- Frequency
- Phase
- Amplitude
- MeanIntensity
- DutyCycle
- OffsetTime
- Mean
- Variance
- LowerCutoffFrequency
- UpperCutoffFrequency
- Filter

Figure A.18 Definition of the *Stimulus* archetype part 2, (*Cluster* RM), which corresponds to the union of the various stimuli sections (Figure A.19 - A.30) in the odML terminology



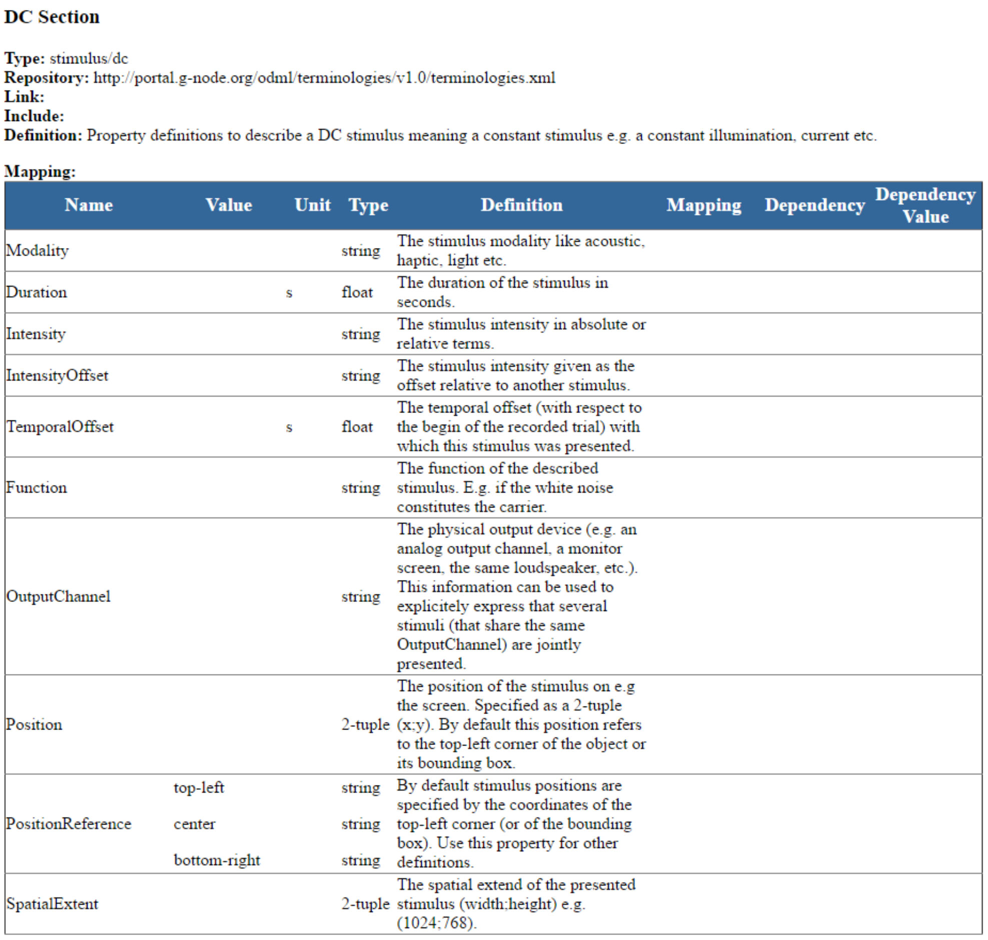
Stimulus Section

Type: stimulus
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: General description of an applied stimulus. This section is basis of various related sections that specify more specific stimuli.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Description			text	A textual description of the stimulus.			
Comment			text	A comment on this specific stimulus.			
Author			person	Who is the author of this stimulus.			
Duration		s	float	The duration of the stimulus.			
StartTime			time	The time the stimulus started.			
EndTime			time	The time the stimulus ended.			
Intensity			string	like the current in case of electrical stimulation.			
Location			string	Describes the site of the stimulus application.			
Modality			string	Visual, acoustic, haptic, electrical, etc. stimulation.			
Repetitions			int	The number of repetitions of the described stimulation.			
InterstimulusInterval		s	float	The time between two subsequent stimulations.			
StimulusFile			binary	The stimulus file used. Using it will blow up the the size of the metadatafile and we recommend to rather define the URL of underlying stimulus file.			
StimulusFileURL			URL	The URL of a an applied stimulus file. This is the recommended alternative to explicitely including the actual stimulus via StimulusFile.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left		string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
	center		string				
	bottom-right		string				
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.19 Specification of the odML *Stimulus* section



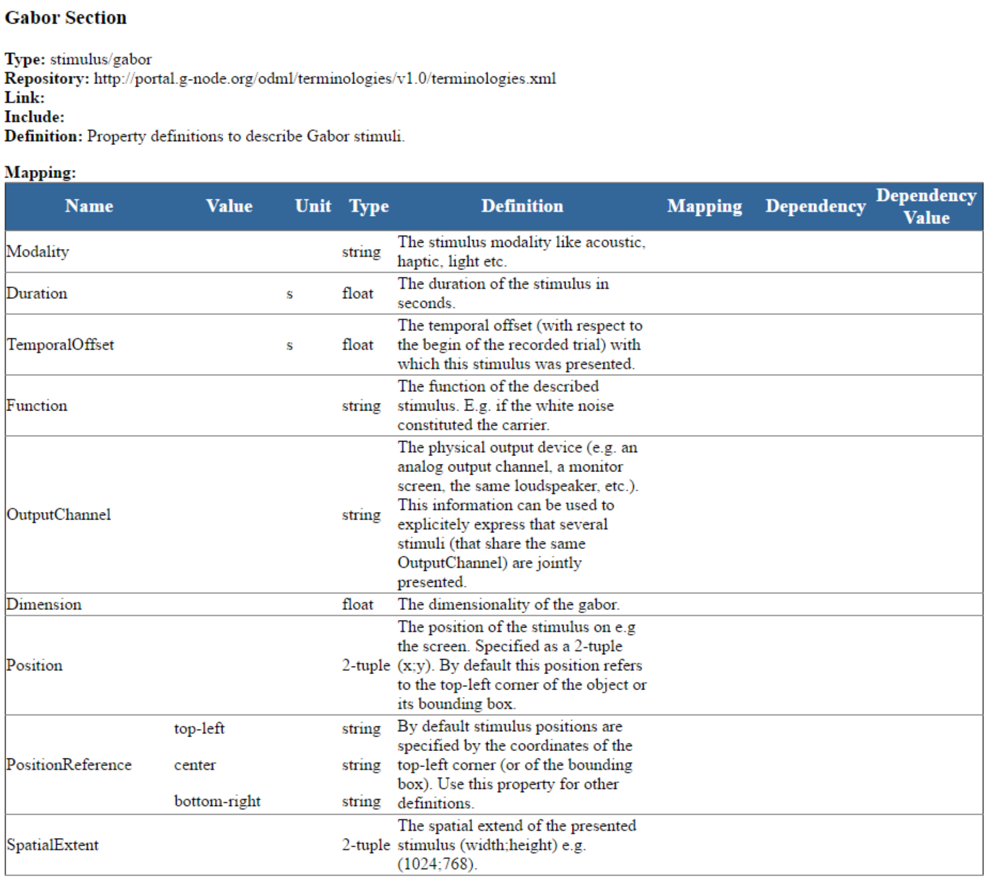
DC Section

Type: stimulus/dc
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Property definitions to describe a DC stimulus meaning a constant stimulus e.g. a constant illumination, current etc.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
Intensity			string	The stimulus intensity in absolute or relative terms.			
IntensityOffset			string	The stimulus intensity given as the offset relative to another stimulus.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constitutes the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.20 Specification of the odML *DC stimulus* section



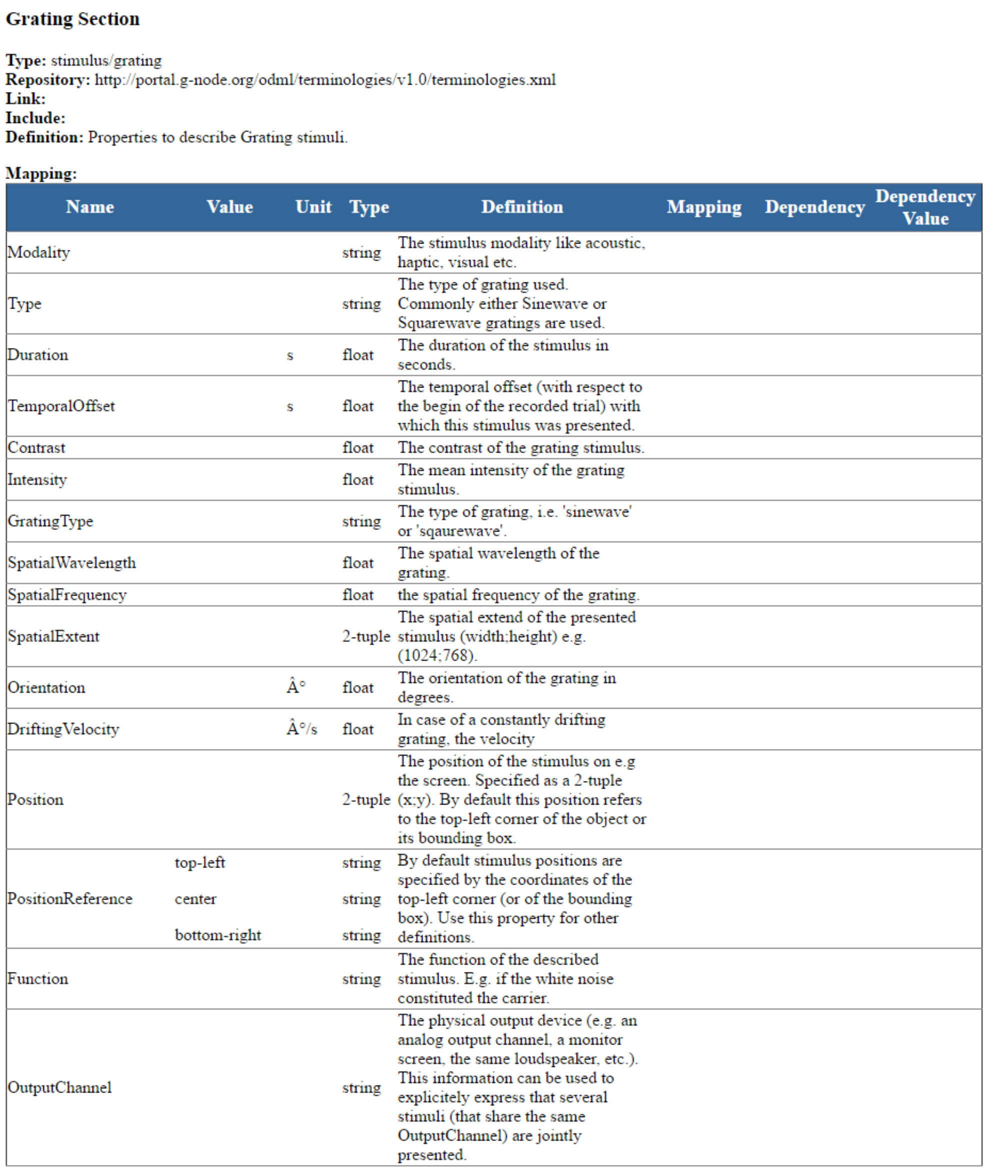
Gabor Section

Type: stimulus/gabor
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Property definitions to describe Gabor stimuli.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Dimension			float	The dimensionality of the gabor.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.21 Specification of the odML *Gabor stimulus* section



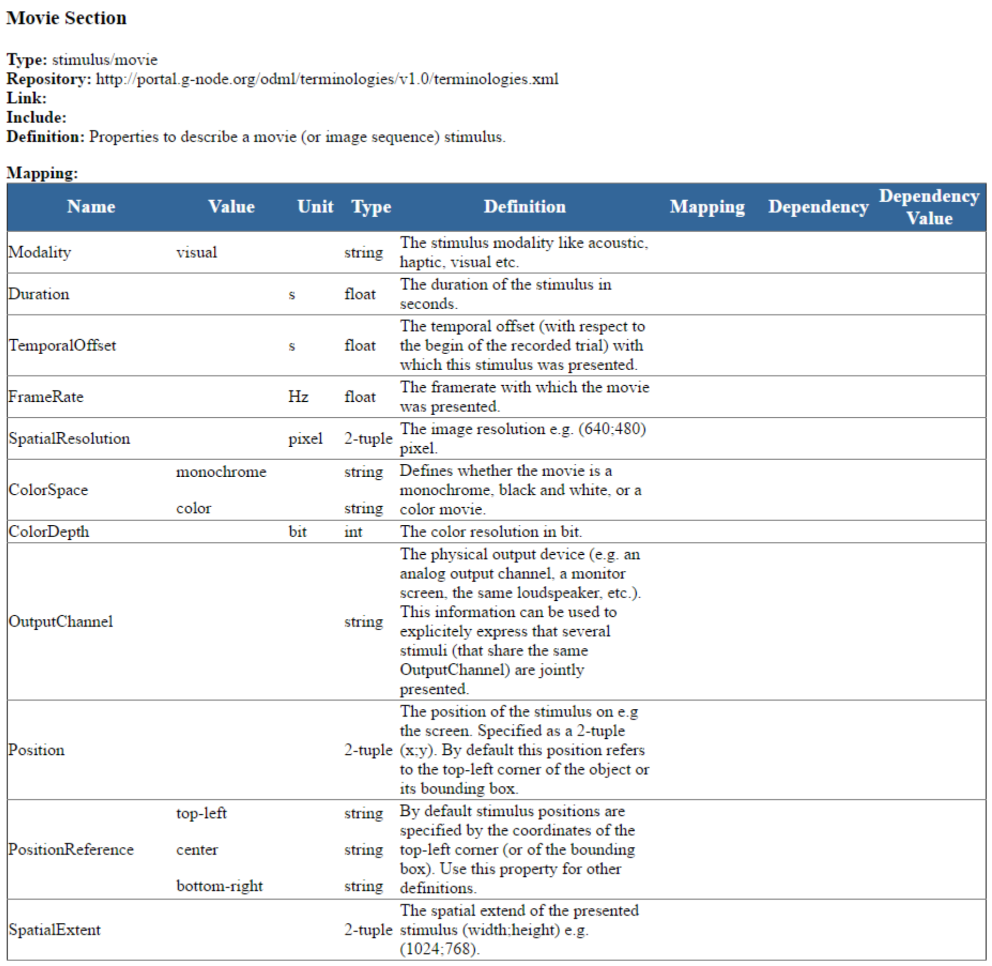
Grating Section

Type: stimulus/grating
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Properties to describe Grating stimuli.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, visual etc.			
Type			string	The type of grating used. Commonly either Sinewave or Squarewave gratings are used.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Contrast			float	The contrast of the grating stimulus.			
Intensity			float	The mean intensity of the grating stimulus.			
GratingType			string	The type of grating, i.e. 'sinewave' or 'sqaurewave'.			
SpatialWavelength			float	The spatial wavelength of the grating.			
SpatialFrequency			float	the spatial frequency of the grating.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			
Orientation		Â°	float	The orientation of the grating in degrees.			
DriftingVelocity		Â°/s	float	In case of a constantly drifting grating, the velocity			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			

Figure A.22 Specification of the odML *Grating stimulus* section



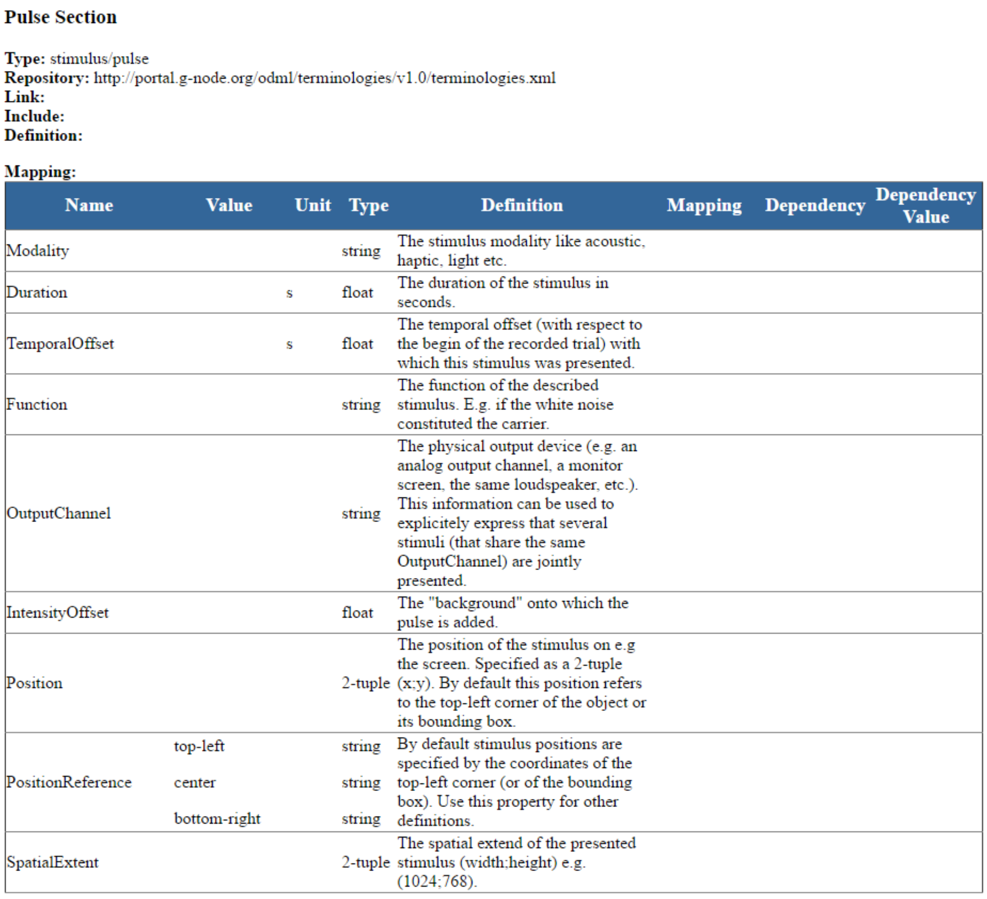
Movie Section

Type: stimulus/movie
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Properties to describe a movie (or image sequence) stimulus.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality	visual		string	The stimulus modality like acoustic, haptic, visual etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
FrameRate		Hz	float	The framerate with which the movie was presented.			
SpatialResolution		pixel	2-tuple	The image resolution e.g. (640;480) pixel.			
ColorSpace	monochrome		string	Defines whether the movie is a monochrome, black and white, or a color movie.			
	color		string				
ColorDepth		bit	int	The color resolution in bit.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left		string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
	center		string				
	bottom-right		string				
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.23 Specification of the odML *Movie stimulus* section



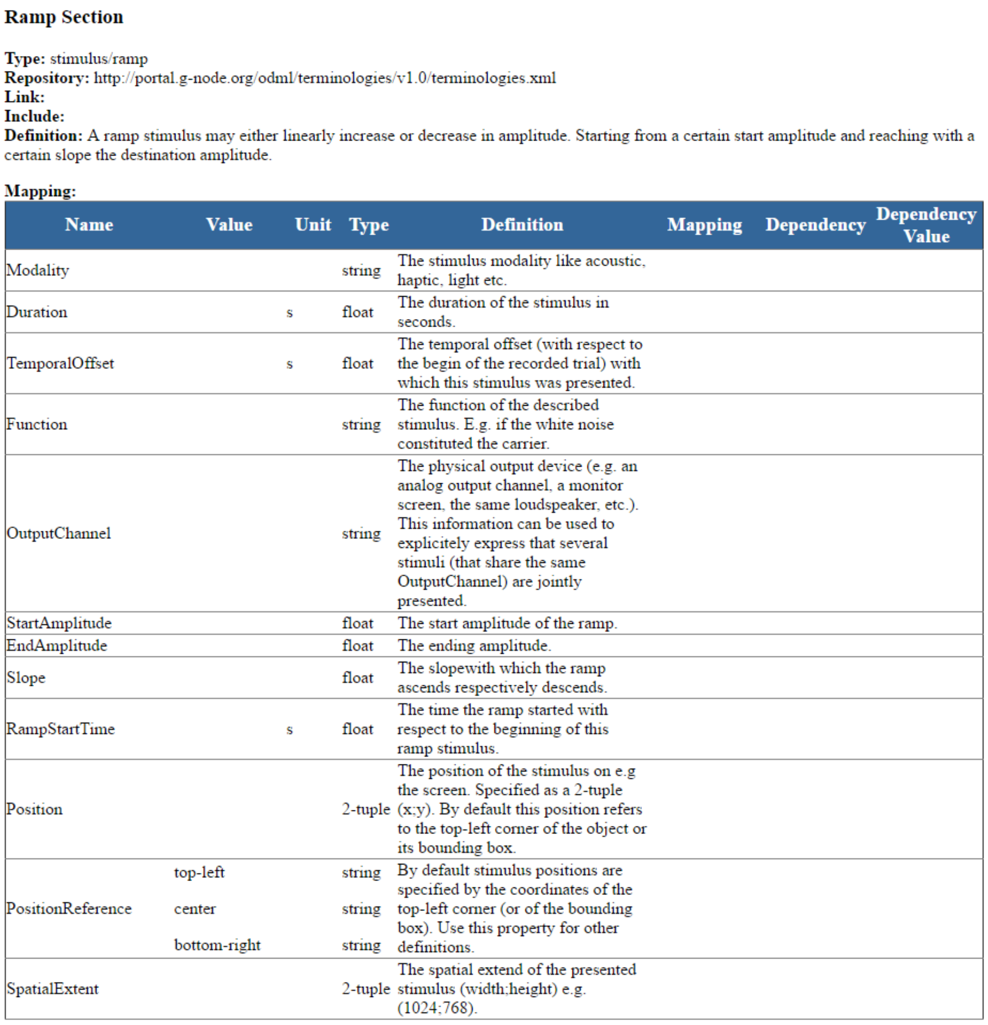
Pulse Section

Type: stimulus/pulse
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition:

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
IntensityOffset			float	The "background" onto which the pulse is added.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.24 Specification of the odML *Pulse stimulus* section



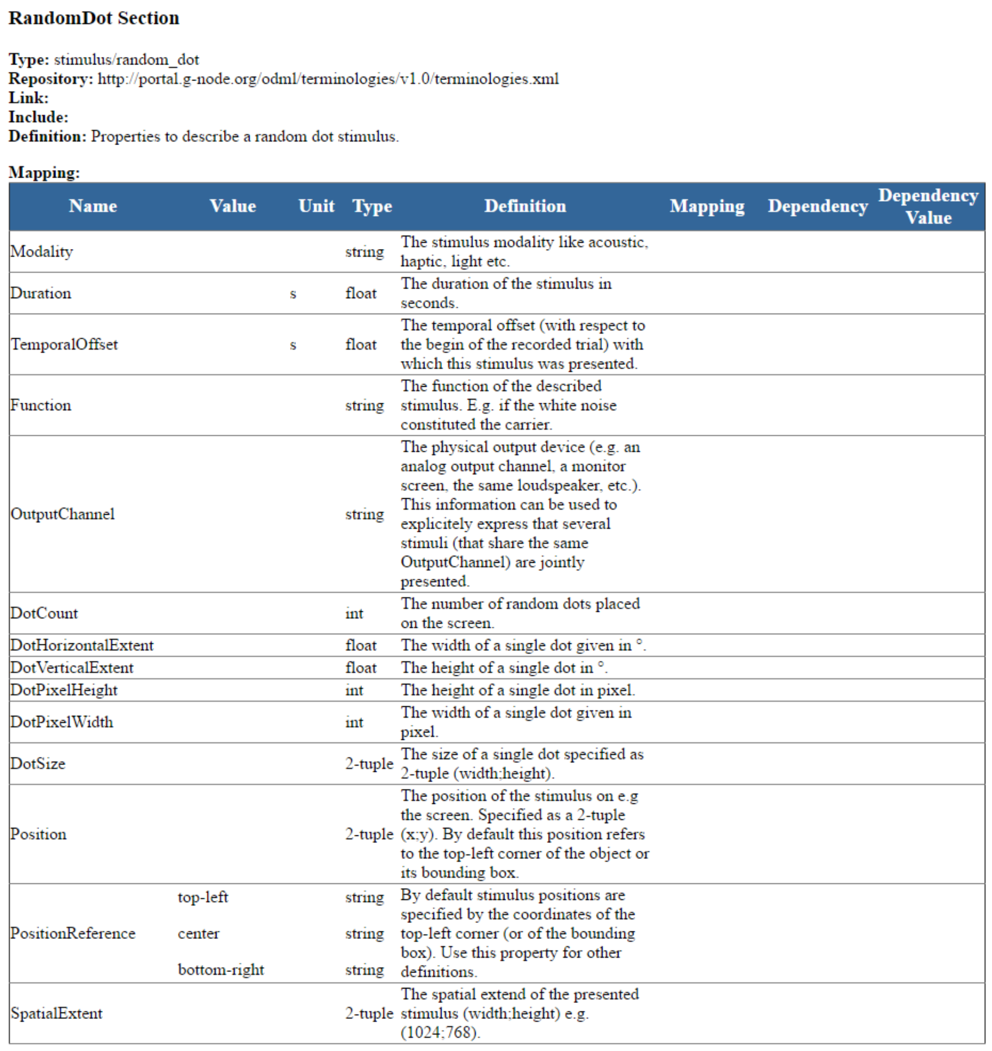
Ramp Section

Type: stimulus/ramp
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: A ramp stimulus may either linearly increase or decrease in amplitude. Starting from a certain start amplitude and reaching with a certain slope the destination amplitude.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
StartAmplitude			float	The start amplitude of the ramp.			
EndAmplitude			float	The ending amplitude.			
Slope			float	The slopewith which the ramp ascends respectively descends.			
RampStartTime		s	float	The time the ramp started with respect to the beginning of this ramp stimulus.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.25 Specification of the odML *Ramp stimulus* section



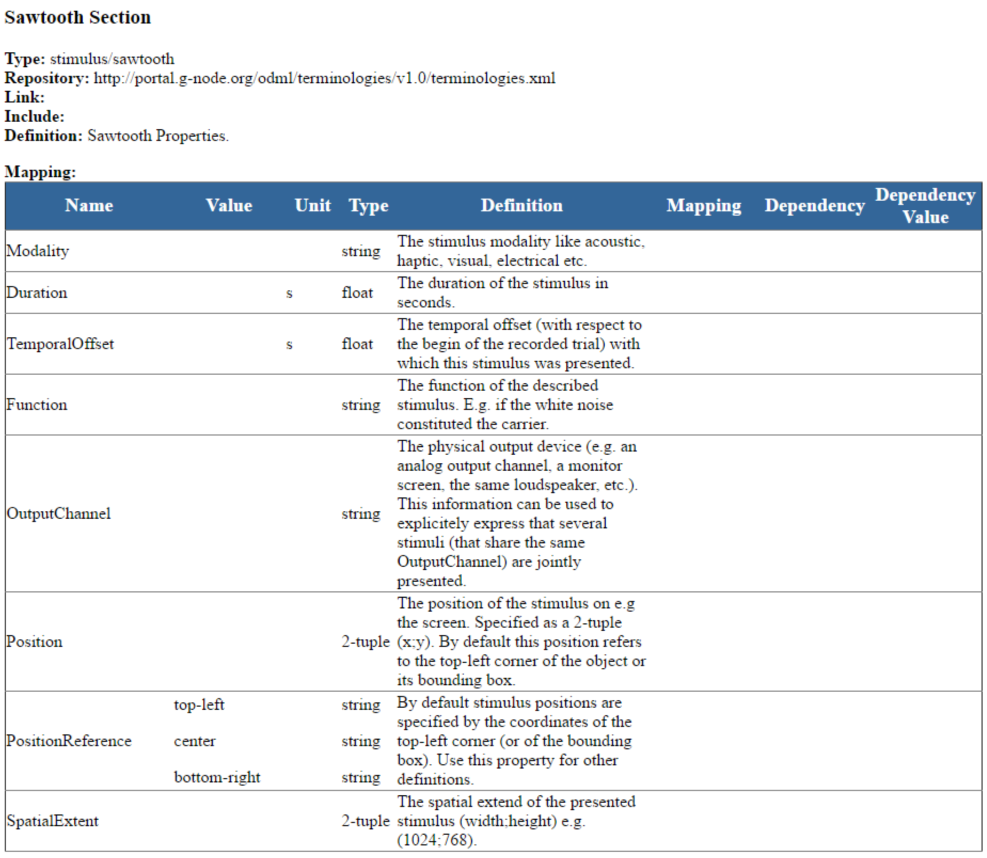
RandomDot Section

Type: stimulus/random_dot
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Properties to describe a random dot stimulus.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
DotCount			int	The number of random dots placed on the screen.			
DotHorizontalExtent			float	The width of a single dot given in °.			
DotVerticalExtent			float	The height of a single dot in °.			
DotPixelHeight			int	The height of a single dot in pixel.			
DotPixelWidth			int	The width of a single dot given in pixel.			
DotSize			2-tuple	The size of a single dot specified as 2-tuple (width;height).			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.26 Specification of the odML *Random dot stimulus* section



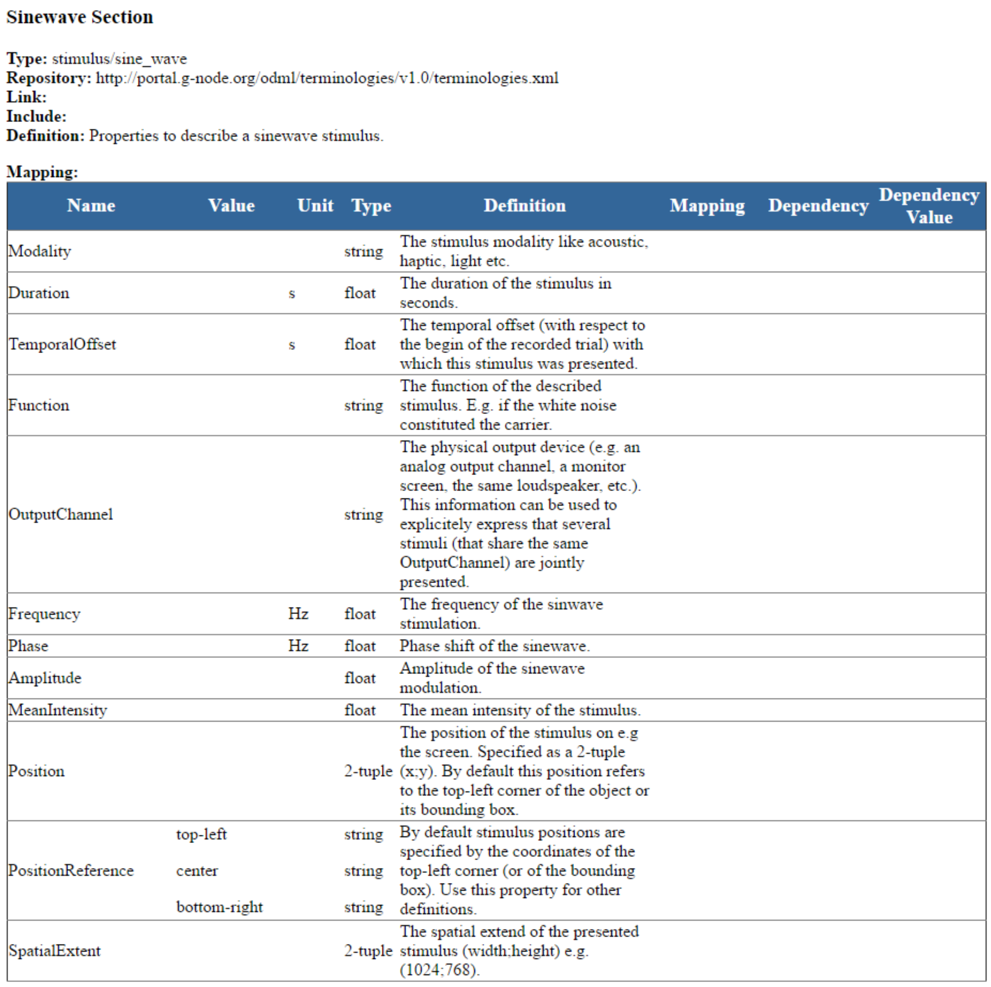
Sawtooth Section

Type: stimulus/sawtooth
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Sawtooth Properties.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, visual, electrical etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.27 Specification of the odML *Sawtooth stimulus* section



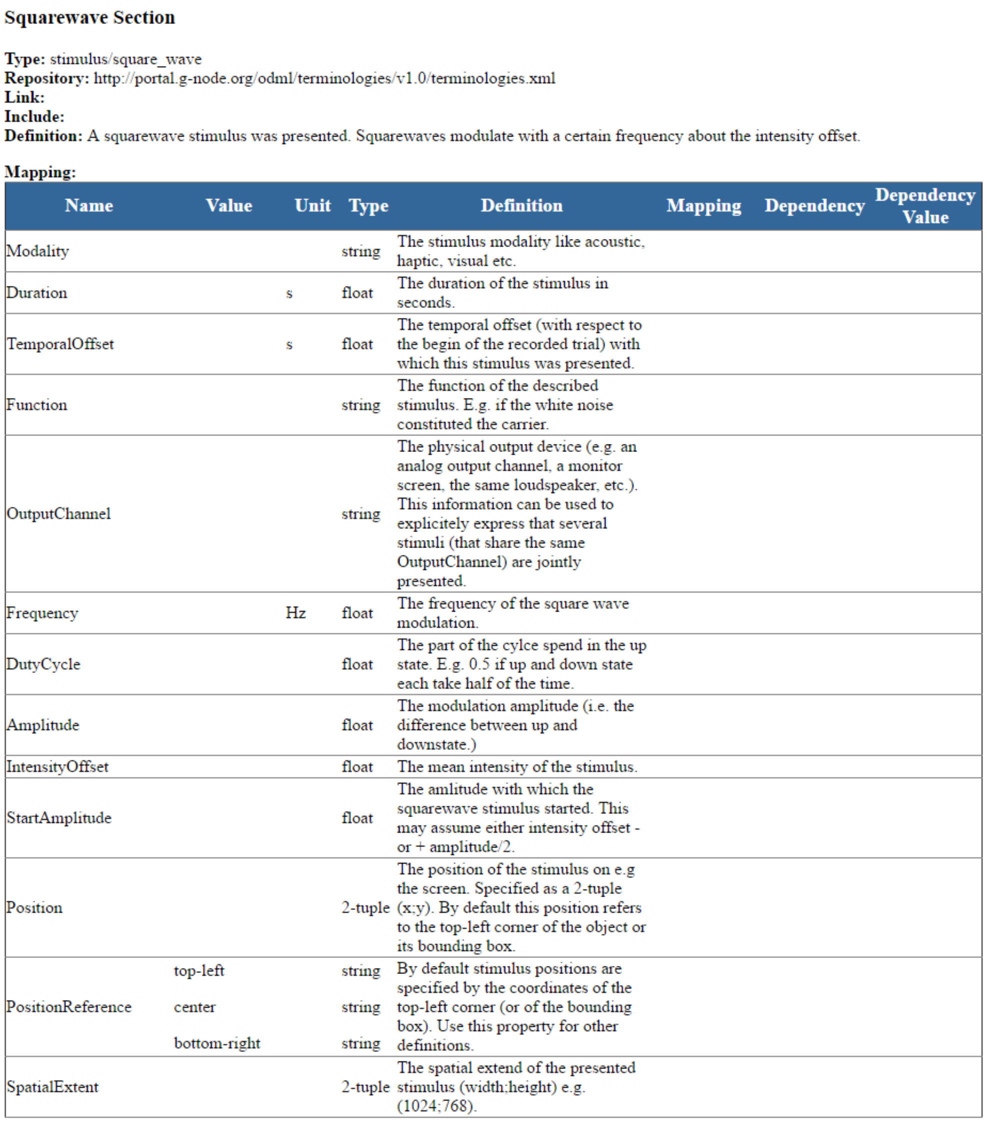
Sinewave Section

Type: stimulus/sine_wave
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Properties to describe a sinewave stimulus.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Frequency		Hz	float	The frequency of the sinwave stimulation.			
Phase		Hz	float	Phase shift of the sinewave.			
Amplitude			float	Amplitude of the sinewave modulation.			
MeanIntensity			float	The mean intensity of the stimulus.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.28 Specification of the odML *Sinewave stimulus* section



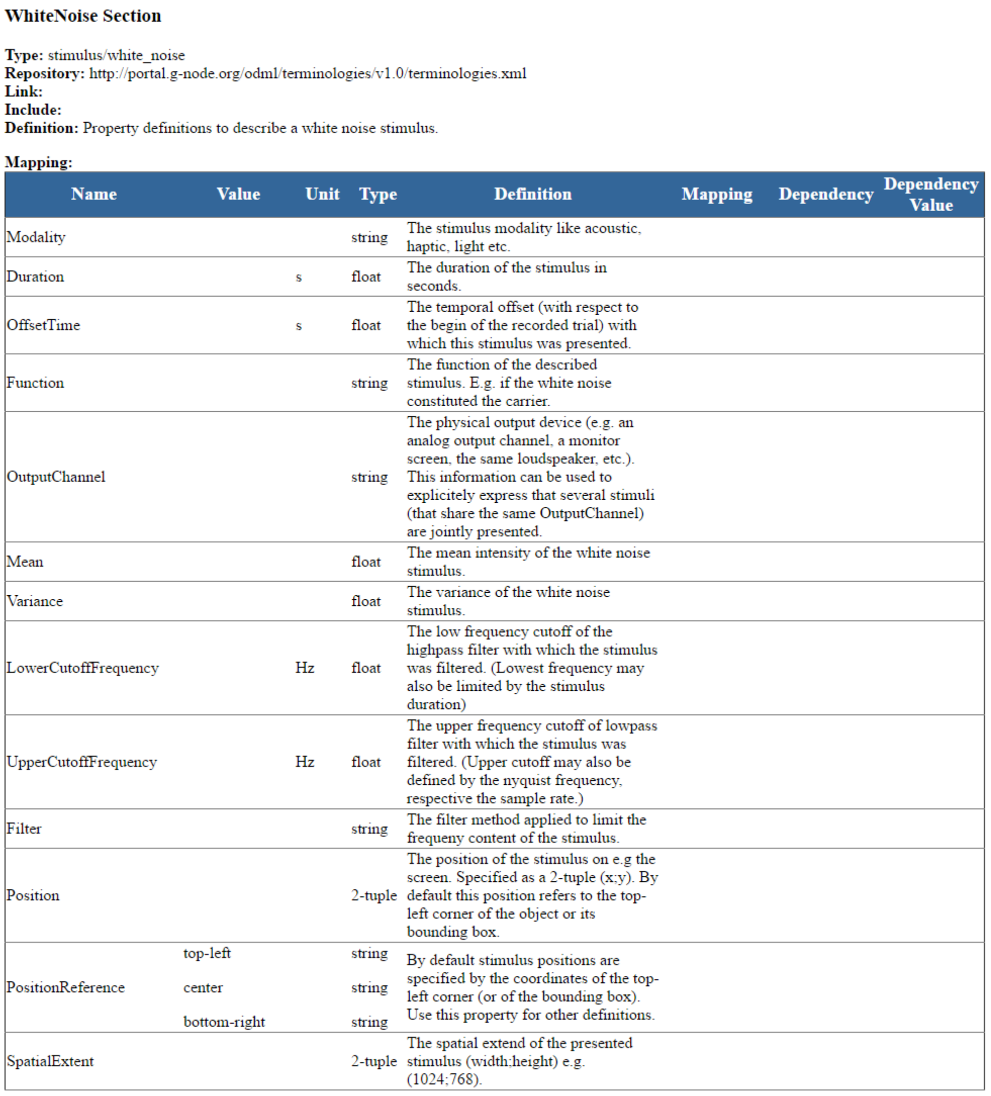
Squarewave Section

Type: stimulus/square_wave
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: A squarewave stimulus was presented. Squarewaves modulate with a certain frequency about the intensity offset.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, visual etc.			
Duration		s	float	The duration of the stimulus in seconds.			
TemporalOffset		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Frequency		Hz	float	The frequency of the square wave modulation.			
DutyCycle			float	The part of the cylce spend in the up state. E.g. 0.5 if up and down state each take half of the time.			
Amplitude			float	The modulation amplitude (i.e. the difference between up and downstate.)			
IntensityOffset			float	The mean intensity of the stimulus.			
StartAmplitude			float	The amlitude with which the squarewave stimulus started. This may assume either intensity offset - or + amplitude/2.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left center bottom-right		string string string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.29 Specification of the odML *Squarewave stimulus* section



WhiteNoise Section

Type: stimulus/white_noise
Repository: http://portal.g-node.org/odml/terminologies/v1.0/terminologies.xml
Link:
Include:
Definition: Property definitions to describe a white noise stimulus.

Mapping:

Name	Value	Unit	Type	Definition	Mapping	Dependency	Dependency Value
Modality			string	The stimulus modality like acoustic, haptic, light etc.			
Duration		s	float	The duration of the stimulus in seconds.			
OffsetTime		s	float	The temporal offset (with respect to the begin of the recorded trial) with which this stimulus was presented.			
Function			string	The function of the described stimulus. E.g. if the white noise constituted the carrier.			
OutputChannel			string	The physical output device (e.g. an analog output channel, a monitor screen, the same loudspeaker, etc.). This information can be used to explicitely express that several stimuli (that share the same OutputChannel) are jointly presented.			
Mean			float	The mean intensity of the white noise stimulus.			
Variance			float	The variance of the white noise stimulus.			
LowerCutoffFrequency		Hz	float	The low frequency cutoff of the highpass filter with which the stimulus was filtered. (Lowest frequency may also be limited by the stimulus duration)			
UpperCutoffFrequency		Hz	float	The upper frequency cutoff of lowpass filter with which the stimulus was filtered. (Upper cutoff may also be defined by the nyquist frequency, respective the sample rate.)			
Filter			string	The filter method applied to limit the frequeny content of the stimulus.			
Position			2-tuple	The position of the stimulus on e.g the screen. Specified as a 2-tuple (x;y). By default this position refers to the top-left corner of the object or its bounding box.			
PositionReference	top-left		string	By default stimulus positions are specified by the coordinates of the top-left corner (or of the bounding box). Use this property for other definitions.			
	center		string				
	bottom-right		string				
SpatialExtent			2-tuple	The spatial extend of the presented stimulus (width;height) e.g. (1024;768).			

Figure A.30 Specification of the odML *White noise stimulus* section

Appendix B

Author’s bibliography

Author’s bibliography

[1] Brůha, P., Papež, V., Bandrowski, A., Grewe, J., Mouček, R., Tripathy, S., et al. (2013). The ontology for experimental neurophysiology: a first step toward semantic annotations of neurophysiology data and metadata. In *Frontiers in Neuroinformatics*, volume 26. (Abstract, poster).

[2] Ježek, P., Mouček, R., Novotný, J., Papež, V., and Řondík, T. (2014). Infrastructure for electrophysiology research. In *SfN Neuroscience 2014 congress*, pages 611–616. (Abstract, poster).

[3] Ježek, P. and Papež, V. (2010). Systém pro správu erp experimentů. In *Kognice a umělý život X*, pages 177 – 180. Slezská univerzita v Opavě. Local conference, poster.

[4] Le Franc, Y., Bandrowski, A., Brůha, P., Papež, V., Grewe, J., Mouček, R., Tripathy, S. J., and Wachtler, T. (2014). Describing neurophysiology data and metadata with oen, the ontology for experimental neurophysiology. In *Frontiers in Neuroinformatics*, volume 8. (Abstract, poster).

[5] Mouček, R., Brůha, P., Ježek, P., Mautner, P., Novotný, J., Papež, V., Prokop, T., Řondík, T., Štěbeták, J., and Vařeka, L. (2014). Software and hardware infrastructure for research in electrophysiology. *Frontiers in Neuroinformatics*, 8(20). (**Impact jounal**).

[6] Mouček, R., Jaroš, P., Ježek, P., and Papež, V. (2011a). Software infrastructure for eeg/erp research. In *3rd International Conference on Knowledge Engineering and Ontology Development (KEOD)*, pages 478–481. (**ISI, Scopus conference**, poster).

[7] Mouček, R., Ježek, P., and Papež, V. (2010a). Deriving semantic web structures from eeg/erp data source. In *INCF Neuroinformatics congress 2010*. (Abstract, poster).

[8] Mouček, R., Ježek, P., and Papež, V. (2011b). Semantic web technologies in eeg/erp domain: Software solution. In *HEALTHINF 2011*, pages 618–621. (**ISI, Scopus conference**, poster).

[9] Mouček, R., Papež, V., and Ježek, P. (2010b). Prostředky sémantického webu v oblasti eeg a evokovaných potenciálů. In *Kognice a umělý život X*, pages 259–262. Slezská univerzita v Opavě. Local conference, poster.

[10] Papež, V. (2008). System for czech archery association. Master’s thesis, University of West Bohemia, Pilsen, Czech Republic. Bachelor thesis.

[11] Papež, V. (2010). Neuroinformatic database and semantic web. Master’s thesis, University of West Bohemia, Pilsen, Czech Republic. Master Thesis.

[12] Papež, V. (2012). Data and metadata models in eeg/erp domain. Technical report, University of West Bohemia, Pilsen, Czech Republic. Technical report.

[13] Papež, V. and Denaxas, S. (2017). Evaluating openehr for defining machine readable electronic health record phenotypes: lessons from the caliber resource. In *Informatics for Health 2017*, pages 539–543. (Abstract, presentation).

[14] Papež, V., Denaxas, S., and Hemingway, H. (2017a). Evaluating openehr for storing computable representations of electronic health record phenotyping algorithms. In *IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*. (**IEEE conference**, poster).

[15] Papež, V., Denaxas, S., and Hemingway, H. (2017b). Evaluation of semantic web technologies for storing computable definitions of electronic health records phenotyping algorithms. Unpublished, but accepted to American Medical Informatics Association (AMIA) 2017 Annual Symposium, (presentation).

[16] Papež, V., Jaroš, P., Suárez Araujo, C. P., Pérez del Pino, M. A., and García Báez, P. (2011). A new cognitive component of visualization tool for diagnosis and monitoring alzheimer's disease and other dementias: Cogvit. In *INCF Neuroinformatics 2011 congress*, volume 10. (Abstract, poster).

[17] Papež, V. and Ježek, P. (2010). Neuroinformatická databáze a sémantický web. In *Kognice a umělý život X*, pages 269 – 273. Slezská univerzita v Opavě. Local conference, presentation.

[18] Papež, V., Ježek, P., and Mouček, R. (2010). Nástroj pro automatický převod relační databáze do prostředků sémantického webu dbtransformer. Authorised software.

[19] Papež, V. and Mouček, R. (2012). Model of storage of erp protocols in eeg/erp portal. In *2012 5th International Conference on BioMedical Engineering and Informatics*, pages 1275–1279. (**Scopus conference**, poster).

[20] Papež, V. and Mouček, R. (2013). Data and metadata models in electrophysiology domain: Separation of data models into semantic hierarchy and its integration into eegbase. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 539–543. IEEE. (**IEEE, ISI, Scopus conference**, presentation).

[21] Papež, V. and Mouček, R. (2015a). Archetypes development in electrophysiology domain: Electroencephalography as a personal ehr system module. In *HEALTHINF 2015 - 8th International Conference on Health Informatics, Proceedings; Part of 8th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2015*, pages 611–616. (**Scopus conference**, poster).

[22] Papež, V. and Mouček, R. (2015b). Development and usage of odml based openehr archetypes in electroencephalography. In *INCF Neuroinformatics 2015 congress*, pages 539–543. (Abstract, poster).

[23] Papež, V. and Mouček, R. (2017). Applying an archetype-based approach to electroencephalography/event-related potential experiments in the eegbase resource. *Frontiers in Neuroinformatics*, 11. (**Impact jounal**).

[24] Papež, V., Mouček, R., Ježek, P., and Řondík, T. (2014). Effects of a personal electronic health record system on a reasoning of neuro-electrophysiology experiments. In *SfN Neuroscience 2014 congress*, pages 611–616. (Abstract, poster).