University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Data and Metadata models in EEG/ERP domain

The State of the Art and the Concept of Ph.D. Thesis

Václav Papež

Technical Report Number: DCSE/TR-2012-01

May 2012

Distribution: public

Abstract

This work summarizes the current state of data modelling in EEG/ERP domain. The progress of research investigation depends on sharing information, on proper terminology and on precisely described problems or tasks. EEG/ERP research is not an exception. Therefore the organization called INCF was established. Nowadays, INCF covers the national nodes all over the world and one of its purposes is to make proposals to share data across the nodes. This work describes the EEG/ERP domain and INCF solutions and efforts. Then it describes various ways to model data, to store data and to make transformation between data models. It deals with standard technologies (such as relational databases) as well as new technologies (such as ontologies and semantic web). Finally, it evaluates various data models and proposes how to make a standardized description of EEG/ERP domain. The raw data from measurement cannot be interpreted without the set of metadata. The current descriptions of a metadata structure do not cover all the necessary levels of abstraction. Therefore, the proposal of complete description of EEG/ERP domain will be the scope of Ph.D. thesis.

Content

1 Introduction

Electroencephalography (EEG) becomes during more than one hundred years of its existence very popular method in the domain of brain activity research. EEG is non-invasive and relatively cheap therefore it is possible to perform experiments not only in hospitals or special laboratories but also at universities or in small labs. Big boom of computers has opened new possibilities for EEG measurement and a new scientific field - Neuroinformatics. Neuroinformatics deals with experiments, signal processing and data management. Data management solves problems what data to store and how to store them. Nowadays, when the Internet is a common part of our lives, data sharing has to be solved as well. It is clear that data storage and data sharing go together. International Neuroinformatics Coordinating Facility (INCF) was established to cover important neuroinformatics research groups and to propose various data sharing solutions. INCF is a non-commercial organization; therefore, their goal is also an open access to data, metadata and analysis results.

ERP (Event-Related Potentials) is based on research of short brain activity evoked by stimulus. At the University of West Bohemia in Pilsen EEG/ERP is investigated – for example driver`s attention or development coordination disorders in children. Moreover, neuroinformatics group is a member of INCF.

During the experiment a lot of data have to be stored. Basically, data can be divided into two groups – output of measurement (raw signal or data decomposed in some complex structure) and metadata (information about an experiment protocol, participating persons, used hardware, conditions etc.). These data are stored in neuroinformatics databases. The databases are often accessible by the neuroinformatics portals. The EEG/ERP portal is developed at the University as well.

The core of the portal / database is a proper data model. Moreover, not only delimitation of the EEG/ERP model but also the technologies must be solved. Metadata are the key factor for data sharing. There are tendencies to make standardized metadata models, unified terminology or unified sharing systems. The progressing technology of semantic web and ontologies give new possibilities in the field of data sharing. Proposals of ontologies are now one of the main scopes of INCF.

This work first describes what EEG/ERP and Neuroinformatics is. Then it briefly describes neuroinformatics portals of other INCF nodes. The part dealing with data model follows. It describes the ways for data modelling. Chapter 5 discusses current state of metadata storages and it evaluates their suitability. The implementation of storage of EEG/ERP protocols is discussed as well. Chapter 6 describes transformation mechanism between mentioned models. In chapter 8 the scope of the Ph.D. thesis is described.

2 EEG/ERP Research

This chapter briefly describes what EEG is and how it is measured. Then we focus on EEG/ERP experiments.

2.1 Electroencephalography (EEG)

Human nervous system is divided into two parts – central nervous system (CNS) and peripheral nervous (PNS) system. CND contains brain and spinal cord; PNS contains the rest. Majority of nervous system is created by neurons.

Neuron is composed from three major parts (cell body, axon, dendrites) - Figure 1.

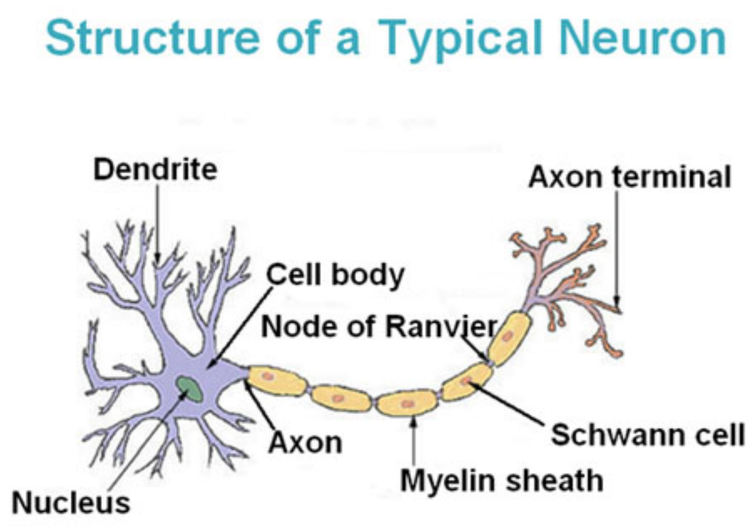

Figure 1: Neuron schema [1]

Very brief description of neural system function follows. Cell body contains enzymes and genetic material. Through the axon goes electrical charge. At the end of axon it activate the transmitters – chemicals that arouse dendrite of neighbour neuron across synapses. Next neuron transmits an excitation to another neuron etc. Basically the neuron has two states – it receives the signal or it is charged and it sends the signal. Some calculations say that total power of human nervous system equals to 60W bulb. Therefore in human brain is enough electrical charge that it can be measured across the bones of skull.

During the experiment the pairs of electrodes are attached to the scalp. Than the voltage difference between paired electrodes is measured and recorded in time. The rhythmic fluctuation of this potential difference is shown as peaks. The output of measurement is a waveform from each channel (pair of electrodes) [2].

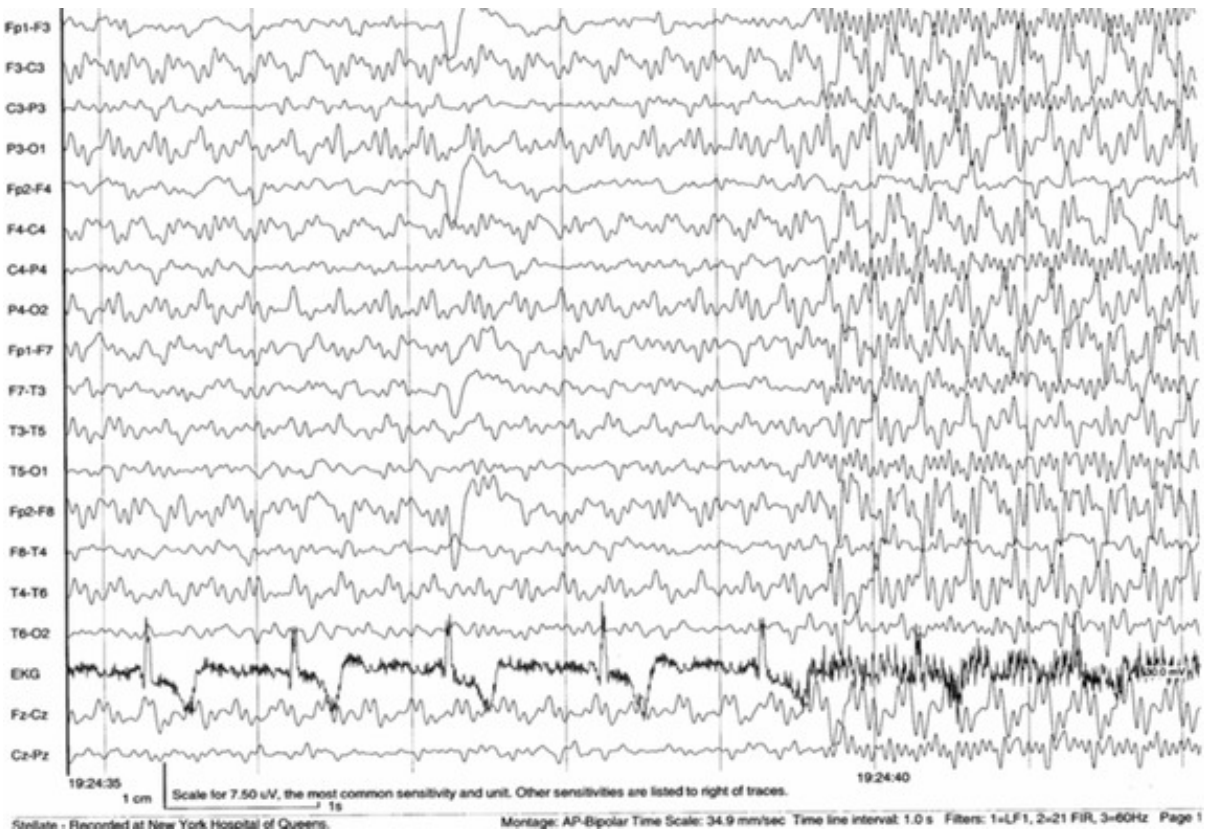

Figure 2: Example of EEG output [3]

2.2 Event-Related Potentials (ERP)

EEG provides very rough data because of its non-invasiveness. The signal on the scalp is muffled by skull. Moreover, the electrodes cannot be directed on one point, therefore they scan a lot of signals from wider area of brain at once. Due to this fact the signal has to be processed and important information mathematically extracted. One of the methods is Event-Related Potential (ERP)[1].

ERP method is based on evocation of some sense, cognitive or motoric potential and scans the brain reaction on this impulse. In 1964 neurophysiologist Grey Walter discovered the first cognitive component CNV (cognitive negative variation) that is evoked circa half of the second before the subject realizes the movement that he wants to do. Other components had been discovered soon.

The components are marked by letters - P (positive), N (negative) and C (polarity is not stable). After the letter the number follows. This number specifies delay between stimulus and occurred component (e.g. N1 – 100ms after stimulus; P3 – 300ms after stimulus). Each component is linked to the type of stimulus. If the component is evoked by more than one stimulus, they are incomparable (e.g. P1 evoked by acoustic stimulus and P1 evoked by visual stimulus). A list of important components can be found in [4]. Two frequently discussed components will be mentioned. N2 is negative component arising 200ms after stimulus. It is oriented to frequently repeated non-target stimulus (N2a) and non-frequently repeated target stimulus. P3 is a positive component arising 300ms after stimulus. It is oriented to unexpected event (P3a) or expected event, which is not frequent (P3b).

[1] Sometimes also Evoked Response Potential

Processing of output signal (the raw measured data) has following steps (in simplified presentation). A measured sample from an experiment is divided into epochs (areas around expected component). More epochs of the same type are averaged. Then mathematical apparatus for reducing artefacts (e.g. winking) is applied. In ideal case, a clear signal is obtained. Nevertheless, the signal has so much noise and artefacts and the output is not perfect. Therefore the research of signal processing is very important in this domain.

Over the time new technologies were developed such as Positron-Emission Tomography (PET) or Functional Magnetic Resonance Imaging (fMRI). Despite that EEG/ERP have not lost its place. Both previously mentioned methods have better results, but they are invasive. EEG/ERP with its non-invasiveness and low cost of laboratory is the only way to test humans safely (extracted from [4]).

2.3 EEG / ERP Laboratory

In order to perform experiments the laboratory has to contain special equipment. Ideal laboratory contains area where the subject is absolutely eliminated from all disturbing elements. At least two persons are presented during experiment – experimenter and subject. At the University of West Bohemia the laboratory consists of a soundproof cabin, 32 channels EEG recorder BrainAmp, BrainVision recording software, PresTi software for presenting experimental protocols, computer for playing protocols, computer for storing EEG data, USB adapter for connecting computers, EEG caps (active and passive) and seats. Apart this, new equipment is developed such as a car simulator. The following Figure 3 shows the schema of the laboratory.

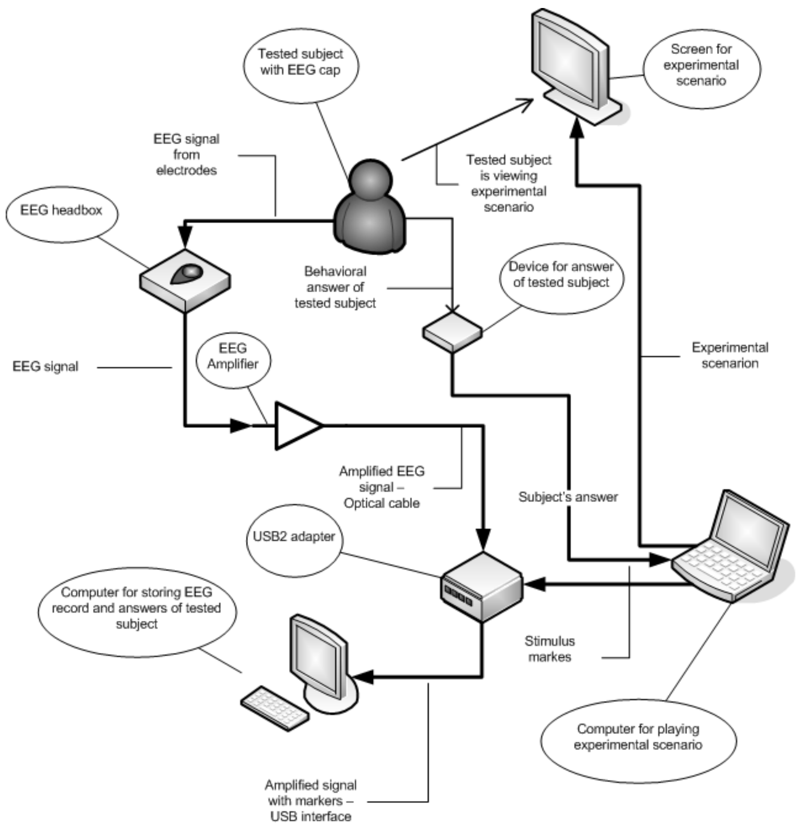
Tested subject with EEG cap
Screen for experimental scenario
EEG signal from electrodes
Tested subject is viewing experimental scenario
EEG headbox
Behavioral answer of tested subject
Device for answer of tested subject
EEG Amplifier
EEG signal
Experimental scenarion
Amplified EEG signal – Optical cable
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 3: EEG / ERP laboratory schema [5]

3 Neuroinformatics

Neuroinformatics is the scientific field combines the neuroscience and informatics. It is focused especially on application of computational methods and analytical tools and data storing. One of the clear definition of neuroinformatics specified International Neuroinformatics Coordination Facility.

3.1 International Neuroinformatics Coordination Facility (INCF)

INCF was established in 2005 in Stockholm. Its purpose is to lead and coordinate neuroinformatics research groups. INCF divides neuroinformatics into three groups.

- 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

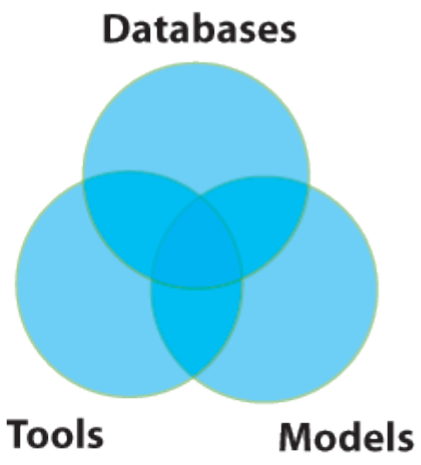

Figure 4: Development directions of neuroinformatics by INCF [6]

This work deals with the first point – data management and data sharing. On the 1st INCF Workshop on Sustainability of Neuroscience Databases [7] the following recommendations were proposed:

I. *INCF establishes a moderated web-based infrastructure with specific issues for discussion by the community*
II. *INCF engages peer-reviewed journals in the process of identifying domain-specific minimal information recommendations for the sharing and sustainability of neuroscience data*
III. *INCF 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*
IV. *INCF 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*
V. *INCF investigates existing neuroscience data/ tools/models clearinghouses and examines how they can engage in coordinating dissemination activities*
VI. *INCF examines how to serve as an accreditation body*
VII. *INCF can facilitate grass-roots recognition of need for data/database sustainability*

3.2 Neuroinformatics databases

Data from neuroinformatics research are stored in neuroinformatics databases. Basically the outputs of the experiment are data and metadata that describe the experiment (no matter what field of neuroinformatics is discussed).

3.2.1 Data

Data captured by recording machine are not strictly raw; they are stored in more or less structured formats. Two frequently used formats are the following ones (the list and theory is taken from [8]).

- **European Data Format**

The European Data Format (EDF and EDF+ for its extension) is a format for storage and exchange of multichannel signals. A data file consists of header including the metadata (ID of subject, ID of recording, time, number of records, number of signals in each records, type of signal, amplitude calibration or number of samples in each data record) and uninterrupted digitized polygraphic recording.

- **Vision Data Exchange Format**

Vision Data Exchange Format (VDEF) is focused on EEG. The record consist of three files:

  - Header file – describes the EEG in a text form
  - Marker file - describes marks in the signal
  - Raw data – contains raw data

3.2.2 Metadata

Except raw data the database must contain metadata about the experiment. Metadata could be divided into three groups in general:

- Character and content – general information about domain
- Context – information about data context with their environment
- Structure – information about internal data structure

The following example shows a comparison of part of metadata of EEG/ERP experiments and project based on prognosis of cortical dementias (such as Alzheimer`s disease). The first set of metadata is taken from the EEG/ERP portal at the *University of West Bohemia in Pilsen*. The second set of metadata is taken from the portal *Edevitalzh*, which is developed by research group *Comciencia* under *Universidad de las Palmas de Gran Canaria* in cooperation with *Universidad de La Laguna*.

Table 1: Metadata of EEG/ERP database and Edevitalzh comparison

• Metadata about experiment	• Metadata about consultations
○ **Attendant**	○ **Physician**
○ **Subject**	○ **Patient**
○ **Protocol**	○ **Test**
○ Weather	○ **Date**
○ Temperature	○ Dementia
○ Hardware	○ Institution
○ **Time of beginning**	○ **Notes**
○ **Time of the end**	• Metadata about patient

○ **Notes**	○ **Name**
• Metadata about subject	○ **Surname**
○ **Name**	○ **Date of Birth**
○ **Surname**	○ **Sex**
○ **Date of Birth**	○ **Contact**
○ **Sex**	○ Profession
○ **Contact**	○ Social background
• Metadata about attendant	• Metadata about physician
○ **Name**	○ **Name**
○ **Surname**	○ **Surname**
○ **Research group**	○ Sex
○ **Contact**	○ **Contact**
○ Privileges	○ Specialization
• Metadata about protocol	○ **Institute**
○ Research group	• Metadata about test
○ Owner	○ **Name**
○ **Name**	○ Number of answers
○ Description	
○ File containing protocol	
• Metadata about research group	
○ Owner	
○ Name	
○ Description	

Bolded metadata find their equivalent in both databases.

From the comparison of metadata sets is seen that a lot of metadata the databases have in common. The same situation occurs for the other neuroinformatics researches as well. Therefore metadata could be divided in the part, which the neuroinformatics has in common and should be standardized; and the second part, which is specified for a special purpose (e.g. EEG/ERP). The second part should be standardized as well but it is much more difficult to find proper solution for all research groups.

3.2.3 Protocols

Protocols are part of metadata nevertheless they deserve their own chapter.

It is not possible to precisely define how protocols look like. It depends on concrete experiment. The portal has to be prepared for protocols of every format and length. In fact, a lot of protocols are describable by a short XML[2] file (with a well-known structure). In general, we can divide protocols into two groups.

3.2.3.1 XML

XML protocols are protocols kept in a standard XML file. Majority of all protocols are in XML because it is the simplest and most comfortable way to work with them. XML protocols can be divided into two sub-groups.

[2] Extensible Markup Language - http://www.w3.org/XML/

- **With schema**

This group contains protocols where the structure of the file is known in advance. In that case it is possible to describe the structure by XSD[3] or DTD[4] files. Schemas specify data types terminology, logical structure, cardinalities and integrity constraints.

- **Without schema**

Protocols without schema are not so common. If some protocol has not got a schema, it means that it is a single unique experiment. It is expected that for more than one protocol of the same structure the schema will be created.

3.2.3.2 Non-XML

Non-XML protocols group contains every other protocol, which could not be categorized into the previous ones. This protocol type is not in the XML format (e.g. map description in racing game[5]). The protocol could be e.g. a plain text file or a binary file.

3.3 Neuroinformatics Portals

The following portals and databases are developed under INCF nodes. They provide a possibility to upload data into them. This list is just subset of all INCF registered portals.

3.3.1 CARMEN

CARMEN is the project of the UK Neuroinformatics Node. Its name is an abbreviation of *Code Analysis, Repository & Modelling for e-Neuroscience*. CARMEN is not a simple portal but it is virtual laboratory. It accompanies experimenter from the measurement till the analysis. What CARMEN provides is illustrated on structure schema in Figure 5.

[3] XML Schema Definiton - http://www.w3.org/XML/Schema
[4] Document Type Definition - http://www.w3.org/TR/SVG/svgdtd.html
[5] Real protocol from experiment – measurement of driver's attention

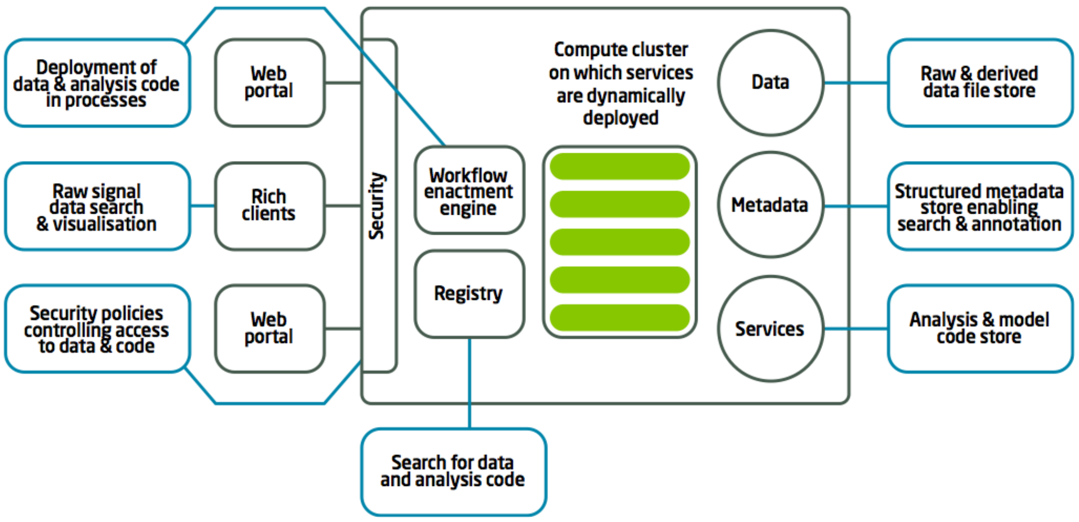
Deployment of data & analysis code in processes
Raw signal data search & visualisation
Security policies controlling access to data & code
Web portal
Rich clients
Web portal
Security
Workflow enactment engine
Registry
Compute cluster on which services are dynamically deployed
Data
Metadata
Services
Raw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

Figure 5: Structure of CARMEN [9]

In additional to other portals, CARMEN provides storage of services. Collaborators can uploaded their routines as web services and share them. These routines can be run on the stored data. The computing is executed on CARMENS machine.

3.3.2 G-Node Portal

G-Node portal is the portal developed by the German Neuroinformatics Node. Except expected functionality as storing and sharing experiments it provides an international neuroinformatics discussion forum. More about G-Node can be found at [10]. One of the principal activity of G-Node itself is proposing data and metadata models.

3.3.3 J-Node

INCF Japan Node provides a set of several platforms.

- Visiome Platform
- Brain Machine Interface Platform
- Invertebrate Brain Platform
- Neuro-Imaging Platform
- Dynamic Brain Platform
- Cerebellar Platform

More information about platforms can be found at the website of J-Node[6].

The software structure of J-Node is described in Figure 6.

[6] http://www.neuroinf.jp/platforms

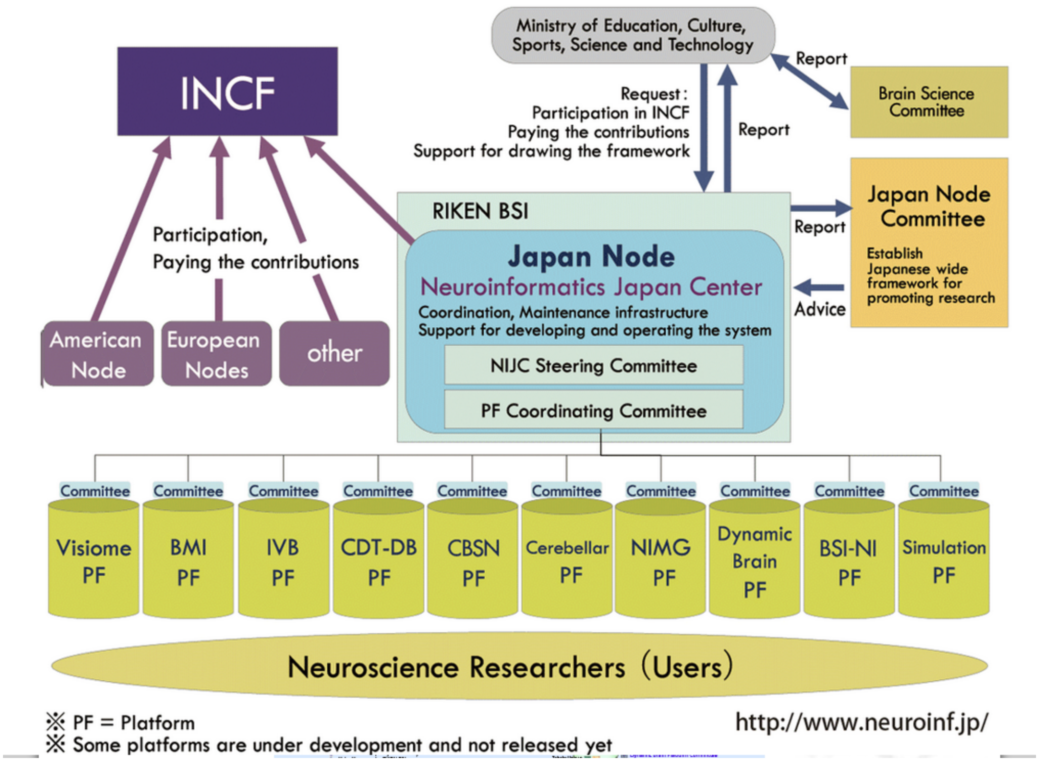
Ministry of Education, Culture, Sports, Science and Technology
Report
Brain Science Committee
INCF
Request : Participation in INCF Paying the contributions Support for drawing the framework
Report
RIKEN BSI
Report
Japan Node Committee
Establish Japanese wide framework for promoting research
Advice
Participation, Paying the contributions
Japan Node
Neuroinformatics Japan Center
Coordination, Maintenance infrastructure
Support for developing and operating the system
NIJC Steering Committee
PF Coordinating Committee
American Node
European Nodes
other
Committee
Visiome PF
BMI PF
IVB PF
CDT-DB PF
CBSN PF
Cerebellar PF
NIMG PF
Dynamic Brain PF
BSI-NI PF
Simulation PF
Neuroscience Researchers (Users)
※ PF = Platform
※ Some platforms are under development and not released yet
http://www.neuroinf.jp/

Figure 6: J-Node Infrastructure [11]

Cerebellar Platform is the data managing and sharing portal as previously mentioned ones.

3.3.4 BrainInfo

BrainInfo portal is focused on helping to identify structures in the brain by name and vice versa. BrainInfo Contains ontologies of human and mice brain atlases, downloadable templates, and atlas creation methods.

3.3.5 Neuroscience Information Framework (NIF)

NIF is the main project of the American National Node. It is a dynamic storage of data, which allows searching data across registered resources. The project was stated at 2005 and now it contains thousands of data resources.

NIF itself is a multi-institutional project. It consists of three parts: the NIF Resource Registry, the NIF Document Archive, and the NIF Database Mediator.

Pilot query interface is developed by NIF. Searched terms on this interface are selected from the standardized dictionary based on NIF standardized ontology (NIFSTD). Therefore the query interface is concept-based. This approach uses the idea of semantic web in combination with more strictly dictionary. The search results are relevant data. Todays search engines (such as Google) obtain text string, they parse it and they return back relevant and irrelevant data as well. The NIF query engine is applicable on every database (through NIF), which respects a standardized data model, ontologies and INCF recommendations.

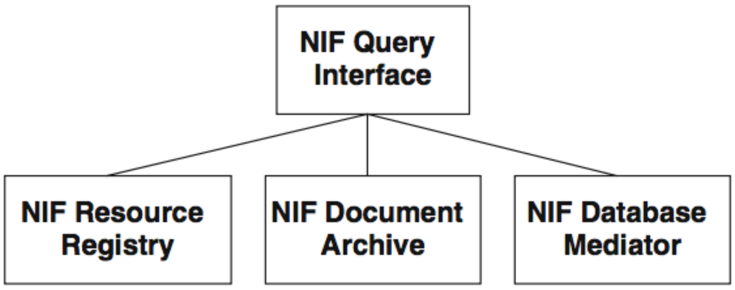

Figure 7: Pilot NIF Concept-Based Search Interface [12]

NIF Resource registry is the database containing information about registered databases or web-based resources. The registry contains description, contact, URL, and mapping to NIFSTD for each data resource. The mapping deals only with the list of terms; it is not mapped to the database itself. Therefore it is possible to find a proper data source but it is not possible directly search in the database through the NIF interface.

NIF Document Archive stores documents and articles dealing with neuroinformatics and provides text search inside them.

NIF Database Mediator provides the way to search in the database through the NIF interface. The database has to be fully mapped on the NIFSTD ontology.

Registration of new resource into NIF is divided into three levels [13].

- **Level one**

URL of the referenced resource is registered into NIF. Then NIF knows the resource and provides its URL but it does not provide access to dynamic content.

- **Level two**

It uses XML-based script to provide a wrapper to a web site that allows searching for key details about a requested data source including dynamic content. [8] The web-service using DISCO solves providing this XML[7].

- **Level three**

The last level of the registration includes technology called the semantic web (more in Chapter 4.3). On this level, the data from a resource are accessible. Data from the resource are mapped by the Content-Based Query Interface and by the standardized ontology NIFSTD. Databases, which are in the third level and which have good mapping can be compared with distributed databases. Each resource looks different but from the point of view NIF Query Interface, it is a huge distributed database.

[7] The Web Service Discovery Tool - http://msdn.microsoft.com/en-us/library/cy2a3ybs(v=vs.80).aspx

3.3.6 EEG/ERP Database at the University of West Bohemia

Neuroinformatics research group at the University of West Bohemia as INCF member is developing their own EEG/ERP portal with neuroinformatics database. The main purpose and already developed part of the portal is to manage experiments and store the data and metadata. The portal is a web-based application developed in JAVA language[8] using frameworks Spring MVC[9] and Spring Security[10]. The database part is built on Oracle 11g[11] and communication between the application and the data layer is provided by relational-object mapping (the Hibernate[12] framework).

The basic functions of the portal are [8]:

- *User authentication*
- *Storage, update and download of EEG/ERP data and metadata*
- *Storage, update and download of EEG/ERP experimental protocols*
- *Storage, update and download of data related to testing and subjects*

Metadata which are stored in EEG/ERP database and which give the meaning to the protocol can be divided into several groups [8]:

- *Protocol of experiment (name, length, description…)*
- *Experimenters and testing persons (given name, surname, contact, experiences, handicaps…)*
- *Used hardware and software (laboratory equipment, type, description…)*
- *Actual surrounding conditions (weather, temperature…)*
- *Description of experiment (name, start time, end time, project…)*
- *Research groups (members, name…)*
- *Articles (name, author…)*

The ER model of the database is included in Appendix A.

There is an effort to register the portal into the NIF and share our data with the others. The current state of registration is level 2. Because of the sharing the following user roles are specified [8]:

- Anonymous user
  - Access to homepage
  - Access to registry form
- Reader
  - Access to account setting
  - Access to public experiments
  - Forbidden access to personal data
  - Forbidden upload of experiments
- Experimenter (In addition to *Reader*)
  - Ability to upload experiment
  - Full access to experiments

[8] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[9] http://www.springsource.org/
[10] http://www.springsource.org/spring-security
[11] http://www.oracle.com/cz/products/database/index.html
[12] http://hibernate.org/

  - Member at least of one group
- Group administrator
  - Access to group administration (to group of which he is administrator)
- Supervisor
  - Access to global administration of user groups

4 Data Models

A proper data model is an alpha-omega of each portal. A badly proposed data model could break all application logic in MVC[13] architecture. There are various ways to construct data models. Basically, data models could be divided by their internal logic

- Based on relational algebra
- Based on object-oriented concept
- Based on description logic

This chapter describes these models.

4.1 Entity-Relational model

The first model is an entity-relational model (ER model). ER models are the cores of relational databases (RDB).

4.1.1 Relational databases

One of the mostly used concepts for storing data from early 70’s are relational databases. RDB are popular for many reasons. They are very fast. There are lots of commercial and non-commercial implementations. They could be modified for special purposes (e.g. spatial databases). For majority of modern programing languages libraries and drivers for comfort access into DB exist. The main advantage is its simplicity. It is very easy to understand how to propose DB model. The concept of relational databases is based on entity-relational models (ER models).

Before the ER model can be described, it is necessary to define the relational model (R model). The formal definition of R model is described as constitution of following fundamental stones (cited from [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 $D_1, D_2, \dots, D_n$, be n (n > 0) domains (not necessarily distinct). The Cartesian product $x\{D_i: i= 1,2, \dots ,n\}$ is the set of all n-tuples $<t_1, t_2, \dots ,t_n>$ 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.

[13] Model-View-Controller Architecture is a way to design software (especially web applications) into more or less separated levels

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.

Let suppose that all domains are simple and the relation has the following tabular representation:

1. No two tuples (in case of tabular representation rows) are duplicated
2. Row order is irrelevant
3. Column order is irrelevant
4. Table entries are atomic values (we still deal with relational databases, not object-relational DB)

Now the term RDB can be formulated.

- **Relational Database**

A relational database is collection of data represented by collection of time-varying tabular relations.

In case we have a database, we need an apparatus to identify relations and their attributes. This identification has to be applied to columns and also to rows because we have a tabular structure.

- **Primary and candidate key**

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. The set of all primary keys is called a primary domain. No primary key has to be null – entity integrity.

Suppose an attribute A of a compound (i.e. multiattribute) 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. – reference integrity.

- **Schema**

A schema for relation R with degree n is a list of unique attribute names.

With these specifications we can define the relational model and the entity-relationship diagram.

Relational model consists of:

- *A collection of time-varying tabular relations.*
- *Insert – Update – Delete rules.*
- *The relational algebra.*

Differences between R model and ER model are based on what the models are focused on. The R model is focused on the relations while the ER Model is focused on relationships between entities. In other words, R model deals with the structure of data values and ER model deals with structure of entities. R model uses Cartesian product of domains to define relations. ER model uses the Cartesian product of entities to define relationships. ER Model contains more semantic information than the R model. It is caused by additional semantics of data in a data structure. ER Model has explicit linkage between entities but R model has it implicit. In addition, the cardinality information is explicit in the ER model, and some of the cardinality information is not captured in the R model. [15].

4.1.2 Relational algebra

The relational databases are based on the relational algebra. Therefore operators UNION, INTERSECTION and SET DIFFERENCE exist. They are applicable with one constraint – the pair of relations, which are used, has to be from the same domain. These constraints guarantee that the result of operation is relation as well.

There are few operators for the manipulation with relations. A list and brief description of these operators follow. More information and examples can be found in [14].

THETA-SELECT (RESTRICT)

Theta-select operator is a binary operator applicable above relations and attributes: <, ≤, =, ≥, >, ≠. If binary operator is "=", then the *theta-select* operator is called SELECT.

PROJECTION

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

THETA-JOIN

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

DIVIDE

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. [14]

4.1.3 ER model construction and graphical representation

In the following chapter only the ER model (the model) will be considered. Despite the fact the term *Entity-Relational Diagram* is frequently used, there are no strict rules how to represent the model. Nevertheless, some convention exists. There are two usual ways to make the diagram.

One way is to use UML. This case will be discussed in chapter 4.2.4. Second way is to use a diagram based on conventions. Consider the following example.

University Example: In one state some universities exist. Each University has several faculties. Each Faculty has n students. One student is able to study more than one University or/and Faculty. Student has some ID, which is unique across the University, and he has name and surname. Faculty and University have some name, dean/rector and students. Dean/rector has name, surname, employee's ID (unique on the University).

We will use this example in further chapters. Sometime it will be extended or modified but the base instructions are static.

First of all, we will create a model and than we will create a diagram. In easy cases like this example is possible to consider subjects as entities, predicates as relationships and nouns as attributes. This could be a basic skeleton of the model. More information has to be specified by the task context. During construction of the model we have to transform a clear task and context for human into form readable by computers. This could be a non-trivial task. Everything what is obvious for human has to be explicitly specified.

As it was said above, we can look on the subjects as on entities and nouns as on attributes. For example, we have the following entities and attributes in parentheses:

- University (name, rector, faculties)
- Faculty (name, dean, students)
- Dean (id, name, surname)
- Rector (id, name, surname)
- Student (id, name, surname)

According to definitions in chapter 4.1.2, every attribute has to be atomic. As we see above, attribute *rector* and *dean* are not atomic. Attributes *faculties* and *students* are non-atomic and moreover they are the lists of another entities. We can easily correct it by using the relational algebra operator – THETA-JOIN, concretely NATURAL-JOIN (now for *dean* and *rector* only). We will substitute attribute *rector* by rector's ID (the same for *dean*) and thanks to NATURAL-JOIN we still have all needed information in case of knowledge the *University*. Entities now look as follows:

- University (name, rector_id, faculties)
- Faculty (name, dean_id, students)
- Dean (dean_id, name, surname)
- Rector (rector_id, name, surname)
- Student (student_id, name, surname)

Now we have to specify candidate keys and select a proper primary key. In this case candidate keys equal primary keys. Keys are highlighted.

- University (**name, rector_id,** faculties)
- Faculty (**name, dean_id,** students)
- Dean (**dean_id**, name, surname)
- Rector (**rector_id**, name, surname)
- Student (**student_id**, name, surname)

First two primary keys are composed from two attributes. This will work in the case we trust that there cannot be two universities with the same name and same rector. But except the fact that database cannot rely on probability we need some simple identifier for JOIN between *University* and *Faculties* (*Faculty* and *students*). Therefore we change the primary key and create a new artificial identifier.

- University (**university_id**, name, rector_id, faculties)
- Faculty (**faculty_id**, name, dean_id, students)
- Dean (**dean_id**, name, surname)
- Rector (**rector_id**, name, surname)
- Student (**student_id**, name, surname)

For this moment we will consider entities as done. Now we have to define relationships between entities.

- ONE University has MANY Faculties
- ONE University has ONE rector
- ONE University has MANY students
- ONE Faculty belongs to ONE University
- ONE Faculty has ONE dean
- ONE Faculty has MANY students
- ONE dean direct ONE Faculty
- ONE rector direct ONE University
- ONE student study MANY Faculties
- ONE student study MANY Universities

Notice that the relationship consists of three parts – subject, predicate and object. It is nearly the same case as the triples in predicate logic.

Lets design relationships more schematically. Numbers in relationships define cardinality - minimum allowed entities. Moreover we dropped out redundant relationships like the reverse ones (e.g. ONE *University* has ONE *rector* and ONE *rector* directs ONE *University*) and nested ones (ONE *University* has MANY *students*; ONE *Faculty* has MANY *students*; ONE *University* has MANY *Faculties*). Exclamation mark means that the value of the attribute cannot be null.

• University	1!…………n!	Faculty
• University	1!…………1!	Rector
• Faculty	1!…………1!	Dean
• Faculty	m………...n	Student

This could be considered as the ER model. Now, when we have elementary parts of ER model, we can construct ER diagram.

Entities are usually interpreted as rectangular boxes.

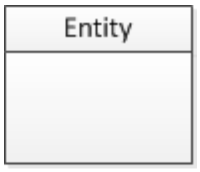

Figure 8: Entity

Attributes are written inside these boxes.

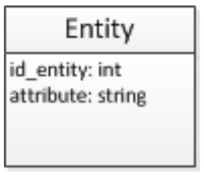

Figure 9: Attributes

Relationships are lines between boxes; their names are in diamonds; cardinality is represented by a single (1) or multiple (n) arrow and filling of arrows defines if the attribute value could be null (empty arrow) or not (full arrow).

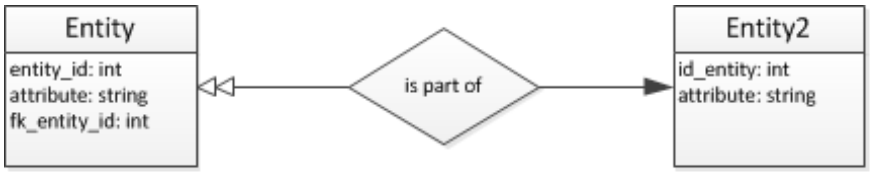

Figure 10: Relationship and cardinality

The following diagram shows the model of the example. Dashed relationship represents m:n relationship. Relationships containing table *Faculty_student* represent the same relation as dashed one, only decomposed.

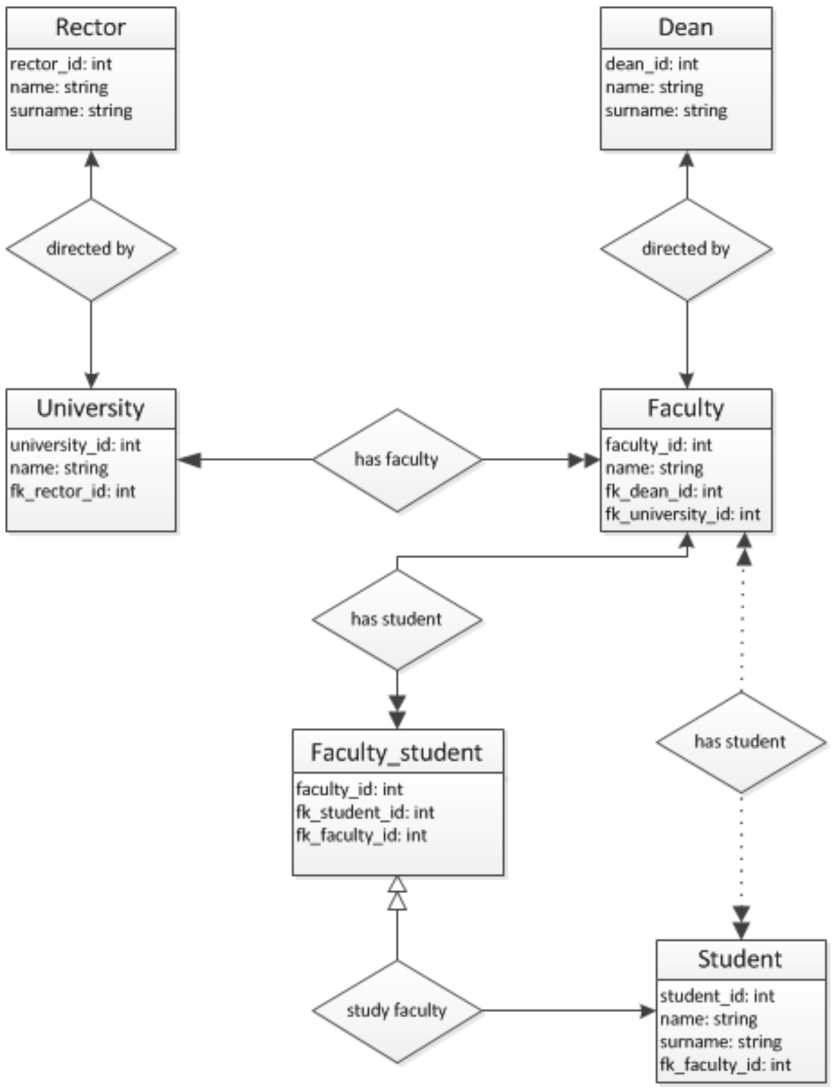

Figure 11: *Example 1* ER diagram

This diagram is quite similar to UML class diagram although we have got neither one object here. Now we return back to entity declaration and we will diagnose it deeper. *University* and *Faculty* have some attributes in common ditto *rector*, *dean* and *student*. Let substituted these entities by more general ones – *institution* and *person* – and recognize the type of the entities only by flag (e.g. number where its value=1 means *University*; value=2 means *Faculty* etc.).

- Institution (**institution_id**, name, director_id, flag)
- Person (**person_id**, name, surname, flag)

And adequate relationship:

- Institution m!…………n! Person

Problems are clearly seemed. Now we cannot restrict, that institution has just one director; faculty could be without students etc. For this purpose the object orient data model was proposed

4.2 Object-oriented model

This chapter deals with the object-oriented model. At first the major characteristics of the OO model will be specified. Then the model of the "University example" will be created. Finally, all will be designed by UML.

At the end of the previous chapter we have tried to use generalization of entities to make the solution more universal. However, the ER model had lower expression power than we had needed.

Object-oriented approach solves this problem. There is no precisely specified what OO concept is but regardless the implementation (programming language, database, artificial intelligent) the following properties has in common. Won Kim called these common features *The Core* [16].

4.2.1 The Core

- **Object**

Object is an abstract representation of any real-world entity. Therefore we can imagine any real world bordered area as a finite set of mutual connected objects. As it is not possible to have two absolutely same entities (in real world) neither in OO concept it is possible to have two absolutely same objects. To avoid this, each object has its own identifier - OID, which is unique at least in a specific domain where object exists. It is a non-trivial task to create an OID. A lot of implementations exist but in general, we can divide the OIDs in two groups

- Logical
    - Instance identifier + class identifier – It is clearly seemed which class is instance member, but in case of reclassification OID becomes invalid
    - Class identifier (e.g. GemStone) – the object has unique id, but it is not implicitly known of which class is instance member
- Physical (e.g. O2) – contains information depending on physical storage (e.g. memory address); object is persistent, but it is not possible to migrate to another physical storage

Complex object is a collection of other objects (e.g. Library could be complex object. It is collection of books. Book could be complex object as well. It is collection of pages, bookbinding and bookmarks.) The following constructors are defined (depend on the structure of the complex object).

- Set constructor
- List constructor
- Tuple constructor

Constructors are orthogonal (i.e. each constructor could be used for any object) [17].

- **Attributes**

Each object has its state. The state is defined by the values of object`s attributes. In general, each attribute is also object.

- **Methods**

Methods define behaviour of the object. Through these methods is possible to change state of objects (defining and changing values of attributes). These methods are encapsulated in the object and they are evoke from outside.

- **Class**

Class could be understood as a template for the object. Each instance of the class has the same structure (set of attributes and methods).

- **Inheritance**

Inheritance is ability of class to derive another class – subclass – that has the same attribute and method set. The subclass can contain additional attributes and methods but at least the same set of attributes and methods as the parent has. Original class is called superclass from the subclass's point of view. Inheritance could be single or multiple. If the single inheritance is supported it is possible to construct class hierarchy (tree). In case of multiple inheritances it is possible to construct more complex graphs. This is sometimes called a class lattice [16]. The process from root to leaves is called specialization; the process from leaves to root is called generalization.

4.2.2 OO model proposal

Because complex attributes and inheritance are allowed, we do not need explicitly formulated relations (at this time). The following candidates for classes were proposed like entities in ER model.

- Class University (Rector, Name, List of Faculties)
- Class Faculty (Dean, Name, List of students)
- Class Rector (ID, name, surname)
- Class Dean (ID, name, surname)
- Class Student (ID, name, surname)

This proposal leads to the valid data model but without advantages of OO concepts. At the end of chapter 4.1.3, generalization was mentioned for the first time. During primary model proposal the generalization is mainly done by knowledge of semantic meaning of the objects. It is almost impossible to realize it automatically. In our example we will generalize every persons (*students*, *rector*, *dean*) and institutions (*university*, *faculty*).

- Class Institute (name, director)
- Class Person (name, surname, ID)
- Class Rector extends Person()
- Class Dean extends Person()
- Class Student extends Person()
- Class University extends Institute (List of Faculties)
- Class Faculty extends Institute (List of students)

Now we have 7 classes and their instances characterizing our example. It is clearly seemed that expressive power of OO model is higher than expressive power of ER DB because we are able to generalize into Institute and Person without loosing information.

4.2.3 Unified Modelling Language - UML

Contrary to ER model, for OO model strictly defined modelling language exists. UML was developed as a reaction to growing trend of OO languages and complex development methodologies. In 90's various modelling methods existed. First UML version was created as a combination of three popular methods - OMT (Rumbaugh), Booch, and OOSE (Jacobson). New ideas were added into language and new versions were released. In 1997 Object Management Group (OMG)[14] took UML 1.1 as a standard. In 2009 UML 2.0 was released. Figure 12 shows evolution of UML. Only UML 2.0 will be discussed further.

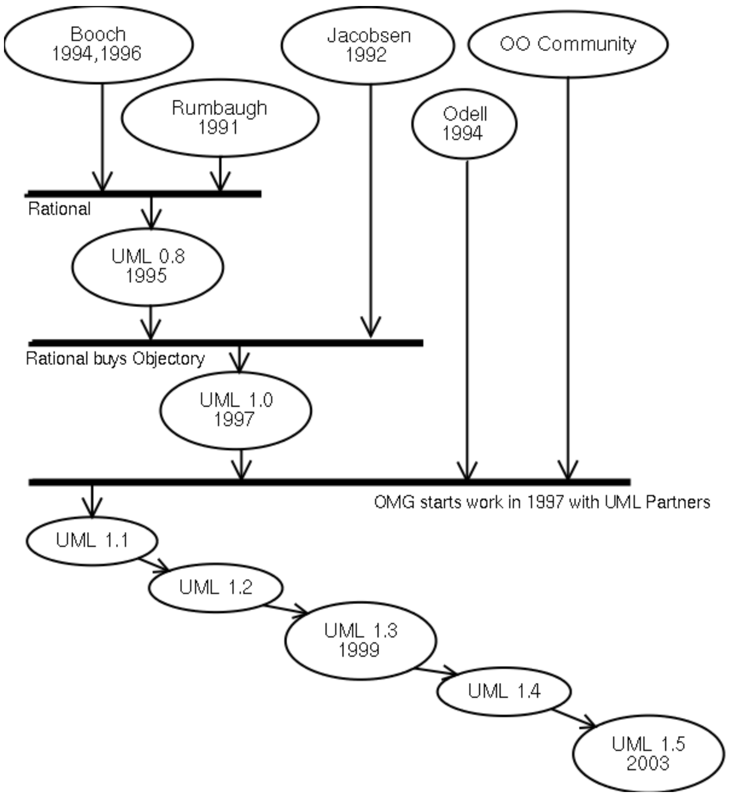

Figure 12: UML History [18]

UML is a language for graphical representation of all aspects of software proposal; it means not even the structure of software but the behaviour as well. UML 2.0 contains 14 basic diagrams. These diagrams are often adapted for special purpose (e.g. UML profile for CORBA). Purpose of the diagrams is to create a model. The model should precisely define OO system independently on its further implementation. Moreover, it is possible to generate parts of the code when the model is well done (e.g. through plugin in Eclipse IDE).

The diagrams can be divided into following two groups [19]:

- Structure diagrams
  - Class diagram
    - Class diagram defines static structure of system
    - It allows to define data model, specific classes with all attributes, methods, relations, inheritance and encapsulation

[14] http://www.omg.org/

        - Only this diagram is oriented on data model and will be discussed further
    - Component diagram
        - This diagram allows to divide the system into smaller parts with the delimited functions
        - The diagram describes how the classes are divided into components
    - Composite structure diagram
        - Defines internal structure of classes; how the methods and functions cooperate
    - Deployment diagram
        - Defines how the components will be physically distributed in IT environment
    - Object diagram
        - Shows state of the object in specified state / time of the system
    - Package diagram
        - Shows how the system is divided into packages
    - Profile diagram
        - *Operates at the metamodel level to show stereotypes as classes with the <<stereotype>> stereotype, and profiles as packages with the <<profile>> stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what metamodel element a given stereotype is extending.* [20]
- Behaviour diagrams
    - Activity diagram
        - Defines activity of process
        - It looks on activity from more general point of view than *state machine diagram* (e.g. it allows conditions)
    - State machine diagram
        - Defines progress of process activities
        - It solves movement from one state of system to another one; it does not solve sequence of activities as *sequence diagram*
    - Use case diagram
        - *Use case diagram* defines how the system will be used, who are the actors and what role they have
    - Communication diagram
        - Defines how the objects communicate together during specified activity
    - Sequence diagram
        - Defines how the system works in various activities (what happens inside the system between classes or components)
    - Timing diagram
        - Shows system behaviour on the time line (very similar to the *sequence diagram*)

Except the diagrams UML specifies exchange formats (CORBA IDL[15], XMI[16]) and OCL[17] (language for expression more restriction which cannot be represented graphically).

[15] Interface definition language
[16] XML Metadata Interchange

UML describes itself as an UML model as well.

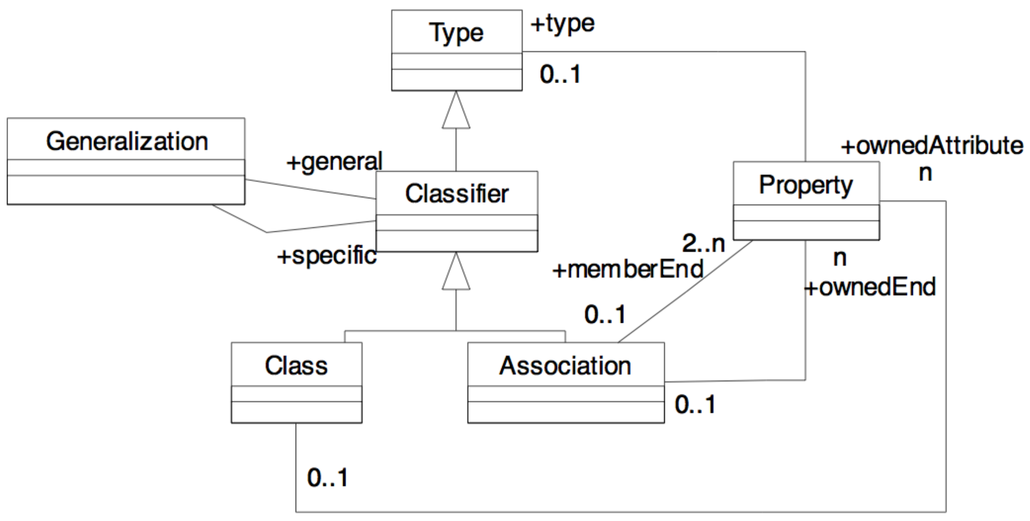

Figure 13: UML Kernel [21]

4.2.4 UML Class diagram construction

The following part focuses on the class diagrams in the way of data modelling. It means that some constructs are not mentioned (e.g. encapsulation).

UML has the following graphical syntax for *The Core*.

- **Class**

Class is a rectangle divided into three parts. The first on includes name of the class.

- **Attribute**

Attribute is an item in the second part of class`s rectangle. The first symbol +/-/# defines encapsulation of the attribute – public, private, protected. The name of the attribute follows. A data type (another object) of the attribute is specified after colon.

- **Method**

Method is an item in the third part of class`s rectangle. Syntax is same as in case of the attribute.

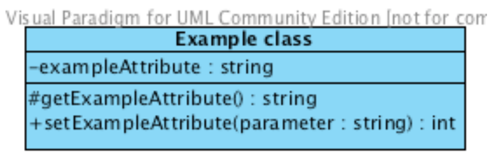
Visual Paradigm for UML Community Edition [not for com
Example class
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 14: Example class

- **Inheritance / Generalization**

Generalization is defined as an arrow *descendant -> parent.*

[17] Object constraint language

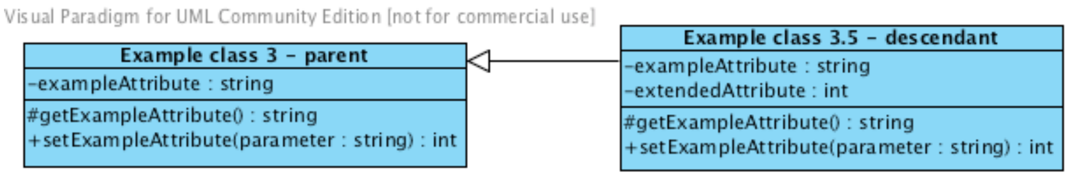
Visual Paradigm for UML Community Edition [not for commercial use]
Example class 3 – parent
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int
Example class 3.5 – descendant
-exampleAttribute : string
-extendedAttribute : int
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 15: Example generalization

- **Association / aggregation / composition**

Association / *aggregation* / *composition* represents alternative to relationships in ER model. But this relationship is divided into three forms based on their semantics. *Association* is the relationship in most general form and can be equals to relationship in ER model. It is a line between two classes. *Association* can be binary or ternary as well.

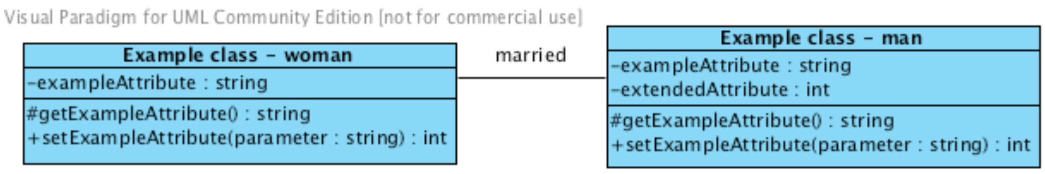
Visual Paradigm for UML Community Edition [not for commercial use]
Example class – woman
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int
married
Example class – man
-exampleAttribute : string
-extendedAttribute : int
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 16: Example association

Aggregation is stronger. It says that one classes is component of the second one. It is a line with an empty diamond on its end.

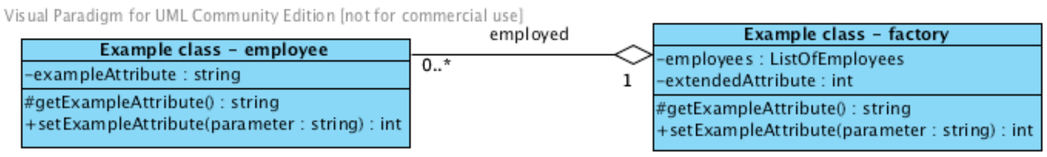
Visual Paradigm for UML Community Edition [not for commercial use]
Example class – employee
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int
employed
0..*
1
Example class – factory
-employees : ListOfEmployees
-extendedAttribute : int
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 17: Example aggregation

Composition is stronger form of *aggregation*. *Composition* defines that one class cannot exist without another class which it includes. *Composition* has the same syntax as *aggregation* but the diamond is filled.

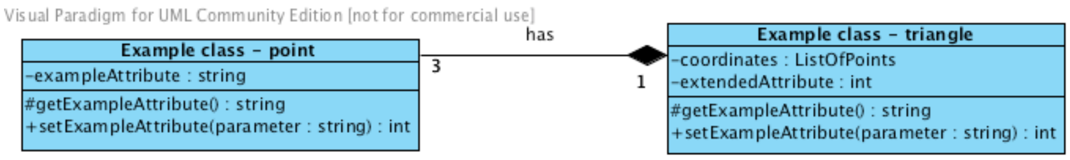
Visual Paradigm for UML Community Edition [not for commercial use]
Example class – point
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int
has
3
1
Example class – triangle
-coordinates : ListOfPoints
-extendedAttribute : int
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 18: Example composition

- **Multiplicity**

In ER diagram *cardinality* has the following types: 1:1; 1:n; n:m. UML provides more general form. It is possible to precisely define *from* and *to* for each side of *association*. Infinity is specified as symbol * (e.g. 1 – 0..*; 1..5 – 1..* etc). Example is in Figure 17 and Figure 18

- **Dependencies** [19]

Dependencies are represented by dashed lines, which characterise any form of dependency in general.

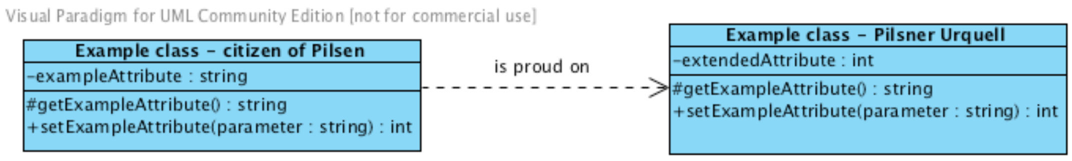
Visual Paradigm for UML Community Edition [not for commercial use]
Example class – citizen of Pilsen
-exampleAttribute : string
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int
is proud on
Example class – Pilsner Urquell
-extendedAttribute : int
#getExampleAttribute() : string
+setExampleAttribute(parameter : string) : int

Figure 19: Example dependency

UML class diagram provides more - e.g. multiple inheritance (multiple classification), stereotypes, parameterized classes etc., but it is not scope of this work.

With these fundamental constructs the following diagram was proposed. Figure 20 shows how the University example will look like in the OO data model described by UML.

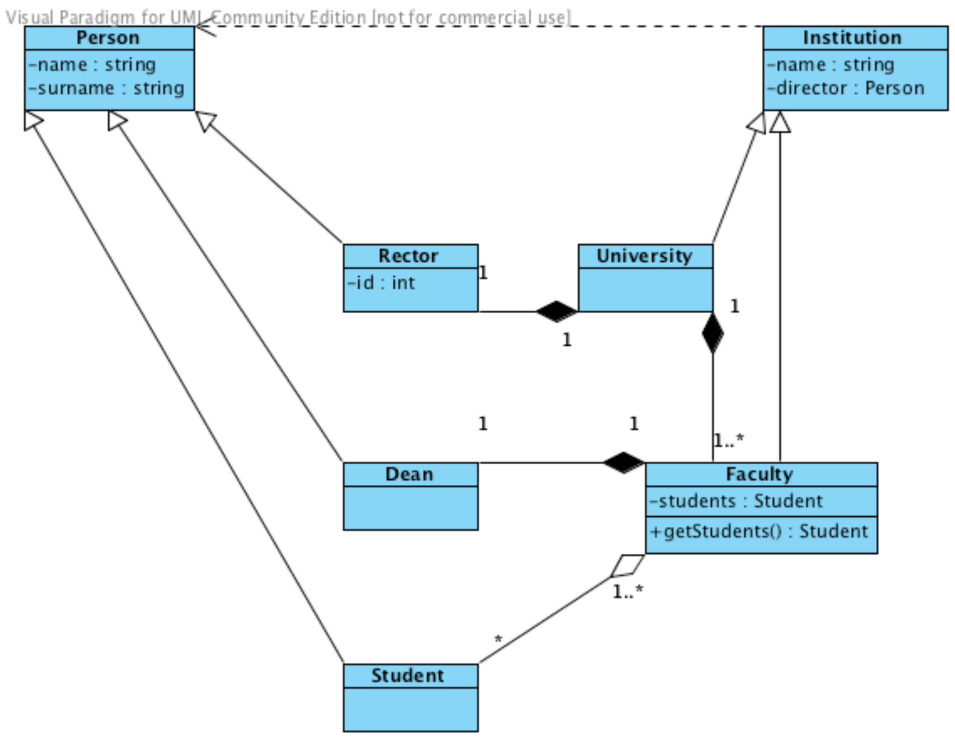

Figure 20: University Example modelled by UML

4.2.5OO model summary

It is clearly seemed that OO model has higher expression power than ER model. On the other hand, we still commented models and diagrams but not a realization. UML does not solve implementation at all. That means, that almost every ER model is realizable in almost every relational database. But in case of UML the situation is more complicated. UML provides a lot of features collected from various OO languages and databases, but there is no language that has it all in common. So, although UML is language independent it can be sense as a set of instructions for programmers (who want implement the model). Therefore, designer of the model has to know what features implementation language / database can implement.

Except this fact one more disadvantages is here. OO languages had great improvement over the time. But OODB does not and ORDB is sill number one in the field of data storing.

4.3 Semantic web

In 1989 Sir Tim Berners-Lee introduced the idea of hypertext. During five years the protocol HTTP was widely spread to public and the World Wide Web Consortium (W3C) was established. In 2001 the range of web was so big that its limitation was revealed and Berners-Lee introduced a new idea – Linked Data and semantic web.

4.3.1 Linked Data

Linked data are the fundamental stone of the semantic web. The meaning of hypertext is to link up the documents. One page leads to another. This approach is oriented just on documents. The meaning of the linked data is to link up data itself – the approach oriented on data. In general can be said that the data can be everything. In other words everything can be linked up with everything by URI (Uniform Resource Identifier).

In [22] four basic rules for Linked Data are specified.

- *Use URIs as names for things*
- *Use HTTP URIs so that people can look up those names*
- *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)*
- *Include links to other URIs. so that they can discover more things.*

In [23] Berners-Lee speaks about the importance of URI. Not even documents can be identified by URIs but e.g. persons or places as well. The result of searching across linked data is not even string of letters or number but it is identified data. Berners-Lee is using comparison that the data are the relationships. Linked data leads to huge collection of logically connected information[18].

Linked data does not speak about semantics expression power. For this purpose the ontologies exists.

4.3.2 Ontologies

The term ontology has many meanings and many definitions, for this work following two definitions are relevant.

- *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 various (non-disjunctive) groups. The article [24] defines three basic groups.

- Terminological and lexical ontologies – extended thesauruses
- Information ontologies – extends database schemas

[18] Information is collection of data interpreted by logical context

- Knowledge ontologies – representation of knowledge; ontologies are mentioned as logical theories

The representation of ontologies is very similar to the OO data models. The ontologies could be represents as an oriented graph. Therefore the elements as entities, classes, attributes and relationships are presented. Moreover the ontologies provide much more restrictions and possibilities, which could not be graphically represented. One of the frequently using editors for partial graphical representation is editor Protége[19].

Many standardized languages for ontology description exist – e.g. *Cyc, Ontolingua, OCML, OKBC and XOL, SHOE, Ontobroker, RDF, RDFS, DAML+OIL, OWL*. This text will deal only with *RDF, RDFS* and *OWL* as the most using ones. Information about the others provides e.g. [24]. These languages can be stored in various formats – will be discussed later.

4.3.2.1 Resource Description Framework

The Resource Description Framework (RDF) is the technology developed by W3C. Its purpose is simple description of statements. The statements are stored in triples – subject; object; predicate. Subject is the resource, object is the described object and predicate is relationship between subject and predicate. Object is another subject or the literal – final value. Subject and object are identified by their URI. There is similarity between relationships in relational model and inheritances in OO model. These triples and URIs can comfortably construct the Linked Data. The triples can be described in various syntaxes, mostly common is RDF/XML, format based on XML.

RDF basic primitives

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

The RDF can realize the linked data. The RDF schema (RDFS) provides an additional semantics.

4.3.2.2 RDF Schema

RDFS add some predefined semantic specifications and restrictions into RDF. It allows a realization of generalization and class hierarchy above the linked data. RDFS allows defining resources, classes, relationships, types and properties. Moreover some predefined classes exist such as collections, containers or comments. RDFS has following basic primitives:

- rdfs:Class
- rdfs:label
- rdfs:comment
- rdfs:domain

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

- rdfs:range
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:Container
- rdfs:Literal
- rdfs:Property

4.3.2.3 Ontology Web Language

The Ontology Web Language is the language fully focused on the ontologies. Nowadays OWL 2.0 is under development of W3C. During the creation of the ontology has to be done the compromise between expression power and performance. Therefore the three dialects of OWL exist.

- OWL Lite – the simplest one; suitable for simple ontologies where the complex restrictions are not necessary
- OWL DL – Fully based on description logic; suitable for description of complex classes
- OWL Full – Contains all constructs of OWL without any restrictions; very complicated for derivate the results of queries.

It is seemed that with growing of complexness and expressiveness the software restriction and support decrease.

OWL Lite is subset of OWL DL. OWL DL is subset of OWL FULL. Except the dialects the OWL is divided into profiles [25] (i.e. Figure 21).

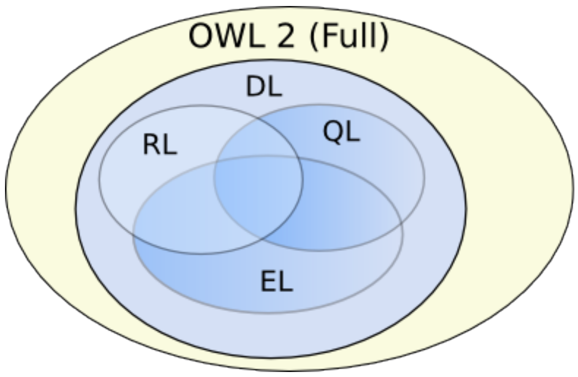

Figure 21: Venn diagram of OWL 2.0 profiles [25]

- OWL EL – Suitable for application using big ontologies; provides querying by polynomial algorithms
- OWL QL – Suitable for simply ontologies with large number of individuals; facilitate the data access in relational database
- OWL RL – Suitable for implementation of polynomial algorithms working directly with database based on RDF triples

Complete structure of OWL 2 is shown on Figure 22 [25].

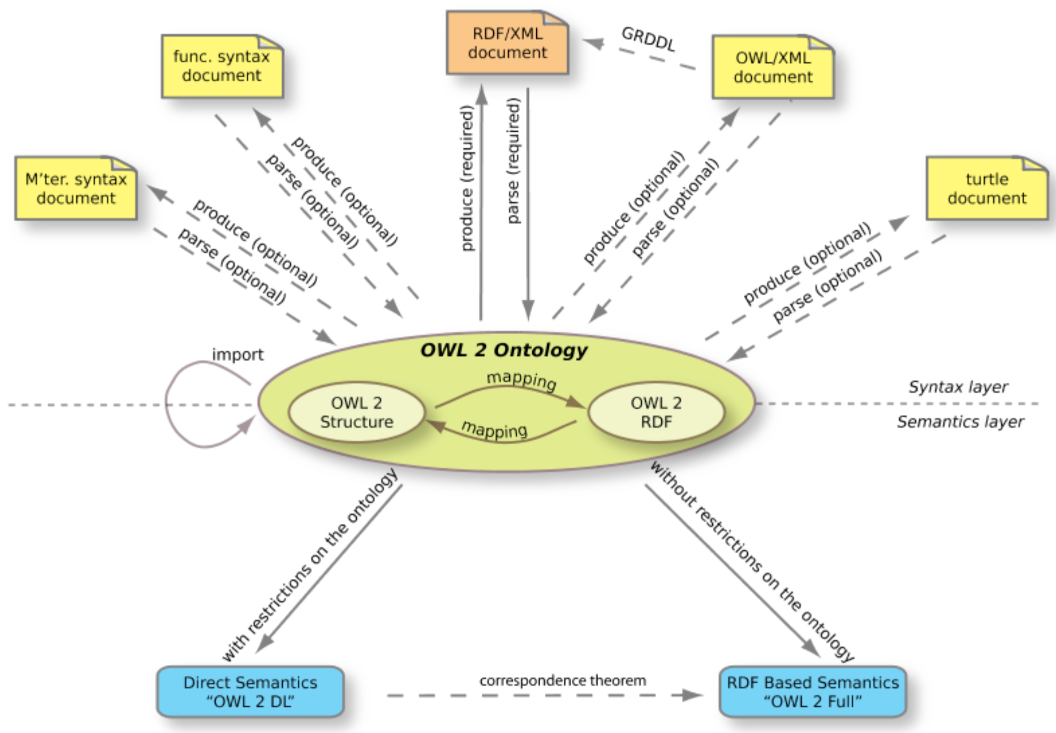

Figure 22: Structure of OWL 2.0 [25]

Based on the dialect and profile, proper syntax can be chosen.

Table 2: OWL syntax review

Syntax	Usage
RDF / XML	Universal storage; exchange format
OWL / XML	Suitable for processing by XML tools
Functional Syntax	Suitable for view the formal structure
Manchester Syntax	Suitable for read/write OWL DL
Turtle	Suitable for read / write RDF triples

4.3.3 Architecture of semantic web

The semantic web is an extension of current web using previously mentioned concepts. Following Figure 23 shows the architecture of the semantic web.

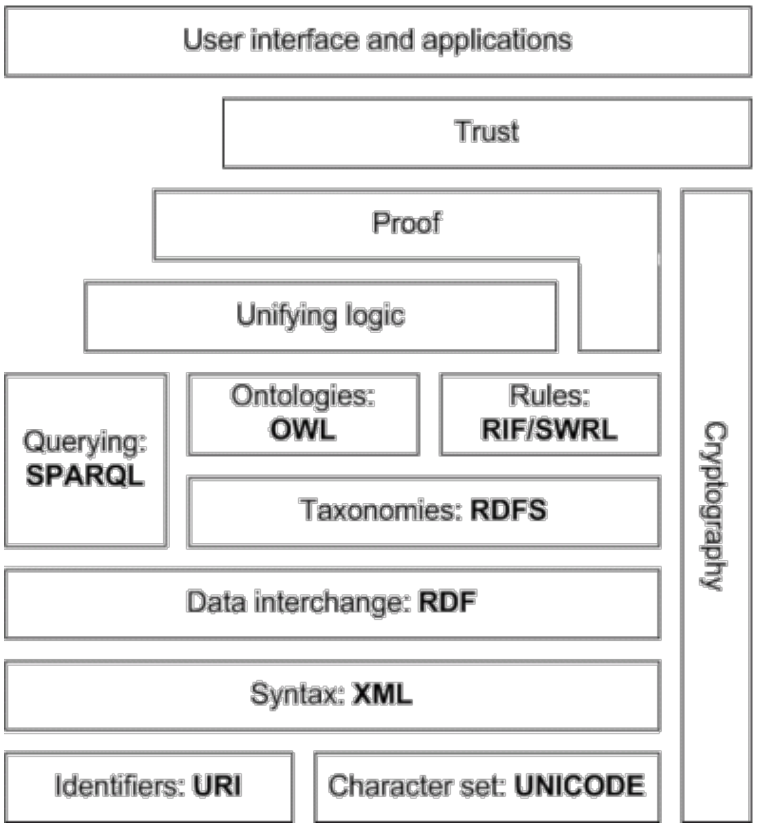

Figure 23: Architecture of the semantic web [26]

This model includes still non-implemented layers such a *Trusts*.

The first layer specifies the identifiers or resources, which are described by an URI syntax (RFC 3986) and the plain text, which is encoded by Unicode characters set. Second layer specifies the syntax of documents – XML. This layer contains not even XML, but the namespaces and the XML schemas as well. These two layers are found in the current web design. Third layer is the first layer oriented on data storing. RDF allows storing resources (data identified by URI) in triples. RDF has various syntaxes but the most common one is XML.

The RDF Schema defines the usable terms, relationships, properties and restrictions. It provides vocabulary of predefines keywords, which brings deeper semantics into RDF triples (e.g. data type or range).

On the same level the layer of querying starts. As the relational databases have SQL (Structured Query Language) RDF needs the query language as well. For this purpose the SPARQL (SPARQL Protocol And RDF Query Language) was developed. The syntax of SPARQL queries is not in the scope this thesis.

Above RDFS and parallels to SPARQL is an OWL. The Ontology Web Language is described in chapter 4.3.2.3. From the architecture of semantic web is seemed that OWL is and extension of RDFS.

RIF (Rule Interchange Format) is a support of predicate logic into RDFS. It allows adding more logical specifications (e.g. sequence of implication, forward chaining etc.).

Unifying logic layer deals with automatic deduction of information from ontologies. It is based on software agents – reasoners – that return relevant data depending on the query.

Proof layer deals with verity of deduced expressions. Thrust layer provides queries on the trustworthiness of returned data.

Thrust layer is in the form of vision only. Proof and logic layer are still only the experimental ones.

5 Storage Technologies

5.1 Relational databases

Relational databases were already defined in chapter 4. They are most wide spread technologies for storing data. Because majority of todays WWW stands on the RDBs, it is easier to look for “on fly” transformation mechanism than force millions of current databases to cross to another technology.

5.2 Object-relational databases

Object-relational databases (ORDB) are RDB with additional OO features. Tables in ORDB are able to store complex objects such *arrays*, not only atomic data types like in RDB. ORDB represents the most popular way today. It provides higher level of abstraction on already function databases. Moreover object support is in almost every complex database system.

5.3 Object-oriented databases

Object-oriented databases (OODB) are fully based on OO concept. They provide inheritances, associations, aggregations etc. Nevertheless they never become so popular as OO languages. Their implementations exist but they are not widespread. Therefore the support for them is very low.

5.4 eXtensible Markup Language

XML (Extensible Mark-up Language) is a powerful tool for storing structured data. It is used for data exchange purpose. XML is derived from SGML (Standard generalized mark-up language) – both developed by W3C[20]. The following Figure 24 shows dependencies of a few mark-up languages.

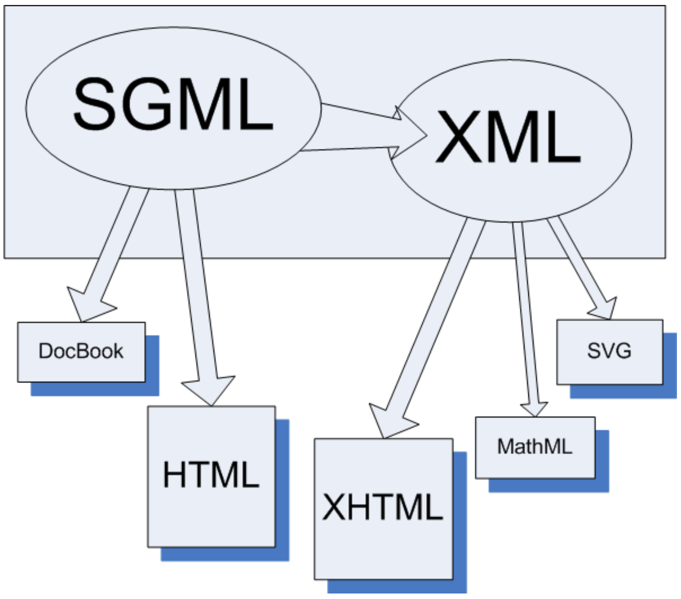

Figure 24: Markup languages dependencies [27]

XML can be validated by XSD (XML schema definition) or DTD (Document type definition). DTD is an older standard, which is not commonly used today. The main advantage of XSD is data type support and cardinality support. XML itself describes structured data. In the ER model, data have a tabular structure, so by definition of XML, it is possible to describe ER model by XML. Moreover XSD allows

[20] The World Wide Web Consorcium - http://www.w3.org/

defining inheritance, encapsulation and substitution and therefore it is possible to describe OO model by XSD. [28]

XML describes instances, not the classes. Contrary to XML, XSD focuses on classes.

The latest version of XML is XML 1.1. Important changes between XML 1.0 and 1.1 are described in the following paragraph.

The overall philosophy of names has changed since XML 1.0. Whereas XML 1.0 provided a rigid definition of names, wherein everything that was not permitted was forbidden, XML 1.1 names are designed so that everything that is not forbidden (for a specific reason) is permitted. Since Unicode will continue to grow past version 4.0, further changes to XML can be avoided by allowing almost any character, including those not yet assigned, in names. [29]

As was previously mentioned, UML give us except diagrams, the exchange formats based on XML. It means that every UML model could be stored as XML. If UML can be defined by UML and UML has XML exchange language, it is possible to describe UML by XML in general. XML document that describes the *University Example* was generated by *Visual Paradigm for XML 9.0*[21]. It is a part of electronic supplement B. XML document describes not even the classes and relationships between them but it specifies also what class or association is in general. Moreover, it carries huge amount of metadata as author`s name or colour of diagram. Supplemented XML specifies all UML diagrams and contains all information that graphical interpreter has to known. On the other hand, it is hardly readable for human.

There can be described just the model without UML metadata by XML. Because transformation from OO model is shown in the generated XML, following lines deal with transformation from ER model. The first step is a suitable proposal of the XSD. The schema describes the model. XML itself describes just the specific data in the prescribed structure. XSD defines how the elements follow each other. Transformation mechanism from ER model to XSD follows.

- **Entity**

Entity is defined as pair declaration-specification. Declaration is done by *element.* Specification is done as a *sequence* of attributes. In this sequence could be another elements and/or attributes. It is big difference contrary to ER model – attributes can be defined by more than one construct.

- **Attributes**

In ER model is one way to describe an attribute. XML provides two basic ways - definition of attribute by *attribute* or by *element.* The following example shows how the entity *Rector* could be described.

```
<xs:complexType name="rector">
            <xs:sequence>
                  <xs:element name="name" type="name"/>
                  <xs:element name="surname" type="surname"/>
            </xs:sequence>
            <xs:attribute name="id" type="id"/>
</xs:complexType>
```

[21] http://www.visual-paradigm.com/

This is only specification what the *Rector* is but not the declaration. Information where the *Rector* stands in the document must be declared in the right place - in this case as the attribute of *University*.

```
<xs:element name="university" type="university"/>

       <xs:complexType name="university">
              <xs:sequence>
                     <xs:element name="name" type="name"/>
                     <xs:element name="rector" type="rector"/>
                     <xs:element name="faculty" type="faculty" minOccurs="1"
maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
```

By this way it is possible to build the hierarchy and structure of the document until the primitive type is reached. Than the *simpleType* is defined instead of *complexType.*

```
<xs:simpleType name="name">
        <xs:restriction base="xs:string">
        </xs:restriction>
</xs:simpleType>
```

Restriction defines data type.

- **Relationships**

Relationships are defines by the proper nesting of elements.

- **Cardinality**

Cardinality is defined by element`s attribute *minOccur* and *maxOccur.*

5.5 XML Type

RDB, ORDB and OODB help us to store data and metadata in general. But there can be cases where it is suitable to use some specific technology. One of these cases is storing of experiment protocol described in XML.

XMLType is an extended data type in Oracle 9i and further. It works above CLOB but it is adapted especially to XML files. It means that it allows us to use it not only for large object storing but also for querying and validating XML itself. XML Type gives us three options to use it depending on knowledge of the XML structure – structured, unstructured and binary storage. Analogously to large objects types (CLOB, BLOB), there can be only one XML Type column per table.

5.5.1 Structured XML Type

Structured XML Type also called *Object-Relational* is designated for frequently repeated XML files with the same schema. XSD schema is required for table creation. It must be specified in table creating script.

Moreover, schemas need their own storage. This storage is provided by Oracle itself (schema must be registered into database). The following PL / SQL code is a registration method.

```
--procedure for XSD
create or replace
PROCEDURE
REGISTER_XSD_SCHEMA_BLOB (
  xsd_url      IN VARCHAR2,
  xsd_content IN BLOB)
AUTHID CURRENT_USER AS
BEGIN
  dbms_xmlschema.registerschema(
    SCHEMAURL => xsd_url,
    SCHEMADOC => xsd_content,
    LOCAL       => TRUE,
    GENTYPES    => TRUE,
    GENBEAN     => FALSE,
    GENTABLES   => TRUE,
    FORCE       => FALSE,
    OWNER       => USER
  );
END REGISTER_XSD_SCHEMA_BLOB;
/

--procedure for DTD
create or replace
procedure create_dtd_schema(
    dtd_path   IN VARCHAR2,
    dtd_content  IN BLOB)
    AUTHID CURRENT_USER
    AS res BOOLEAN;
BEGIN
    IF (dbms_xdb.existsResource(dtd_path)) THEN
      dbms_output.put_line('Deleting existing schema');
      dbms_xdb.deleteResource(dtd_path);
    END IF;
    res := dbms_xdb.createresource(dtd_path, dtd_content);
END;
/
```

Example 1: Registration schema procedures [30]

This registration is not trivial and is easier to use some application to it e.g. OracleSchemaRegister from Miroslav Král [30].

5.5.2 Unstructured XML Type

Unstructured XML Type is most similar to traditional CLOB. The major difference is possibility to query XML by XPath. For storing XML it does not need a schema. If user has it, it is used only for XML validation. There is no problem with migration or with creating special scripts for schema registration, but it has the worst performance. Moreover, we have to know explicitly how the selected protocol looks like during querying because we have got various XML in one column. XML validity is only restriction. This storage is useful when the user has not got protocols schemas or while he does not know if he has it. On the other hand, if we have guaranteed that every protocol is a valid XML file, unstructured XML Type is better than normal CLOB.

5.5.3 Binary XML Type

Binary XML Type is combination of previous ones. It uses schemas and pre-parsing of XML but it does not require schema registration. XML protocol here is parsed and stored in a special binary format. In other words, the user can have various protocols with different schemas in one column, but the protocols still could be validated. Querying is faster than in unstructured storage because of schema presence, but the performance is not as high as structured storage has. This solution is the best in the case we have many protocols with different schemas, we have not got lot of protocols from the same schema or variability of schemas quickly grows.

Following Table 3 comparison of each type.

Table 3: Kinds of XML Type comparison

Test	Structured	Unstructured	Binary
More than one XML type per table permitted	No	No	No
Schema required before table is created	Yes	No	No
Pre-parsed schema	Yes	No	No
XML Type 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 schema	No	No	Yes
Suitability for non-schema XMLs	No	Yes	Yes

5.6 RDF, OWL and NoSQL Databases

For the resources of semantic web a lot of storage types exist but none of them can be taken as mainstream. Very common is storing data in the form of text files. Various syntaxes exist (i.e. Table 2). This way to store the data is very unsuitable for huge databases – each file has to be parsed, stored in some internal structure in memory and than the data can be effectively used. Some tools for such processes exist – e.g. JENA[22]. Other approach is to walk through the text file step by step for each query. This is not memory consuming but it is very slow.

The new type of databases appears – noSQL. This chapter mention two database alternatives, which could be useful from this noSQL family. Second way is to use the transformations, which are discussed in Chapter 6.

5.6.1 Mongo DB

Mongo is document-oriented database. Data in Mongo are stored in the form of pairs – key and value. The advantage of Mongo is supported set of data types. Not even primitive types and binary types can be stored as value but the structures as well. Through this is possible to build a hierarchical structure. Basically can be said that this approach is quite close to ORDB and XML type. There can be found similarities with RDB – databases (schemas) contain collections (tables). Collections are composed from documents (table records) and the documents contain fields (columns). But there is a main difference – the document of database is not so strictly restricted as the record in ORDB. If this property is advantage or disadvantage depends on the manner of usage. Unlike the RDB or ORDB there is no DDL (the Data Definition Language). The database structure can be modified “on fly” without any ALTER TABLE or similar commands. Again, this approach can be very useful and also very dangerous. Access to the database is realized directly through the OO language (through the drivers). Mongo does not support clause JOIN, it has to be solved in application layer – denormalization is risked. Next, mongo does not support transactions and maximum size of document is 16MB.

[22] http://jena.apache.org/

Mongo is not suitable database for single usage, but it can be used as a polyglot storage with RDB. RDB stores the almost invariable data structure and Mongo stores the frequently modified parts. In case of EEG/ERP database this frequently modified part is storage of experiment protocols in XML. Unfortunately this DB does not seem as helpful tool for better storage of semantic web resources.

More about Mongo can be found in [31] and on the official website[23].

5.6.2 Neo4J

Neo4J has a graph approach. It is based on construction of oriented graphs. Therefore it is close to OO models and very close to the RDF concept. Neo4J provides transactions. Then it provides support for Java, Ruby, Python and other OO languages. Two primitives – Node and Relationship, compose the graph. Both can be described by graph.

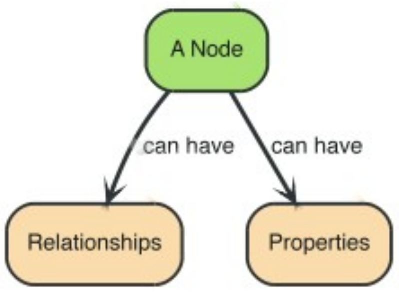

Figure 25: Node [32]

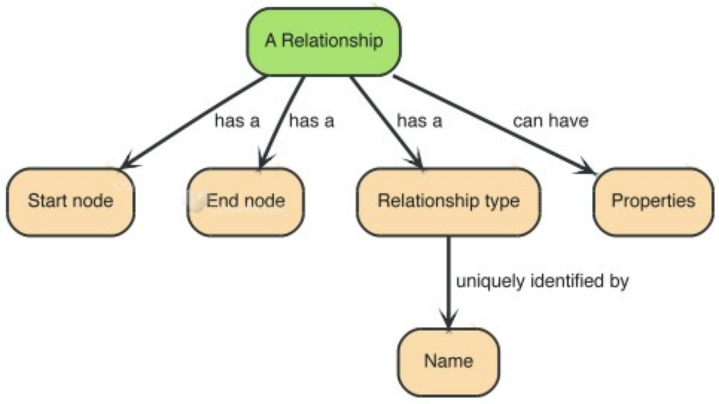

Figure 26: Relationship [32]

Neo4J could be very powerful database in future development and it could solve the problem with non-trivial transformations from RDB to RDF. Realization of EEG/ERP database by Neo4J is still in investigation.

[23] http://www.mongodb.org/

6 Transformation mechanisms

This part of the text describes transformation rules and un-transformable parts of previously mentioned models. These transformations are important for data sharing. New ways to easy share and search data rise (semantic web) but old technologies are and for long time will be still in use (RDB). Therefore at least semiautomatic transformations are necessary.

6.1 Spoken word to ER model

Table 4: Spoken word to ER model transformation

Spoken word	ER model
Subject	Entity / attribute
Noun	Attribute
Verbs	Relationships
Numbers	Cardinalities
Other	Stronger restrictions, dependencies etc. have to be derived from semantic meaning.

A spoken word has an unlimited expression power (from the point of view of ER model) therefore it is impossible to describe everything. Moreover, domain borders are defined very strictly in case of modelling but it is hard to surround natural language in the same way.

6.2 ER model to spoken word

The same rules (in opposite) as in the previous paragraph are valid in general. This transformation can be good test how much semantics of solved problem is captured in the model. In case the output sentences are without additional semantics meaningless, the model is probably badly proposed.

6.3 ER model to UML Class diagram

ER model itself has not got any standardized graphical representation, but it is possible to describe it by UML without any restrictions.

Table 5: ER model to UML Class diagram transformation

ER model	UML Class diagram
Entity	Class
Attributes	Attributes
Relationships	Associations
Cardinality	Multiplicity

6.4 UML Class diagram to ER model

UML has stronger expression power than ER model; therefore not everything can be transformed.

Table 6: UML class diagram to ER model transformation

UML class diagram	ER diagram
Class	Entity
Attribute	Attribute
Encapsulation	N/A

Methods	N/A (there is still possible to use DB methods and triggers, but it is not part of the ER model)
Multiplicity	Cardinality; multiplicity has to be simplified as follows ("?" Substitute any natural number except 1 or 0) • 1 -> 1 • ? –> n (restriction NOT NULL) • * -> n • 0..? -> n • ?..? –> n
Association	Relationship
Aggregation	Aggregation is transformed like an association with loosing the information "is part of"
Composition	Same as aggregation; the entity of composite class has to carry foreign key
Dependency	N/A
Specialization / Generalization	N/A

Other UML Class diagram constructs (e.g. stereotypes) will not be discussed because they are not a part of the data model.

6.5 UML Class diagram to XSD

Table 7: UML Class diagram to XSD transformation

UML Class diagram	**XSD**
Class	Complex type
Attribute	Element / sequence type / complex content / complex type
Simple attribute	Element / attribute / simple content / simple type
Multiplicity	Minimal occurrence / maximal occurrence
Generalization / Specification	Extension
Association	Group
Aggregation / Composition	Extension / Group / Group attribute

In XSD we have various ways to represent the same thing. This could be useful for extended semantics. If we define some UML attribute by XML attribute or XML element, there is no semantic difference. On the other hand, if we choose some self-defined rules where to use element and where attribute, we can subsequently recognize deeper semantics by these rules.

6.6 XSD to UML Class Diagram

Table 8: XSD to UML Class Diagram transformation

XML to UML Class diagram	
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 (alternative – dependency / association / generalization)

More XSD elements can be found at [33]. Their usage is very specialized and will not be described.

6.7 Relational Databases to RDF

One of the approaches how to transform common data models into RDF is from the level of RDB.

As was mentioned in chapter 4.1.3 the relationships between two relations could be sensed as the triples – subject, predicate, and object. In the same meaning it is possible to look on the columns of one table – two columns as subject and object connected by predicate "member of table XY". With this assumption is possible to build the raw RDF graph. It is important to notice that the tables have to include atomic row ID and proper foreign keys. Otherwise the relationships between tables will be lost.

The tables themselves represent the classes in RDF. Finally the tables with their relationships (foreign keys) can make the structure of RDF graph. Then the cells of tables form the literals. More about this transformation is in [34].

The transformation mechanisms are already implemented. The very useful solution for accessing the relational databases as the virtual RDF graph provides the tool D2RQ[24]. Architecture of D2RQ describes following Figure 27: Architecture of D2RQ Figure 27.

[24] http://d2rq.org/

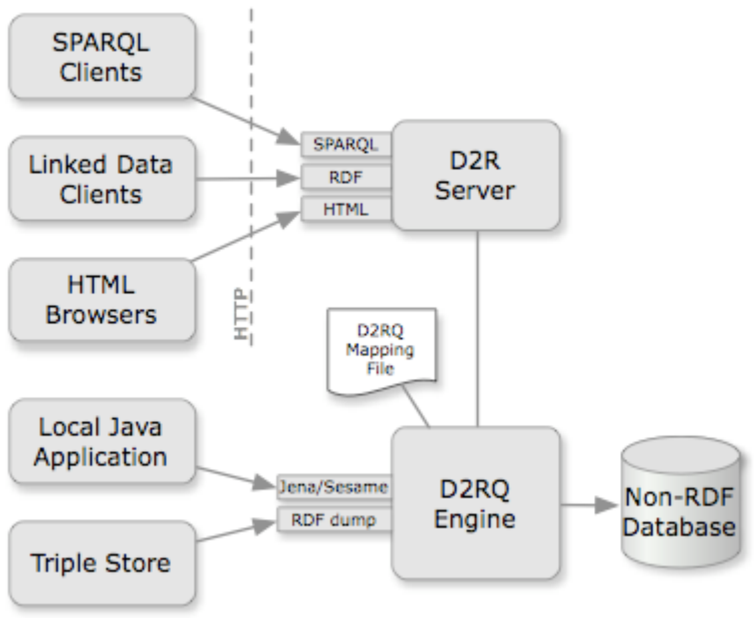

Figure 27: Architecture of D2RQ [35]

The main purpose of D2RQ is to allow possibility to query the RDB by SPARQL. During the creation of RDF graph the additional ontology could be included. This ontology has to 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 used for description of the relationships between relational model and the ontology. The engine is constructed as a plugin for Jena framework. D2RQ provides the access into RDB and in upper layers the user can use Jena tool on the RDF graph. And finally server provides the presentation of data (by the SPARQL, HTML or RDF).

D2RQ is used in the transformation tool developed on our University [36] in combination with Jena and OWL API[25]. D2RQ accesses to the database and creates virtual RDF graph in JENA framework. Jena is able to save this graph into RDF/XML syntax. Finally OWL API allows converting RDF/XML into other syntaxes. It is clearly seemed that all expression power is fully depending on the data model of RDB – no additional information is added.

6.8 OO model to RDF

The second approach how to transform relational database into RDB is through the OO model.

OO model and RDF graph have some similarities – classes, instances, inheritance, restrictions, cardinalities and attributes (literals). On the other hand, there is also the major difference. In OO model classes contain the set of attributes and their vales; in RDF the class has connection to another class (inheritance) up to final value – literal.

[25] http://owlapi.sourceforge.net/

A lot of modern web applications use the OO languages and the OR mapping such as Hibernate framework. The principle is based on manually inscription of a java annotation into Java Beans. This solution cannot be done automatically but it proposes better integration of complex ontologies – everything is described by annotations and not in external mapping file. The disadvantage is double transformation – the RDB to OO model and the OO model to RDF graph. Nowadays this solution was chosen for RDF generating of our EEG/ERP portal.

One of the tools that provide this function is JenaBean[26]. Another one deals with POJO objects instead of JavaBeans – Sommer.

6.9 XML and XSD to RDF

XML and XSD transformation into RDF is also useful. One case when it can be useful is for experiment protocols in EEG/ERP database. When the RDF graph is generated from the database the protocols are still just text strings. After the transformation it can be a part of the big RDF graph.

Transformation of XML into RDF triples is basically quite easy process. XML file is gone trough until the deepest elements (leaves) are reached and the URI is created from the path. Through this approach is possible to make the RDF graph where the literals are included and each subject and object has the URI. Nevertheless, by this approach is not possible to catch any semantics. Neither the data types of literals are recognized. Therefore the support of XSD has to be integrated. One of the possibilities is to find the literal in XML and then to find its prescription in XSD and to add founded semantics. This approach is at least time-consuming.

Another way could be starting from XSD. At first the class hierarchy will be created and than the literals will be inscribed. This solution allows more sophisticated parsing of XML document and it allows partial parallelism because it is known what has to be found in advance.

6.10 Transformation summary

Transformation between spoken word, relational model and object-oriented model was presented. Than it was shown how to store data in the XML format. Finally some basic rules and facts about transformation of RDB into RDF graph were presented. Deeper look on transformations of various data models into RDF will be part of the scope of Ph.D. thesis. The purpose of this chapter was to find the basic elements, which have each representation in common, and to specify the additional information value of each one. Continuous work will be focused on the backward steps and on the way to substitute higher semantics in the lower expressional models.

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

7 Current terminologies, data models, data formats and tools

In this chapter few projects are presented. A lot of other ontologies, formats or models can be found on the website of INCF[27].

7.1 odML

Open Metadata Markup Language (odML) is proposed by German Neuroinformatics Node. Data are stored in pairs *key-value* in hierarchical structure. Basic principle of odML is to separate the structure and the content. Therefore it is possible to use predefined structure of metadata without predefined terminology. Terminology itself is also defined but beside of structure. The domain of EEG experiments can be divided into two levels – measurement and analysis. Authors of odML have an effort to strictly separate structure and terminology on the both levels. This approach was built due to higher flexibility of metadata models – ontologies themselves are very binding.

7.2 NEMO

Neural ElectroMagnetic Ontologies (NEMO)[28] is a project founded by National Institutes of Health in Oregon. NEMO is fully focused on EEG and MEG (Magneto-encephalography) domains. NEMO project focuses on three parts – raw data, ontologies describing metadata and database storage, which stores data and metadata. Moreover it provides analysis toolkit for ERP. NEMO is registered by INCF and NIF.

7.3 HDF5

HDF5[29] is a data model, library, and file format for storing and managing data. HDF5 can be the way to store electrophysiology data in other way.

7.4 NIFSTD and NeuroLex

NeuroLex project is supported by NIF[30]. It is a lexicon of neuroscience terms. The main purpose of NeuroLex is a unified terminology and basic definition of terms. It is useful for relatively easy data searching and data comparisons. Values of NeuroLex were taken from NIFSTD ontologies. NIFSTD includes a set of modular ontologies that provide a comprehensive collection of terminologies (more than 60000 concepts) to describe neuroscience data and resources.

7.5 Dublin Core

Dublin Core is one of the first standardized ontologies. It is oriented on description of electronic information resources. It is quite far from the EEG/ERP domain but on the other hand, it can be useful for the part of the EEG/ERP portal, which describes the articles. More information about Dublin Core can be found on official website[31].

[27] http://datasharing.incf.org/ep/Resources
[28] http://nemo.nic.uoregon.edu/wiki/
[29] http://www.hdfgroup.org/HDF5/
[30] http://neurolex.org/wiki/Main_Page
[31] http://www.dublincore.org/metadata-basics/

7.6 Signal ML

SignalML[32] is not a data format at all, but it allows describing other formats for storing biomedical data. Due to this tool the data can be read from the original source by unified manner without any conversion. It is focused on EEG and MEG and the output is described by XML.

7.7 Extended FMA

Foundational Model of Anatomy Ontology (FMA) is an ontology that describes anatomic parts of human body. The extended form proposed in [37] is focused on the nervous system. This ontology can be useful in combination with various brain activity measures (EEG, fMRI, PET).

7.8 Converters

Various converters between different formats (of raw data and metadata) exist. Useful list of converting tools can be found on the website of INCF[33]. Unfortunately some of these projects (e.g. EEG data formats converting tool) are currently unreachable (due to the end of SVN repository Origo[34]).

[32] http://www.signalml.org/
[33] http://datasharing.incf.org/ep/Converters
[34] http://origo.ethz.ch

8 Scope of the Ph.D. Thesis

8.1 Metadata model

In this work, various neuroinformatics portals were mentioned. Each of them has its own solution to store data. The majority of them are based on relational database and they provide an ontology e.g. for the NIF registration. Nevertheless the data models are different. There is no standard metadata structure how to store data. The standard metadata structure could be solved separately on different layers.

8.1.1 Metadata without specified model

Basically can be said that metadata are only enumerated. The necessary ones should be grouped into a finite set. Various finite sets could form the standards. This can give us the borders of described domain and the unified terminology. It does not provide the deeper semantics – it does not describe relationships and restrictions. Different implementations can grow above these enumerations but the final result is inhomogeneous (very similar to current state).

8.1.2 Metadata in the relational model

This metadata model could be very useful for the relational database implementations. It provides unified terminology and unified data storage. On the other hand, the standard could be too strict. There is not an option to change anything in it and it could bring the problems for dissimilar implementations. Moreover it can easily lead to performance loss. In case of using e.g. object-oriented database there could be problem with various transformation of relational model to OO model.

8.1.3 Metadata in the object-oriented model

Unified object model provides except unified terminology and restrictions more sophisticated view on the domain. There is a higher level of abstraction and differences between various implementations can be ignored. It is possible to make e.g. "irregular" relational model and map it by the Hibernate framework into standardized object model. The problem occurs in the case of transformation into ontologies because it could be very various. And another problem can occur in case the relational model bellow is diametrically different than the potential standardized object-oriented one and the mapping to OO model is too difficult or impossible. In this case the performance could go down very fast.

8.1.4 Metadata in the ontologies

The highest level of abstraction could be realized by ontologies. A lot of portals use ontologies for e.g. registration to NIF. NIF itself provides the finite vocabulary for ontology creation called NeuroLEX. This vocabulary is some kind of standard but it is very voluminous and it is hard to separate just EEG/ERP domain. In case of construction of own ontology a lot of parts (sub-ontologies) could be adopted. Object *Person* could be one of them. The problem is that there are a lot of ontologies describing persons and none of them is standardized. Therefore each solution can finally have different structure with different terminology. Moreover, there is maybe no way to automatically transform an RDB or OODB into standardized ontology. The ontologies for the EEG/ERP domain exist (such NEMO).

8.2 Metadata model hierarchy proposal

The previously mentioned possibilities and disadvantages lead to proposal to solve this problem. There should be a standard on each level of abstraction and semantic expression power - from simple enumeration to sophisticated ontology. If there will be such hierarchy, it is possible to choose one model and implement it (depending on used technologies). Moreover the transformations between each layer can be clearly specified and the migration to another storage technology will be more comfortable. The solution with such hierarchy of "standards" can be fully accessible from each level e.g. for data sharing through NIF. The Following Figure 28 shows the levels.

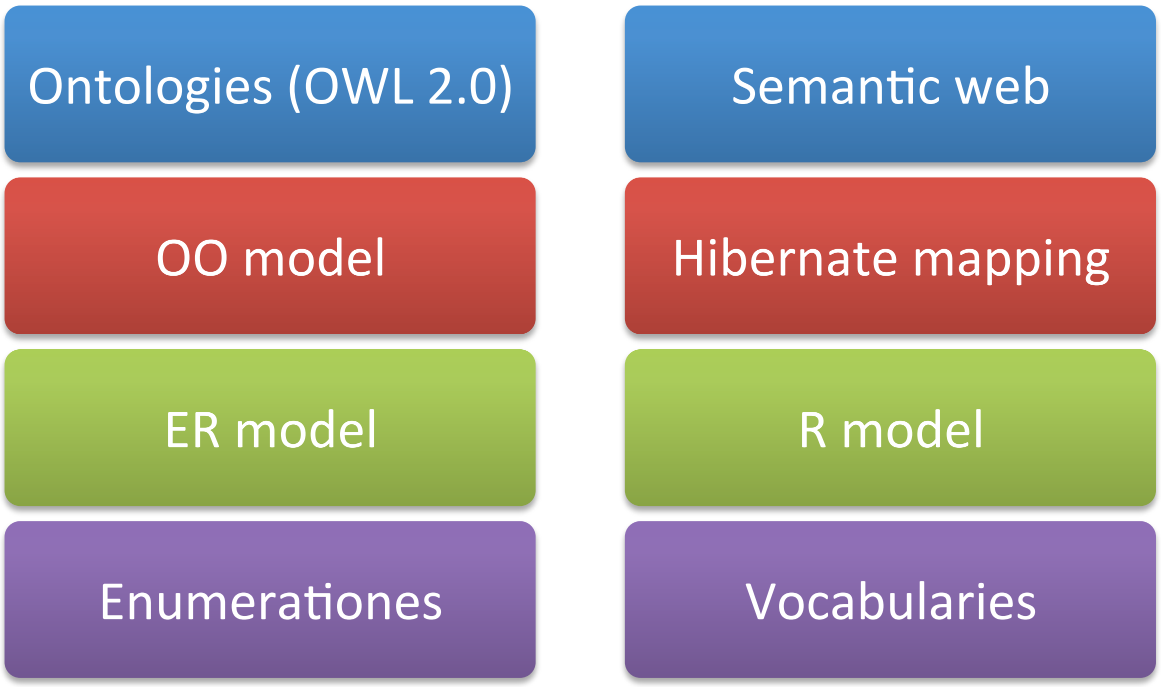

Figure 28: 4 levels of metadata standardization

This proposal more or less corresponds with the effort of G-Node. Their tendencies described in 7.1 are to separate the data model structure and the vocabulary. For this purpose the cooperation between the University of West Bohemia and the Ludwig-Maximillians Universtät München has been established. Substance of the cooperation is following.

The experiments are divided into the following structure.

- Recording
  - Data + odML + terminology
  - Semantic layer, ontologies
- Analysis
  - Data + odML + terminology
  - Semantic layer, ontologies
  - Visualization

Our task is to:

- Define terminology for driver's domain

- Specify the semantic layer at the recording level – description and construction
- Define metadata structure at the analysis level
- Classification of electric fish communication

8.3 Transformation between different implementations

Transformations between levels are derived from semantic expression power of lower level. It is clearly seemed that there will be a semantic loss in one direction (transformation from ontologies to lower levels). In the second direction (from enumerations/dictionaries to ontologies) the semantics will be kept. Theory of transformation is in the scope of Ph.D. thesis as well.

8.4 Proposal as a standard for electrophysiology domain

In ideal case, the Ph.D. thesis will provide the potential standardizable metadata models for electrophysiology domain. These models/proposals will describe the domain of EEG/ERP research area with properly selected borders. The borders of the domain will be determined by the consultations with other research groups (especially with G-Node) and the physicians and pathologies (mouse and fish brain activity domain).

9 Conclusion

There have been a lot of data standardization efforts in EEG/ERP domain. A lot of sharing concepts and implementation exist. But there is still a lot of space for improvement. International Neuroinformatics Coordinating Facility (INCF) covers national neuroinformatics nodes all over the world and invites the nodes for data sharing. The sharing here is not only the publication of raw data, but sharing metadata, information about experiments or the results of analysis as well. However, sharing is a non-trivial task. It is not only about data publication, it is about publication of information. All knowledge and research progress depends on information and information depends on proper data, which are logically structured and have in ideal case a machine-readable meaning.

This work brought the summary of current state of EEG/ERP domain in the field of data modelling. There were described significant data models, how they are proposed and how they are realized. Then the transformation rules between these models were drafted. Except traditional relational models and object-oriented models the work dealt with the semantic web. This new concept of representing and linking data becomes very popular. It opens brand new possibilities in the field of data sharing.

The work briefly described already realized ontologies and standards. As the result it proposed a complex solution of metadata models. The proposal solves more than one data model. It builds a hierarchy of semantic expressions. This solution will provide unified terminology, transformation rules between levels, various expressiveness and restrictions, but separated vocabulary, data model and implementation.

9.1 Aims of Ph.D. Thesis

- Define the terminology for driver`s domain
- Specify the semantic layer at the level of data recording
- Define the metadata structure at the level of data analysis
- Propose a metadata model at each layer at the level of data recording
- Propose the transformation rules and transition steps between the hierarchical models
- Verified the proposed solution at least on two specific domains (driver`s domain and most probably domain provided by the Ludwig-Maximillians Universtät München)

References

[1] Epilepsy Foundation. Epilepsy Foundation. [Online]. http://www.epilepsyfoundation.org/aboutepilepsy/whatisepilepsy/science/brainfunctionsandmakeup.cfm

[2] Encyclopædia Britannica Inc. (2012) Encyclopædia Britannica Online. [Online]. http://www.britannica.com/EBchecked/topic/183075/electroencephalography

[3] N. Schaul, D. Kolesnik, D. Labar, P. Sethi N. Sethi, "Laptop artifact during electroencephalography," *The Internet Journal of Neuromonitoring*, vol. 5, no. 2, 2007.

[4] Stephen J. Luck, *An introduction to the event-related potential technique*.: MIT Press, 2005.

[5] Roman Mouček and Pavel Mautner, "Driver attention while double stress - EEG/ERP experiment," in *Kognice a umělý život IX*, Opava, 2009.

[6] International Neuroinformatics Coordinating Facility. [Online]. http://www.incf.org/about

[7] Jaap van Pelt Jack Van Horn. (2007, Dec.) International Neuroinformatics Coordinating. [Online]. http://www.incf.org/documents/workshop-reports/incfworkshop-1st-SustainabilityDatabases.pdf

[8] Petr Ježek, "Database of EEG/ERP experimetns," Department of Computer Science and Engineering, University of West Bohemia in Pilsen, Pilsen, May 2010.

[9] (2012) CARMEN. [Online]. http://www.carmen.org.uk/about/carmen-factsheet.pdf/view

[10] Ralph Meier,Martin P. Nawrot,Willi Schiegel,Tiziano Zito Andreas V.M. Herz, "G-Node: An integrated tool-sharing platform to support cellular and systems neurophysiology in the age of global neuroinformatics," *Neural Networks*, vol. 21, no. 8, pp. 1070-1075, 2008.

[11] Neuroinformatics Japan Center. INCF Japan Node. [Online]. http://www.neuroinf.jp/jnode

[12] Yuli Li, Maryann E. Martone, Paul W. Sternberg, Gordon M. Shepherd, Perry L. Miller Luis Marenco, "Issues in the Design of a Pilot Concept-Based Query Interface for the Neuroinformatics Information Framework," *Neuroinformatics*, vol. 12021, pp. 229–239, Oct. 2008.

[13] Petr Brůha, "EEG portál a prostředky sémantického webu," Department of Computer Science and Engineering, University of West Bohemia in Pilsen, Pilsen, Diploma Thesis 2011.

[14] San Jose, CA E. F. Codd IBM Research Lab, "Extending the database relational model to capture more meaning," *ACM Transactions on Database Systems (TODS)*, vol. 4, no. 4, p. 38, 1979.

[15] Peter P. Chen, "Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned,".

[16] Won Kim, "Object-Oriented Databases: Definition and Researche Directions," in *TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 2, Austin, TX, 1990, pp. 327-341.

[17] Oscar Díaz Mario Piattini, *Advanced database technology and design*.: Artech House, 2012.

[18] Ralph L. Vinciguerra. (2004, Mar.) Vinci.org. [Online]. http://vinci.org/rlv/d/uml/history.html

[19] Karel Richta, "Unifikovaný modelovací jazyk UML".

[20] (2012, Apr.) Wikipedia. [Online]. http://en.wikipedia.org/wiki/Unified_Modeling_Language

[21] Object Management Group. (2009, May) www.omg.org. [Online]. http://www.omg.org/spec/ODM/1.0/PDF/

[22] Tim Berners-Lee. (2006, July) W3C. [Online]. http://www.w3.org/DesignIssues/LinkedData.html

[23] Tim Berners-Lee. (2009) Tim Berners-Lee on the next Web. [Online]. http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

[24] Vojtěch Svátek, *Ontologie a WWW*. Brno: DATAKON, 2002.

[25] (2009, Mar.) W3C. [Online]. http://www.w3.org/TR/2009/WD-owl2-overview-20090327/

[26] Marek Obitko. (2007) Ontologies and Semantic Web. [Online]. http://obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

[27] David P. Heitmeyer. (2012) Harvard University Extension School. [Online]. http://cscie12.dce.harvard.edu/lecture_notes/2007-08/20080130/slide26.html

[28] Michael Jervis. (2002, Nov.) Sitepoint. [Online]. http://www.sitepoint.com/xml-dtds-xml-schema/

[29] Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan Tim Bray. (2006, Sep.) W3C Recommendation. [Online]. http://www.w3.org/TR/xml11/

[30] Miroslav Král, "Diplomová práce - Experimentální lékařský informační systém - zpracování heterogenních nestrukturovaných dat," Plzeň, 2010.

[31] Jan Koreň, "Alternativní databázové systémy (frameworky) pro EEG/ERP portál," Department of Computer Science and Engineering, University of West Bohemia, Pilsen, 2012.

[32] Neo4J Technology. (2012) Neo4J. [Online]. http://docs.neo4j.org/chunked/milestone/index.html

[33] W3School. (2012) WXSchool. [Online]. http://www.w3schools.com/schema/default.asp

[34] Yanhui Lv and Senior Member, IEEE Z. M. Ma, "Transformation of Relational Model to RDF Model," , Shenyang.

[35] Chris Bizer. (2009, Feb.) Freie Universität Berlin. [Online]. http://www4.wiwiss.fu-berlin.de/bizer/D2RQ/#feedback

[36] Václav Papež, *Neuroinformatická databáze a sémantický web*. Plzeň: Západočeská Univerzita v Plzeni - Katedra informatiky a výpočetní techniky, 2010.

[37] Mejino JLV, Brinkley JF, Detwiler LT, Lee HJ, Martone ME, Rubin DL Turner JA, "Application of neuroanatomical ontologies for neuroimaging data annotation," *Frontiers in Neuroinformatics*, Apr. 2012.

[38] Petr Ježek, *Ontology Development in EEG/ERP Domain*. Pilsen, Czech Republic, 2012.

[39] Richard Hutcheson. (2011, June) The Ohio State University. [Online]. http://www.cse.ohio-state.edu/~hutcheso/classes/670/notes/relational-model.html

[40] Václav Papež, "Diplomová práce - Neuroinformatická databáze a sémantický web," Plzeň, 2010.

List of figures

List of figure

List of tables

List of Abbreviations

API – Application Programming Interface

BLOB – Binary Large Object

CLOB – Character Large Object

DB – Database

DTD – Document Type Definition

EEG – Electroencephalography

ERP – Even-Related Potentials

fMRI – Functional Magnetic Resonance Imaging

INCF – International Neuroinformatics Coordinating Facilities

MEG - Magneto Encephalography

NIF – Neuroinformatics Information Framework

OO – Object-oriented

OODB – Object-oriented database

OR – Object-relational

ORDB – Object-relational database

OWL – Ontology Web Language

PET – Positron-Emission Tomography

POJO – Plain Old Java Object

R – Relational

RDB – Relational database

RDF – Resource Description Framework

RDFS – Resource Description Framework Schema

RIF - Rule Interchange Format

SQL – Structure Query Language

SPARQL - SPARQL Protocol and RDF Query Language (Recursive acronym)

SVN – Subversion repository

List of Abbreviations

UML	Unified Modeling Language
URI	Unified Resource Identifier
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	eXtensible Markup Language
XSD	XML Schema

Appendix A – Current ER model of EEG/ERP Portal

Appendix A is due to its size only a part of electronic supplement.

Appendix B – Generated XML

An XML file describes the UML model of the University example by Visual Paradigm is part of electronic supplement only.