

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

## **Bachelor Thesis**

# **Semantic web - format for the exchange of EEG/ERP data**

# Statement

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

Pilsen, 6th May 2012

Jakub Daněk

# Abstract

This thesis provides a general description of the most common use-cases of the semantic web and its technologies and provides theoretical background for semantic web extension of the EEG/ERP Portal.

It compares expressivity of OWL 2, Java and UML and describes possibilities of mapping between these languages. The thesis also briefly covers possible performance problems and describes known solutions.

As a result, the thesis presents an ontology for the EEG/ERP experiment.

# Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Context</b>	<b>2</b>
2.1. Semantic Web . . . . .	2
2.1.1. OWL 2 . . . . .	2
2.1.2. OWL 2 Document Syntax . . . . .	3
2.1.3. Existing Projects . . . . .	4
2.2. NIF Portal . . . . .	6
2.3. EEG/ERP Portal . . . . .	7
<b>3. OWL 2, Java and UML Comparison</b>	<b>8</b>
3.1. Basic Elements . . . . .	8
3.2. Property Restrictions . . . . .	11
3.3. Complex Classes . . . . .	14
<b>4. Reasoning Performance</b>	<b>18</b>
4.1. OWL 2 Profiles . . . . .	18
4.1.1. OWL 2 EL . . . . .	20
4.1.2. OWL 2 QL . . . . .	21
4.1.3. OWL 2 RL . . . . .	22
<b>5. Ontology for EEG/ERP Portal</b>	<b>23</b>
<b>6. Conclusion</b>	<b>26</b>
<b>A. Additional Samples</b>	<b>31</b>
A.1. Basic Elements . . . . .	31
A.2. Property Restrictions . . . . .	33
<b>B. The EEG/ERP Portal's Ontology</b>	<b>35</b>
B.1. Ontology File Excerpt . . . . .	35
B.2. Ontology Tree Preview . . . . .	36

# 1. Introduction

At the moment there are many laboratories all over the world where neurological research is being commenced. Each of them has its own way of persisting records related to their measurements. Such case is also the EEG/ERP Portal (referenced as “the Portal” from now on) at the Department of Computer Science and Engineering at the Faculty of Applied Sciences, University of West Bohemia in Pilsen. It is designed for online sharing of measured data among laboratories with the same or similar aim of research.

One of the current development goals is to register the Portal in the NIF<sup>1</sup>, which serves as an international search engine and access point to neuroscience data and resources. To achieve that, both the data and its model must be presented in a format of the semantic web.

The aim of this thesis is to compare expressive power of UML, Java programming language and OWL<sup>2</sup>, which is the main language for technologies based on the idea of the semantic web. Another goal is to provide unified insight into common use-cases, possible utilisation and main benefits resulting from adapting the idea of “Web of Data”. As a result an OWL-based format is prepared for the Portal. It will represent the same or even enhanced semantic information about the data as current, conventional, entity model.

The first chapter provides general overview of the semantic web and its technologies and describes several cases in which these technologies have been successfully used.

The second chapter contains description of the OWL 2 expressive power and provides mapping possibilities between OWL 2 and Java or UML. The following chapter briefly describes complexity problems related to the semantic web technologies and known solutions leading to acceptable performance.

The last chapter presents the implementation details of the ontology for the Portal.

---

<sup>1</sup>Neuroscience Information Network

<sup>2</sup>Ontology Web Language

## 2. Context

### 2.1. Semantic Web

The current World Wide Web contains a huge amount of data: corporate information, company presentations, community portals, social networks, all kinds of multimedia, web-stores etc. However, all these data are controlled by applications and it is not possible to combine and use them efficiently.

For example if you want to search for a list of books by your favourite author, current search engines return, among others, a lot of pages which only mention the author's name. The problem is that search algorithms do not have any information about the semantics of the data and therefore must search for every invocation of the wanted term. The Semantic Web is supposed to be the solution for this issue. It is all about data and connections between them. By providing additional information (metadata) about relationships between data, applications should be able to provide enhanced functionality.

Another, and not completely unrelated, purpose of semantic web technologies is an extensible and powerful exchange format for sharing pieces of information among applications. E.g. current tagging systems work only within the universe of a particular application - even if the tag value is the same, there is no proof that applications understand and handle it in a compatible way. Semantic web technologies create bridges between these universes.

For more detailed description of basic concepts of the Semantic Web see [1].

#### 2.1.1. OWL 2

OWL 2 is a language for expressing ontologies. An ontology is a set of precise descriptive statements about some part of the world (usually referred to as the domain of interest or the subject matter of the ontology).[2]

OWL is supposed to describe a particular state of the domain of interest by storing data and their metadata (structure, relationships among data, etc). One of the major differences between OWL documents and standard databases (except for the used technology) is so called "open-world" assumption. If a statement is not present in a database, we presume it is false. While for an ontology, we have to assume that the statement might be true, only missing.

OWL documents are based on RDF<sup>1</sup>, which is a framework for describing information about resources on the World Wide Web. The metadata written in RDF are machine-readable and the framework is suitable for cases, when information should be processed by computers rather than displayed to a user. It has been designed for data exchange between applications.

The main concept of RDF works with object-predicate-subject tripples, which form RDF graphs. The following example and more information on the topic can be found in [3]. The example describes several facts:

- “me” is a Person
- Full name of the person is “Eric Miller”
- “Eric Miller” has a personal title “Dr.”

```
<contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
  <contact:fullName>Eric Miller</contact:fullName>
  <contact:personalTitle>Dr.</contact:personalTitle>
</contact:Person>
```

The OWL works on the same principle. In its full version it is a semantic extension of RDF. OWL 2 Profiles (described in chapter 4) put restrictions on the structure of OWL 2 ontologies [4]. Because of these restrictions, not all RDF documents comply with each of OWL 2 Profile specifications.

### 2.1.2. OWL 2 Document Syntax

Ontology documents can be, by specification, written in five defined syntaxes. As one can see in the Table 2.1, only RDF/XML is mandatory and therefore all OWL 2 conformant tools must be able to parse ontologies written in this syntax.

Name of Syntax	Status	Purpose
RDF/XML	Mandatory	Interchange
OWL/XML	Optional	Easier to process by XML tools.
Functional Syntax	Optional	Easier to see the formal structure of ontologies.
Manchester Syntax	Optional	Easier to read/write DL ontologies.
Turtle	Optional	Easier to read/write RDF tripples.

Table 2.1.: Table of OWL 2 Syntaxes [5]

The most significant are RDF/XML, OWL/XML and Functional Syntax. The RDF/XML syntax results in a document containing an RDF graph with OWL vocabulary. As it is the only mandatory syntax, it has been commonly used.

<sup>1</sup>RDF - Resource Description Framework

The functional syntax has been created for easy and compact description of various OWL definitions - overall specification, profiles, etc. it is not meant for use in software implementations (though it is not prohibited to use it). It is the main syntax in all W3C Recommendation documents regarding OWL 2.

The OWL/XML syntax is designed as an alternative ontology exchange media to the RDF/XML. it has been created with emphasis on easy parsability by XML tools.

The XML serialization mirrors the structural specification of OWL 2 and is defined by means of an XML schema plus some additional constraints in prose. [6]

The XSD schema definition can be found in [6]. The OWL/XML syntax has been used for all OWL code examples in this document (unless stated otherwise), because it is more transparent for the reader than RDF/XML.

### **2.1.3. Existing Projects**

Around 45 use-case studies can be found in [7], from which more than 30 have been already deployed within a production system, while the rest remain in a prototype or planning state. Projects vary from a music archive through a diagnostic system to the integration of separated web portals.

Most of the studies promote the open standard and easy extendability of the semantic web format. These attributes make it suitable for data integration and metadata frameworks. Only a few projects try to use reasoners' inference capabilities. Extracts from several case studies have been included below to provide a general picture of semantic web technology usage.

#### **Digital Music Archive [8]**

The project of Digital Music Archive for the Norwegian National Broadcaster aims on digitalizing the content of less resistant media as tapes or CD's to ensure its persistence for the future. The main goals of the project were:

- provide easy access to the archive's content
- show relations among the data which were not obvious or easy to deduce
- implement comprehensive view on the archive's content

Semantic web technologies have not been, however, used as a data storage medium in this case. Because a large amount of data had been expected, a scalable RDBMS was deployed and semantic web functionality was provided by the presentation layer.



According to the study the goals have been achieved successfully, providing an easily searchable archive with vast support for structured metadata (music file formats, links to related pictures or interviews etc.). Plus there is the possibility of integration with other systems or databanks without significant complications.

### **Detection of Patients at Risk of Organ Failure [9]**

The project of Detection of Patients at Risk of Organ Failure through Immune Rejection tries to utilise the ability of semantic-web-based technologies to make inferences from the data in a domain. The system is supposed to provide a solid knowledge base for decision-making assistance. The data format integrity should ensure coherence and extensibility. That makes possible to create, qualify and validate hypotheses despite huge complexity of medical data and resources.

The project has not been applied to the common healthcare systems yet, as it faces several problems (complexity of the domain, different data standards in the community, etc.).

### **Ordnance Survey [10]**

Ordnance Survey is the Great Britain's national mapping agency. Their geographical data are important for large scale of customers, including the government of GB. Common usecase requires integration with customer's own data. Semantic Web technologies allow users to see differences between the two domains (Ordnance Survey's and customer's data) and special ontologies can be created as an interface between them.

The interfaces can be divided into two categories. They are either created for cases when one object has different names in each domain, or for cases when a name stands for something different in each databank. An example has been mentioned in the study:

... is the Ordnance Survey definition of a field the same as the way the Government Department for the Environment, Food and Rural Affairs (DEFRA) would understand it? Our definition is spatially delineated by barriers such as fences or ditches, while DEFRA might distinguish a field's extent by the kind of crops that were grown, as part of their task of calculating farmers' subsidies.[10]

## BBC Web Sites [11]

The content published by BBC online used to be scattered over many microsites, which, however, had not been connected together until 2007. The case study describes the situation:

These sites can be very successful, but tend not to link together, and so are less useful when people have interests that span programme brands or domains. For example, we can tell you who presents Top Gear, but not what else those people have presented. As a user it is very difficult to find everything the BBC has published about any given subject, nor can you easily navigate across BBC domains following a particular semantic thread. For example, until recently you weren't able to navigate from a page about a programme to a page about each artist played in that programme. [11]

Using semantic web standards, the BBC sites have been interconnected allowing users to navigate from one webpage to another following the given topic. Another advanced functionality is extended search capability coming from the data integration. The BBC web developers have adapted the idea, that the Internet is its own CMS<sup>2</sup>, which seems to fully correspond with the idea of the semantic web - web of data.

## 2.2. NIF Portal

As can be found in [12], Neuroscience Information Framework is a web portal designed for sharing neuroscience resources, including data, tools and other materials or documents. NIF tries to advance neuroscientific research by enabling worldwide access to public research data.

The portal provides the following services[12]:

- A search portal for researchers, students, or anyone looking for neuroscience information, tools, data or materials.
- Tools for resource providers to make resources more discoverable, e.g., ontologies, data federation tools vocabulary services
- Tools for promoting interoperability among databases
- Standards for data annotation
- Best practices for creating discoverable and interoperable resources

and more...

---

<sup>2</sup>CMS - Content Management System

## 2.3. EEG/ERP<sup>3</sup> Portal

The EEG/ERP Portal (from now on referenced as “the Portal”) is a web application developed at the Department of Computer Science and Engineering, University of West Bohemia in Pilsen. Neuroscientific research at the department, focused on event-related potentials, produces a huge amount of test data, which need to be accessible by various research teams or individuals. The portal aims to provide a sharing platform for research data, metadata, scenarios and more. The main goal is to make cooperation among team members or even whole teams easier and therefore more efficient.

As the portal tries to join the current neuroscience community, it follows INCF<sup>4</sup> effort to standardize database models used for storing neuroscientific research data[13]. The next step is registering the Portal as a recognized resource of NIF. One of the conditions is to provide data and metadata in the format of the semantic web - OWL. Therefore the Portal’s developers have been experimenting with automatic data transformation from the object-oriented model into the ontology model. Services for generating ontology files containing either the whole content of the database, or the data model only, are available.

The semantic web technologies have a vast potential. The OWL 2 represents an open, consistent and extensible data format. It has been used in various projects as data exchange format or metadata description format. Because of both implementation complexity and performance issues, it was not commonly used as a data storage medium. Therefore it seems that the technology is not ready to replace the current World Wide Web at the moment. However, there are use-cases in which the OWL 2 format is a suitable solution.

---

<sup>3</sup>ERP - Event-related Potential

<sup>4</sup>INCF - International Neuroinformatics Coordinating Facility

## 3. OWL 2, Java and UML Comparison

This section covers description of OWL 2 elements and their comparison with Java programming language and UML. If an OWL expression does not have any equivalent in Java, a recommendation for replacement is provided. Examples are given for Java or UML (if exist) and OWL. For more examples and comparison with UML see appendix A. All the OWL 2 samples are given in OWL/XML syntax. The examples used in this chapter have been taken from [2].

### 3.1. Basic Elements

Basic units of OWL 2 model are individuals, classes and properties. All of the elements have an equivalent in both UML nad Java.

#### Individuals

An individual in OWL 2 is an instance of a particular class. All the individuals belong to the universal class `Thing`.

The main difference between OWL and UML in respect of instances is that in OWL an individual can be an instance of `Thing` and not necessarily of any other class, so could be outside the system in a UML model.[14]

In Java, instances represent actual data which are referenced from the code (by variables). Therefore the data have to be fetched from a database before they are saved into an OWL file.

#### Individuals and Unique Name Assmption

By default, OWL 2 lacks the unique name assumption. It means that unless explicitly stated, the reasoner supposes that two individuals of different names may actually be the same instance (in the case when it is not made impossible by another axiom - e.g. they could belong to different classes which are complements of each other). To precisely state

which names refer to the same instance and which do not, two elements are provided by OWL 2 - `DifferentIndividuals` and `SameIndividuals`.

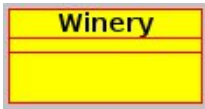
In case we are using OWL to describe an entity schema, all the instances in the document (or namespace) do not have to be explicitly stated as mutually different (using `DifferentIndividuals` element) to enforce unique names. If we presume that all individuals belong to precisely one class (entity), and all classes of the model are stated to be disjoint, then their individuals cannot be the same objects. All what must be done is to explicitly mark that all instances of each class are mutually different.

## Classes

A class represents a set of individuals. In OWL 2, each class is a sub-class of the universal class `Thing`. Classes are represented in both UML and Java. In all three languages tree-shaped class hierarchy is supported, however, Java does not allow full multiple inheritance - only interfaces are provided (no attributes).

### Sample

Definition of the class `Winery`.

Java: <code>class Winery {...}</code>	OWL 2: <code>&lt;Class IRI='Winery'/&gt;</code>
UML: 	

## Properties

In OWL 2 properties represent relationships between individuals or between an individual and a literal. In UML they are represented by association connection and in Java they are replaced by class attributes.

Properties are not necessarily tied to classes. By default, a property is a binary relation between `Thing` and `Thing`.<sup>[14]</sup>

There are two basic Property sub-classes - `ObjectProperty` and `DataProperty`. The first one represents a relation between two individuals and the latter connects a class to an elementary datatype (i.e. literal).

Properties are defined using two elements:

- **domain** - defines the classes which the individuals having this property belong to.
- **range** - defines the classes or datatypes the values of this property belong to.

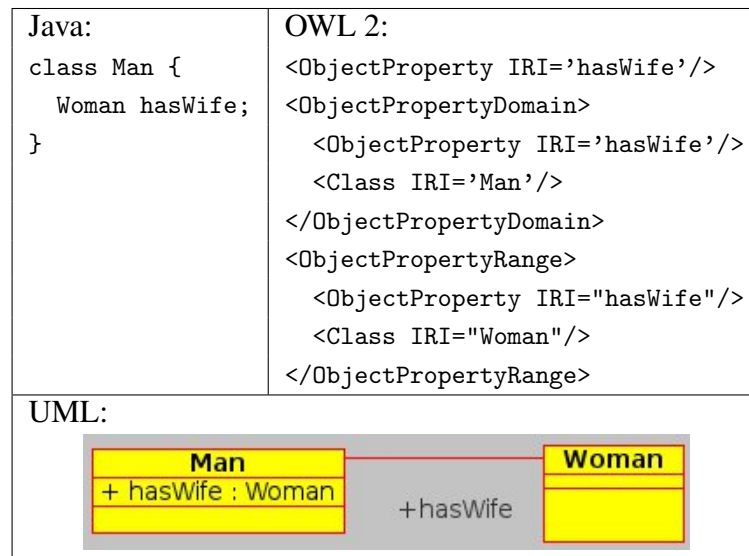
It is important to realize that domain does not serve as a restriction. It just allows the reasoner to deduce more information. If we say that an individual had been made from a grape (property `madeFromGrape`), it automatically becomes an instance of the class `Wine`, even though it's actually an instance of the class `Cognac`.

Similar logic applies to range. Consider having a property `hasPet` with its range set to `Dog` and `Cat`. Then expression `Pete hasPet Fluffy` implies that `Fluffy` is a dog and a cat at the same time.

Further property restrictions must be used to fully represent Java or UML class attributes, including type safety or cardinality restrictions. This will be covered in the section 3.2.

## Sample

The class `Man` has the property `hasWife`, which is a reference to the class `Woman`.



## Sub-Properties

In OWL 2 properties can be, as well as classes, arranged in a tree-shaped hierarchy. That can be used for encapsulation support. E.g. if the class `Mother` extends the class `Parent`, then the property `hasMother` is a sub-property of the property `hasParent`.

Depending on the usecase, one has to be careful with encapsulation. As an example consider having a property `hasParent` with both domain and range set to `LivingThing`. Without any further restriction, this statement would e.g. allow the expression that parents of a human child are a squirrel and a crocodile.

When mapping the Java entity model into OWL 2, sub-properties do not have to be used, because class hierarchy contains all required information. However, if methods were used for mapping instead of attributes, sub-properties would be used for describing the connection between interface methods and their implementation.

## Sample

The property `hasWife` is a sub-property of the property `hasSpouse`.

```
OWL:
<SubObjectPropertyOf>
  <ObjectProperty IRI='hasWife' />
  <ObjectProperty IRI='hasSpouse' />
</SubObjectPropertyOf>
```

## 3.2. Property Restrictions

Because it is possible to infer from the properties of an individual that it is a member of a given class, we can think of the complex classes and property restrictions as a sort of predicate definition language.[14]

While Java does not have any means for predicate definition, UML standard provides a predicate definition language OCL<sup>1</sup>.

The Object Constraint Language (OCL) is a textual sublanguage of the Unified Modelling Language (UML). It can be used to express additional constraints on UML models that cannot be expressed, or are very difficult to express, with the graphical means provided by UML. OCL is based on first-order predicate logic but it uses a syntax similar to programming languages and closely related to the syntax of UML. It is, thus, more adequate for every-day modelling than pure first-order predicate logic.[15]

For majority of OWL 2 elements described in this section and section 3.3 applies that there is basically no equivalent in Java and very poor or none support in UML. OCL might supply the described semantics for the UML models, however its description lays beyond the scope of this thesis. The specification can be found in [16].

## Quantification

Quantification property restrictions can be used to enforce that property values are of the demanded class or type and/or that each individual of a class has particular number of the property's values assigned.

---

<sup>1</sup>Object Constraint Language

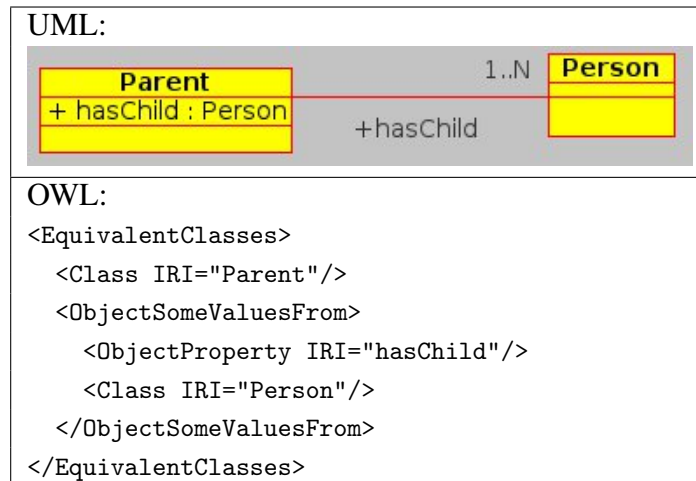
## Existential Quantification

Existential quantification in OWL is represented by a tuple of elements - `ObjectSomeValuesFrom` and `DataSomeValuesFrom`. If used in a class definition, they specify that each individual belonging to the class must have at least one property value of the given class or data range (e.g. to be a parent, one must have at least one child).

In UML the equivalent is minimum cardinality set to one. Java does not provide any native way to describe such rule (except for the keyword `final`, which, however, brings other restrictions too). However, it can be implemented using annotations (for cardinality restrictions the annotations have been already implemented in `javax.persistence` package).

## Sample

Example describing that every parent must have at least one child.



## Universal Quantification

Universal quantification in OWL is represented by a tuple of elements `ObjectAllValuesFrom` and `DataAllValuesFrom`. In class definition it specifies for all the individuals of the class that all values of the property must belong to a given class or data type. Unlike the range attribute of properties this one enforces the actual restriction of property data type. That is also the natural functionality of Java and UML class attributes.



## Sample

The code in this example specifies, that every child of a human must be also a human.

```
OWL:
<EquivalentClasses>
  <Class IRI="Human"/>
  <ObjectAllValuesFrom>
    <ObjectProperty IRI="hasChild"/>
    <Class IRI="Human"/>
  </ObjectAllValuesFrom>
</EquivalentClasses>
```

## Cardinality Restrictions

Cardinality specifies a number of individuals involved in the relation. OWL provides elements to set minimum, maximum or exact cardinality on either property range or domain. All variations also have a qualifiedCardinality version, which allows to restrict cardinality on a specified class (as mentioned above, the class of property values is not strictly given unless universal quantification is used). E.g. different cardinality restriction can be used for sons and daughters while defining hasChild property.

Java has some limited means to set cardinality - e.g. array size for maximum cardinality or key word `final` for minimum cardinality etc. However, there are not any means to specify cardinality for collections. Enforcing usage of arrays would be too restrictive. Special annotations could be designed and used (and are in the `javax.persistence` package). On the other hand, UML has specified means to state multiplicity on both ends of an association.

## Functional Properties

A special case of property restrictions in OWL are `functionalProperty` and `inverseFunctionalProperty`. Functional property has maximum cardinality restriction set to 1 on its range, while inverse functional property has it on its domain.

## Other Restrictions

OWL also provides an option to give an exact property value for the given class using `hasValue` element. E.g. in definition of the class `JohnsChildren` the property `hasParent` can be set to be always of the value `John`. Such behaviour is unknown to UML or Java and could be simulated by annotations and enums. However, it is very difficult to find a reasonable and generic use for similar functionality.

Another case is definition of inverse properties (e.g. `hasChild` and `hasParent`). This element can be used for mapping of a binary association with two navigable ends in UML. Annotations have to be used in Java to implement such feature.

### 3.3. Complex Classes

As mentioned in section 3.2, most of the elements mentioned in this chapter do not have any direct equivalent neither in Java nor UML.

#### Class Expression

In OWL 2, classes and property expressions are used to construct class expressions, sometimes also called descriptions, and, in the description logic literature, complex concepts. Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions.[17]

In other words a class is only a named set of individuals without any further restrictions or specifications. To define special attributes of individuals, class expressions have to be used. Class expressions do not have names by default, however they can be stated as equivalent to a class and further referenced through the class' name.

Axioms of the form `EquivalentClasses( C, CE )`, where `C` is a class and `CE` is a class expression, are often called definitions, because they define the class `C` in terms of the class expression `CE`. [17]

`EquivalentClasses` element is used to declare that two classes (or class expressions) have the same semantic value - they contain the same set of individuals.

The element `DisjointClasses` has the opposite meaning which specifies that every individual can be an instance of maximum 1 class included in the axiom - classes do not share individuals. Instances in Java always belong to exactly one class, therefore `DisjointClasses` element should be used for all the classes at the same level in a hierarchy tree. Making two classes at the same hierarchy branch (a class and its sub-class) disjoint will result in an inconsistent ontology.

UML provides the `disjoint` constraint which can be added to a set of generalisation relationships. It states, that an instance of the parent entity can be an instance of only one of the children within the set.

## Complex Class Definitions

As mentioned earlier, `EquivalentClasses` element is used for definitions of complex class expressions.

### ObjectIntersectionOf

An intersection class expression `ObjectIntersectionOf( CE1 ... CEn )` contains all individuals that are instances of all class expressions `CEi` for  $1 \leq i \leq n$ . [17]

E.g. presume that the class expression `Mother` is an intersection of the classes `Parent` and `Woman`. Using this element the reasoner can infer that `Mother` is a subclass of `Parent` and `Woman`, therefore any instance of the class `Mother` is also an instance of both the classes. Unlike the `subClassOf` element, intersection allows inference of the fact that any individual which is an instance of the classes `Parent` and `Woman` is also an instance of class expression `Mother`.

This element cannot be used when representing the entity model, because it is impossible for an object to be an instance of two different classes in Java (unless Java interfaces are mapped as OWL classes as well).

### Sample

The example describing that every mother is a woman and a parent and vice versa.

```
OWL:
<EquivalentClasses>
  <Class IRI="Mother"/>
  <ObjectIntersectionOf>
    <Class IRI="Woman"/>
    <Class IRI="Parent"/>
  </ObjectIntersectionOf>
</EquivalentClasses>
```

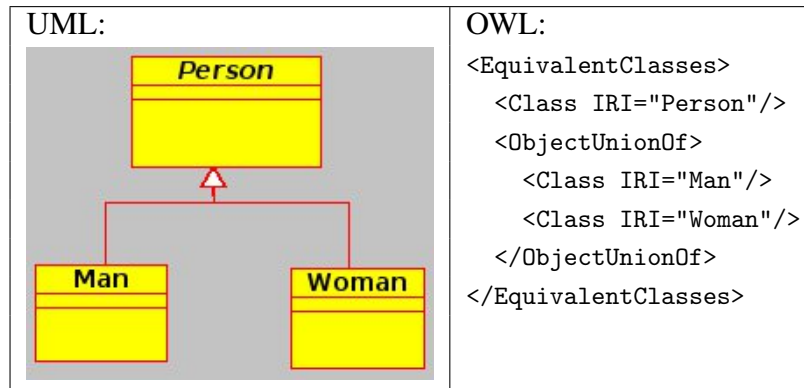
### ObjectUnionOf

A union class expression `ObjectUnionOf( CE1 ... CEn )` contains all individuals that are instances of at least one class expression `CEi` for  $1 \leq i \leq n$ . [17]

In UML and Java, all abstract superclasses can be considered unions of their subclasses.

## Sample

Class expression Person can be described as the union of classes Man and Woman.



## ObjectComplementOf

A complement class expression ObjectComplementOf( CE ) contains all individuals that are not instances of the class expression CE. [17]

ObjectComplementOf is the logical negation of the union element. E.g. the class ChildlessPerson can be represented as intersection of a class Person and the complement of class Parent.

This element could be used to state that each individual belongs to exactly one class by making all the classes complement of the others. However, if  $n$  was the number of classes in the domain, it would result in  $\frac{n \times (n-1)}{2}$  ObjectComplementOf element occurrences. Therefore it is much more efficient to use DisjointClasses element for this purpose.

## Sample

The code in this example states that all individuals of the class Person who are not parents belong to the class ChildlessPerson and vice versa.

```
OWL:
<EquivalentClasses>
  <Class IRI="ChildlessPerson"/>
  <ObjectIntersectionOf>
    <Class IRI="Person"/>
    <ObjectComplementOf>
      <Class IRI="Parent"/>
    </ObjectComplementOf>
  </ObjectIntersectionOf>
</EquivalentClasses>
```

## ObjectOneOf


An enumeration of individuals  $\text{ObjectOneOf}(a_1 \dots a_n)$  contains exactly the individuals  $a_i$  with  $1 \leq i \leq n$ . [17]

Defining a class by enumerating all its individuals ensures (together with unique name assumption), that all not-mentioned elements are automatically considered being instances of the complement class expression of the given class.

This element has, unlike others in this chapter, an equivalent in both Java and UML. In Java, they are represented by the enum keyword and UML has the `<<Enumeration>>` literal.

### Sample

Example of cardinal directions' enumeration.

<b>Java:</b> enum Directions { North, East, South, West }	<b>OWL:</b> <EquivalentClasses> <Class IRI="Directions"/> <ObjectOneOf> <NamedIndividual IRI="North"/> <NamedIndividual IRI="East"/> <NamedIndividual IRI="South"/> <NamedIndividual IRI="West"/> </ObjectOneOf> </EquivalentClasses>
<b>UML:</b> 	

## 4. Reasoning Performance

So-called reasoners are applications capable of processing ontologies. Their performance is measured in respect to their most common and important tasks [18]. Those are classification, consistency check, satisfiability, subsumption query and soundness and completeness.

**Classification** is ontology domain hierarchy computing. it is one of the most important reasoning tasks. Especially during development, which requires often check for unwanted or unexpected subsumptions.

**Consistency check** is performed to see whether all the statements about individuals satisfy the axioms regarding the ontology domain structure.

**Satisfiability check** tests whether it is possible to find instances satisfying the ontology concept.

**Subsumption query** checks whether a concept can be inferred from another one or returns all concepts subsumed from (or inferring) a concept.

**Soundness and completeness** represents the quality of a reasoner's output. It depends on whether all subsumptions are sound based on the underlying semantics and whether they are complete [18]. That is whether all facts have been inferred. For many ontologies set of all statements based upon given concept is unknown. Therefore soundness and completeness of a reasoner can be very often evaluated only by comparing results of multiple reasoners.

### 4.1. OWL 2 Profiles

OWL 2 provides a high expressive power, which together with the open-world concept brings huge performance problems. As it brings almost no general assumptions and no computational guarantees into the ontology, it is more likely a theoretical concept than a usable technology. In order to gain acceptable reasoning times, subsets of the language, called profiles, have been specified. Each of the profiles aims for different use cases and keeps different parts of OWL expressiveness. it is very important to choose an appropriate profile for the considered project.

At the moment there are three main OWL 2 profiles defined - OWL 2 EL, OWL 2 QL and OWL 2 RL. However, as stated in [4], there are other profiles, e.g extending one of those mentioned above. OWL 2 is fully backward-compatible with OWL 1. Therefore OWL DL and OWL Lite, which have been specified for OWL 1, can be considered as profiles of OWL 2. However, this thesis focuses only on the profiles created directly for OWL 2.

In order to describe computational complexity of each of the profiles, several aspects have been considered [4]:

- **Data Complexity:** the complexity measured with respect to the total size of the assertions in the ontology.
- **Taxonomic Complexity:** the complexity measured with respect to the total size of the axioms in the ontology.
- **Query Complexity:** the complexity measured with respect to the total size of the query.
- **Combined Complexity:** the complexity measured with respect to both the size of the axioms, the amount of assertions and, in the case of conjunctive query answering, the size of the query as well.

Most common reasoning problems are ontology consistency, class expression satisfiability, class expression subsumption and instance checking. It will be referenced as T in tables further in this document. Conjunctive query answering will be referenced as Q.

The tables contain the following complexity classes [4]:

- **P**TIME: problems solvable by a deterministic algorithm using time that is at most polynomial in the size of the input (i.e., roughly  $n^c$ ,  $n$  is the size of the input and  $c$  is a constant).
- **P**SPACE: problems solvable by a deterministic algorithm using space that is at most polynomial in the size of the input (i.e., roughly  $n^c$ ,  $n$  is the size of the input and  $c$  is a constant).
- **NP**: problems solvable by a nondeterministic algorithm using time that is at most polynomial in the size of the input (i.e., roughly  $n^c$ ,  $n$  is the size of the input and  $c$  is a constant).
- **NLogSpace**: problems solvable by a nondeterministic algorithm using space that is at most logarithmic in the size of the input (i.e., roughly  $\log(n)$ ,  $n$  is the size of the input).

- **AC<sup>0</sup>**: the class of problems definable using a family of circuits of constant depth and polynomial size, which can be generated by a deterministic Turing machine in logarithmic time (in the size of the input).

A problem can be **complete** within one of the classes mentioned above. It means that an algorithm which solves the problem is known to be inside the class and it had been proved that the problem is at least as hard as any other within the class.

#### 4.1.1. OWL 2 EL

OWL 2 EL is a profile recommended for simple but large ontologies and polynomial time reasoning.

Besides negation and disjunction, OWL 2 EL disallows universal quantification on properties. This fact disallows implementation of “type-safe” properties, because the ontology cannot contain axioms restricting properties only to a given type. Also, there is no way of specifying that, say, `parentOf` and `childOf` are invert to each other.

Other restrictions involve (see [4] for complete specification):

- cardinality restrictions are not allowed
- enumerations involving more than one individual are not allowed
- functional properties are not allowed
- anonymous individuals (without name) are not allowed
- boolean type is not allowed
- floating point number types from `xsd` have been replaced by `owl:real` and `owl:rational`

**Reasoner support:** CB, CEL, ELLY, Pellet, SHER, snorocket

Computing complexity values for OWL 2 EL are in the Table 4.1:

-	Data Complexity	Taxonomic Complexity	Query Complexity	Combined Complexity
T	P <sub>TIME</sub> -complete	P <sub>TIME</sub> -complete	-	P <sub>TIME</sub> -complete
Q	P <sub>TIME</sub> -complete	P <sub>TIME</sub> -complete	NP-complete	P <sub>SPACE</sub> -complete

Table 4.1.: OWL 2 EL complexity



### 4.1.2. OWL 2 QL

This profile is suitable for rather light ontologies with large amount of data (individuals). It provides good interoperability with relational databases. Its expressive power can be used to represent key features of entity-relationship and UML diagrams.

Different restrictions apply to axiom definition for superclasses and subclasses. E.g. existential quantification is not allowed for a class expression or data range in the subclass position.

It is very important that only atomic classes are allowed in class assertions. Therefore individuals can belong to a class, but not to an anonymous class expression. Nevertheless, classes can be defined (see section 3.3 in this document) in terms of a class expression. All individuals must have a name, anonymous instances are not allowed. These rules try to follow the common relational model, where anonymous entities are not permitted and each entity instance can be recognized via primary key.

Other restrictions involve (see [4] for complete specification):

- enumerations of individuals and literals are not allowed
- cardinality restrictions are not allowed
- property chains are not allowed
- functional properties are not allowed
- existential quantification to an individual or a literal is not allowed
- individual equality assertions are not allowed (can not state that two individuals are equal)
- boolean type is not allowed
- floating point number types from `xsd` have been replaced by `owl:real` and `owl:rational`

**Reasoner support:** Owlgres, OWLIM, Quill, QuOnto

Computing complexity values for OWL 2 EL are in the Table 4.2:

-	Data Complexity	Taxonomic Complexity	Query Complexity	Combined Complexity
T	$AC^0$	NLogSpace-complete	-	NLogSpace-complete
Q	$AC^0$	NLogSpace-complete	NP-complete	NP

Table 4.2.: OWL 2 QL complexity

### 4.1.3. OWL 2 RL

OWL 2 RL is a profile that can easily operate, using rule extended DBMSs, on data in RDF tripples.

It is designed to accommodate both OWL 2 applications that can trade the full expressivity of the language for efficiency, and RDF(S) applications that need some added expressivity from OWL 2.[4]

OWL 2 RL provides quite extensive expression power, because scalable reasoning is achieved more by specifying particular conditions under which elements can be used than by constraining the set of elements available. This makes it a little more complicated to maintain than other profiles. It is, however, extremely useful in cases where it is necessary to work on the data in RDF tripples directly[5].

**Reasoner support:** ELLY, Jena, Oracle Database 11g OWL Reasoner, OWLRL Computing complexity values can be found in the Table 4.3.

-	Data Complexity	Taxonomic Complexity	Query Complexity	Combined Complexity
T	PTIME-complete	PTIME-complete	-	PTIME-complete
Q	PTIME-complete	PTIME-complete	NP-complete	PSPACE-complete

Table 4.3.: OWL 2 RL complexity

# 5. Ontology for EEG/ERP Portal

Several decisions were made while preparing the ontology format for the Portal. Those included:

- which OWL 2 syntax to use
- which OWL 2 Profile to choose
- what semantics provide in the ontology in addition to the entity model schema semantics

## Syntax

RDF/XML was chosen as the major syntax for the project, because it is the only syntax, which has been specified as mandatory for OWL conformant tools. This should make the service processable by most of the neuroscience community members. The ontology is generated automatically using JenaBean Extension, a project developed at our university, which is based on JenaBean<sup>1</sup> library [19]. The current version of the library supports only OWL 1, therefore the OWL API framework<sup>2</sup> [20] has been used for transformation into the OWL 2<sup>3</sup> format. Support of other syntaxes depends on capabilities of these frameworks.

## Profile

The decision, which profile should be used for the Portal, was based on several criteria - size and complexity of the data model, expected amount of instances and main use-case for the data written in the ontology format. The first one rejected was OWL 2 RL, as it is designed for data stored in RDF tripples, which is not the case of the Portal.

The data model of the Portal contains around 60 entities. It is a very small and simple ontology, especially compared to large medical ontologies like SNOMED CT<sup>4</sup>, which contains more than 300 000 concepts connected by over a 951 000 relationships[21].

---

<sup>1</sup>JenaBean - Java Library for persisting JavaBeans to RDF

<sup>2</sup>OWL API - Java framework for managing OWL ontologies.

<sup>3</sup>Transformation from OWL 1 into OWL 2 is possible, because OWL 2 is fully backwards-compatible with OWL 1.

<sup>4</sup>SNOMED CT - large clinical terminology maintained by International Health Terminology Standards Development Organisation.

Compared to the amount of classes in the ontology, a large number of instances is expected. Data, measured by various research teams including students, are saved in the Portal for future research and evaluation. The Portal is also planned to store not only the experiment's data retrieved at the University of West Bohemia, but also become a sharing platform for the related community worldwide. In this case the number of data stored would grow even faster.

The ontology format has been prepared due to NIF's requirements for integration. It is therefore supposed to reflect the Portal's data model (and the relational database schema). All these facts support the choice of OWL 2 QL profile, as it has been designed for interoperability with relational databases and for representing simple ontologies with large amount of individuals. All the restrictions made to the OWL 2 syntax and semantics in this profile are supposed to provide compatibility with relational database's semantics together with ensuring certain computing complexity.

Therefore OWL 2 QL has been chosen as the profile for the Portal's ontology service. Its full specification can be found in [4].

## Semantics

Due to lack of requirements specification from the NIF, it has been decided for the moment that the ontology format for the Portal should reflect the semantics of its data model in Java and/or its representation in the relational database. This has been achieved by using Class and Property elements. Additionally, all classes have been specified as disjoint sets of instances.

Also some of XSchema datatypes have been prohibited and replaced by the profile specification. Resulting mapping rules are described in Table 5.1:

Java	OWL 2
Class	owl:Class
extends, implements	owl:SubClassOf
attributes	owl:ObjectPropertyOf or owl:DataPropertyOf
double	owl:real
boolean, long, int	xsd:integer

Table 5.1.: EEG/ERP Portal Java-OWL 2 mapping table

The selected profile specification bans usage of anonymous individuals, therefore each instance must have a name. Entity Id attribute represents instance's name.

The profile usage makes implementation of some of the entity model semantics impossible. Type safety and cardinality restrictions are not available. The type safety can be ensured by owl:ObjectAllValuesFrom and owl:DataAllValuesFrom elements - that is by universal quantification, which is banned in both OWL 2 QL and OWL 2 EL pro-

files as well as all cardinality-restrictive elements. In both cases it is due to performance drawbacks.

However, because the ontology is created automatically by the Portal, type safety is ensured by the Portal's implementation. Cardinality specification can be omitted without any major complications as well. The ontology is designed for data exchange and cardinality restrictions may vary among the recipients' applications. Proper documentation should suffice to allow correct processing of the ontology by recipients.

## Testing

Testing of the model consisted of three parts: validation against the profile specification, data model correctness and syntax the test.

Profile validity check was done using OWL 2 Validator service, which has been developed and run by the University of Manchester at the following URL:

`http://owl.cs.manchester.ac.uk/validator/`

The validator is capable of making a test for all three OWL 2 profiles, giving a human-readable feedback with highlighted specification violations in case any occurred.

The correctness of the data model was the most difficult to confirm. The testing consisted from manual checks of the ontology structure in Protege<sup>5</sup> against the entity model in UML, including inferred relationships. Because the ontology was generated automatically from annotated entity classes using services provided by JenaBean library, it has been considered generally correct. However, more sophisticated testing process should be developed in the future. Model consistency and satisfiability have been checked during the process by reasoners, which are distributed as a part of Protege.

The test for the correct syntax was done as a part of both profile validation and model correctness checks. Both the applications, the profile validator and Protege, make check for the correct structure and syntax of owl documents they are provided.

See appendix B for an excerpt from the ontology file and for a part of the ontology visualisation.

---

<sup>5</sup>Protege - an ontology editor. See <http://protege.stanford.edu/>

## 6. Conclusion

The Semantic Web brings an extension to the World Wide Web (WWW) as we know it. It enables more precise search, complex metadata description, easier interapplication communication and data integration, “hidden” relations discovery or data-driven machine-based decision-making. Use-case studies mentioned in this paper prove that it is possible to successfully implement this functionality. It is suitable for integration of applications working with data from the same or similar domains. Such case is also the NIF portal, which tries to provide search options across various neuroscience-related sites (as is e.g. the EEG/ERP Portal).

However, it is questionable where the bounds of the technology concept lie. While it might work quite well in these, in the context of WWW rather small, cases, the complexity of such solution for the whole web would be huge.

As one can see from the language comparison in this thesis, modelling of an ontology for an application requires different approach than conventional entity modelling. Partly it is because the expressive power of OWL is higher compared to conventional technologies, partly it is because of the “open-world” concept. The designer cannot rely on any default rules applied to the data domain.

Another obstacle is the computational complexity. The defined subsets of the OWL (i.e. profiles) provide, at cost of loosing expressivity, boundaries for both time and space required. However, for large ontologies, it might still make some of the semantic web functionality (especially inferring) difficult to use in real time.

For all these reasons it seems valid to presume that the Semantic Web will not replace the classic web concept anytime soon.

Ontology for the EEG/ERP Portal has been designed to represent similar semantics as the Portals entity model. This has not been achieved completely because some of required elements were prohibited by the OWL 2 QL profile which we used to ensure complexity boundaries. The lost pieces of functionality are type-safety and cardinality restrictions. Also the boolean (binary) type had to be replaced by the integer type. However, these are just minor parts of the semantics and can be omitted.

The Portal provides the basic ontology model, which can be easily extended - some further requirements may be requested by the NIF during the registration process. Therefore the main goal of this thesis has been fulfilled.

# List of Abbreviations

**API** Application Programming Interface

**CMS** Content Management System

**EEG** Electroencephalography

**ERP** Event-Related Potential

**INCF** International Neuroinformatics Coordinating Facility

**NIF** Neuroscience Information Framework

**RDBMS** Relational Database Management System

**RDF** Resource Definition Framework

**OCL** Object Constraint Language

**OWL** Ontology Web Language

**UML** Unified Modeling Language

**XML** Extensible Markup Language

# Bibliography

- [1] W3C. *W3C Semantic Web Activity*, Nov. 2001  
<http://www.w3.org/2001/12/semweb-fin/w3csw>, (accessed 15.1.2012)
- [2] W3C. *OWL 2 Web Ontology Language Primer*, rev. 27.10.2009  
<http://www.w3.org/TR/owl2-primer/>, (accessed 12.4.2012)
- [3] W3C. *RDF Primer*, rev. 10.2.2004  
<http://www.w3.org/TR/rdf-primer/>, (accessed 2.4.2012)
- [4] W3C. *OWL 2 Web Ontology Language Profiles*, rev. 27.10.2009  
<http://www.w3.org/TR/owl2-profiles/>, (accessed 16.3.2012)
- [5] W3C. *OWL 2 Web Ontology Language Document Overview*, rev. 27.10.2009  
<http://www.w3.org/TR/owl2-overview/>, (accessed 17.1.2012)
- [6] W3C. *OWL 2 Web Ontology Language XML Serialization*, rev. 27.10.2009  
<http://www.w3.org/TR/owl2-xml-serialization/>, (accessed 4.4.2012)
- [7] W3C. *Semantic Web Use Cases and Case Studies*  
<http://www.w3.org/2001/sw/sweo/public/UseCases/>, (accessed 10.4.2012)
- [8] Dr. Robert H.P. Engels, Jon Roar Tønnesen  
*Case Study: A Digital Music Archive (DMA) for the Norwegian National Broadcaster (NRK) using Semantic Web techniques*, Sep. 2007  
<http://www.w3.org/2001/sw/sweo/public/UseCases/NRK/>, (accessed 10.4.2012)
- [9] Robert Stanley, Bruce McManus, Raymond Ng, Erich Gombocz, Jason Eshleman, Charles Rockey  
*Case Study: Applied Semantic Knowledgebase for Detection of Patients at Risk of Organ Failure through Immune Rejection*, Mar. 2011  
<http://www.w3.org/2001/sw/sweo/public/UseCases/IOInformatics/>,  
(accessed 10.4.2012)
- [10] Catherine Dolbear. *Case Study: Semantic Web Technology at Ordnance Survey*, Mar. 2007  
<http://www.w3.org/2001/sw/sweo/public/UseCases/OrdSurvey/>,  
(accessed 10.4.2012)



- [11] Yves Raimond, Tom Scott, Patrick Sinclair, Libby Miller, Stephen Betts, Frances McNamara  
*Case Study: Use of Semantic Web Technologies on the BBC Web Sites*, Jan. 2010  
<http://www.w3.org/2001/sw/sweo/public/UseCases/BBC/>, (accessed 10.4.2012)
- [12] *Neuroscience Information Framework*, 2012  
<http://www.neuinfo.org/>, (accessed 20.1.2012)
- [13] Petr Ježek, Roman Mouček  
*Database of EEG/ERP Experiments*, in proc. of the Third International Conference on Health Informatics, Valencia, Spain, 2010
- [14] Lewis Hard, Patrick emery, Bob Colomb, Kerry Raymond, Sarah Taraporewalla, Dan Chang, Yiming Ye, Elisa Kendall, Mark Dutra  
*OWL Full and UML 2.0 Compared*  
version 2.4, 12. 3. 2004
- [15] *OCL Portal*, 2012  
<http://www-st.inf.tu-dresden.de/ocl/>, (accessed 20.1.2012)
- [16] Object Management Group. *Object Constraint Language*, ver. 2.2  
<http://www.omg.org/spec/OCL/2.2>, (accessed 20.1.2012)
- [17] W3C. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*, rev. 27.10.2009  
<http://www.w3.org/TR/owl2-syntax/>, (accessed 22.3.2012)
- [18] Kathrin Dentler, Ronald Cornet, Annette ten Teije, Nicolette de Keizer  
*Comparison of Reasoners for large Ontologies in the OWL 2 EL Profile*  
2011, Semantic Web, vol. 2, p. 71-87, ISSN 1570-0844
- [19] *JenaBean*, 2012  
<http://code.google.com/p/jenabean/>, (accessed 10.4.2012)
- [20] *OWL API*, 2012  
<http://owlapi.sourceforge.net/> , (accessed 10.4.2012)
- [21] The International Health Terminology Standards Development Organisation  
*SNOMED Clinical Terms® (SNOMED CT®) International Release*, July 2009  
[http://www.nlm.nih.gov/research/umls/Snomed/SNOMED\\_CT\\_July\\_2009\\_Scope\\_Memo.doc](http://www.nlm.nih.gov/research/umls/Snomed/SNOMED_CT_July_2009_Scope_Memo.doc),  
(accessed 10.4.2012)

# Appendices

# A. Additional Samples

## A.1. Basic Elements

### Class

#### OWL

```
<Class IRI='Wine' />  
<Class IRI='Drink' />  
<SubClassOf>  
  <Class IRI='Drink' />  
  <Class IRI='Wine' />  
</SubClassOf>
```

#### Java

```
class Drink {...}  
class Wine extends Drink {...}
```

#### UML

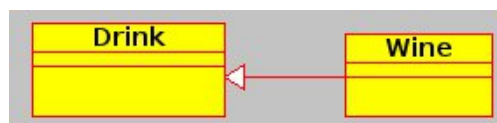


Figure A.1.: UML representation of class hierarchy

## Property

### OWL

```
<ObjectProperty IRI='grape' />  
<ObjectPropertyDomain >  
  <ObjectProperty IRI='grape' />  
  <Class IRI='Wine' />  
</ObjectPropertyDomain >  
<ObjectPropertyRange >  
  <ObjectProperty IRI='grape' />  
  <Class IRI='WineGrape' />  
</ObjectPropertyRange >
```

### Java

```
class Wine {  
  WineGrape grape;  
}
```

### UML

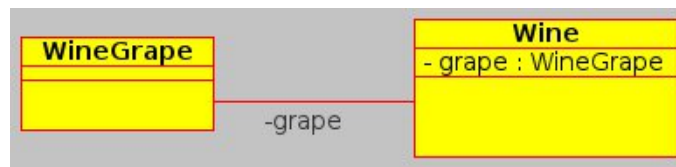


Figure A.2.: UML representation of attributes

## A.2. Property Restrictions

### Min/Max Cardinality

#### OWL

```
<EquivalentClasses >  
  <Class IRI='Car' />  
  <ObjectIntersectionOf >  
    <ObjectMaxCardinality cardinality="5" >  
      <ObjectProperty IRI="hasDoors" />  
      <Class IRI="Door" />  
    </ObjectMaxCardinality >  
    <ObjectMinCardinality cardinality="3" >  
      <ObjectProperty IRI="hasDoors" />  
      <Class IRI="Door" />  
    </ObjectMinCardinality >  
  </ObjectIntersectionOf >  
</EquivalentClasses >
```

#### UML



Figure A.3.: UML representation of min/max cardinality

## Exact Cardinality

### OWL

```
<EquivalentClasses >  
  <Class IRI='Bike' />  
  <ObjectExactCardinality cardinality="2">  
    <ObjectProperty IRI="hasWheels" />  
    <Class IRI="Wheel" />  
  </ObjectMinCardinality >  
</EquivalentClasses >
```

### UML

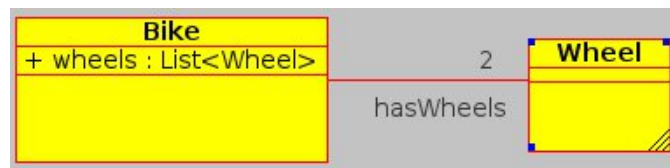


Figure A.4.: UML representation of min/max cardinality

# B. The EEG/ERP Portal's Ontology

## B.1. Ontology File Excerpt

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:j.0="http://thewebsemantic.com#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://kiv.zcu.cz/eegbase#">
  <owl:Ontology rdf:about="http://kiv.zcu.cz/eegbase"/>
  <owl:Class rdf:ID="Person">
    <j.0:javaclass>cz.zcu.kiv.eegdatabase.data.pojo.Person</j.0:javaclass>
  </owl:Class>
  <owl:Class rdf:ID="History">
    <j.0:javaclass>cz.zcu.kiv.eegdatabase.data.pojo.History</j.0:javaclass>
  </owl:Class>
  <owl:Class rdf:ID="IScenarioType">
    <j.0:javaclass>cz.zcu.kiv.eegdatabase.data.pojo.IScenarioType</j.0:javaclass>
  </owl:Class>
  <owl:Class rdf:ID="Scenario">
    <j.0:javaclass>cz.zcu.kiv.eegdatabase.data.pojo.Scenario</j.0:javaclass>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="histories">
    <rdfs:domain rdf:resource="http://kiv.zcu.cz/eegbase#Person"/>
    <rdfs:range rdf:resource="http://kiv.zcu.cz/eegbase#History"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="scenarioType">
    <rdfs:domain rdf:resource="http://kiv.zcu.cz/eegbase#Scenario"/>
    <rdfs:range rdf:resource="http://kiv.zcu.cz/eegbase#IScenarioType"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="scenarioLength">
    <rdfs:domain rdf:resource="http://kiv.zcu.cz/eegbase#Scenario"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  </owl:DatatypeProperty>
  <owl:AllDisjointClasses>
    <owl:members rdf:parseType="Collection">
      <owl:Class rdf:about="http://kiv.zcu.cz/eegbase#Scenario"/>
      <owl:Class rdf:about="http://kiv.zcu.cz/eegbase#Person"/>
      <owl:Class rdf:about="http://kiv.zcu.cz/eegbase#History"/>
      <owl:Class rdf:about="http://kiv.zcu.cz/eegbase#IScenarioType"/>
    </owl:members>
  </owl:AllDisjointClasses>
</rdf:RDF>
```

